UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE ESTUDIOS DE POSGRADO

TESIS

TRÁNSITO DE AVENIDAS EN CAUCES MEDIANTE REDES NEURONALES ARTIFICIALES

PRESENTADA POR

ING. JUAN PABLO MOLINA AGUILAR

PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA (HIDRÁULICA)

DIRIGIDA POR

DR. FRANCISCO JAVIER APARICIO MIJARES

CAMPUS MORELOS
AGOSTO 2005

Índice general

Agradecimientos y Dedicatorias

Agradezco primeramente a **Dios** por la vida y la oportunidad de crecer como persona.

A la **Universidad Nacional Autónoma de México (UNAM)** y al **Instituto Mexicano de Tecnología del Agua (IMTA)** por brindarme los conocimientos y herramientas necesarios para el crecimiento como profesionista.

Al **Consejo Nacional de Ciencia y Tecnología (CONACYT)** por el invaluable apoyo recibido, sin el cual no hubiera sido posible el desarrollo de mis estudios de posgrado.

A mis **Maestros** por su compromiso, trabajo y experiencias en el campo del conocimiento así como su amistad.

Al personal Administrativo del Campus Morelos por todas las facilidades y afecto durante el proceso del posgrado

Un agradecimiento muy especial a mi tutor el Dr. Fco. Javier Aparicio Mijares por su tiempo, ideas y recomendaciones plasmadas en este trabajo.

Agradezco sinceramente al Dr. Aldo Iván Ramírez Orozco, Dr. M. Alfonso Gutiérrez López, M. I. Juan Francisco Gómez Martínez y M. I. Roberto Mejía Martínez por los comentarios y sugerencias para lograr un mejor desarrollo del tema de tesis.

Finalmente gracias al grupo de trabajo del área de Hidrología y Mecánica de ríos del IMTA, por su apoyo y atenciones, de manera muy particular a la M. I. Margarita Preciado.

Dedico el esfuerzo realizado en la obtención de este grado académico a mis **Padres** Juan Molina Ávila y Cristina Aguilar Hurtado por su confianza, cariño y apoyo incondicional. A mis **Hermanos** Cesar, Claudia, Christián y Monzerrath por acompañarme siempre a lo largo de esta desafiante, enriquecedora y bella aventura.

Ofrezco este logro y mi profundo agradecimiento a todos mis familiares, amigos y compañeros en Morelia, de igual manera para todos y cada uno de mis amigos de posgrado de las áreas de Hidráulica, Gestión Integral del Agua e Ingeniería Ambiental por brindarme su confianza y amistad a todos y cada uno de ustedes la mejor de las suertes en la vida.

Si en la lid el destino te derriba, si todo tu camino es cuesta arriba,
Si tu sonrisa es ansia insatisfecha, si a tu caudal se interponen diques,
Date una tregua pero… NO CLAUDIQUES

1. Introducción

Las Redes Neuronales Artificiales (RNA) comenzaron a ser desarrolladas tratando de simular los sistemas naturales utilizando estructuras análogas a las Redes Neuronales Biológicas (RNB). Las neuronas del cerebro humano están interconectadas por medio de sinapsis. La complejidad de las conexiones entre neuronas son las responsables de las características atribuidas a la inteligencia.

Al comienzo de la década de los cuarenta la comunidad científica intentó entender el funcionamiento del cerebro humano. La agilidad y la eficiencia con que el cerebro realiza sus funciones constituyen un objetivo a ser alcanzado por los sistemas de Inteligencia Artificial (IA), que basan sus estudios de simulación computacional en aspectos de la inteligencia humana. Con el desarrollo de la IA surgió la idea de representar por medio de determinados programas computacionales o funcionamiento de procesos de aprendizaje al cerebro humano. Asimismo, un intento de simular una red neuronal del cerebro da origen a las RNA. Por lo tanto, una RNA puede ser definida como una estructura computacional que tiene como objetivo permitir la implementación de modelos matemáticos que representen, de forma simplificada, la forma en que el cerebro humano procesa la información que adquiere.

Las RNA pueden ser entendidas como conjuntos bien estructurados de unidades de procesamiento, interconectadas por canales de comunicación, cada uno de los cuales tiene un determinado peso correspondiente a un valor numérico. Las RNA consisten en varias unidades de procesamiento (neuronas artificiales) interconectadas entre sí, formando una determinada disposición estructural de capas (entrada, oculta y salida) y conexiones entre capas.

El primer trabajo sobre RNA surgió en la década de los cuarenta con el neurofisiólogo McCulloch y el matemático Pitts cuyas ideas fueron publicadas en 1943 en el articulo "*A logic calculus of the ideas immanent in nervous activity*" (McCulloch y Pitts, 1943). En este trabajo, establecieron una analogía entre el proceso de comunicación de las células nerviosas y el proceso de comunicación por transmisión eléctrica. De acuerdo con este trabajo en un instante de tiempo la neurona está activa cuando la influencia de los

impulsos exteriores (excitaciones eléctricas) rebasa el valor de umbral que tiene asignado de acuerdo con el tipo y características propias de la neurona y esta inactiva si la influencia de los impulsos eléctricos se halla por debajo del valor del umbral, con este principio apareció la formación de las neuronas artificiales. En 1947 ellos consiguieron demostrar que era posible conectar las neuronas y generar una red capaz de ejecutar funciones complejas.

La neurona artificial creada por McCulloch y Pitts trató de asemejar el funcionamiento que realiza una neurona en una RNB; sin embargo, la RNA propuesta para llevar a cabo dicho objetivo no pudo asemejar totalmente el sistema de procesamiento paralelo que ocurre entre las células biológicas.

Las investigaciones en RNA tratan de simular al cerebro humano principalmente en sus capacidades de aprender y de adaptarse a eventuales cambios. Asimismo, las RNA pueden ejecutar tareas que los programas convencionales no consiguen realizar, debido a esas características de aprendizaje y adaptabilidad.

Para 1949 el psicólogo Donald Hebb en su publicación *"The organization of behavior"* contribuyó significativamente en los estudios de redes neuronales. Hebb elaboró una teoría fundamentada en el proceso de aprendizaje que ocurre en el cerebro humano, que sirvió de base para el aprendizaje de las redes neuronales. Hebb sugirió que la conectividad del cerebro cambia continuamente cuando un organismo aprende a diferenciar diversas funciones, y que las conexiones neuronales son generadas por ese cambio. El aprendizaje es generalmente un proceso iterativo de adaptación aplicando los parámetros de una red (pesos y umbrales), donde los conocimientos son almacenados a través de cada iteración.

El estudio de Hebb inspiró el desarrollo de modelos computacionales de aprendizaje y sistemas adaptables. En 1956 Nathaniel Rochester *et al.* desarrollaron un modelo de RNA el cual simulaba la interconexión de centenas de neuronas y un sistema para verificar el comportamiento de la red durante dos estímulos externos. Tal vez éste haya sido el primer intento de simulación en computadora para probar la teoría de las RNA basándose en lo aprendido por Hebb.

Una red perceptrón creada por Frank Rosenblat (1958) se volvió muy popular. Su libro *"Principles of neurodynamics"* de 1962 fortaleció varias ideas sobre los perceptrones, que son redes de neuronas fundamentadas en el modelo de McCulloch y Pitts. Rosenblat demostró que si se colocan sinapsis con la capacidad de ajustarse en las neuronas artificiales de McCulloch y Pitts, las neuronas artificiales podían ser entrenadas para clasificar cierto tipo de patrones en dos categorías, dada su salida binaria.

Widrow y Hoff en 1960 desarrollaron la primera red capaz de imitar al cerebro humano utilizando procesadores paralelos (en lugar de un único procesador), con una estructura de red Adaline (Adaptative Linear Element). Más tarde ellos estructuraron una nueva red denominada MADALINE (Many Adaline).

La credibilidad de las redes perceptrón no duró mucho Minsky y Papera en 1969 publicaron el libro *"Perceptrón"*, el cual criticó severamente las redes neuronales, argumentando que los perceptrones presentaban limitaciones en sus aplicaciones y no poseían capacidad de aprendizaje para resolver problemas simples con una adecuada sustentación matemática. Además, probaron formalmente que una red formada por una única capa de neuronas, independiente del algoritmo de aprendizaje, era capaz apenas de resolver problemas de asociación de patrones, cuando el conjunto de pares de patrones es linealmente separable. Ellos sabían que las redes perceptrón con más de una capa de neuronas, llamadas Perceptrón Multicapa o también redes feedforward, tienen el poder computacional de aprender patrones linealmente dependientes; sin embargo, no se conocía un algoritmo de aprendizaje que pudiera realizar tal asociación.

A pesar de esto surgieron trabajos significativos en las décadas de los sesentas y setentas, como los de Werbos, Anderson y Grossberg (Haykin, 1994), los resultados y observaciones hechas por Minsky y Papera fueron devastadores, dejando así las RNA en segundo plano durante estas décadas. Pero en el inicio de la década de los ochentas, las investigaciones acerca de las RNA volvieron a recuperar su credibilidad con los trabajos del físico y biólogo John Hopfield (1982).

La incapacidad de las redes perceptrón para resolver problemas de asociación de patrones para un conjunto de patrones no lineales fue eliminada por Rumelhart *et al.* (1986). la solución encontrada fue la *"Regla Delta Generalizada"*, más conocida como *"Algoritmo de corrección del error por retropropagación"*, en 1986, para redes perceptrón de multicapas de neuronas con entradas y salidas analógicas. Las funciones de activación fueron substituidas por funciones continuas sigmoidales. Un resultado similar llamado *"The Learning Logia"* ya había sido encontrado independientemente por Parker (1985), sin repercusión en esa época.

Aunque las redes neuronales se han desarrollado en muchas esferas cientificas y tecnológicas, incluyendo la hidrología, aún hay diversos temas en que su posible aplicación no está completamente estudiado, ya que por ejemplo es difícil saber cuánto tiempo necesita una red para aprender cierta tarea, cuántas neuronas se requiere como mínimo para realizar cierta tarea, etc.

Las principales características de las redes neuronales son:

- la capacidad de aprendizaje a partir de la experiencia, debido a que las redes neuronales son entrenadas para realizar una determinada tarea sin necesidad de representar con detalles los mecanismos intrínsecos de dicha tarea, ni de programarla usando un lenguaje de programación. Además, las redes neuronales pueden ser reentrenadas para ajustarse a nuevas necesidades de la tarea que van realizar, sin tener que reescribir o realizar el código, cosa que sí es muy frecuente en programas tradicionales.

- Su buena velocidad de respuesta una vez concluido el entrenamiento.

- Se comportan como lo hace el cerebro: los seres humanos no necesitamos pensar mucho para reconocer objetos, palabras, etc. una vez que hemos aprendido a hacerlo.

- Su robustez en el sentido de que el conocimiento adquirido se encuentra repartido por toda la red, de forma que si se lesiona una parte se continúa generando cierto número de respuestas correctas.

Además las redes neuronales artificiales tienen como principal virtud el poder reproducir largos registros de datos una vez que han sido entrenadas de manera adecuada. Como resultado, se tiene la capacidad de simular escenarios con buena aproximación por medio de datos que pueden presentarse una vez realizado el entrenamiento correcto de la red.

El método de Muskingum que es comúnmente utilizado para el tránsito de avenidas debido a su facilidad de calibración presenta limitantes en el análisis de grandes registros hidrométricos como por ejemplo un tren de avenidas, por tanto las características mencionadas de las redes neuronales representan una posibilidad a explorarse para la aplicación de las redes neuronales artificiales al tránsito hidrológico de avenidas en general y en particular en aquellas zonas donde existen entradas o salidas de gasto lateral o bien uniones de ríos.

Por tanto el objetivo de la tesis es determinar la posibilidad de aplicar las redes neuronales artificiales para el tránsito hidrológico de avenidas con presencia de aportaciones y/o extracciones laterales o bien en confluencias de ríos.

El desarrollo de la tesis consistirá primeramente en aplicar las redes neuronales artificiales a un ejemplo numérico para conocer si es factible que reproduzca un hidrograma de salida a partir del conocimiento de un hidrograma de entrada cualquiera, una vez realizado el entrenamiento de la red.

Posteriormente se procederá a aplicar la estructura de la RNA entrenada en el caso anterior a datos reales y determinar su capacidad de respuesta ante avenidas registradas en un cauce, donde se ha podido apreciar la presencia de aportaciones o extracciones laterales.

Finalmente se observará el comportamiento de las redes neuronales artificiales en dos casos de mayor dificultad. El primero de ellos tratará el análisis de trenes de avenidas observados en un cauce de la región hidrológica 30 Grijalva – Usumacinta y por último se enfocará la atención en el caso de una unión de dos ríos en la región hidrológica 28 Papaloapan pertenecientes a la cuenca del río Tesechoacán.

2. Aplicaciones de redes neuronales en hidrología

La lluvia es un elemento del ciclo hidrológico cuya gran variabilidad espacial y temporal hace que sea difícil realizar su predicción. La medida de lluvia generalmente se realiza puntualmente. Los métodos para predicción de lluvia dependen de la aplicación de las escalas temporales y espaciales de interés. Las aplicaciones hidrológicas a grandes escalas temporales están asociadas a grandes escalas espaciales.

French *et al.* (1992) utilizaron una RNA para realizar una predicción del campo de intensidades de lluvia en el espacio y en el tiempo, utilizando una red típica de tres capas: capa de entrada, capa oculta y capa de salida. El algoritmo de entrenamiento utilizado fue el de retropropagación, donde los campos de lluvia de entrada y salida fueron presentados a la red como un conjunto de entrenamiento. Después de que la red fue entrenada, la RNA fue utilizada para predecir las intensidades de lluvia con intervalo de tiempo de 1 hora.

Crespo y Mora (1993) utilizaron una RNA multicapas de alimentación hacia adelante para desarrollar un método de cálculo del escurrimiento a partir de precipitaciones, para gastos pequeños, de manera que se pudiera hacer un mejor análisis sobre el periodo seco. El algoritmo utilizado fue el de retropropagación, utilizando un ajuste "on – line". La función de activación utilizada fue la tangente hiperbólica. La predicción de escurrimientos basándose en la precipitación histórica es importante, mas también lo es la evapotranspiración y la infiltración. Por esto, las entradas a la red fueron utilizadas las precipitaciones de los instantes t y t-1, y también el número de días de lluvia en dicho periodo.

Como las RNB usan millones de neuronas en sus tareas, mientras que las RNA están restringidas a algunos centenares solamente, los trabajos utilizando RNA para resolver problemas en ingeniería civil iniciaron después de los años 80 a raíz de los trabajos de John Hopfield, los cuales devolvieron la credibilidad en el uso de las RNA. Flood y Kartman (1994) mostraron conceptos, métodos para usar las RNA, y el potencial para aplicaciones en ingeniería civil. Las aplicaciones de las RNA utilizando series temporales crecieron mucho dentro del área de recursos hídricos, obteniendo buenos resultados en predicción de escurrimientos en ríos.

Karunanithi *et al.* (1994) utilizaron un algoritmo de correlación en cascada, que es un algoritmo constructivo que puede automáticamente sintetizar la arquitectura de una red como parte del proceso de entrenamiento. El objetivo de Karunanithi *et al.* era validar la capacidad de predicción de un modelo de red neuronal diferente, antes de concentrarse en la selección de la arquitectura apropiada para la red de convergencia del algoritmo de entrenamiento.

Los modelos de RNA también han sido útiles y eficientes cuando son utilizadas para resolver problemas cuyos procesos involucrados son difíciles de ser descritos por ecuaciones físicas. Hsu *et al.* (1995) utilizaron este tipo de modelo para simular procesos hidrológicos no lineales en una cuenca. Para identificar la estructura y los parámetros de una red de alimentación hacia adelante con tres capas fue utilizado un nuevo procedimiento, mostrando cuál era el potencial de esta nueva herramienta en la predicción de escurrimientos. Para ellos se hizo una comparación entre modelos basados en la capacidad individual de cada una de ellas, para representar los procesos hidrológicos.

La estimación de lluvia por radar se obtiene por medio de la relación del factor de reflectividad, cuya expresión más utilizada es la propuesta por Prober - Jones. Las relaciones usadas para obtener la lluvia dependen de dos parámetros del radar y de la lluvia media. Xiao y Chandrasekar (1995), utilizaron RNA para obtener una estimación de lluvia sin considerar alguna relación de reflectividad a lluvia. Una técnica de RNA mapea directamente los múltiples parámetros del radar con las medidas del pluviómetro. Una red aproxima las relaciones de entradas – salidas basándose en un grupo de medidas del radar y del pluviómetro por medio de pesos. Entonces cuando la red consigue ser entrenada adecuadamente con los datos dados se pueden generalizar las relaciones para nuevos conjuntos de datos.

Otro problema que ha sido abordado a través de RNA es la predicción del escurrimiento en una estación hidrométrica en instantes de tiempo posteriores al presente $(t + 1)$, $(t + 2),\ldots,(t + r)$ a partir de los datos diarios en el tiempo (t), y conocidos los valores en tiempos anteriores $(t - 1)$, $(t - 2),\ldots,(t - r)$ de n estaciones hidrométricas y de la estación considerada, utilizando como criterio para evaluar el desempeño la diferencia entre el valor calculado y el observado, el cual debía ser menor de un valor predeterminado. Para este problema, Souza *et al.* (1955) utilizaron una RNA definiendo dos arquitecturas de la red. La primera estaba totalmente conectada, esto es, cada neurona poseía conexiones con cada una de las neuronas de la capa anterior. En la segunda, utilizaron una arquitectura donde cada conjunto de entrada fue conectado a una neurona lineal, obteniendo de esta manera entradas conectadas a neuronas lineales y éstas conectadas a una capa de neuronas no lineales de tamaño variable. La función de transferencia utilizada en ambas arquitecturas fue una función sigmoidea del tipo tangente hiperbólica. En este trabajo concluyeron que la RNA era altamente eficiente para el mapeo de problemas no lineales, pudiendo tener ventaja sobre los métodos estadísticos.

Con el pasar de los años también han sido desarrollados varios modelos para la predicción de la relación lluvia – escurrimiento de diferentes grados de complejidad y sofisticación. Shamseldin *et al.* (1996) desarrollaron un sistema general para predicciones de lluvia – escurrimiento, en el cual se estimaban las descargas obtenidas de diferentes modelos que debían de ser combinadas eficientemente, tornándose una buena herramienta en sistemas operacionales de predicción de escurrimientos en ríos. La esencia de este concepto es que la salida de cada modelo capte ciertos aspectos importantes de información disponible sobre el proceso a ser modelado.

Sarmento (1996) demostró la posibilidad del uso de la técnica de RNA en la modelación de variables hidrológicas mostrando dos aplicaciones. La primera consistió en la simulación de la relación lluvia – escurrimiento mensual de una cuenca de la región semiárida de Nordeste de Brasil, y la segunda en la predicción de escurrimientos medios diarios con intervalos de 1 y 2 días haciendo uso de las series pluviométricas de una cuenca en la región central de Alemania. La serie histórica de escurrimientos utilizada en la primera aplicación fue dividida en dos sub-series, siendo una la utilizada para el entrenamiento y la otra para la validación de la red. Los resultados obtenidos de la validación de la red fueron comparados con los resultados de la calibración de un modelo lluvia – escurrimiento conocido como MODHAC (Lanna y Schwarzbach, 1988). En la segunda aplicación, los escurrimientos medios diarios previstos con intervalos de 1 y 2 días se fundamentaron en los escurrimientos medios diarios registrados en dos estaciones localizadas en la cuenca y el escurrimiento en el propio puesto para el cual se realizó la predicción. Sarmento *et al.* concluyeron que las RNA deben ser usadas en el modelado de fenómenos naturales en los casos para los cuales no se dispone de una formulación matemática explicita, capaz de reproducir las relaciones entre las diversas variables involucradas, y que para la aplicación de una RNA en la transformación lluvia – escurrimiento en regiones semiáridas la extensión de la serie requerida para la fase de entrenamiento de la red debe ser muy superior a aquellas que sería la adecuada en cuencas de ríos perennes.

Ballini (1996) realizó estudios relativos a la predicción de escurrimientos en tiempo real, concentrándose en el uso de dos RNA: una Red de Kohonen y una red multicapa con algoritmo retropropagación. Estas redes fueron utilizadas para el reconocimiento de las informaciones relativas de los datos de precipitación y escurrimiento que debían ser recuperadas para la realización de predicciones de escurrimientos diarios, observándose cuál era el potencial de la utilización de ambas redes. Se realizó en este trabajo un análisis comparativo de los resultados obtenidos por la red multicapa y el modelo de Kohonen del desempeño de ambas en la tarea de predicción lluvia – escurrimiento. Los resultados mostraron que la red multicapa presenta capacidad superior a la red de Kohonen para la predicción lluvia – escurrimiento.

Valencia (1997) hizo un análisis de aplicaciones de redes neuronales perceptrón multicapa de simulación de transformación de lluvia en escurrimiento y la predicción de escurrimientos medios mensuales. Valencia concluyó que una gran ventaja de la técnica de RNA está en su versatilidad de permitir, en el proceso lluvia – escurrimiento, la

incorporación de la representatividad de cada estación pluviométrica en vez de trabajar con la lluvia media. Otro aspecto fue la posibilidad de considerar una regionalización teniendo en cuenta aspectos diversos de la cuenca, tipo de suelo, etc. en la predicción mensual de lluvia – escurrimiento. Los resultados también comprobaron la eficacia de la técnica.

Shamseldin (1997) utilizó una técnica de RNA para hacer el modelado de la transformación de lluvia – escurrimiento. El objetivo era evaluar el desempeño de la técnica comparándola con algunos modelos tradicionales de lluvia – escurrimiento. Dependiendo de la naturaleza del problema, la RNA puede tener múltiples entradas y múltiples salidas. Shamseldin en sus trabajos utilizó múltiples entradas y sólo una salida. Como en otros modelos empíricos, la red neuronal debe ser calibrada usando datos observados de entrada y salida. Los valores óptimos de los parámetros son obtenidos por un método de búsqueda. La red utilizada fue una unidireccional multicapa, por ser considerada como buena función aproximadora (Nielsen, 1991). La forma de la red fue probada usando diferentes tipos de información de entrada, como ejemplo, lluvia, lluvia histórica de estaciones e información de la cuenca vecina más próxima. Con los datos de seis cuencas hidrográficas, aplicó la técnica para cuatro escenarios diferentes en donde se utilizará algunos de estos tipos de entradas. Un desempeño de la técnica fue comparada con modelos que utilizan información de entrada similar: el modelo simple lineal (SLM), el modelo de perturbación lineal basado en la estacionalidad (LPM) y el modelo de perturbación lineal del vecino más próximo (NNLPM). Los resultados con las RNA son prometedores en el contexto de la modelación lluvia – escurrimiento, los resultados varían para las distintas formas de la RNA teniendo un valor mas alto de R^2 (valor que aumenta con el número de neuronas en la capa oculta) que el que presentan los demás modelos en las seis cuencas analizadas, comparados los desempeños de los modelos la generalización de la RNA es mejor en cinco de las seis cuencas analizadas.

Ballini *et al.* (1997) utilizaron una RNA y compararon sus resultados con los de los métodos de Box – Jenkins. Ellos utilizaron datos de escurrimientos mensuales de los ríos Furnas, Itumbiara y Sobradinho en Brasil. Los resultados indican la importancia de la estandarización de las series temporales, mostrando las ventajas de los modelos de RNA cuando está hecha la estandarización. Pero el desempeño de las redes neuronales está afectado por factores como la topología, los parámetros de entrenamiento y la naturaleza de las series temporales. Tang y Fishwick (1991) demostraron que existe una arquitectura de capas ocultas óptima para cada tipo de serie temporal. Por lo tanto Ballini *et al.* (1997) utilizaron un algoritmo de retropropagación y trabajaron con dos aprovechamientos para la predicción de escurrimientos medios mensuales.

La mayor parte del tiempo que se invierte en el montaje de una RNA es en el proceso de identificación de su mejor estructura (topología) y su entrenamiento. Para la determinación de una buena estructura y una forma de disminuir ese tiempo Kadowaki *et al.* (1997) observaron que existe una relación entre una estructura de autocorrelación de serie temporal y la mejor selección de lo nudos de entrada de una red neuronal. Con esto consiguieron proponer una metodología para la determinación del número de entradas de las RNA utilizadas para la predicción de escurrimientos medios mensuales, basada en los

estudios de las funciones de autocorrelación (*f.a.c.*) y autocorrelación parcial (*f.a.c.p.*) de los datos de escurrimientos históricos. Observaron que es relativamente más fácil escoger un modelo de red neuronal que construir un modelo de Box – Jenkins. Con algunas pruebas verificaron que la RNA era capaz de competir con los métodos convencionales como el de Box – Jenkins.

Analizar y prever eventos futuros fundamentándose en registros de nivel de un río es una tarea difícil porque varias variables como lluvia, infiltración y características del suelo influencian el nivel del río de una manera no lineal. La predicción del nivel puede hacerse por métodos directos o indirectos. La predicción del escurrimiento debe antes convertirse en un nivel medio con ayuda de la curva característica del río elevaciones – gastos, por lo que se debe contar con una buena curva. La predicción del nivel por los métodos directos utiliza técnicas de correlación estadística. Thirumalaiah y Deo (1998) utilizaron una RNA con este propósito. Ellos utilizaron datos de nivel medidos en una estación hidrométrica además de sus datos históricos. La red fue entrenada usando tres algoritmos: retropropagación, correlación en cascada y gradiente conjugado. En seguida se hizo una comparación entre ellos. Los resultados mostraron que es posible hacer una predicción del nivel en tiempo real a través de las RNA y que existe una buena correlación entre los valores de nivel observados y los de salida en la red independientemente del algoritmo de entrenamiento. Una red dinámicamente adaptable produce una predicción de nivel más satisfactoria que una no adaptable.

Las RNA proveen una forma rápida y flexible para la creación de modelos para la predicción de escurrimientos en ríos y que tiene un mejor desempeño que los métodos tradicionales. Imrie *et al.* (2000) crearon un método que tratara de quitar la limitante de que las RNA no tienen un buen funcionamiento para aquellos eventos que quedan fuera del rango utilizado en el entrenamiento mediante una arquitectura de aprendizaje de correlación en cascada, tomado para ello dos casos en el Reino Unido.

Yitian y Gu (2003) desarrollaron un modelo para predecir los escurrimientos diarios y las descargas anuales de sedimentos transportados en un sistema de ríos en la región media del río Yangtze, China. Estos autores demostraron que la técnica de las RNA era una herramienta poderosa para la predicción de flujo y transporte de sedimento en tiempo real aplicado a una red fluvial compleja. La metodología propuesta mostró que es posible integrar principios físicos fundamentales en la información manejada por la técnica de modelación y que es posible adaptar un sistema natural a la construcción de una RNA.

Chang *et al.* (2002) presentaron una alternativa para evitar el uso de las RNA clasificadas como estáticas debido a que solamente pueden simular estructuras de memoria a corto plazo dentro de los procesos y que consistía en una red con aprendizaje recurrente en tiempo real (RTRL) para predicción de escurrimientos dada la variación de las características del proceso hidrológico con respecto del tiempo. Sus resultados demostraron que la red RTRL tuvo una capacidad de aprendizaje de alta eficiencia además de ser un modelo adecuado para la predicción de series de tiempo. El estudio de la red RTRL se aplicó a la información lluvia – escurrimiento del río Da – Chia en Taiwán teniendo buena exactitud.

Cigizoglu (2003) realizó una investigación de los ríos en la región Media Este de Turquía, donde estimó, previó y extrapoló escurrimientos de dichos ríos mediante el uso de una red perceptrón multicapa. Este estudio tenía como finalidad observar la capacidad de generalización fuera del rango de entrenamiento de la RNA con la información hidrométrica de cuatro estaciones. Cigizoglu concluyó de los gráficos y los estadísticos que la solución dada por la RNA provee un mejor ajuste de la información que los métodos convencionales.

Campolo *et al.* (2003) llevaron a cabo un estudio en el río Arno tratando de predecir la evolución de su nivel de agua en tiempo real a partir de información de lluvia, hidrométrica y de operación de presas, con la finalidad de implementar un sistema de alertamiento en tiempo real que disminuyera el riesgo para la población y las pérdidas económicas por efecto de las inundaciones. El modelo fue basado en una RNA utilizada exitosamente en trabajos previos para la predicción de escurrimientos en cuencas no reguladas. Capolo *et al.* observaron un buen funcionamiento de la RNA en la predicción de la evolución del nivel en el río Arno.

3. Redes Neuronales Artificiales

3. 1 Comparación entre el cerebro y una red neuronal artificial

Las computadoras electrónicas comparten muchas características del Sistema Nervioso Central (SNC) humano. La computadora tiene un circuito de entrada similar a la parte sensorial del SNC y los circuitos de salida son similares a la parte motora del SNC. Las vías de conducción entre las entradas y las salidas son los mecanismos para ejecutar diferentes tipos de cálculos.

Según Guyton y Hall (1997), las computadoras simples, las sinapsis de salida son directamente controladas por las sinapsis de entrada, operando de un modo semejante a los reflejos simples de la médula espinal humana. En las computadoras más complejas, la salida es determinada tanto por las sinapsis de entrada tanto por la información que fue almacenada en la memoria de la computadora, o que es análogo a los mecanismos de reflejos y los mecanismos de procesamiento de nuestro sistema nervioso superior.

A medida que las computadoras se tornaron más complejas, fue necesario aumentar otra unidad, llamada unidad procesadora central, que determina la secuencia de todas las operaciones. Esta unidad es análoga a los mecanismos existentes en nuestro cerebro que nos permite dirigir la atención primero a un pensamiento o sensación o actividad motora, después a otro, y asimismo por tanto, ocurren secuencias complejas de pensamiento o acción.

La figura 3.1 presenta un esquema simplificado de una computadora moderna. Este esquema se asemeja al SNC. Los componentes básicos de una computadora común son análogos a los del SNC humano y, por tanto, el cerebro realiza un trabajo similar al de una computadora que continuamente selecciona información sensorial y las utiliza juntamente con la información almacenada para realizar el curso diario de actividades corporales.

En la tabla 3.1 se muestra una comparación entre el cerebro humano y la computadora. El sistema nervioso está formado por cerca de cien mil millones de neuronas (Tafner *et al.* 1996), donde cada neurona se comunica con otras 10 mil. Su respuesta puede ser modificada dependiendo del comportamiento de los axones.

Parámetros	Cerebro	Computadora
Material	Orgánico	Metal y plástico
Velocidad	Milisegundos	Nanosegundos
Eficiencia energética	10^{-16} J/operación/s	10^{-16} J/operación/s
Tipo de procesamiento	Paralelo	Secuencial
Almacenamiento	Adaptativo	Estático
Control de procesos	Distribuido	Centralizado
Número de elementos procesados	10^{11} a 10^{14}	10^{5} a 10^{6}
Liga entre elementos procesados	10,000	< 10

Tabla 3.1 *Cuadro comparativo entre el cerebro humano y la computadora (Fuente Tafner et al, 1996)*

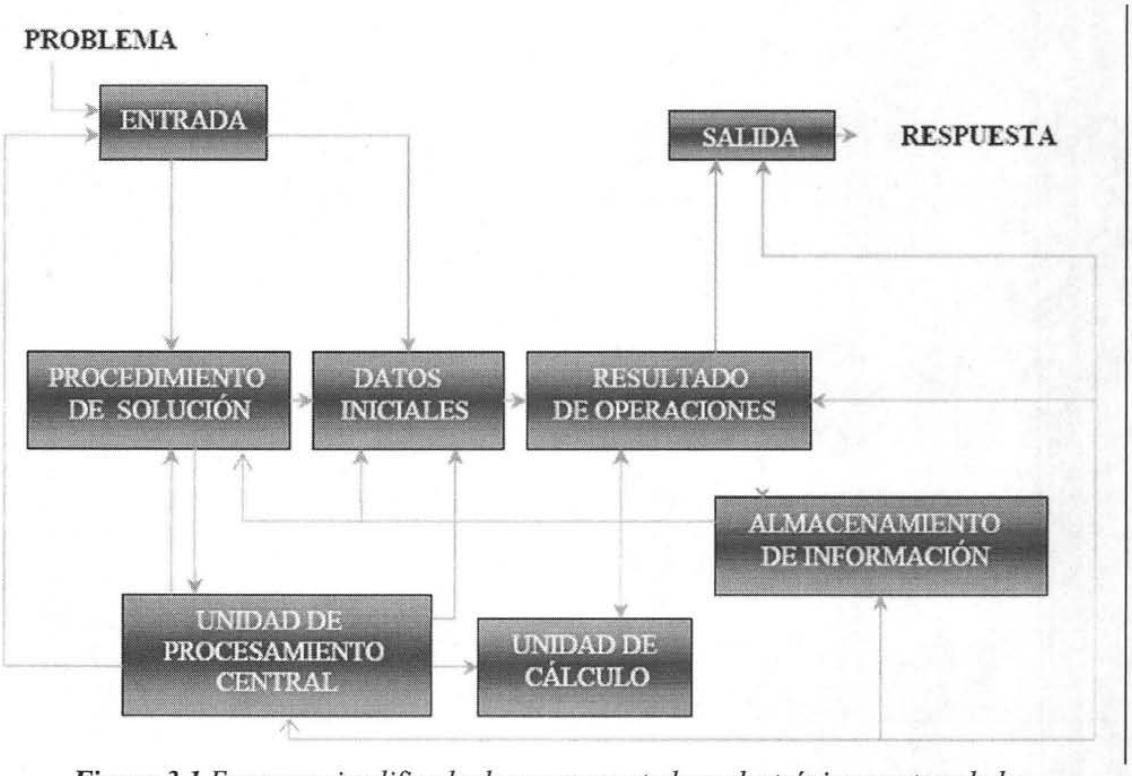


Figura 3.1 *Esquema simplificado de una computadora electrónica mostrando los componentes básicos y sus interrelaciones (adaptado de Guyton y Hall, 1997)*

Según Carvalho Filho y Marar (1993), las dificultades de comprender el sistema nervioso es millones de veces superior al del funcionamiento de las computadoras más avanzadas existentes. Los procesos que ocurren a nivel biológico presentan un gran número de variables correlacionadas de forma corporal, provocando una explosión exponencial de posibilidades de comportamiento neuronal y haciendo que los sistemas biológicos presenten una apariencia no determinista.

Las computadoras convencionales son adecuadas para la realización de cálculos y procesos secuenciales, siendo más apropiadas para realizar cálculos que los seres humanos. Por otro lado, los seres humanos realizan con más facilidad tareas como el reconocimiento de imágenes y caras, lectura de manuscritos o un análisis de escenas.

Las computadoras que utilizan los sistemas de Inteligencia Artificial (IA), buscan resolver problemas de reconocimiento de patrones de fácil procesamiento tal como el neurofisiológico. Asimismo, los modelos neuronales pueden aproximar el procesamiento de las computadoras al del cerebro humano, con un grado de interconexión similar a la estructura del cerebro. La tabla 3.2 muestra una comparación entre las computadoras y las redes neuronales.

Las redes neuronales toman un enfoque diferente para la solución de problemas que el utilizado por las computadoras convencionales. Las computadoras convencionales usan un enfoque algorítmico, es decir, la computadora sigue un grupo de instrucciones para resolver un problema. A menos de que los pasos específicos que la computadora necesita seguir no sean conocidos, no podrá resolver el problema. Eso restringe la capacidad de solucionar problemas por parte de las computadoras convencionales que sólo nosotros entendemos y sabemos resolver. Pero las computadoras serían más útiles si pudieran hacer cosas que nosotros no sabemos exactamente cómo hacer.

Computadoras	Redes Neuronales Artificiales
Ejecuta programas	Aprende
Ejecuta operaciones lógicas	Ejecuta operaciones no lógicas, Transformaciones, comparaciones
Depende del modelo o del programador	Descubre las relaciones o reglas de los datos y ejemplos
Prueba una hipótesis por vez	Prueba todas las posibilidades en paralelo

Tabla 3.2 *Cuadro comparativo entre computadoras y neurocomputadoras (Fuente Tafner et al, 1996)*

Las RNA procesan la información de una manera similar de cómo lo hace el cerebro humano. La red está compuesta de un gran número de elementos procesadores interconectados entre si (neuronas) trabajando en paralelo para resolver un problema específico. Las redes neuronales aprenden por medio de ejemplos, los cuales deben seleccionarse cuidadosamente o de otra forma se pierde tiempo útil o peor aún, la red podría estar funcionando incorrectamente.

Por otro lado, las computadoras convencionales usan un enfoque cognitivo para la solución de problemas; la manera en que el problema es resuelto debe conocerse y debe declararse en pequeñas instrucciones concretas que no sean ambiguas. Estas instrucciones son convertidas entonces en un lenguaje de programación de alto nivel y entonces en un código que la computadora puede entender. Estas máquinas son totalmente predecibles; si algo sale mal es debido a la falla de un software o hardware.

Las RNA y las computadoras convencionales no están en competencia pero se complementan unas a otras. Hay tareas que se adaptan más a un enfoque algorítmico que a operaciones aritméticas y tareas que se adaptan más a las redes neuronales. Más aun, un gran número de tareas, requiere de sistemas que usan una combinación de los dos enfoques (normalmente una computadora convencional es usada para supervisar la red neuronal) para obtener la eficiencia máxima.

3.1.1 Neurona biológica

Se estima que el cerebro humano contiene más de cien mil millones de neuronas. Los estudios sobre la anatomía del cerebro humano concluyen que hay más de 1000 sinapsis a la entrada y a la salida de cada neurona. Es importante notar que aunque el tiempo de conmutación de la neurona (unos pocos milisegundos) es casi un millón de veces menor que en los actuales elementos de las computadoras, ellas tienen una conectividad miles de veces superior que las actuales supercomputadoras.

Las neuronas y las conexiones entre ellas (sinapsis) constituyen la clave para el procesado de la información. Algunos elementos a destacar de su estructura histológica son:

- Las dendritas. Vías de entrada de las señales que se combinan en el cuerpo de la neurona. La neurona elabora una señal de salida a partir de ellas.
- El axón. Camino de salida de la señal generada por la neurona.
- Las sinapsis. Unidades funcionales y estructurales elementales que median entre las interacciones de las neuronas. En las terminaciones de las sinapsis se encuentran unas vesículas que contienen unas sustancias químicas llamadas neurotransmisores, que ayudan a la propagación de las señales electroquímicas de una neurona a otra.

Lo que básicamente ocurre en una neurona biológica es lo siguiente: la neurona es estimulada o excitada a través de sus entradas y cuando se alcanza un cierto valor del umbral, la neurona se activa, pasando una señal hacia el axón, como se observa en la figura 3.2.

Investigaciones posteriores condujeron al descubrimiento de que estos procesos son el resultado de eventos electroquímicos. Así, el secreto de la inteligencia se sitúa dentro de estas neuronas interconectadas y de su interacción.

La forma en que dos neuronas interactúan no está totalmente conocida. En general, una neurona envía su salida a otras por su axón. El axón lleva la información por medio de diferencias de potencial, u ondas de corriente, que depende del potencial de la neurona.

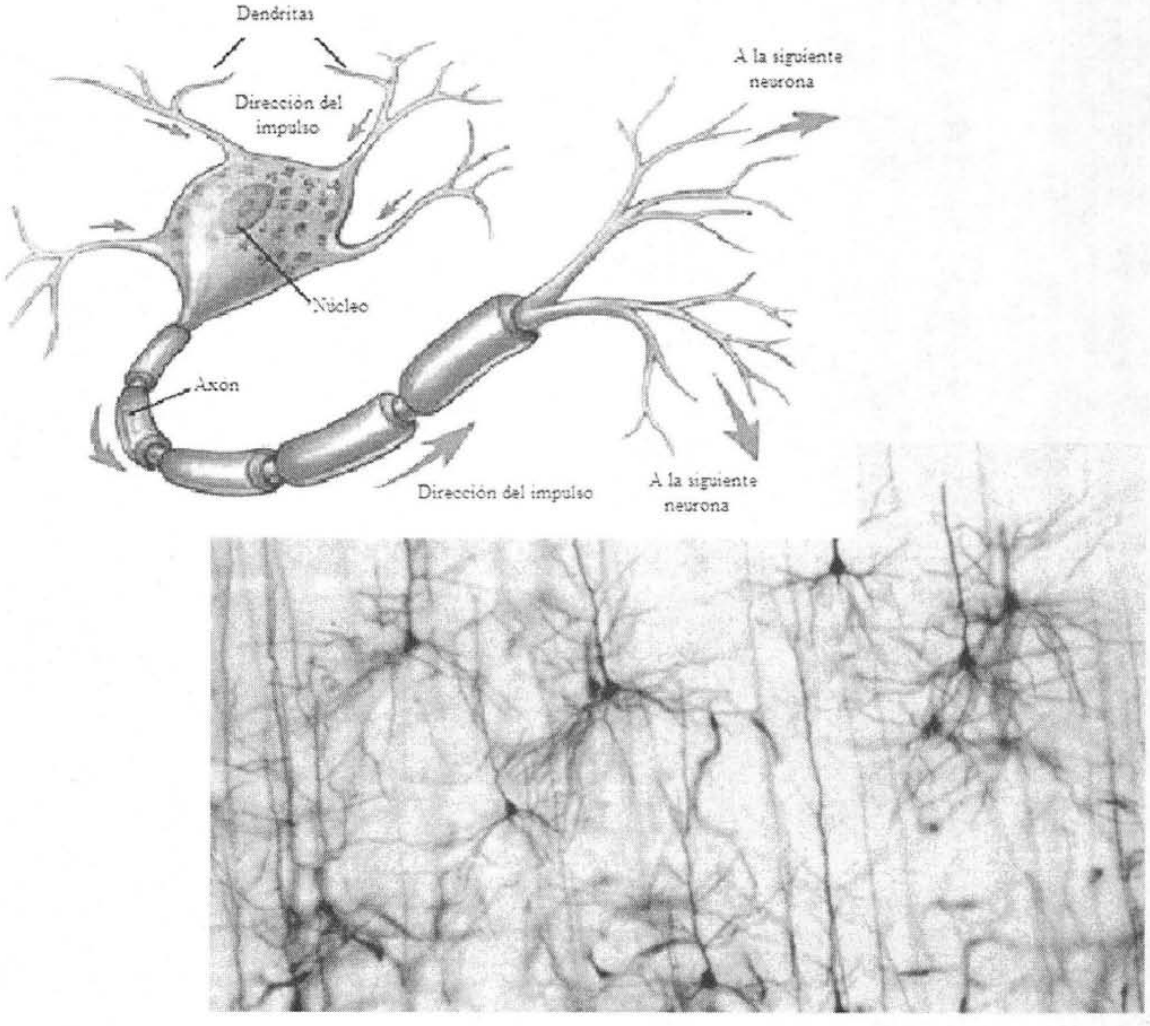


Figura 3.2 *Representación de una neurona biológica y su conexión con otras neuronas*

3.1.2 Neurona Artificial

La unidad básica de una RNA es la neurona artificial. Aunque existen diferentes tipos de neuronas, la más común es la de tipo McCulloch - Pitts. En la figura 3.3 puede verse una representación de la misma.

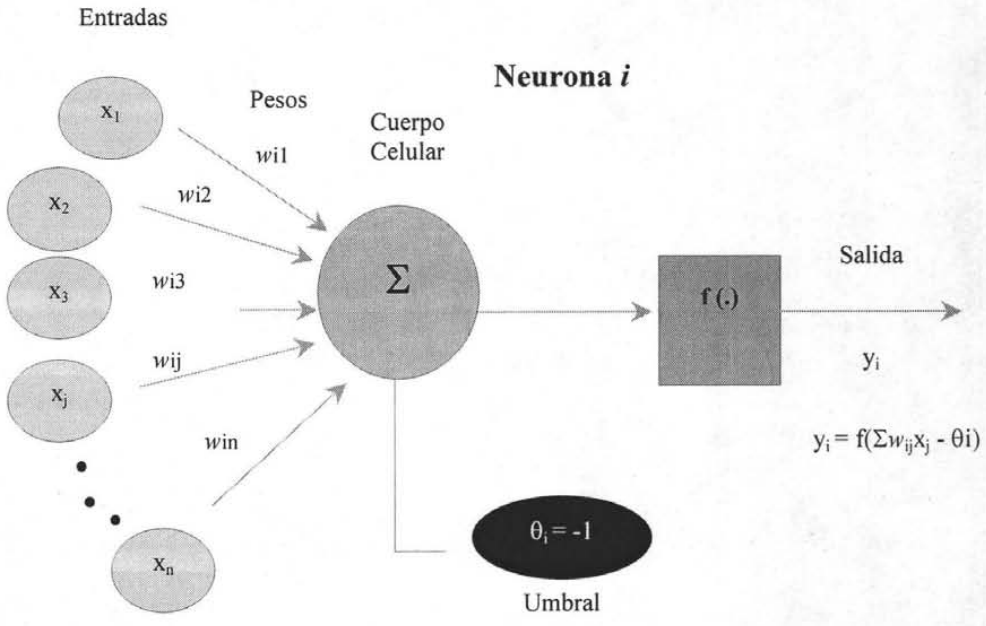


***Figura 3.3** Representación de una neurona artificial tipo McCulloch - Pitts (Fuente Larrañaga et. Al 1997)*

Una neurona artificial es un procesador elemental, en el sentido de que procesa un vector x (x_1, x_2,...x_n) de entradas y produce un respuesta o salida única. Los elementos clave de una neurona artificial los podemos ver en la figura anterior y son los siguientes:

- **Las entradas (x_i).** Reciben los datos de otras neuronas. En una neurona biológica corresponderían a las dendritas

- **Los pesos sinápticos (w_{ij}).** Al igual que en una neurona biológica se establecen sinapsis entre las dendritas de una neurona y el axón de otra, en una neurona artificial a las entradas que vienen de otras neuronas se les asigna un peso, un factor de importancia. Este peso, que es un número, se modifica durante el entrenamiento de la red neuronal, y es aquí por tanto donde se almacena la información que hará que la red sirva para un propósito u otro.

- **Regla de propagación**. Con las entradas y los pesos sinápticos, se suele hacer algún tipo de operación para obtener el valor del potencial post – sináptico (valor que es función de las entradas y los pesos, y que es utilizado en último término para realizar el procesamiento). Una de las operaciones más comunes es sumar las entradas, pero teniendo en cuenta la importancia de cada una (el peso sináptico asociado a cada entrada), es lo que se llama suma ponderada $h_i(t)=\Sigma w_{ij}x_j$, aunque es posibles también realizar otras operaciones.

3.2 Funciones de Transferencia

Este proceso es a menudo modelado como una regla de propagación representada por la función de red u(.). La neurona recoge las señales por su sinapsis sumando todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas dominan, entonces la neurona da una señal positiva y manda este mensaje a otras neuronas por sus sinapsis de salida. En este sentido la neurona puede ser modelada como una simple función escalón f(.).

Existen cuatro funciones de transferencia típicas que determinan distintos tipos de neuronas: Escalón, Lineal, Sigmoidal y Gaussiana.

Función escalón. Se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la transferencia es 1, si es menor a dicho umbral, la transferencia es 0 (ó –1). Las redes formadas por este tipo de neuronas son fáciles de implementar en hardware, pero sus capacidades están limitadas.

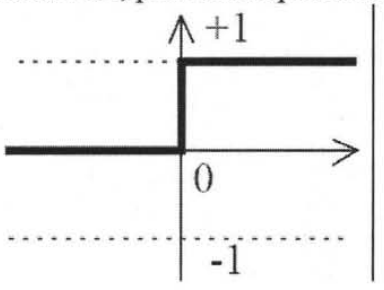


Figura 3.4 *Función de transferencia Escalón*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

Función lineal. La función lineal o mixta corresponde a la función F(x) = x. En las neuronas con función mixta si la suma de las señales de entrada es menor que un límite inferior, el valor de la transferencia se define como –1. Si dicha suma es mayor o igual que el límite superior, entonces el valor de la transferencia es 1. Si la suma de entrada está comprendida entre ambos límites, la transferencia se define como una función lineal de suma de las señales de entrada.

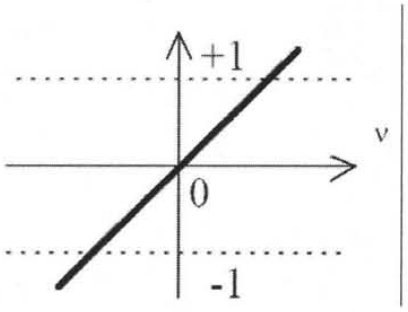


Figura 3.5 *Función de transferencia Lineal*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

Función sigmoidal. Cualquier función definida simplemente en un intervalo de posibles valores de entrada, con un incremento monotónico y que tengan tanto límite superior como inferior (por ejemplo las funciones sigmoidal y arco tangente), podrá realizar la función transferencia de forma satisfactoria.

Con la función sigmoidal, para la mayoría de los valores del estímulo de entrada, el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace posible que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho cuando la pendiente es elevada, esta función tiende a la función escalón. La importancia de esta función es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además toma su valor máximo cuando x es cero. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la función escalón, con la ventaja respecto a esta función de que la función escalón no podía definir la derivada en el punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se usan derivadas.

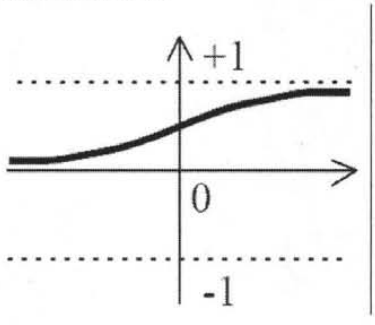


Figura 3.6 *Función de transferencia Sigmoidal*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

Función Gaussiana. Los centros y anchura de estas funciones pueden ser adaptados, lo cual las hace más adaptativas que las funciones sigmoidales.

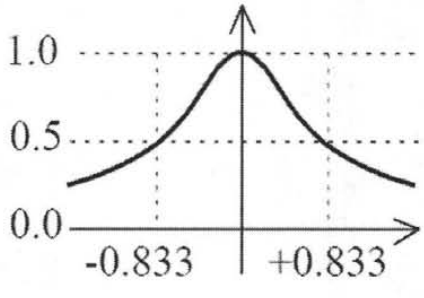


Figura 3.7 *Función de t transferencia Gaussiana*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

La función escalón únicamente se utiliza cuando las salidas de la red son binarias. La salida de una neurona se activa sólo cuando el estado de transferencia es mayor o igual a cierto valor umbral. La función lineal o identidad equivale a no aplicar función de salida. Las funciones mixta y sigmoidal son las más apropiadas cuando queremos como salida información analógica.

La neurona se activa si la fuerza combinada de la señal de entrada es superior a un cierto umbral. En el caso general, el valor de transferencia de la neurona viene dado por una función de transferencia.

Nombre	Función	Rango
Escalón	$y = sign(x)$ $y = H(x)$	$[-1, +1]$ $[\ 0, +1]$
Lineal	$y = x$	$[-\infty, +\infty]$
Sigmoidal	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[\ 0, +1]$ $[-1, +1]$
Gaussiana	$y = Ae^{-Bx}$	$[\ 0, +1]$

Tabla 3.3 *Funciones de transferencia y rango de aplicación*
(Fuente Larrañaga et. Al 1997)

Se puede remarcar que en una RNA si la última capa de la red multicapa tiene neuronas con función de transferencia sigmoidal, entonces las salidas de la red estarán limitadas a un rango muy pequeño. Si se utilizan neuronas con función de transferencia lineal las salidas de la red podrán tomar cualquier valor.

En la retropropagación del error es importante tener la capacidad de calcular las derivadas de cualquier función de transferencia utilizada. Por tal razón en la elaboración de esta tesis se utilizará la función de transferencia sigmoidal en cada neurona de la(s) capa (s) oculta (s) y la función lineal en las neuronas de la capa de salida, dado que en el caso de avenidas se tendrá un rango muy amplio de valores registrados en las estaciones hidrométricas.

3.3 Elementos de una Arquitectura

La elaboración de una RNA depende del tipo específico del problema a resolver. Asimismo, la arquitectura de la RNA restringe el tipo de aplicación de la misma y está determinada por: número de capas de la red, número de neuronas en cada capa y tipo de conexión entre las neuronas.

3.3.1 Número de capas

En cuanto al número de capas las redes se pueden clasificar como monocapa o multicapa.

- Red monocapa: existe solamente un elemento procesador entre cualquier entrada y salida de la red. Esta red recibe tal denominación pues la capa de entrada del elemento procesador de origen no es tomada en consideración porque ningún cálculo es realizado en la misma.

- Red multicapa: existe más de una neurona entre la entrada y salida de una red. Generalmente los organismos procesadores están organizados en capas las cuales pueden ser de tres tipos:

    - Capa de Entrada. Es la responsable de la distribución de la información o patrones provenientes del medio externo para las capas ocultas.

    - Capa oculta. Donde ocurre el procesamiento por medio de las funciones de combinación y transferencia. Por lo tanto en estas capas es donde se realiza la mayor parte del procesamiento a través de las conexiones ponderadas. Pueden ser consideradas como extractoras de características, o sea, sus pesos son una codificación de las características representadas en los patrones de entrada y permiten que la red pueda crear su propia representación de una forma más rica y compleja del problema.

    - Capa de salida. Es la responsable de la respuesta de la red, donde se presenta el resultado final. Recibe los estímulos de las capas ocultas y construye un patrón de respuesta.

3.3.2 Tipo de conexiones

Las neuronas pueden tener conexiones del tipo:

- Unidireccional o acíclica (*feedforward*): la transmisión de la señal por parte de la red es unidireccional. Cada neurona de la capa recibe sus entradas de la capa anterior, y envía las señales de salida en dirección de la capa siguiente.

- Retroalimentación o cíclica (*feedback*): la transmisión de la señal por la red es hecha a través de caminos cerrados (*loops*). Existe flujo de la información para capas anteriores y comunicación lateral. Las neuronas pueden estar conectadas directamente a las entradas externas; además, estas unidades sirven también de salidas de la red.

3.4 Arquitectura de las redes neuronales artificiales

Las entradas a una neurona incluyen su umbral y la suma de los pesos de las entradas. La salida de una neurona depende de sus entradas y de su función de transferencia. Como se ha mencionado previamente existen muchas funciones de transferencia posibles.

Una sola neurona no puede hacer mucho. Sin embargo, muchas neuronas pueden ser combinadas en una capa o múltiples capas teniendo mucho potencial.

La mejor arquitectura a usarse depende del tipo de problema a ser representado por la red. El problema computacional de una red radica en el mapeo de los valores de entrada y salida. El problema particular del mapeo puede ser mejorado seleccionando el número de entradas, así como el número de salidas de la red.

Tanto el número de neuronas de la capa de salida como el número de neuronas en cada capa puede ser modificado, excepto para las redes puramente lineales. Por lo tanto, a mayor número de neuronas en las capas ocultas, mayor capacidad de la red.

Si se necesita representar un mapeo lineal, entonces deberán utilizarse neuronas lineales. Además, las redes lineales no pueden realizar ningún cálculo no lineal. El uso de una función de transferencia no lineal hace capaz a una red de determinar la relación de entradas y salidas.

Un problema muy sencillo puede ser representado por una sola capa de neuronas. Sin embargo, las redes de una sola capa no pueden resolver ciertos problemas. Las redes unidireccionales multicapa dan un mayor grado de libertad a la red en la resolución de problemas. Por ejemplo, cualquier función razonable puede ser representada por una red de dos capas: una capa sigmoidal antecediendo a una capa de salida lineal.

Las redes con umbrales pueden representar relaciones entre entradas y salidas más fácilmente que las redes sin umbrales. (Por ejemplo, una neurona con umbral tendrá siempre una entrada neta a la función de transferencia de cero cuando todas sus entradas sean cero. Además, una neurona con un umbral puede aprender cualquier entrada neta a la función de transferencia bajo las mismas condiciones mediante el aprendizaje para valores apropiados de los umbrales.

Debido al funcionamiento de cada parte de una neurona artificial y los elementos que conforman una arquitectura, aunado con el tipo de problema a solucionarse se ha generado gran cantidad de arquitecturas, las cuales en su mayoría tienen un uso específico o se adaptan mejor al problema en cuestión. A continuación se mencionan las arquitecturas predecesoras a la que es la más utilizada en el campo de la hidrología como lo es el perceptrón multicapa.

3.4.1 Perceptrón

El perceptrón es la red neuronal pionera y la forma más simple de las RNA para clasificar un tipo especial de patrones denominado "linealmente separables". Esto es, patrones que se ubican en lados opuestos de un hiperplano. Esta red es capaz de clasificar apenas patrones linealmente separables, o sea, patrones que caen en determinadas posiciones de un hiperplano los cuales pueden ser separados por una línea recta, como se aprecia en la figura 3.8.

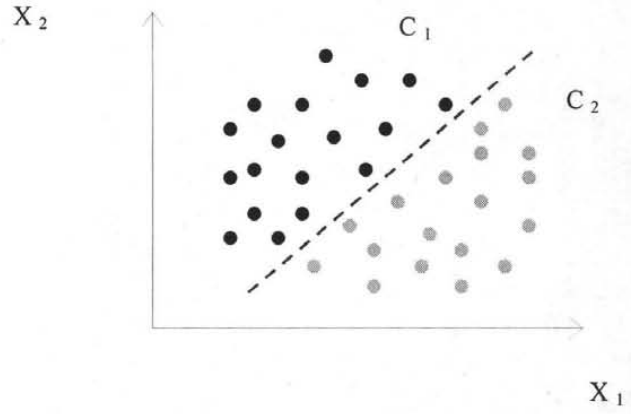


Figura 3.8 *Patrones linealmente separables*
(Fuente Larrañaga et al. 1997)

Básicamente, el perceptrón está constituido de una capa simple con los pesos sinápticos y un umbral. El algoritmo de ajuste de los parámetros libres de esa red fue desarrollado por Rosenblatt en 1958. Rosenblatt probó que el algoritmo converge si el patrón (vector) de entrenamiento es delimitado por dos clases linealmente separables. El algoritmo también establece la superficie de separación de la forma de un hiperplano entre las dos clases.

Asimismo, el algoritmo de entrenamiento del modelo perceptrón permite distinguir clases en los conjuntos de entradas, si éstas fueran linealmente separables en términos de algún espacio de decisión. Los perceptrones toman decisiones y determinan si un patrón entrada encaja o no en un cierto patrón. El algoritmo de entrenamiento del perceptrón corresponde al Teorema de Convergencia Perceptrón (Rosenblatt, 1958). En ese teorema Rosenblatt postula que el perceptrón puede aprender todo lo que se le pueda presentar.

De esta forma, existe diferencia entre representación y aprendizaje. La representación es la capacidad de una red para simular una función específica, en cuanto que el aprendizaje requiere la existencia de un procedimiento sistemático de ajuste de los pesos de la red para producir esta función.

3.4.2 Adaline

La red Adaline (Adaptative Linear Element), fue desarrollada por Bernie Widrow en la Universidad de Stanford (Widrow y Hoff, 1960). Dicha red usa neuronas con función de transferencia escalón, y está limitada a una única neurona de salida.

Adaline utiliza la denominada regla Delta de Widrow – Hoff o regla del mínimo error cuadrado medio (LMS), basada en la búsqueda del mínimo de una expresión del error entre la salida deseada y la salida lineal obtenida antes de aplicarle la función de transferencia tipo escalón. Estas redes pueden procesar información analógica, tanto de entrada como de salida, utilizando una función de transferencia tipo lineal o sigmoidal.

En cuanto a su estructura, está formada por un elemento denominado combinador adaptativo lineal (ALC) que obtiene una salida lineal que puede ser aplicada a otro elemento de conmutación bipolar, de forma que si la salida del ALC es positiva, la salida de la red Adaline es +1; si la salida es negativa, entonces la salida de la red Adaline es –1.

En la figura 3.9 se muestra la red Adaline, compuesta por un combinador adaptativo lineal y una función de salida bipolar.

El ALC realiza el cálculo de la suma ponderada de las entradas:

$$S = w_0 + \sum_{j=1}^{N} w_j x_j$$

El umbral de la función de transferencia se representa a través de una conexión ficticia de peso w_0. Si tenemos en cuenta que para esta entrada se toma el valor de $x_0 = 1$, se puede escribir la anterior ecuación de la forma:

$$S = \sum_{j=1}^{N} w_j x_j = XW^T$$

Donde:

S = Valor de salida de la neurona.
X = Vector de valores de entrada a la neurona.
W^T = Vector transpuesto de los pesos sinápticos.

Esta es la salida lineal que genera el ALC. La salida binaria correspondiente de la red Adaline es, por tanto:

$$y(t+1) = \begin{cases} +1 & S \geq 0 \\ -1 & S < 0 \end{cases}$$

La red Adaline se puede utilizar para generar una salida analógica utilizando un conmutador sigmoidal, en lugar de binario; en tal caso, la salida $y(t)$ se obtendrá aplicando una función tipo sigmoidal, como la tangente hiperbólica o la exponencial.

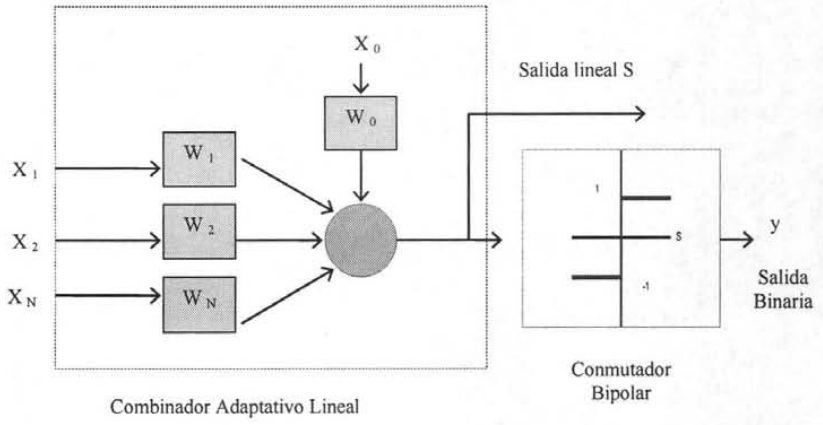


Figura 3.9 *Esquema de la red Adaline*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

3.4.3 Perceptrón Multicapa

Como se vio previamente, las redes de apenas una capa (perceptrón), resuelven apenas problemas linealmente separables. Minsky y Papera (1969) demostraron que el perceptrón no resolvía problemas no linealmente separables. En 1986, Rumelhart *et al.* desarrollaron el algoritmo de entrenamiento llamado "retropropagación", demostrando que era posible entrenar con eficiencia redes con capas intermedias. Esto resultó en el modelo de RNA más utilizado actualmente, las redes perceptrón multicapas (MLP). Por lo tanto, una MLP es una extensión del perceptrón con algunas particularidades.

Esta red se utiliza en el reconocimiento de patrones, filtrado de señales, comprensión de datos y comparación de patrones heteroasociativos, que asocia un patrón con otro. Las redes unidireccionales multicapas poseen un mejor desempeño debido al paso unidireccional de la señal por la red. El procesamiento de información ocurre en sentido progresivo a través de las interconexiones entre las neuronas de las capas adyacentes. Más allá de eso, estas redes pueden ser utilizadas para clasificación de patrones no linealmente separables.

Una red neuronal típica se representa en la figura 3.10 Se observa en la figura una capa de entrada, dos capas ocultas y una capa de salida. En este tipo de redes puede tener una o más capas ocultas. Cada capa está plenamente interconectada con una capa vecina. La salida de cada neurona que no está en la última capa propaga su salida para todas las neuronas de la capa posterior.

Esta red puede hacer gran variedad de mapeos complejos. Esto ocurre porque las neuronas de las capas ocultas aprenden a responder a las características de entrada. Esas características se refieren a las correlaciones de actividades entre las diferentes neuronas de entrada, posibilitando una representación abstracta de la información de entrada de las capas ocultas. Mas allá de la capacidad de abstracción, la red posee la capacidad de generalización, siendo capaz de clasificar correctamente un patrón complejo aún cuando este no pertenezca al conjunto de entrenamiento de la red.

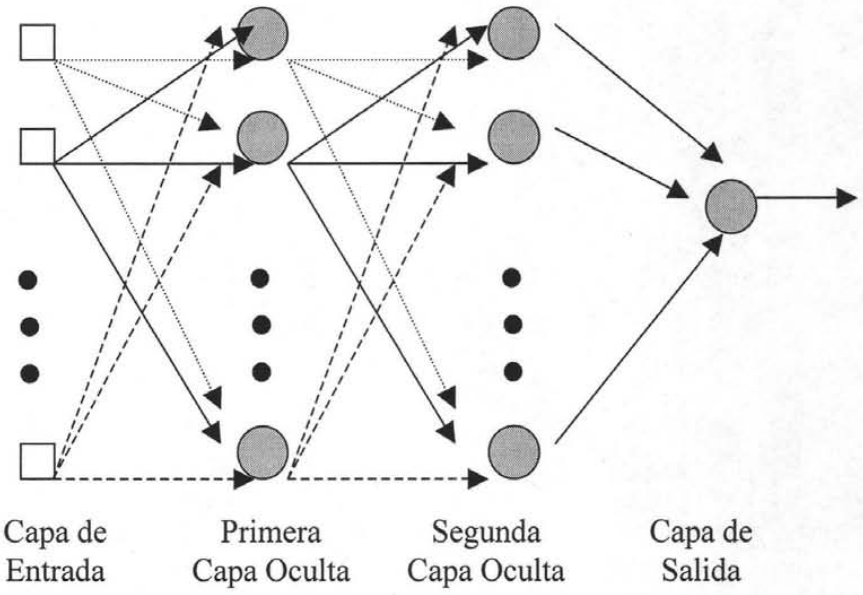


Figura 3.10 *Arquitectura de un MLP con dos capas ocultas*
(Fuente: Neural Network Toolbox, User's Guide version 4.0)

El algoritmo de retropropagación es utilizado para el entrenamiento de esta arquitectura. Durante el entrenamiento con este algoritmo, la red opera en una secuencia de dos pasos:

Primero. Un patrón es representado por la capa de entrada de la red. La actividad resultante fluye a través de la red, capa por capa, hasta que la respuesta sea producida por la capa de salida

Segundo. La salida obtenida se compara con la salida deseada para ese patrón en particular y se calcula el error. El error es propagado a partir de la capa de salida hasta la capa de entrada, y los pesos de las conexiones de las unidades de las capas internas van siendo modificados conforme al error retropropagado.

Las redes que utilizan el algoritmo de retropropagación operan con una variación de la regla delta, denominada regla delta generalizada (nntoolbox, user´s guide version 4.0). Este algoritmo procura minimizar el error obtenido por la red por medio del método del gradiente descendente. El objetivo de este gradiente es buscar un mínimo global. Siendo que, el mínimo global es considerado como una solución teórica óptima, pues representa el menor error posible.

En el entrenamiento de una RNA compleja, la solución obtenida puede no corresponder al mínimo global, porque una función puede presenta mínimos locales (figura 3.11). Cuando el tipo de superficie de error no es conocida a priori, el algoritmo verifica una gran cantidad de posibilidades antes de determinar la mejor solución. Una tasa de aprendizaje de un elemento es muy importante en este proceso, pues ella controla el

tiempo de aprendizaje tomando en cuenta la tasa de convergencia entre la solución actual y el mínimo global.

Por lo tanto, el entrenamiento de las redes multicapas con retropropagación puede requerir un número excesivo de iteraciones, resultando en un tiempo de entrenamiento considerablemente largo. Una manera de optimizar la Regla Delta Generalizada es introducir el término "momentum", que es una constante que determina o afecta los cambios pasados de los pesos en la dirección actual del movimiento en el espacio de los pesos, y tiene como objetivo permitir el aumento de la tasa de aprendizaje, sin que ocurran oscilaciones, además de actuar en el aumento de la velocidad de convergencia.

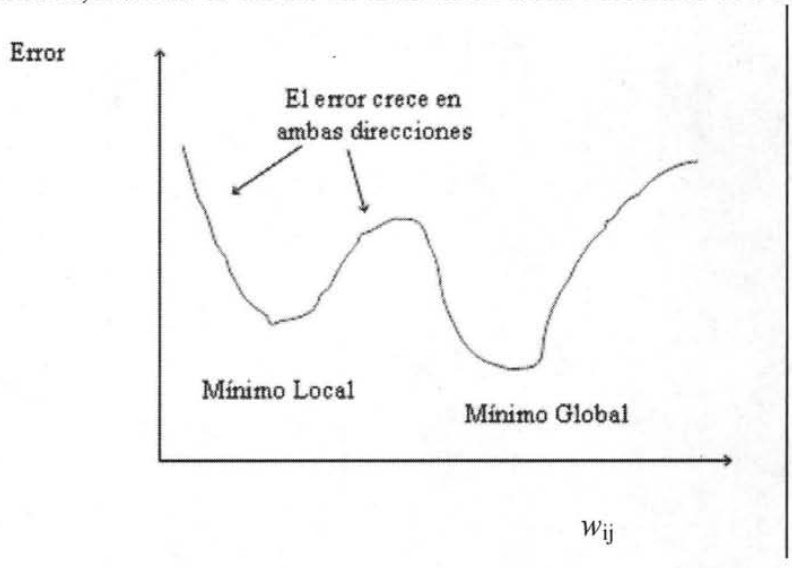


Figura 3.11 *Representación de un mínimo local y el mínimo global (Fuente Larrañaga et al. 1997)*

Las redes perceptrón multicapa pueden ser vistas como aproximadores universales de funciones. Las redes neuronales multicapas con una única capa oculta y la función sigmoidal puede aproximar cualquier función continua arbitraria.

3.5 Aprendizaje

Después de seleccionar la estructura particular de red, el siguiente paso es estimar los pesos de conexión (w_{ij}) y los diferentes valores del umbral de las neuronas en el caso de que exista.

Una red neuronal es entrenada mediante el ajuste de los pesos que conectan sus neuronas. Esto se logra mediante el planteamiento de la red con un número de ejemplos de entrenamiento, cada una de los cuales se compone de una muestra de datos de entrada específicos y por la correspondiente respuesta de salida "correcta". Dependiendo de la naturaleza del algoritmo de entrenamiento utilizado, puede ser necesario plantear la red con los datos de calibración repetidamente hasta que la función haya aprendido.

Sin embargo, se debe tener cuidado para asegurarse que la red no se convierta demasiado familiar a los datos de calibración y de ese modo, pierda su habilidad de generalizar problemas que todavía no se ha encontrado.

Por tanto en los pesos es donde reside el conocimiento de una red, entre los diferentes tipos de aprendizaje podemos mencionar los siguientes:

1. Aprendizaje supervisado. En este tipo de aprendizaje se le proporciona a la RNA una serie de ejemplos consistentes en unos patrones de entrada, junto con la salida que debería dar la red. El proceso de entrenamiento consiste en el ajuste de los pesos para que la salida de la red sea lo más parecida posible a la salida deseada. Es por ello que en cada iteración se use alguna función que dé cuenta del error o el grado de acierto que está cometiendo la red.

2. Aprendizaje no supervisado o auto organizado. En este tipo de aprendizaje se presenta a la red una serie de ejemplos pero no se presenta la respuesta deseada. Lo que hace la RNA es reconocer regularidades en el conjunto de entradas, es decir, estimar una función densidad de probabilidad p(x) que describe la distribución de patrones x en el espacio de entrada Rn.

3. Aprendizaje híbrido. Es una mezcla de los anteriores. Unas capas de la red tienen un aprendizaje supervisado y otras capas de la red tienen un aprendizaje de tipo no supervisado. Este tipo de entrenamiento es el que tienen redes como las RBF.

4. Aprendizaje reforzado. Es un aprendizaje con características del supervisado y con características del autoorganizado. No se proporciona una salida deseada, pero si que se le indica a la red en cierta medida el error que comete, aunque es un error global.

La forma de proceder en los algoritmos de aprendizaje supervisado es definir una función objetivo o error a minimizar. Esta función será siempre una función monótona creciente de la diferencia entre la señal deseada (señal que debería dar la red) y la salida proporcionada por la red. El problema es, pues, de optimización: búsqueda del mínimo de una función y aquí aparecen una serie de procedimientos de búsqueda que, generalmente, se dividen en dos grupos:

- Métodos de búsqueda global. Buscan el mínimo global de la función objetivo. Proporcionan los pesos que dan el valor más pequeño de dicha función sobre todo su dominio.

- Métodos de búsqueda local. Buscan el mínimo más cercano de la función objetivo en relación al estado inicial de los pesos al comenzar dichos algoritmos. Lógicamente dan pesos diferentes que los métodos anteriores, pero son algoritmos mucho más rápidos en la obtención de la solución.

3.6 Función de desempeño

Las funciones de desempeño permiten evaluar el comportamiento de una red. Esto es útil para muchos algoritmos de aprendizaje, como el de retropropagación, el cual opera ajustando los pesos y umbrales de la red neuronal para mejorar el desempeño respecto de los datos dados a la red para el entrenamiento.

Funciones de desempeño		
MAE	Error Medio Absoluto	$\frac{\sum_i^j \left\| y - \hat{y} \right\|}{N}$
MSE	Error Medio Cuadrático	$\frac{\sum_i^j \left(y - \hat{y} \right)^2}{N}$
RMSE	Raíz del Error Medio Cuadrático	$\sqrt{\frac{\sum_i^j \left(y - \hat{y} \right)^2}{N}}$
SSE	Suma del Error Cuadrático	$\sum_i^j \left(y - \hat{y} \right)^2$

Donde:

y = valor observado de la variable (valor objetivo)
$\hat{y}$ = valor calculado de la variable durante el entrenamiento (valor de salida)
N = número total de neuronas en la capa de salida.

Tabla 3.4 *Funciones de desempeño*
(Fuente: Dolling et al.2003)

De acuerdo con Karunanithi *et al.* (1994), los errores cuadráticos (MSE) proporcionan una buena medida de la bondad de ajuste para valores de escurrimientos altos, mientras que los errores relativos medios (RMSE) proporcionan una perspectiva más equilibrada de la bondad de ajuste en valores de escurrimientos moderados. Sin embargo, estas medidas están fuertemente afectadas por las características del cauce y se debe tener cuidado cuando se comparan estudios utilizando estas estadísticas.

3.7 Normalización de los datos

Todos los valores de las variables consideradas en la modelación de un fenómeno hidrológico deben de ser normalizadas, es decir deberán de ser divididos los valores de cada variable por el valor máximo que se presenta en cada una de ellas, para asegurar que recibirán igual atención durante el proceso de entrenamiento.

Esto es particularmente importante en las redes MLP ya que sin la normalización, los valores de las variables de entrada medidas en diferentes escalas dominarán el entrenamiento en una mayor o menor extensión porque los pesos iniciales dentro de una red son aleatorios para el mismo rango finito.

La normalización de los datos también es importante para la eficiencia de los algoritmos de entrenamiento. Por ejemplo, el algoritmo de gradiente descendente (retropropagación del error) con el que suele entrenar el MLP es particularmente sensible a la escala de los datos utilizada. Debido a la naturaleza de este algoritmo, los valores altos hacen lento el entrenamiento porque el gradiente de la función sigmoide en valores extremos se aproxima a cero.

Para evitar este problema, los datos son escalados usando una transformación apropiada. En general, los datos son reescalados a los intervalos [-1, 1], [0.1, 0.9] o [0, 1]. Otra aproximación es reescalar los valores a una función Gaussiana con una media de 0 y una desviación estándar de valor unitario.

La ventaja de utilizar [0.1, 0.9] para modelar el escurrimiento es que los eventos para gastos extremos (altos y bajos), que ocurren fuera del rango de calibración, pueden tomarse en cuenta. Alternativamente, las variaciones en el gasto en lugar de gastos absolutos, puede utilizarse para evitar problemas de saturación, pero Minns y May (1996) informó sólo sobre ventajas limitadas de esta aproximación.

4. Metodología y aplicaciones

4.1 Tránsito de avenidas.

El tránsito de avenidas es un procedimiento para conocer cómo evoluciona un hidrograma a medida que escurre a lo largo de un cauce o a través de un depósito o embalse.

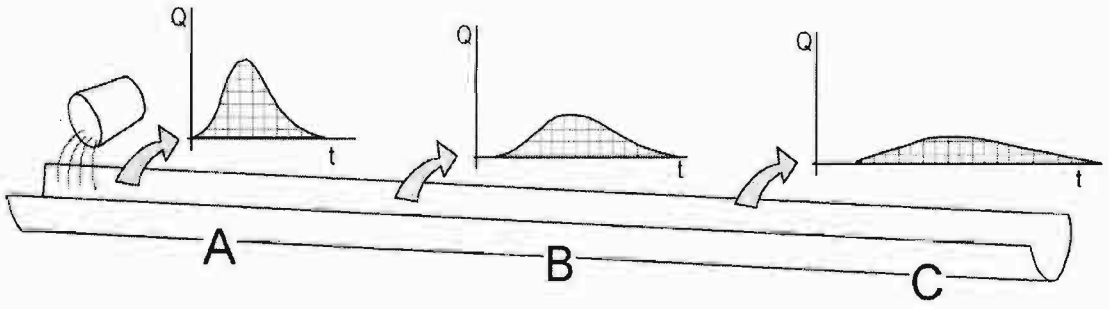


Figura 4.1 *Proceso de tránsito de avenida en un cauce*

Calcular el tránsito de un hidrograma es obtener el hidrograma del punto C a partir del hidrograma del punto A. La utilidad práctica del procedimiento es evidente. Por ejemplo, el carácter catastrófico de una avenida está relacionado directamente con la altura del pico del hidrograma (el caudal máximo), de modo que es fundamental calcular cómo ese pico va disminuyendo a medida que la avenida se mueve aguas abajo.

Si la figura 4.1 evocaba el proceso que se produce en un río, también se estudia el proceso de tránsito de caudales en embalses o cualquier depósito con una entrada y una salida. Observando la figura 4.2 se comprende que un aumento en el caudal de entrada producirá también un aumento en el caudal de salida, pero amortiguado por el depósito. Si en el caudal de entrada (I) se produjera un hidrograma similar al de la figura 4.1-A, en el caudal de salida (O) se produciría un hidrograma similar a la figura 4.1-B ó 4.1-C.

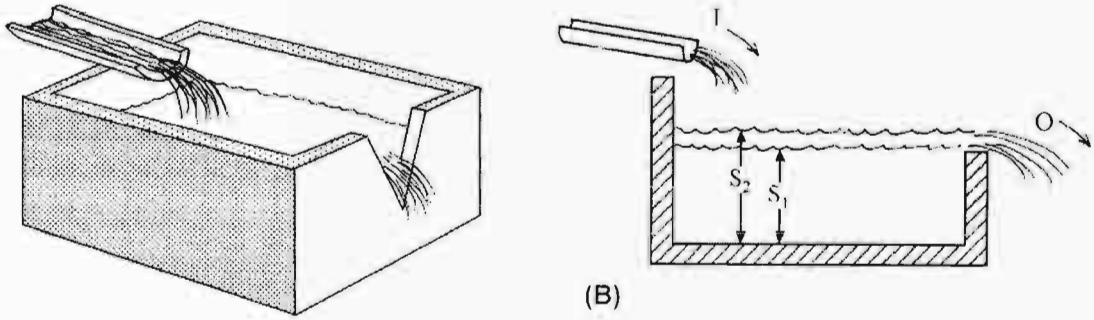


Figura 4.2 *Proceso de tránsito de avenida en un embalse*

Regularmente el lugar en donde se hacen las mediciones de escurrimientos o en donde se encuentra una presa para el control de inundaciones se localiza varios kilómetros aguas arriba del punto donde las avenidas pueden causar daños.

Es necesario contar con métodos que permitan conocer la variación de un hidrograma a lo largo de un cauce, con el objetivo de determinar el efecto que las presas reguladoras tienen aguas abajo, y así poder diseñar bordos de protección contra inundaciones. A la simulación de la variación de un hidrograma al recorrer un cauce como se mencionó previamente se le conoce como tránsito de avenidas en cauces.

Este problema se asemeja al tránsito de avenidas en vasos en el sentido de que el río presenta una especie de almacenamiento alargado y de que la solución se da a través de la ecuación de continuidad y alguna relación entre almacenamiento y gasto de salida. Sin embargo, en este método se suelen presentar algunos problemas adicionales como:

a) No se tienen planos topográficos del tramo y la relación descargas – volúmenes no se conoce.

b) Casi siempre se tienen entradas a lo largo del tramo del cauce, adicionales a las de la sección aguas arriba, que se desconocen.

c) El nivel de la superficie libre del agua en el río no es horizontal, como en el caso de tránsito de avenidas en vasos, lo que implica que un mismo tirante en el extremo final del tramo se puede formar para diferentes gastos de salida.

Existen diversos procedimientos para efectuar estos cálculos, que se agrupan en dos categorías:

4.2 Métodos hidráulicos.

Los métodos hidráulicos se basan en la solución de las ecuaciones de conservación de masa y cantidad de movimiento para escurrimiento no permanente mostradas a continuación (Berezowsky, 1980; Aparicio 2001).

Conservación de masa:

$$h\frac{\partial v}{\partial x}+v\frac{\partial h}{\partial x}+\frac{\partial h}{\partial t}=\frac{q}{B} \tag{4.1}$$

Conservación de cantidad de movimiento:

$$\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial x}+g\frac{\partial h}{\partial x}=g\left(S_0-S_f\right) \tag{4.2}$$

Donde:

h = tirante
v = velocidad
q = gasto lateral
B = ancho de la superficie libre
S_0 = pendiente del fondo
S_f = pendiente de fricción, donde si se emplea la formula de Manning:

$$S_f=\frac{v^2n^2}{Rh^{\frac{4}{3}}} \tag{4.3}$$

Rh = radio hidráulico
n = coeficiente de rugosidad
x = coordenada espacial
t = tiempo

Las ecuaciones de conservación de masa (4.1) y conservación de cantidad de movimiento (4.2), forman un sistema de ecuaciones diferenciales parciales hiperbólicas no lineales, del que no existe una solución analítica conocida, por lo que es necesario resolverlo empleando un método numérico como el de las características, de diferencias finitas o de elemento finito.

Los métodos hidrológicos utilizan simplificaciones derivadas de las ecuaciones (4.1) y (4.2) para la obtención de soluciones más simples, pero desde luego menos exactas que los que se logran con los métodos hidráulicos (Aparicio 2001). El método de Muskingum no requiere de procedimientos complejos, y que sin embargo arroja resultados muy aproximados a los reales.

4.3 Métodos Hidrológicos.

Los métodos hidrológicos se basan en la ecuación de continuidad, que para un tramo de un cauce (o para un embalse) establece que en un intervalo de tiempo Δt:

$$\text{Volumen de entrada en un } \Delta t - \text{Volumen de salida en ese } \Delta t = \Delta almacenamiento \quad (4.4)$$

Dividiendo por Δt:

$$\text{Q entrada - Q salida} = \frac{\Delta almacenamiento}{\Delta t} \quad (4.5)$$

O, lo que es lo mismo (figura 4.2-B):

$$I - O = \frac{\Delta S}{\Delta t} \quad (4.6a)$$

$$I - O = \frac{(S_2 - S_1)}{\Delta t} \quad (4.6b)$$

Siendo:

I = Caudal de entrada medio (durante el tiempo Δt)
O = Caudal de salida medio (durante el tiempo Δt)
$\Delta V = S_2 - S_1$ = incremento del almacenamiento en el tiempo Δt

Para calcular con exactitud los caudales medios de cada Δt deberíamos disponer de un hidrograma continuo, pero si conocemos solamente un dato de caudal para cada Δt, los caudales medios podemos evaluarlos haciendo la media de los caudales de dos Δt consecutivos. Así la expresión resultaría:

$$\frac{I_1 + I_2}{2} - \frac{O_1 + O_2}{2} = \frac{S_2 - S_1}{\Delta t} \quad (4.7)$$

4.3.1. Método de Muskingum.

Entre lo métodos hidrológicos, posiblemente el más utilizado en cálculos manuales por su sencillez sea el de Muskingum (Chow *et al.*, 1994, p.264; Singh, V. P, 1992, p.680; Wanielista, 1997, p.323; Viessman, 1995, p.235; Aparicio, 2001 p. 103)).

Este método utiliza la ecuación de continuidad en su forma discreta:

$$\frac{I_j + I_{j+1}}{2} \Delta t - \frac{O_j + O_{j+1}}{2} \Delta t = \Delta V \quad (4.8)$$

Y una relación algebraica entre el almacenamiento en el tramo (V) y las entradas (I) y salidas (O) de la forma:

$$V = KO + Kx(I - O) = K[xI + (1 - x)O] \tag{4.9}$$

Donde K es una constante llamada parámetro de almacenamiento, y x es un factor de peso que expresa la influencia relativa de las entradas y las salidas del almacenamiento del tramo.

La ecuación 4.9 se planteó pensando en que el almacenamiento de un tramo de río se puede analizar en dos partes como se muestra en la figura 4.3. El primero de ellos es llamado *almacenamiento de prisma*, (KO).

El almacenamiento de prisma depende únicamente de las salidas y sería el único si la superficie de agua fuera paralela a la pendiente del fondo del cauce. Este almacenamiento se puede comparar como el que se tiene en un vaso, si se combina la ecuación de descarga del vertedor

$$Ov = CL(E - E_0)^{\frac{3}{2}}, E > E_0$$

Donde:

Ov = Gasto vertido (m^3/s)
C = Coeficiente de descarga del vertedor (Adim.)
L = Longitud de la cresta del vertedor (m)
E = Elevación de la superficie libre del vaso (msnm)
E_0 = Elevación de la cresta del vertedor (msnm)

con las curvas elevación – volumen y elevación – área de un vaso, con lo que resulta:

$$V_p = f(O) \tag{4.10}$$

Donde f significa una función. En el caso de cauces, se supone que la función $f(O)$ es de la forma:

$$f(O) = KO \tag{4.11}$$

El otro tipo de almacenamiento que ocurre en un río, que por lo general no se representa en el caso de vasos, se debe al efecto que tiene la pendiente de la superficie libre del agua en el gasto. Este tipo de almacenamiento es llamado *almacenamiento de cuña.*

Dicha pendiente depende tanto de las entradas como de las salidas, y en el método de Muskingum el almacenamiento correspondiente a la cuña se toma como una función lineal de la diferencia de ambas:

$$Vc = f(I - O) = Kx(I - O) \tag{4.12}$$

De la ecuación 4.9 se obtiene:

$$\Delta V = V_{i+1} - V_i = K\left[x\left(I_{i+1} - I_i\right) + \left(1-x\right)\left(O_{i+1} - O_i\right)\right] \tag{4.13}$$

Sustituyendo la ecuación 4.13 en la ecuación 4.8 se obtiene:

$$\frac{I_{i+1} + I_i}{2}\Delta t - \frac{O_{i+1} + O_i}{2}\Delta t = K\left[x\left(I_{i+1} - I_i\right) + \left(1-x\right)\left(O_{i+1} - O_i\right)\right] \tag{4.14}$$

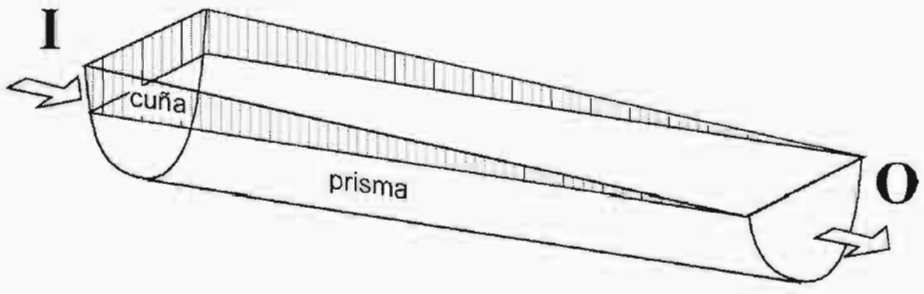


Figura 4.3 *Almacenamiento en un río durante el paso de una avenida (Aparicio, 2001)*

Donde si despejamos O_{i+1}:

$$O_{i+1} = \frac{Kx + \frac{\Delta t}{2}}{K\left(1-x\right) + \frac{\Delta t}{2}} I_i + \frac{\frac{\Delta t}{2} - Kx}{K\left(1-x\right) + \frac{\Delta t}{2}} I_{i+1} + \frac{K\left(1-x\right) - \frac{\Delta t}{2}}{K\left(1-x\right) + \frac{\Delta t}{2}} O_i \tag{4.15}$$

O bien:

$$O_{i+1} = C_1 I_i + C_2 I_{i+1} + C_3 O_i \tag{4.16}$$

Donde:

$$C_1 = \frac{Kx + \frac{\Delta t}{2}}{\alpha} \tag{4.17}$$

$$C_2 = \frac{\frac{\Delta t}{2} - Kx}{\alpha} \tag{4.18}$$

$$C_3 = \frac{K(1-x) - \frac{\Delta t}{2}}{\alpha} \tag{4.19}$$

$$\alpha = K(1-x) + \frac{\Delta t}{2} \tag{4.20}$$

Nótese que:

$$C_1 + C_2 + C_3 = 1 \tag{4.21}$$

Con la ecuación 4.16 es posible analizar cualquier tránsito de avenida por el tramo, dados Δt y los valores de K y x.

El parámetro K tiene unidades de tiempo y su valor es aproximadamente igual al tiempo de viaje del pico de la avenida a lo largo del tramo:

$$K = \frac{L}{\bar{v}} \tag{4.22}$$

Donde L es la longitud del tramo y $\bar{v}$ es la velocidad promedio de pico de la avenida, $\bar{v}$ puede estimarse, en relación con la velocidad media del agua v, como:

$$\bar{v} \approx 1.5v \tag{4.23}$$

El parámetro x puede oscilar entre 0.0 y 0.5.

- Si x = 0.0, el volumen de almacenamiento en el tramo sólo es una función de salida O, lo que significa que no existe almacenamiento en cuña y el tramo se comporta como un vaso cuya curva de gasto está descrita por la ecuación 4.11.

- Si x = 0.5, las entradas y las salidas tienen la misma importancia y no habría ningún abatimiento en el pico.

En términos muy generales se puede decir que x se aproxima a cero en cauces muy caudalosos y de pendiente muy pequeña, y a 0.5 en caso contrario. Cuando no se tengan todos los datos es recomendable tomar x = 0.2 como valor medio.

Cuando se cuenta con al menos una avenida en ambos extremos de cauce, los valores de K y x son calculados con mayor precisión mediante el siguiente razonamiento.

Si se dibuja la ecuación 4.9 en una gráfica considerando V como ordenada y $(xI + (1-x) O)$ como abscisa, se obtendrá una línea recta con pendiente K. por otra parte, el volumen de almacenado en el tramo del río hasta t_0 dado es el área acumulada entre el hidrograma de entrada y salida (ver figura 4.4), o mejor dicho:

$$V = \int_0^{t_0} (I - O)dt \qquad (4.24)$$

Entonces, si se supone algún valor de x, se puede calcular $(xI + (1-x) O)$ y el resultado se grafica contra el volumen almacenado para tiempo en el intervalo $0 < t < t_1$, (ver figura 4.4), y la gráfica deberá describir aproximadamente una línea recta con pendiente K si el valor propuesto de x es el correcto. En caso contrario, es necesario suponer otro valor de x hasta que se obtenga una línea aproximadamente recta. (Ver figura 4.5)

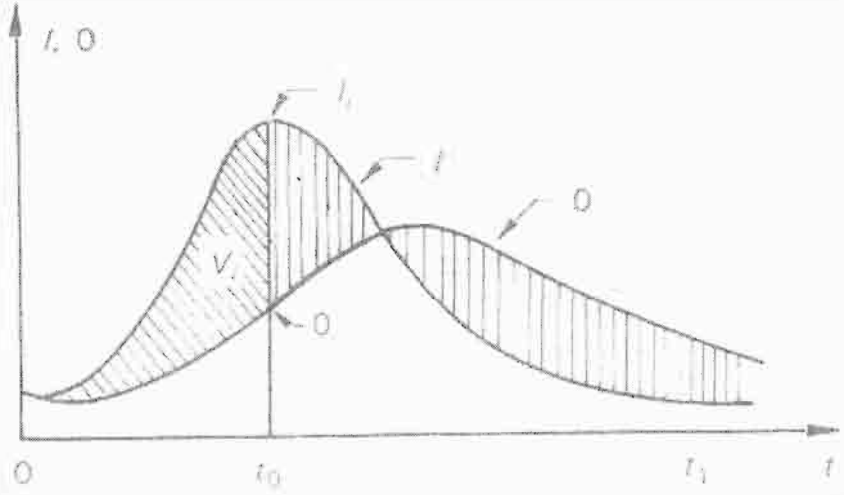


Figura 4.4 *Volumen almacenado durante el tránsito*
(Aparicio, 2001)

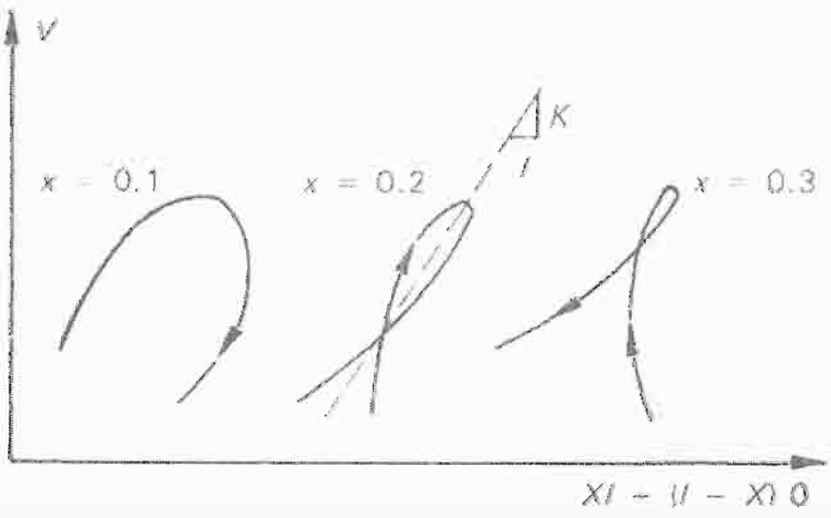


Figura 4.5 *Obtención del parámetro K*
(Aparicio, 2001)

4.3.2. Método de Muskingum – Cunge

Cunge combinó métodos hidráulicos con la simplicidad del método de Muskingum. Calcula las dos constantes utilizadas en el método de Muskingum, K y x, mediante parámetros hidráulicos del cauce (referencia).

$$K = \frac{\Delta x}{c} \tag{4.25}$$

$$x = \frac{1}{2}\left(1 - \frac{Q}{BS_0 c\Delta x}\right) \tag{4.26}$$

Donde:

Δx = longitud del tramo del cauce considerado (m)
c = "celeridad" = velocidad media (m/s)
S_0 = pendiente media del cauce (adimensional)
Q = caudal (m^3/s)
B = anchura del cauce (m)

La correcta aplicación de este método requiere elegir correctamente el Δt y el Δx. Para ello se dividirá el tramo estudiado en subtramos, de modo que el caudal de salida de uno de ellos será el caudal de entrada del siguiente (US Army Corps of Engineers, 1994).

4.4 Método de Muskingum considerando flujos laterales

Posiblemente el modelo más simple para incorporar el gasto o aportación lateral en el método de Muskingum sea considerar una cantidad α proporcional al gasto (I) que entra al tramo (O'Donnell, 1985). Tal consideración fue originalmente planteada por Nerc (1975) en el análisis de crecientes por fusión de nieve y se describe esquemáticamente en la figura 4.6. En tal condición las ecuaciones del método se transforman en:

$$O_{j+1} = \delta_1 I_{j+1} + \delta_2 I_j + \delta_3 O_j \tag{4.27}$$

Por lo que ahora:

$$\delta_1 = (1 + \alpha) C_1 \tag{4.28}$$

$$\delta_2 = (1 + \alpha) C_2 \tag{4.29}$$

$$\delta_3 = C_3 \tag{4.30}$$

El desarrollo de la ecuación 4.27 desde $j = 1$ hasta n, número de parejas de datos de I y O, establece un sistema de n ecuaciones con sólo tres incógnitas (sistema sobredeterminado) el cual tiene solución de mínimos cuadrados (O'Donnell, 1985) que es similar al procedimiento matricial de identificación de ordenadas del hidrograma unitario, que hace

uso de la matriz transpuesta de precipitaciones en exceso para modificar el sistema sobredeterminado de ecuaciones y poder encontrar la inversa (Newton y Vinyard, 1967).

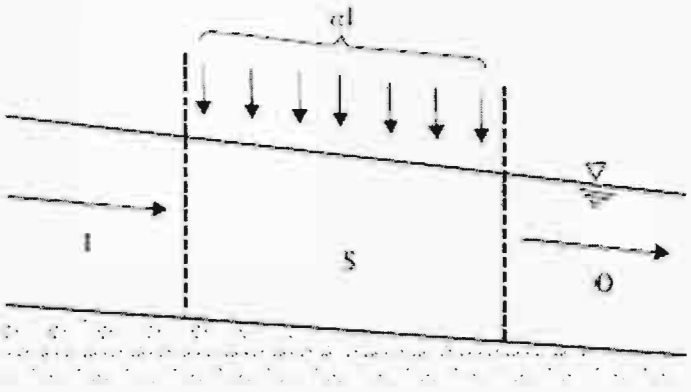


Figura 4.6 *Esquema del modelo de Terence O'Donnell Método de Muskingum de tres parámetros. (Fuente: Campos, 2000)*

Las ecuaciones 4.17 a 4.19 y 4.28 – 4.30 permiten tener una ecuación operativa idéntica para ambos métodos de Muskingum, el convencional y el de tres parámetros (K, x, α).

Por lo anterior, es posible desarrollar el método de mínimos cuadrados a partir de la ecuación 4.16 y aplicar los resultados a la ecuación 4.27. Interesa la solución de mínimos cuadrados del método de Muskingum convencional, porque ésta es la mejor y además tal criterio será contrastado contra la solución propuesta vía optimización restringida.

Entonces, de acuerdo con la ecuación 4.16, si se tienen n intervalos de tiempo Δt entonces se tendrán las n ecuaciones siguientes:

$$O_2 = C_1 I_2 + C_2 I_1 + C_3 O_1$$

$$O_3 = C_1 I_3 + C_2 I_2 + C_3 O_2$$

.

.

$$O_{j+1} = C_1 I_{j+1} + C_2 I_j + C_3 O_j$$

.

.

$$O_{n+1} = C_1 I_{n+1} + C_2 I_n + C_3 O_n \qquad (4.31)$$

El sistema de ecuaciones 4.31, en notación matricial, corresponde a:

$$| O_{j+1} | = | I_{j+1} \; I_j \; O_j \; | \; | C_j | \qquad (4.32)$$

Si se designa por $| P |$ a la matriz $| I_{j+1} I_j O_j |$ que es rectangular con n renglones y tres columnas y se multiplican ambos miembros de la ecuación 4.32 por la transpuesta de $| P |$, es decir:

$$| P |^T | O_{j+1} | = | P |^T | P | | C_j |$$

El producto de la matriz $| P |$ por su transpuesta $| P |^T$ se denomina matriz $| R |$, que es cuadrada de 3x3 y puede ser fácilmente invertida para obtener la solución del vector $| C_i |$, pues el producto de $| P |^T$ por $| O_{j+1} |$ es el vector $| Q |$ de tres renglones, esto es:

$$| C_i | = | R |^{-1} | Q | \tag{4.33}$$

El sistema 4.31 de n ecuaciones ha sido condensado a sólo tres ecuaciones sin perder información. Si los valores de C_i obtenidos con la ecuación 4.33 se utilizan en el sistema 4.32 para evaluar los elementos del vector $| O_{j+1} |$ se obtienen los valores estimados $| \hat{O}_{j+1} |$ y entonces la suma de $(\hat{O}_{j+1} - O_{j+1})^2$ será mínima y se podrá definir el error medio cuadrático (EMC) como:

$$\text{EMC} = \left[\frac{\sum \left(\hat{O} - O \right)^2}{n} \right]^{\frac{1}{2}} \tag{4.34}$$

En donde n representa el número de valores comparados, y $\hat{O}$ y O son los gastos de los hidrogramas de salidas estimados y observados, respectivamente.

A partir de las ecuaciones 4.17 y 4.18 se obtienen las expresiones de los parámetros K y x del método de Muskingum convencional, obtenidos por un procedimiento matemático objetivo, el de mínimos cuadrados, esto es:

$$K = \Delta t \left(\frac{1 - C_1}{C_1 + C_2} \right) \tag{4.35}$$

$$x = \left(\frac{C_1 - C_2}{2 - 2C_1} \right) \tag{4.36}$$

Para obtener los valores de los coeficientes δ_i a partir de los C_j es necesario evaluar el parámetro α y aplicar las ecuaciones 4.28 a 4.30; sin embargo, con fines de comparación y sobre todo de caracterización hidrológica de un tramo de río, es necesario contar con los tres valores de los parámetros K, x y α, cuyas expresiones son:

$$K = \Delta t \left(\frac{C_1 + C_2 C_3}{(1 - C_3)(C_1 + C_2)} \right) \tag{4.37}$$

$$x = 0.50\left[1-\left(\frac{C_2 + C_1C_3}{C_1 + C_2C_3}\right)\right] \tag{4.38}$$

$$\alpha = \frac{C_1 + C_2 + C_3 - 1}{1 - C_3} \tag{4.39}$$

Lógicamente, si no existe flujo o aportación lateral, se tiene que $\alpha = 0$, entonces $\delta_j = C_j$ y $\Sigma I = \Sigma O$ en cambio, cuando $\alpha > 0$ se tendrá que $\Sigma I = \omega \Sigma O$ donde ω es un factor de ganancia diferente en cada evento. O'Donnell (1985) plantea la posibilidad que una descarga o salida lateral ocurra en algunos tramos de río, de manera que entonces α tendrá un valor negativo.

4.5 Casos de estudio

En el desarrollo de la aplicación se analizarán cuatro casos de estudio:

1. El primer caso es la aplicación de una RNA a los datos del experimento numérico similar al "evento Wilson" discutido por O'Donnell (1985) y estudiado por Aldama (1990) en una evaluación comparativa de procedimientos de calibración del método de Muskingum.

2. El segundo caso corresponde a datos históricos de avenidas aisladas de la región hidrológica número 30 Grijalva – Usumacinta, en la parte baja del río Usumacinta. En este ejemplo se tratarán únicamente avenidas aisladas cuya calibración se realizará con los gastos registrados en el lapso de 1964 a 1971, donde se puede apreciar la presencia de flujo lateral.

3. El tercer caso corresponde al estudio de trenes de avenidas, las cuales se registraron en la región hidrológica número 30 Grijalva – Usumacinta durante el lapso de 1964 a 1971, en la parte baja del río Usumacinta, con presencia de flujo lateral.

4. El cuarto caso se enfocará al análisis de gastos (con influencia de flujos laterales) históricos de entrada de la estación hidrométrica Azueta, ubicada en el río Tesechoacán, generados aguas arriba debido al escurrimiento superficial de la confluencia de los ríos Manso y Cajones, donde se encuentran ubicadas las estaciones hidrométricas Zapote y Monte Rosa respectivamente, todas ellas en la región hidrológica 28.

Los resultados del primer caso se compararán con los valores reportados por Aguilar (1995). El desarrollo de las RNA de esta aplicación es realizado en la herramienta NNtool del Toolbox de MATLAB 6.5, herramienta especifica para el desarrollo de RNA.

4.5.1 Caso 1 – Experimento numérico

Considérese un canal trapecial con ancho de plantilla b = 100m, talud k = 2, longitud L = 50,500m, pendiente de plantilla S_0 = 0.0001 y rugosidad de Manning n = 0.08. Para fines de calibración, se supondrá que el evento de avenida discutido por Aldama (1990) (Tabla 4.1), ocurre en dicho canal.

Numero de Intervalo	**Tiempo** (hrs)	**Gasto de entrada I** (m^3/s)	**Gasto de salida O** (m^3/s)
1	0	22.000	22.000
2	6	23.000	22.000
3	12	35.000	22.001
4	18	71.000	22.005
5	24	103.000	22.078
6	30	111.000	22.624
7	36	109.000	25.541
8	42	100.000	36.093
9	48	86.000	57.113
10	54	71.000	76.565
11	60	59.000	85.625
12	66	47.000	86.555
13	72	39.000	82.866
14	78	32.000	76.915
15	84	28.000	70.106
16	90	24.000	63.251
17	96	22.000	56.781
18	102	22.000	50.894
19	108	22.000	45.651
20	114	22.000	41.066
21	120	22.000	37.137
22	126	22.000	33.834
23	132	22.000	31.114
24	138	22.000	28.910
25	144	22.000	27.158
26	150	22.000	25.789
27	156	22.000	24.746
28	162	22.000	23.963
29	168	22.000	23.384
30	174	22.000	22.964
31	180	22.000	22.662

Tabla 4.1 *Datos para el experimento numérico*
(Fuente: Aguilar, 1995)

Es importante resaltar que los gastos del hidrograma de salida fueron obtenidos mediante un método hidráulico.

Con el objeto de evaluar la capacidad predictiva de la metodología propuesta en este primer caso en comparación con otros procedimientos existentes, en situaciones de pronóstico, el hidrograma del citado evento de avenida se multiplica por factores 2, 5 y 10 y cada uno de los hidrogramas resultantes se transitará por el canal empleando un método hidráulico, obteniéndose así los correspondientes hidrogramas de salida.

Los hidrogramas de salida obtenidos mediante el tránsito hidráulico se toman como datos reales y contra ellos se comparan los hidrogramas pronosticados por el método propuesto y por el método de Muskingum calibrado con las técnicas propuestas por Gill (1997) y O´Donnell (1985), que han sido discutidas y comparadas por Aldama (1990) y finalmente con los datos generados por las redes neuronales.

En el desarrollo de la aplicación se seguirá básicamente la metodología descrita en la figura 4.7

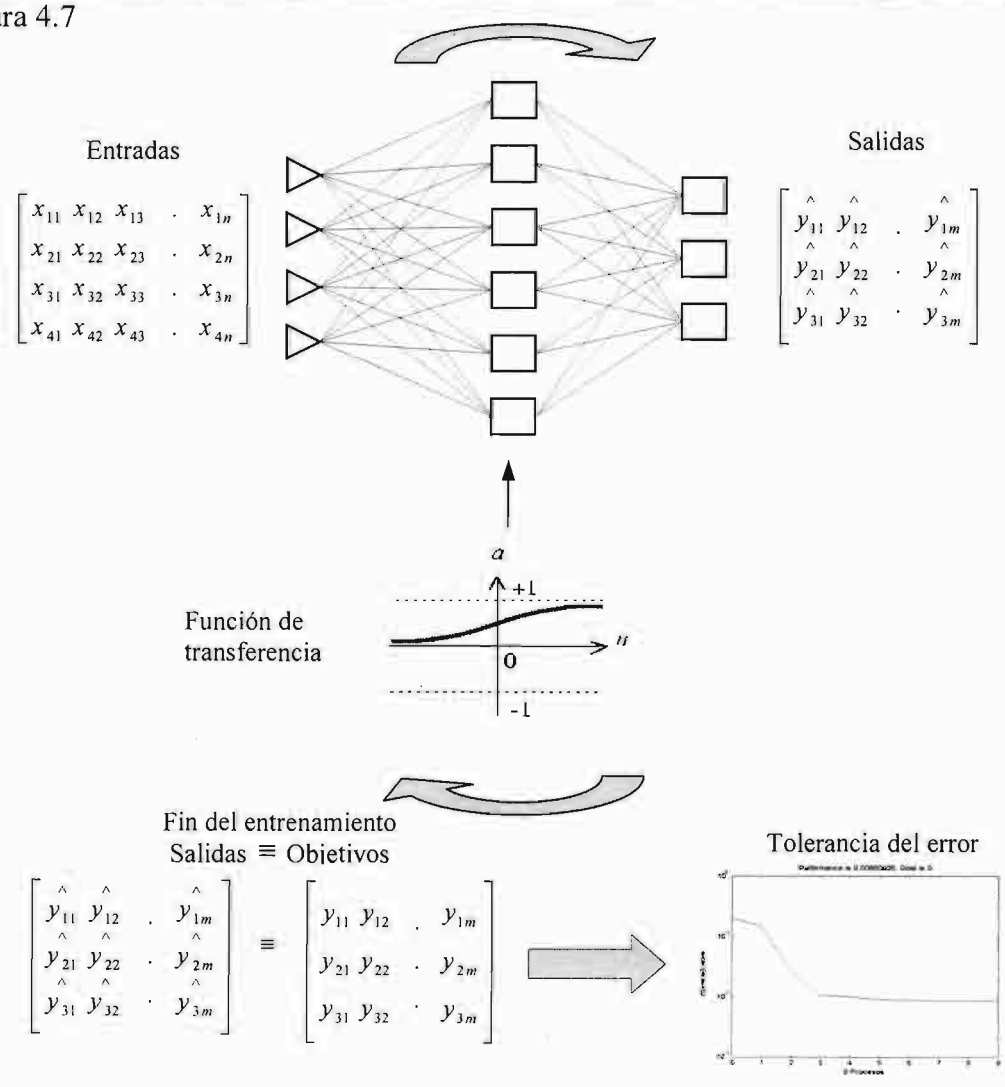


Figura 4.7 *Metodología ilustrativa de RNA*

Una vez que se normalizó tanto el hidrograma de entrada como el de salida de la tabla 4.1 por el gasto pico del hidrograma de entrada, se procedió a realizar diversas arquitecturas de la RNA con los siguientes parámetros:

```
Tipo de red: Unidireccional con retropropagación
Rango de Entradas: [0,1]
Función de entrenamiento: trainlm
Función de aprendizaje: learngdm
Función de desempeño: mse
Número de capas: 3 (1 de entrada, 1 oculta, 1 salida)
Número de neuronas capa oculta: n
Función de transferencia: Logsig
```

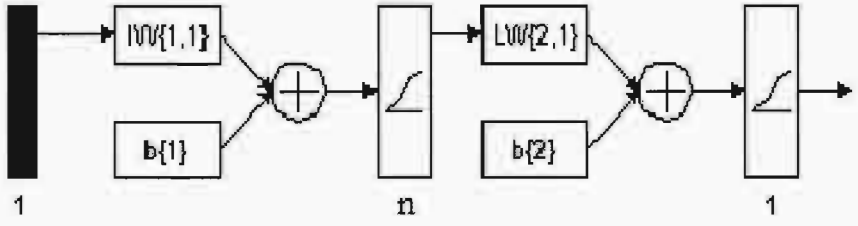


Figura 4.8 *RNA con estructura 1 – n – 1*

Realizado el entrenamiento de la red variando el número de neuronas en la capa oculta (n), Los resultados se pueden apreciar en las figuras 4.9, 4.10 y 4.11:

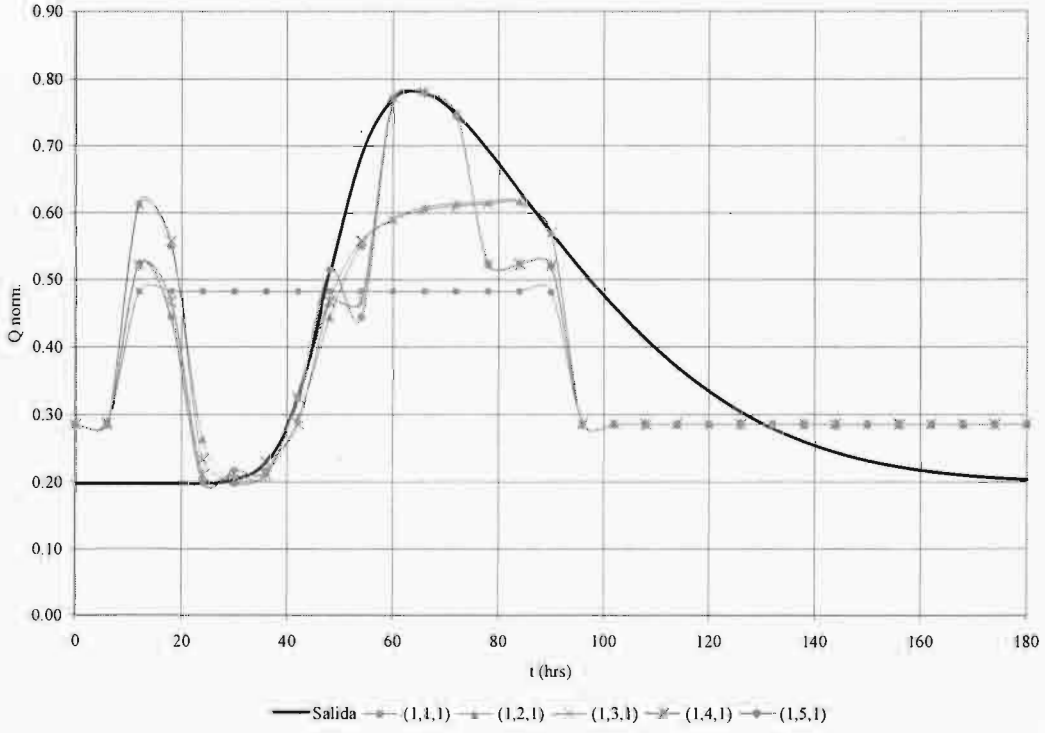


Figura 4.9 *Resultado del entrenamiento de la red con estructura 1 – n – 1; (1<n<5)*

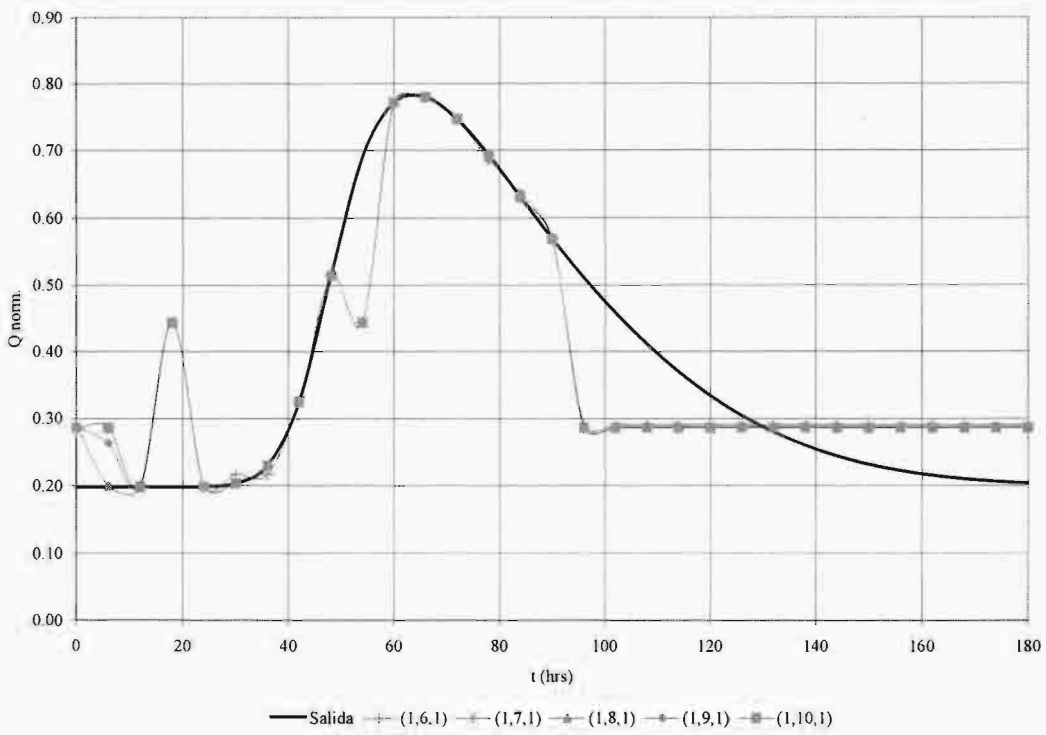


Figura 4.10 *Resultado del entrenamiento de la red con estructura 1 – n – 1; (6<n<10)*

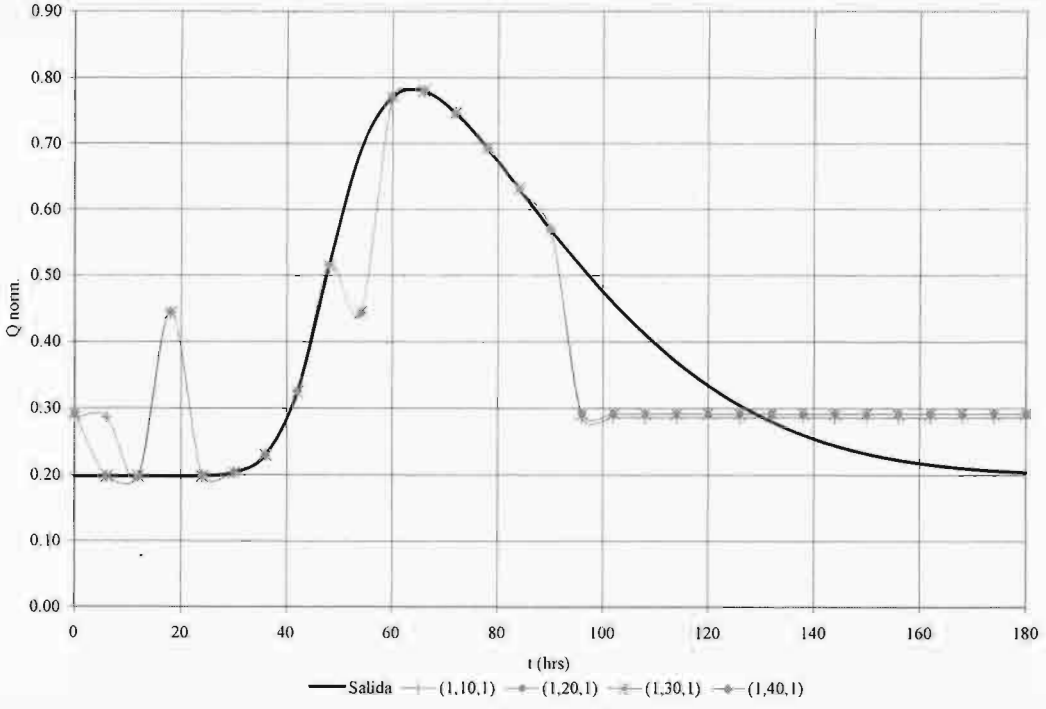


Figura 4.11 *Resultado del entrenamiento de la red con estructura 1 – n – 1; (10<n<40)*

En la figura 4.12 se puede apreciar el comportamiento del error para las primeras 10 estructuras de RNA propuesta (Figuras 4.9 y 4.10), teniendo como función de comparación entre los valores de salida con los objetivos la función `mse`.

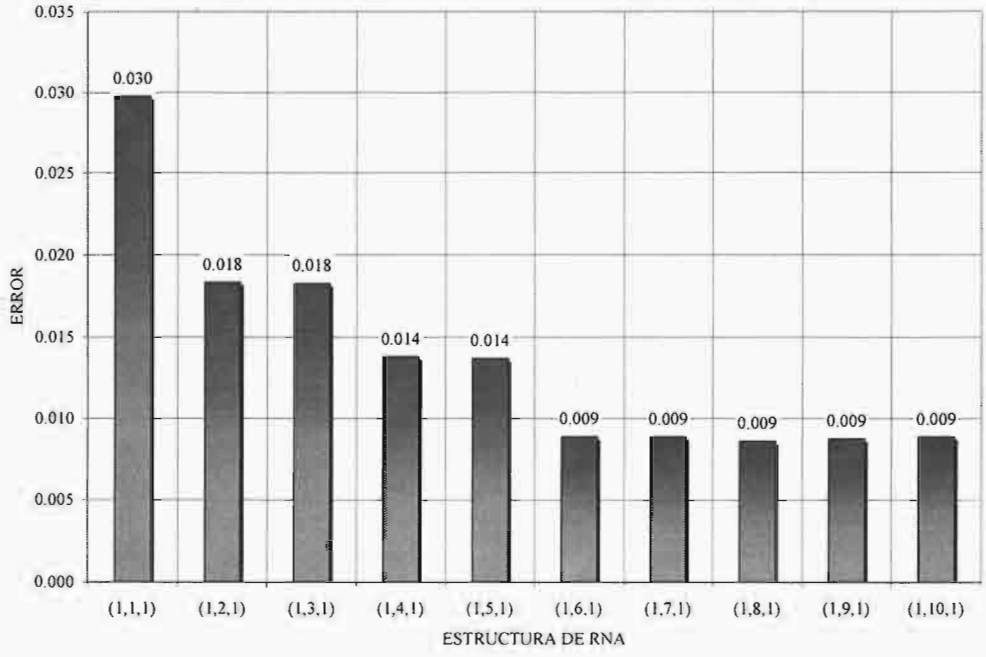


Figura 4.12 *Error durante el entrenamiento con estructura 1 – n – 1*

Como puede apreciarse en las figuras 4.9, 4.10 y 4.11 el entrenamiento de la red no es el adecuado según se aprecia principalmente en las partes inicial y final del hidrograma de salida normalizado, es decir la red neuronal propuesta con estructura *1 – n – 1* no ha aprendido adecuadamente de los datos con los que fue entrenada (entradas y objetivos), por lo que la red no será capaz de generalizar resultados debido al mal desempeño de su entrenamiento. Con la finalidad de lograr un mejor aprendizaje de la red y para obtener un mejor entrenamiento se agregará otra capa oculta (fig. 4.13) con función de transferencia logsig; los resultados se muestran en las figuras 4.14 y 4.15:

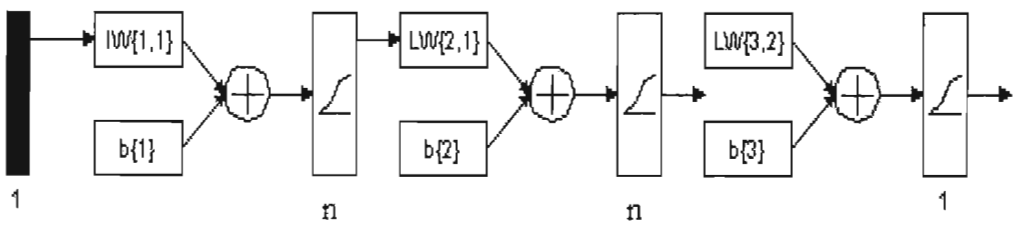


Figura 4.13 *RNA con estructura 1 – n – n – 1*

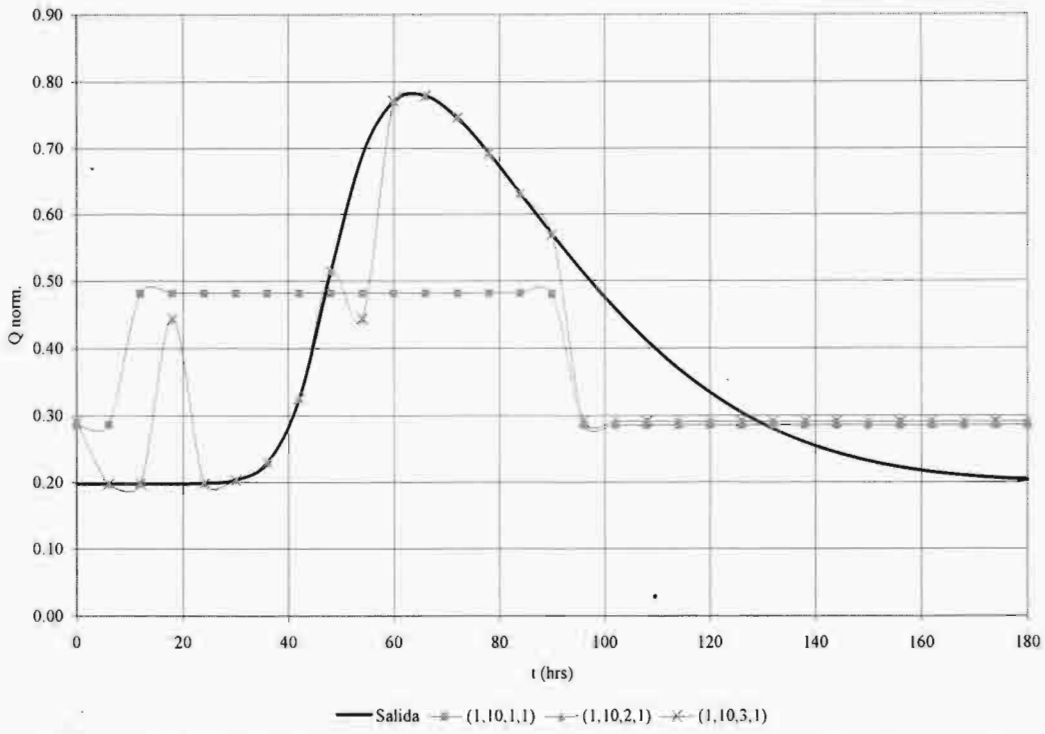


Figura 4.14 *Resultado del entrenamiento de la red con estructura 1 – 10 – n – 1 (1 < n < 3)*

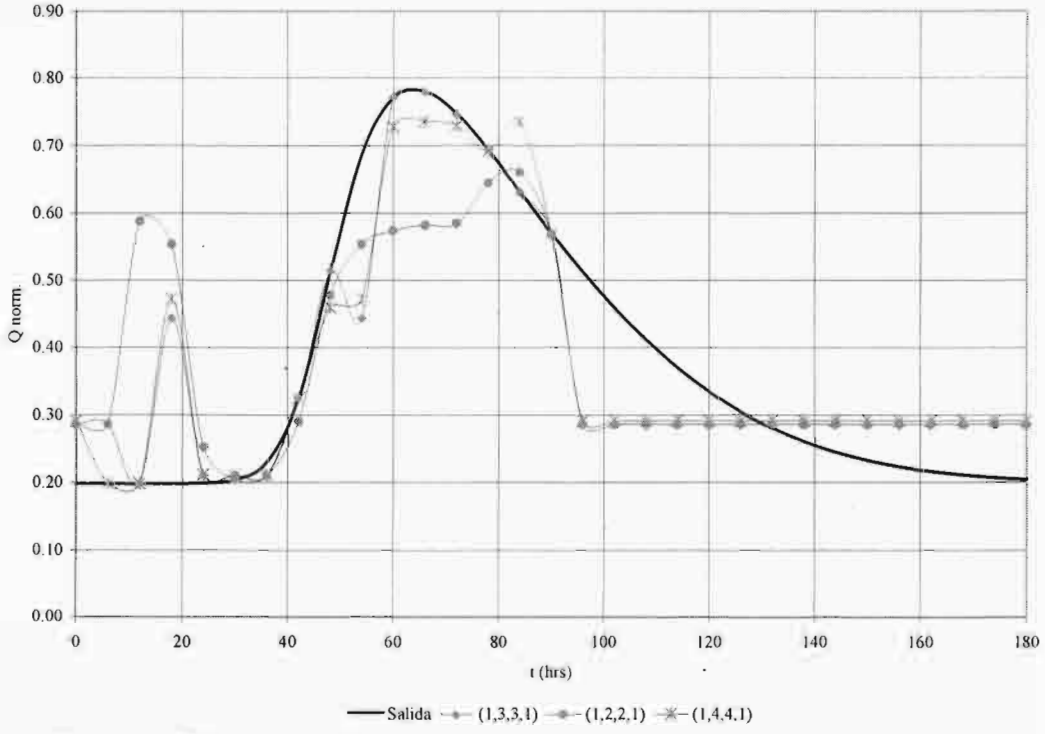


Figura 4.15 *Resultado del entrenamiento de la red con estructura 1 – n_1 – n_2 – 1 ($2 < n_1 < 4$, $2 < n_2 < 4$)*

Como puede apreciarse en este primer desarrollo de aprendizaje de la red aun no existe buen entrenamiento, por lo que los valores de salida obtenidos quedan en muchos casos muy lejos del valor objetivo utilizado para el entrenamiento.

En los siguientes desarrollos se harán cambios en la arquitectura de la red (fig. 4.16) con respecto a la forma de presentar los valores del hidrograma de entrada en la capa de entrada de la RNA y los del hidrograma de salida en los objetivos de la misma, los resultados del entrenamiento se muestran en las figuras 4.17 y 4.18.

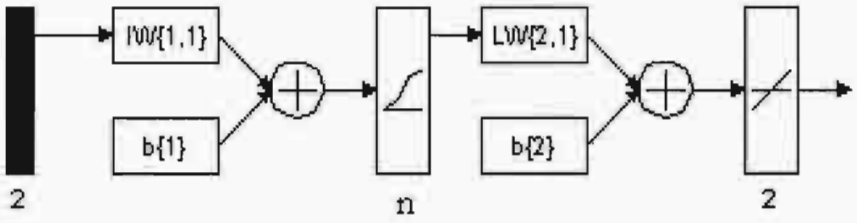


Figura 4.16 *RNA con estructura 2 – n – 2*

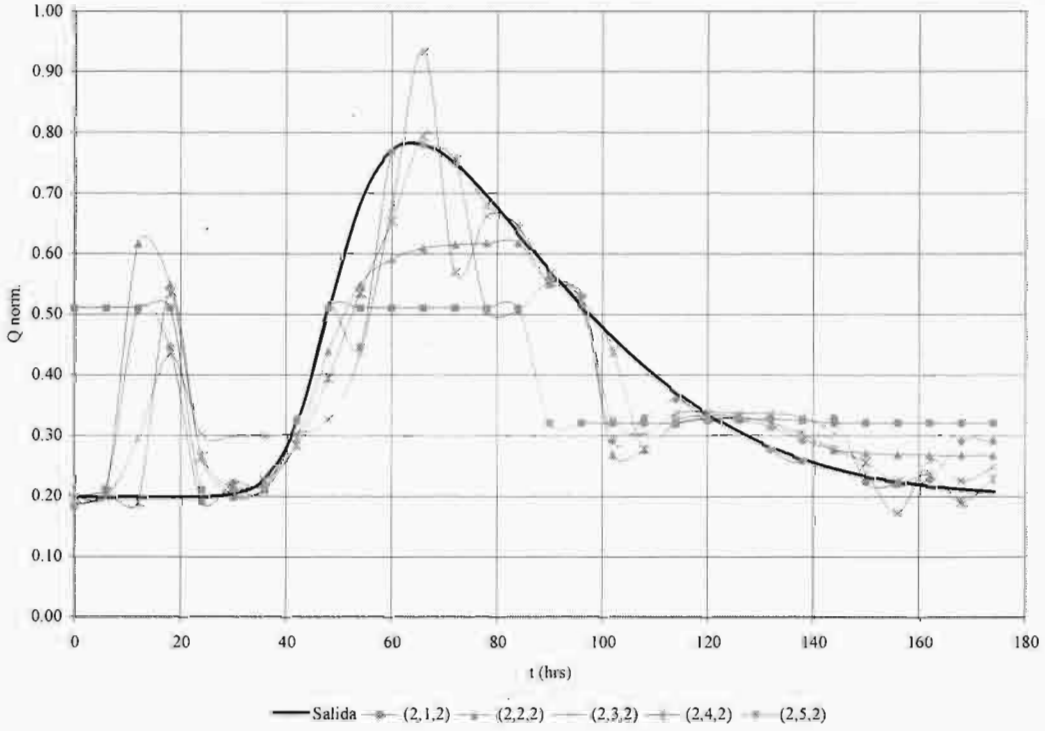


Figura 4.17 *Resultado del entrenamiento de la red con estructura 2 – n – 2; (1 < n < 5)*

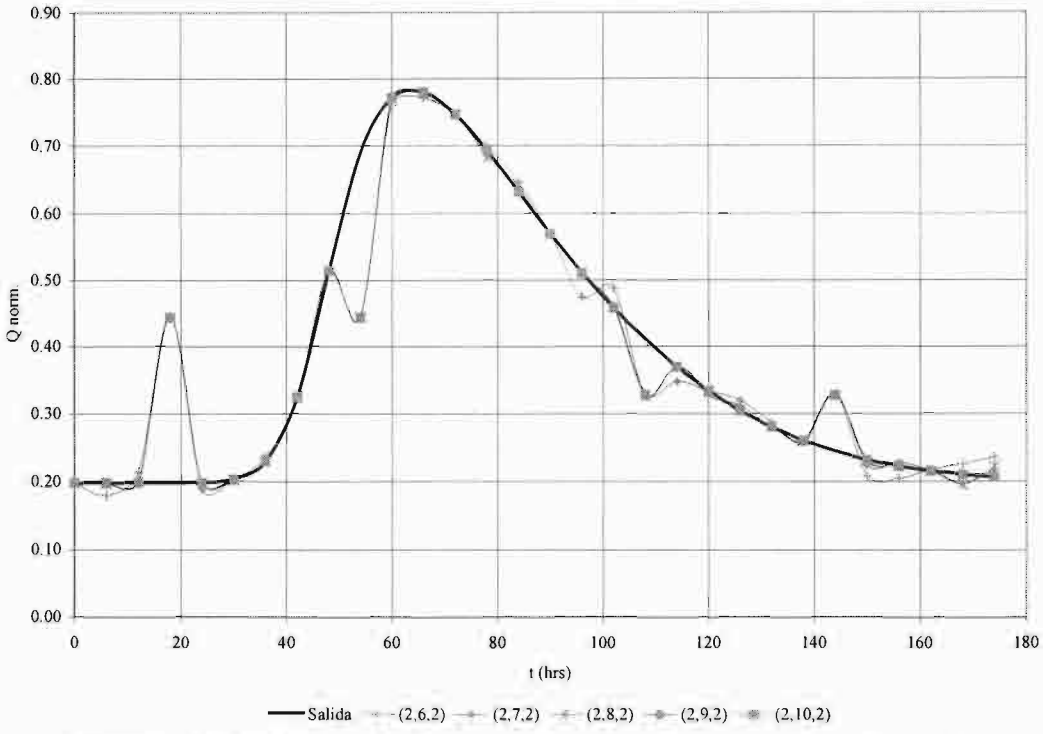


Figura 4.18 *Resultado del entrenamiento de la red con estructura 2 – n – 2 (6 < n < 10)*

Como resultado de presentar el hidrograma de entrada con dos neuronas y el hidrograma de salida con dos neuronas se puede observar que el resultado del entrenamiento es mucho mejor que teniendo una neurona en la capa de entrada y la capa de salida. Sin embargo se puede observar aun la existencia zonas donde el aprendizaje no es aún el deseado, por lo que se presentará tanto el hidrograma de entrada como el de salida mediante tres neuronas (fig. 4.19) y los resultados del entrenamiento de la RNA se muestran en las figuras 4.20 y 4.21.

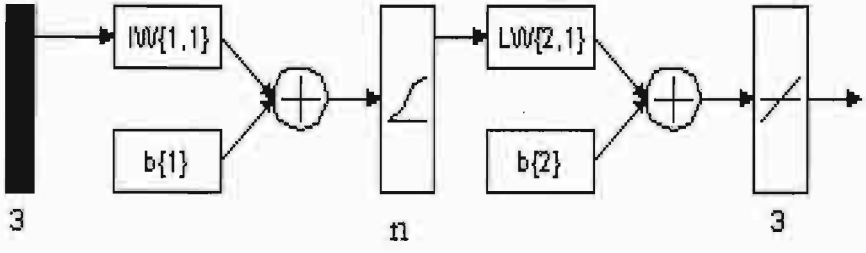


Figura 4.19 *RNA con estructura 3 – n – 3*

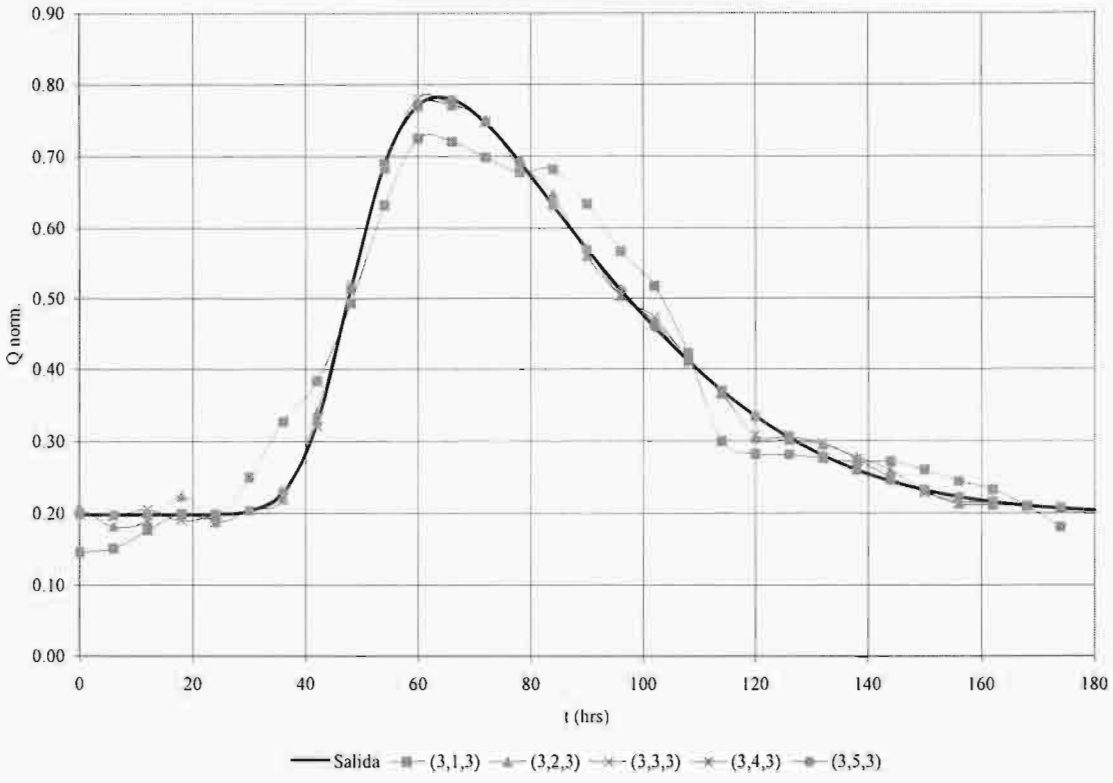


Figura 4.20 *Resultado del entrenamiento de la red con estructura 3 – n – 3;(1 < n < 5)*

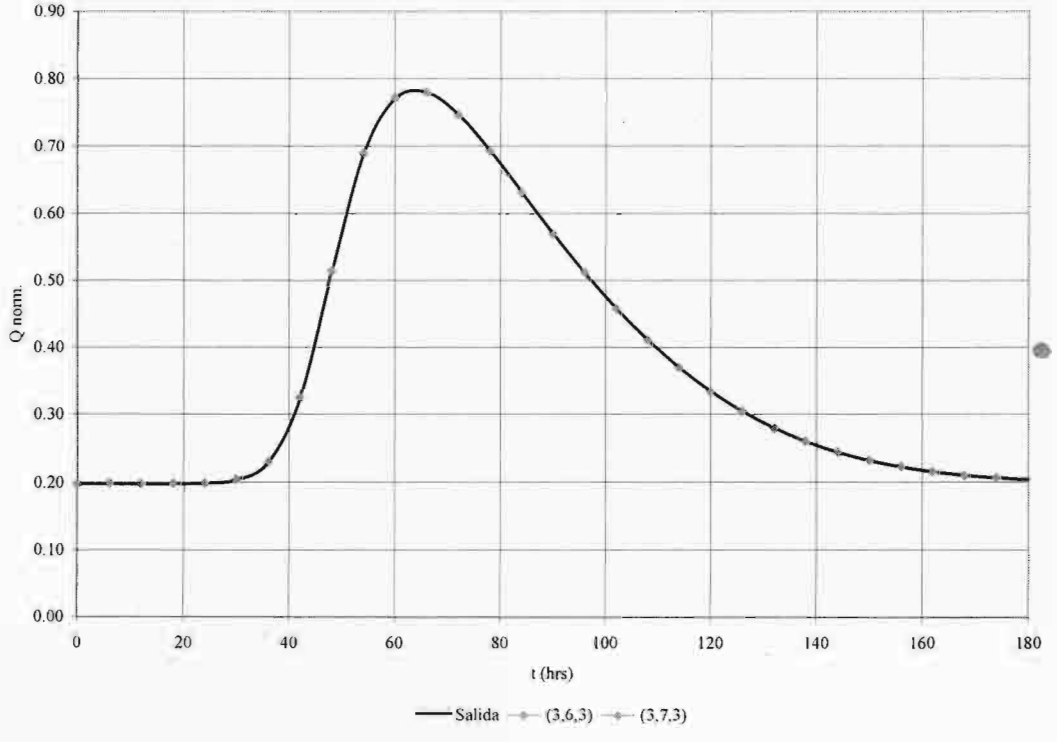


Figura 4.21 *Resultado del entrenamiento de la red con estructura 3 – n – 3;(6 < n <7)*

En figura 4.21 se puede apreciar que se tiene un entrenamiento prácticamente perfecto a partir de una RNA 3 – 6 – 3 y el comportamiento del error en dicha red es el que se muestra en la figura 4.22, donde al tener un buen entrenamiento se necesitan pocas iteraciones para lograr que las salidas sean idénticamente iguales que los objetivos (234 iteraciones (epochs) de las 1000 dadas de instrucción en el toolbox nnet, para esta estructura este caso).

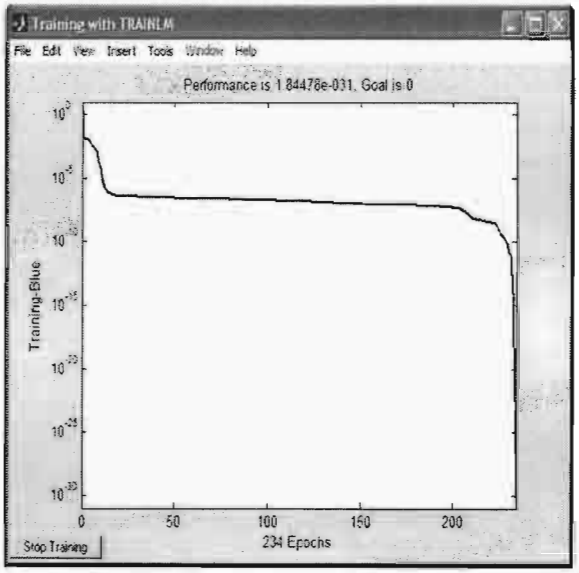


Figura 4.22 *Comportamiento del error en RNA 3 – 6 – 3*

Después de ver los resultados para los casos planteados, podemos ver que una red unidireccional con retropropagación de estructura 3 – 6 – 3 o 3 – n > 6 – 3 se ajusta eficientemente a los datos del tránsito de hidrogramas. Ahora procederemos a introducir a la red neuronal los valores del hidrograma multiplicados por los factores 2, 5 y 10 del cálculo realizado se presenta la comparación con los otros métodos de comparación.

Numero de intervalo	**Tiempo** (hrs)	**I** (m^3/s)	**I*2** (m^3/s)	**I*5** (m^3/s)	**I*10** (m^3/s)
1	0	22.000	44.000	110.000	220.000
2	6	23.000	46.000	115.000	230.000
3	12	35.000	70.000	175.000	350.000
4	18	71.000	142.000	355.000	710.000
5	24	103.000	206.000	515.000	1030.000
6	30	111.000	222.000	555.000	1110.000
7	36	109.000	218.000	545.000	1090.000
8	42	100.000	200.000	500.000	1000.000
9	48	86.000	172.000	430.000	860.000
10	54	71.000	142.000	355.000	710.000
11	60	59.000	118.000	295.000	590.000
12	66	47.000	94.000	235.000	470.000
13	72	39.000	78.000	195.000	390.000
14	78	32.000	64.000	160.000	320.000
15	84	28.000	56.000	140.000	280.000
16	90	24.000	48.000	120.000	240.000
17	96	22.000	44.000	110.000	220.000
18	102	22.000	44.000	110.000	220.000
19	108	22.000	44.000	110.000	220.000
20	114	22.000	44.000	110.000	220.000
21	120	22.000	44.000	110.000	220.000
22	126	22.000	44.000	110.000	220.000
23	132	22.000	44.000	110.000	220.000
24	138	22.000	44.000	110.000	220.000
25	144	22.000	44.000	110.000	220.000
26	150	22.000	44.000	110.000	220.000
27	156	22.000	44.000	110.000	220.000
28	162	22.000	44.000	110.000	220.000
29	168	22.000	44.000	110.000	220.000
30	174	22.000	44.000	110.000	220.000
31	180	22.000	44.000	110.000	220.000

Tabla 4.2 *Datos del experimento numérico y múltiplos del hidrograma de entrada para realizar las pruebas de desempeño de la red entrenada.*

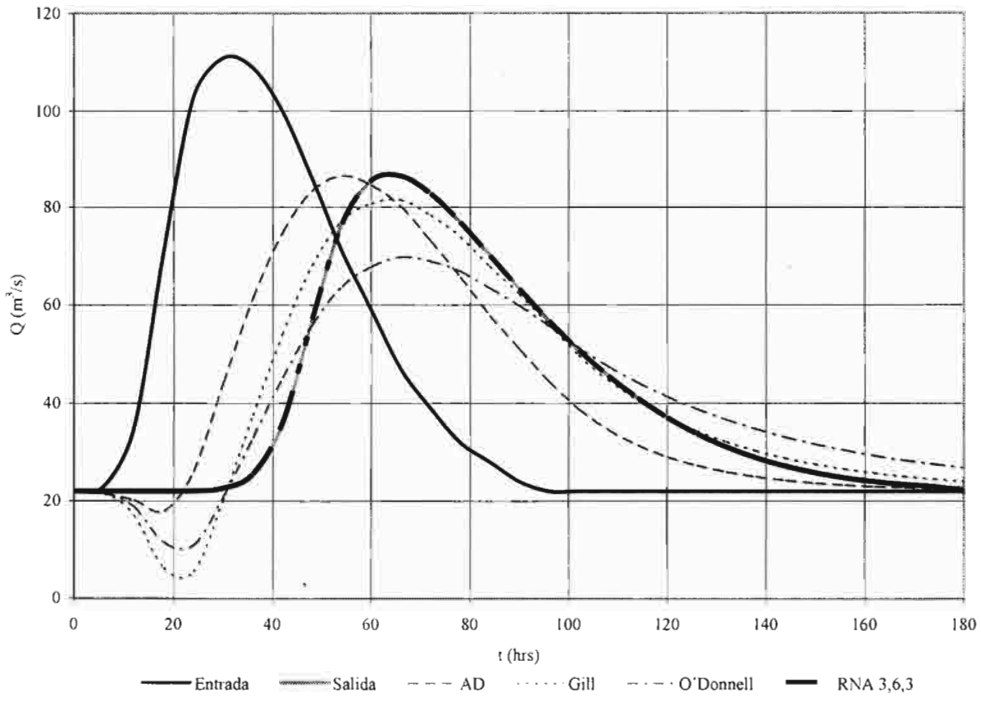


Figura 4.23 *Calibración del método numérico utilizando los métodos de Advección – Difusión, las técnicas de Gill y O´Donnell y RNA*

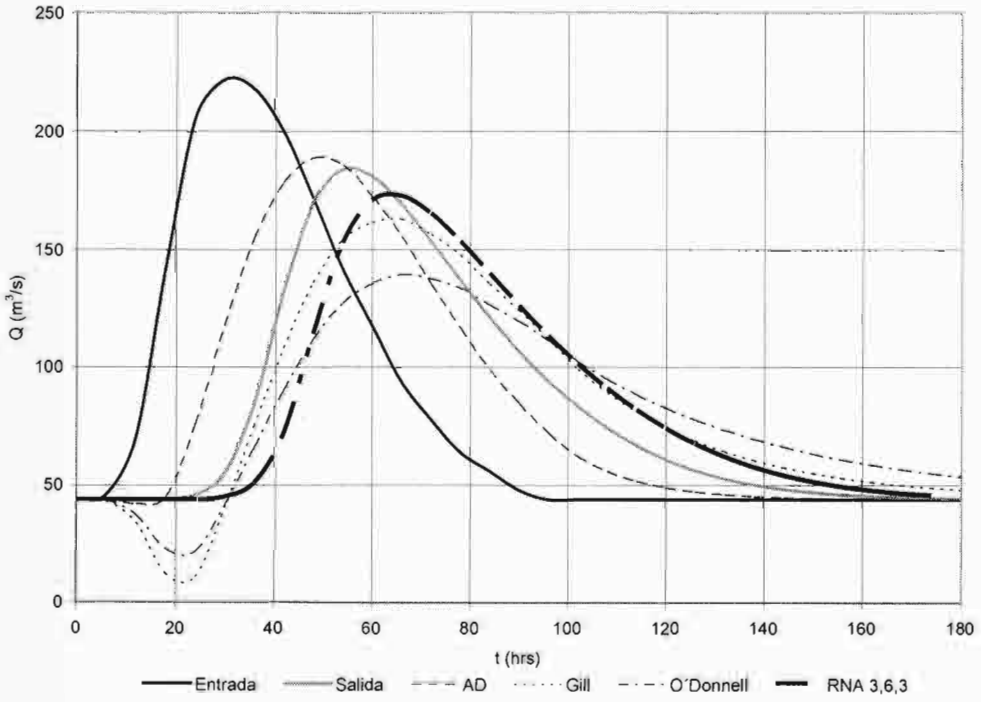


Figura 4.24 *Pronóstico de avenida para dos veces el hidrograma original*

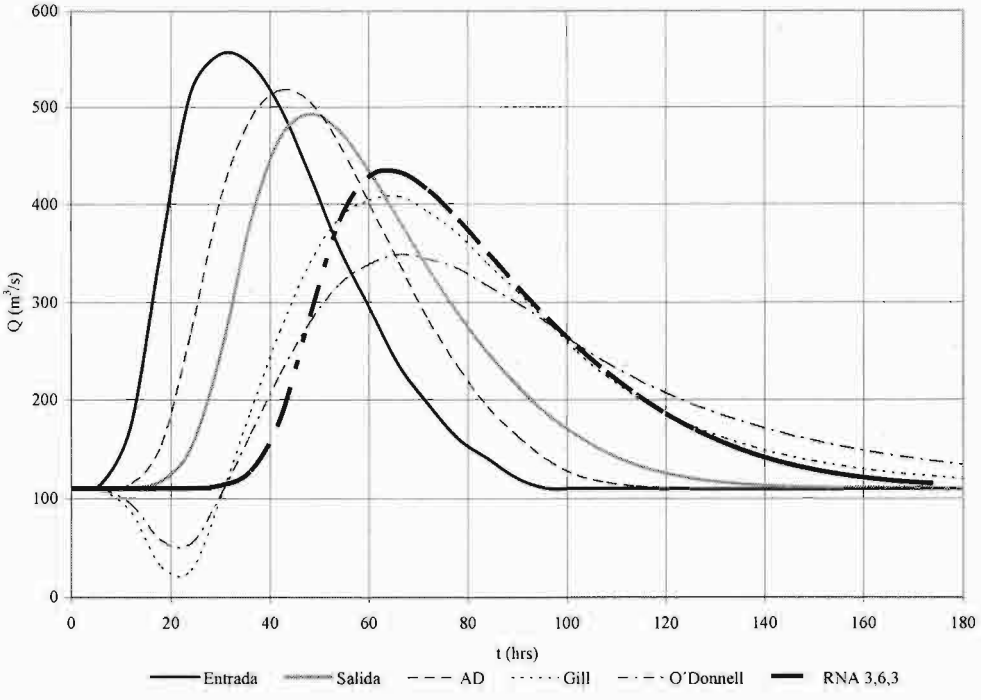


Figura 4.25 *Pronóstico de avenida para cinco veces el hidrograma original*

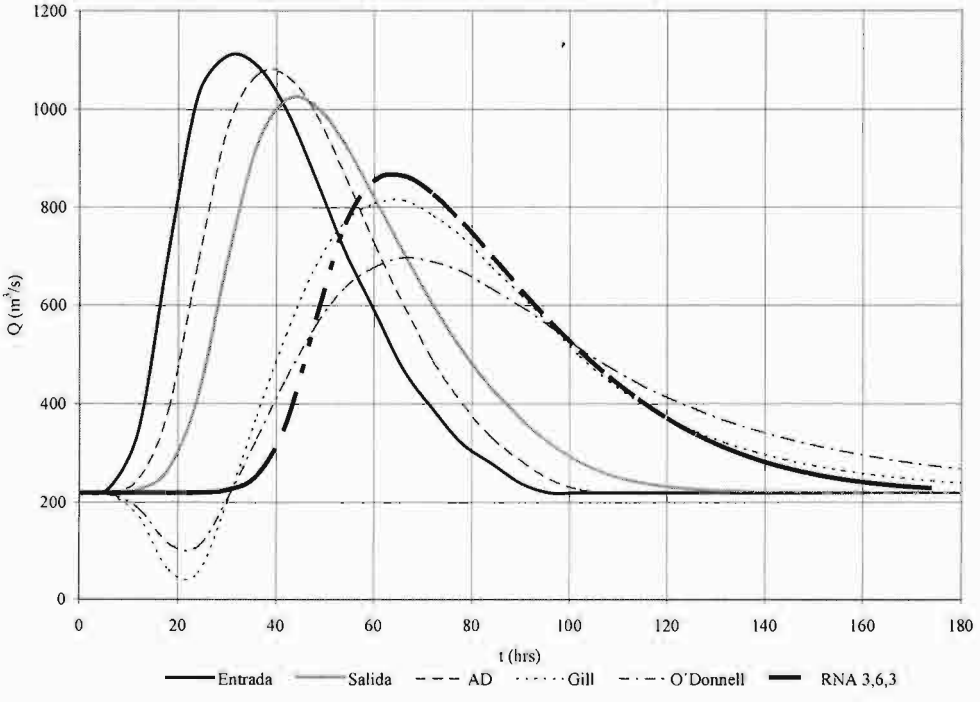


Figura 4.26 *Pronóstico de avenida para diez veces el hidrograma original*

Caso	MÉTODO UTILIZADO				
	Gasto salida Medido	**A – D** pico ajustado	**Gill**	**O´Donnell**	**RNA**
Gasto pico (m^3/s) "Evento Wilson"	86.555	86.556	81.542	69.771	86.555
Diferencia con respecto Al gasto medido (%)	0.000	0.001	-5.792	-19.391	0.000
Diferencia con respecto Al gasto medido (m^3/s)	0.000	0.001	-5.013	-16.784	0
Gasto pico (m^3/s) "Evento Wilson" * 2	183.865	188.968	163.085	139.542	173.110
Diferencia con respecto Al gasto medido (%)	0.000	2.775	-11.302	-24.106	-5.849
Diferencia con respecto Al gasto medido (m^3/s)	0.000	5.103	-20.78	-44.323	-10.755
Gasto pico (m^3/s) "Evento Wilson" * 5	492.756	517.333	407.712	348.855	432.775
Diferencia con respecto Al gasto medido (%)	0.000	4.988	-17.259	-29.203	-12.173
Diferencia con respecto Al gasto medido (m^3/s)	0.000	24.577	-85.044	-143.901	-59.981
Gasto pico (m^3/s) "Evento Wilson" * 10	1019.592	1070.738	815.424	697.71	865.550
Diferencia con respecto Al gasto medido (%)	0.000	5.016	-20.024	-31.570	-15.108
Diferencia con respecto Al gasto medido (m^3/s)	0.000	51.146	-204.168	-321.882	-154.042

Tabla 4.3 *Comparación del gasto pico del tránsito hidrológico correspondiente al experimento numérico mediante las técnicas mencionadas*

Caso	MÉTODO UTILIZADO				
	Tiempo Pico Medido	**A – D** pico ajustado	**Gill**	**O´Donnell**	**RNA**
Tiempo pico (h) "Evento Wilson"	66	54	66	66	66
Diferencia con respecto al tiempo pico (%)	0.00	-18.18	0.00	0.00	0.00
Diferencia con respecto al tiempo pico (h)	0.00	-12	0	0	0
Gasto pico (h) "Evento Wilson" * 2	54	48	66	66	66
Diferencia con respecto al tiempo pico (%)	0.00	-11.11	22.22	22.22	22.22
Diferencia con respecto al tiempo pico (h)	0.00	-6	12	12	12
Gasto pico (h) "Evento Wilson" * 5	48	42	66	66	66
Diferencia con respecto al tiempo pico (%)	0.00	-12.50	37.50	37.50	37.50
Diferencia con respecto al tiempo pico (h)	0.00	-6	18	18	18
Gasto pico (h) "Evento Wilson" * 10	42	36	66	66	66
Diferencia con respecto al tiempo pico (%)	0.00	-14.29	57.14	57.14	57.14
Diferencia con respecto al tiempo pico (h)	0.00	-6	24	24	24

Tabla 4.4 *Comparación del tiempo pico del tránsito hidrológico correspondiente al experimento numérico mediante las técnicas mencionadas*

En la figura 4.23 se aprecia el tránsito tanto hidráulico (método advección – difusión) como del hidrológico (métodos de Gill y O´Donnell) y la RNA realizado para los valores del experimento numérico registrados en la tabla 4.1. Se observa que la RNA se ajusta perfectamente al hidrograma de salida puesto que este fue el caso de entrenamiento. Adicionalmente se puede visualizar cómo el resto de los métodos tienen problemas al inicio de la rama ascendente ya que presentan valores menores que el gasto base (hasta de 18 m^3/s). Posteriormente a dicha zona el método AD es el que más se aproxima al hidrograma de salida mientras que los métodos de Gill y O´Donnell se quedan por debajo del gasto pico en 5.013 y 16.784 m^3/s respectivamente.

Una vez que se duplicaron los valores de los hidrogramas del experimento numérico, en la figura 4.24 se aprecia cómo el método AD disminuye el problema que presentaba en la rama ascendente con respecto a sus ordenadas menores al gasto base, no así los métodos de Gill y O´Donnell los cuales continúan con la misma tendencia que en el caso anterior. Se aprecia como la RNA se aproxima mejor al hidrograma de salida con respecto de los métodos de Gill y O´Donnell, los cuales presentan una diferencia de 5.849%, 11.302% y 24.106% por debajo del gasto pico del hidrograma de salida de forma respectiva.

Quintuplicados y decuplicados los valores de los hidrogramas del experimento numérico se observa que el método AD corrige el problema de las ordenadas del hidrograma transitado en la zona de la rama ascendente, además de que presenta la tendencia a representar valores del gasto mayores a los del hidrograma de salida (4.988% y 5.016% respectivamente) en las primeras 50 horas aproximadamente y el tiempo de pico siempre se adelanta (6 hrs). Se aprecia también la tendencia de los métodos hidrológicos, en el sentido de que los valores del gasto pico permanentemente son menores que los del hidrograma de salida (del orden del 12% al 32%) y el tiempo pico se retrasa para ambos casos en 18 y 24 hrs. respectivamente.

Los métodos hidrológicos y la RNA presentan el mismo retraso en el tiempo pico con respecto al del hidrograma de salida, con la diferencia de que la RNA en los casos en que se duplican, quintuplican y decuplican los hidrogramas tienen un gasto pico mayor y con mejor aproximación con respecto al transitado por los métodos de Gill y O´Donell. Se puede mencionar que dados los resultados del tránsito de la avenida observados en las tablas 4.3 y 4.4, la RNA se comporta mejor que los métodos hidrológicos y presenta cierta deficiencia respecto al tránsito hidráulico.

4.5.2 Caso 2 – Avenidas en la región hidrológica número 30 Grijalva – Usumacinta, en la parte baja del río Usumacinta.

En este segundo caso se evaluará la posibilidad de generalización de una RNA para avenidas históricas registradas en el tramo comprendido entre las estaciones hidrométricas El Tigre (30095) y Boca del Cerro (30019) en la parte baja del río Usumacinta, Figura 4.26.

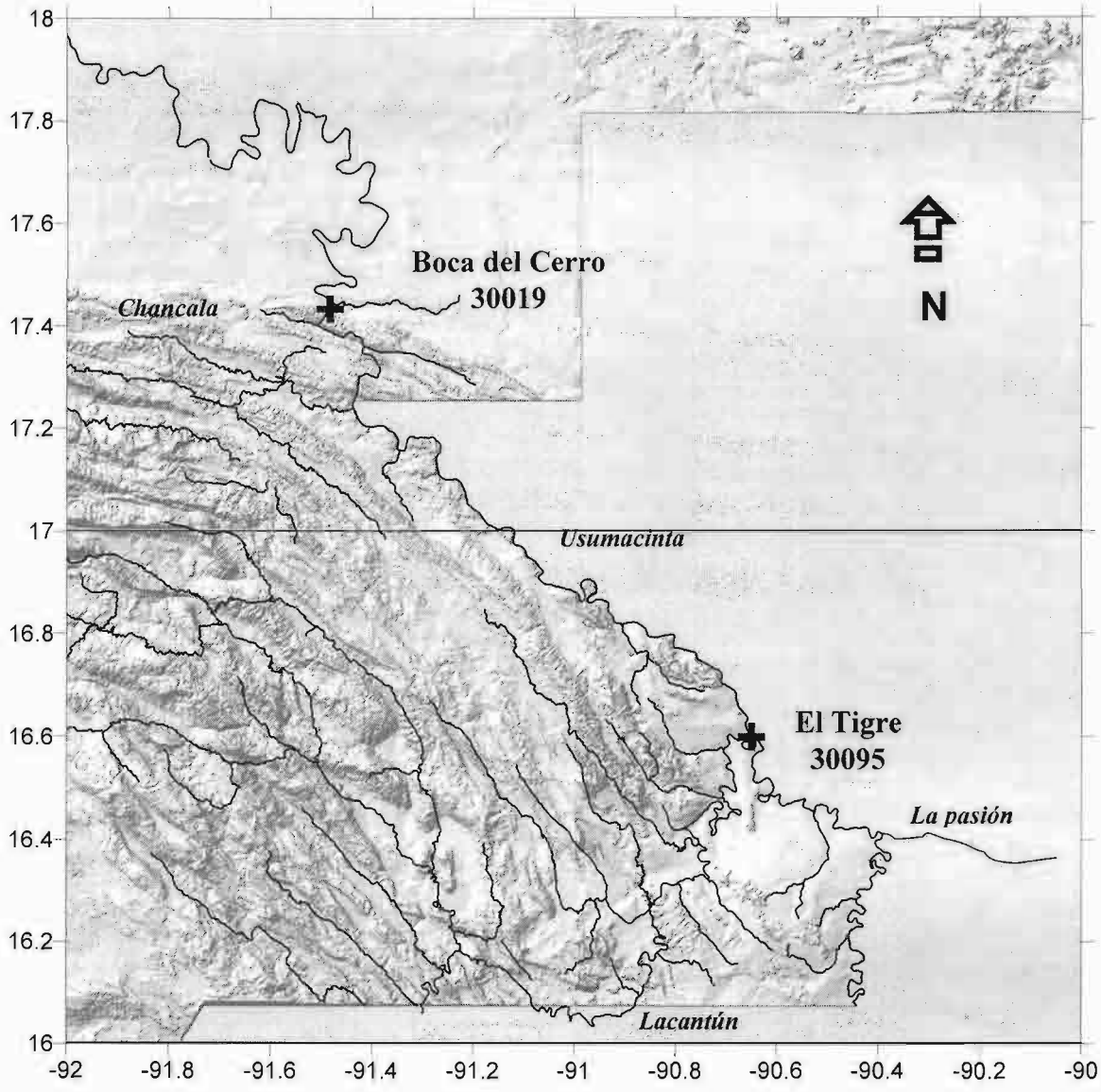


Figura 4.27 *Ubicación de las estaciones de estudio en el río Usumacinta*

Dentro de la región hidrológica número 30 "Grijalva – Usumacinta" se localiza la estación hidrométrica El Tigre sobre el río Usumacinta, formado por confluencia de los ríos La Pasión y Chixoy o Salinas los cuales proceden de territorio guatemalteco, y 500 m aguas arriba de la estación El Tigre confluye el río Lacantún por la margen izquierda.

El área drenada aguas arriba de la estación hidrométrica El Tigre es de 41,852 km^2, y los registros correspondientes permiten conocer el escurrimiento del Usumacinta aguas abajo de la confluencia de los ríos Salinas y Lacantún.

La estación hidrométrica Boca del Cerro se encuentra en la frontera aguas abajo del tramo en estudio, y aguas arriba de la confluencia con el río San Pedro y tiene un área de aportación de 47,697 km^2. Es en este lugar donde empieza la zona denominada Bajo Usumacinta.

Aguas abajo, el río se divide en tres brazos, de los que el occidental va a unirse al Grijalva en un punto que se denomina Tres Brazos; el brazo central se llama San Pedro y San Pablo y desemboca directamente en el golfo de México y el brazo oriental, denominado Palizada, desagua en la Laguna de Términos por la llamada Boca Chica.

Para fines de entrenamiento, se utilizarán cinco avenidas ocurridas entre 1965 y 1971, las cuales tienen aproximadamente la misma duración, y se realizará la validación de la RNA con una avenida de 1972.

Después de la revisión de los datos hidrométricos se eligieron varias avenidas donde la duración fuera prácticamente la misma. Las avenidas seleccionadas se presentaron durante junio – julio de 1965 (figura 4.29), julio de 1966 (figura 4.30), junio – julio de 1968 (figura 4.31), junio – julio de 1969 (figura 4.32) y julio – agosto de 1970 (figura 4.33). La avenida utilizada para fines de validación se presentó en julio – agosto 1972 (figura 4.34).

A continuación se mencionan las características de la RNA que se utilizará en el desarrollo de este caso de estudio:

```
Tipo de red: Unidireccional con retropropagación
Rango de Entradas: [0,1]
Función de entrenamiento: trainlm
Función de aprendizaje: learngdm
Función de desempeño: mse
Número de capas: 3 (1 de entrada, 1 oculta, 1 salida)
Número de neuronas capa oculta: n = 6
Función de transferencia: Logsig
```

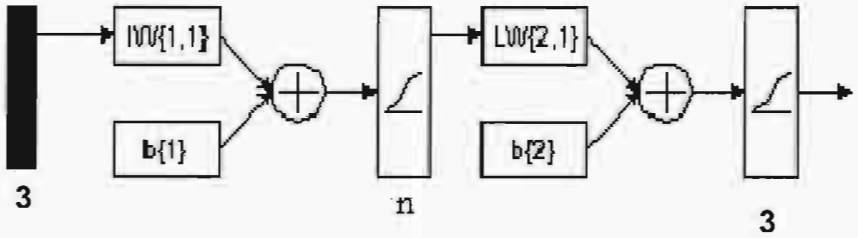


Figura 4.28 *RNA con estructura 3 – n – 3*

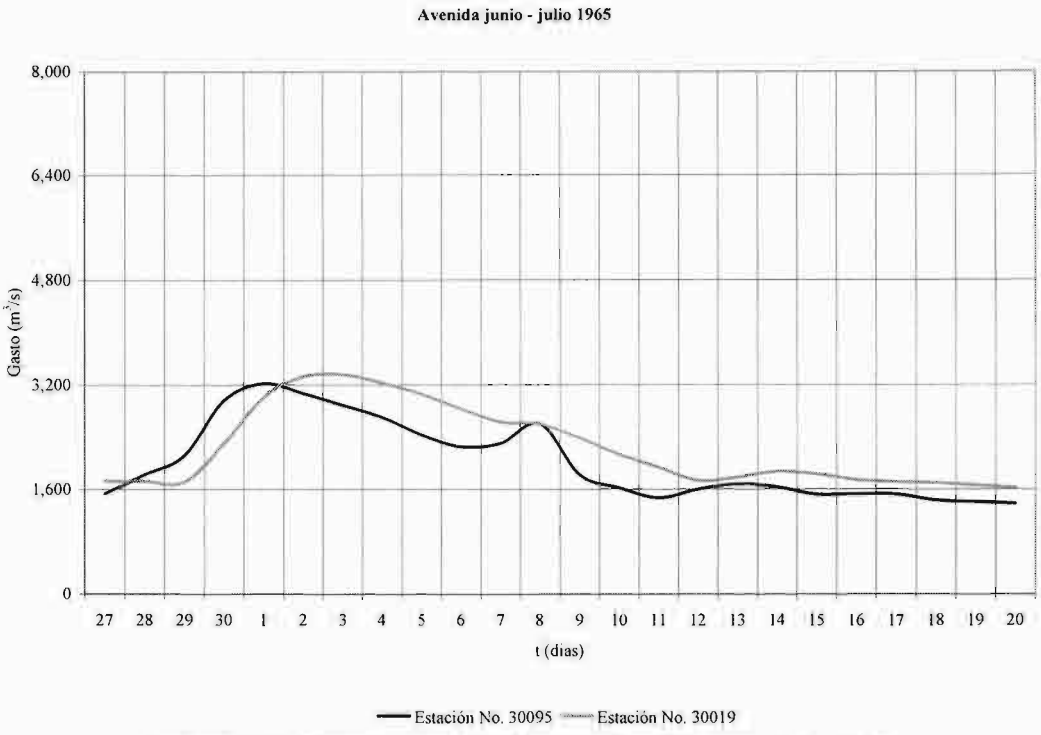


Figura 4.29 *Primera avenida para el entrenamiento de la RNA*

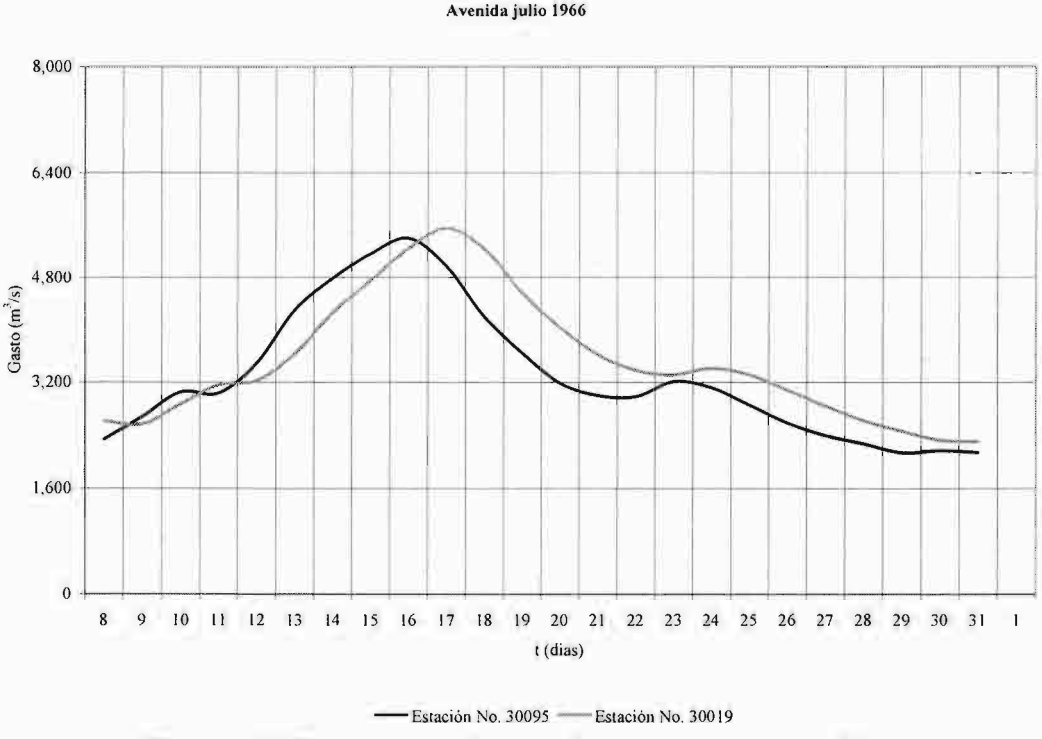


Figura 4.30 *Segunda avenida para el entrenamiento de la RNA*

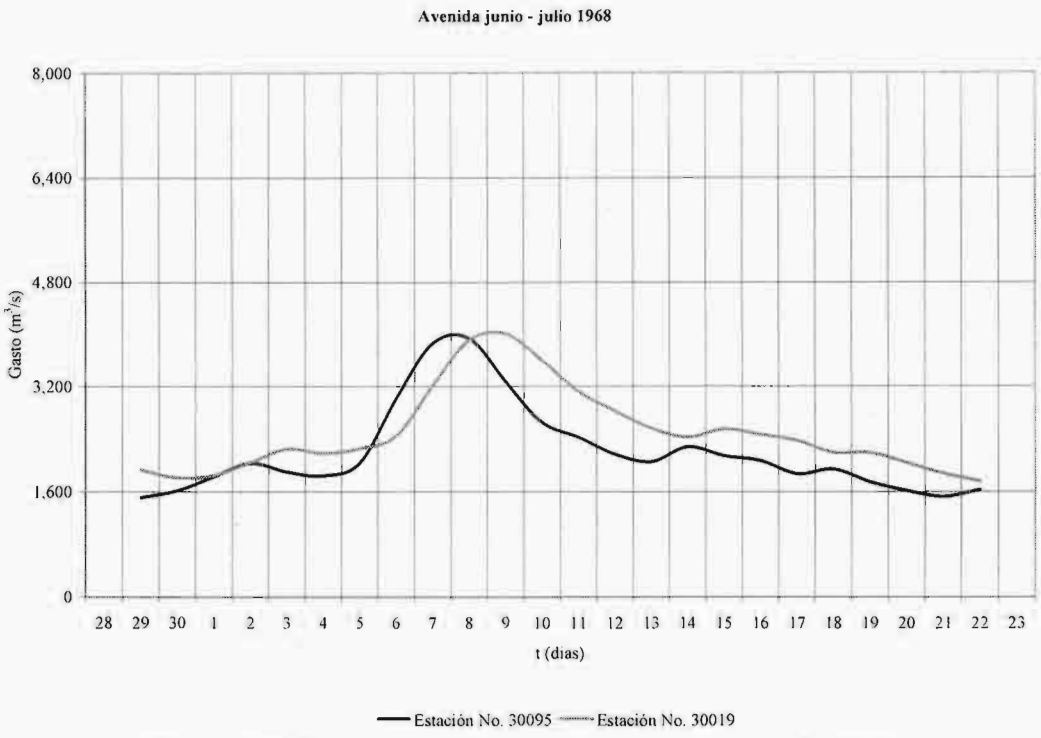


Figura 4.31 *Tercera avenida para el entrenamiento de la RNA*

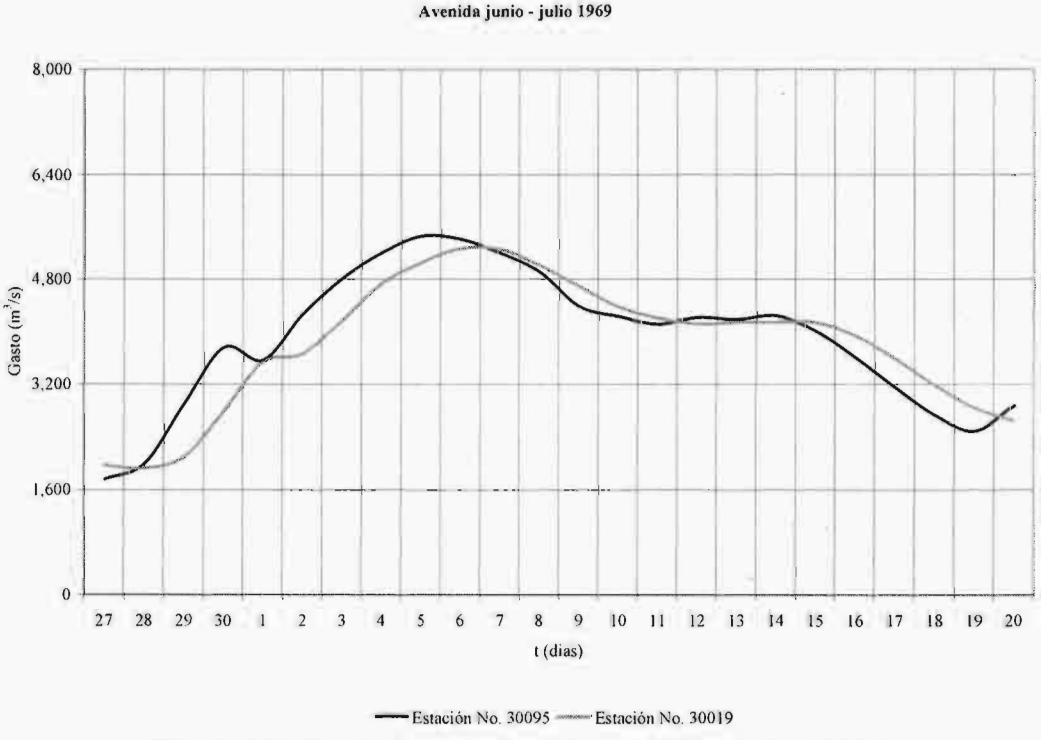


Figura 4.32 *Cuarta avenida para el entrenamiento de la RNA*

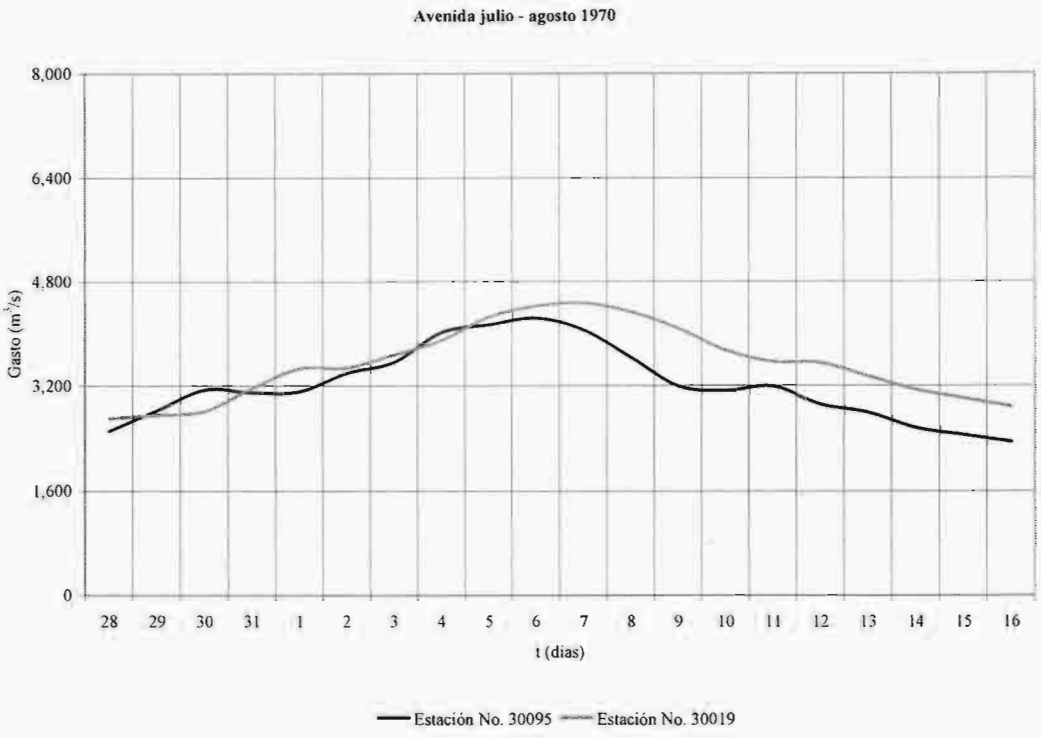


Figura 4.33 *Quinta avenida para el entrenamiento de la RNA*

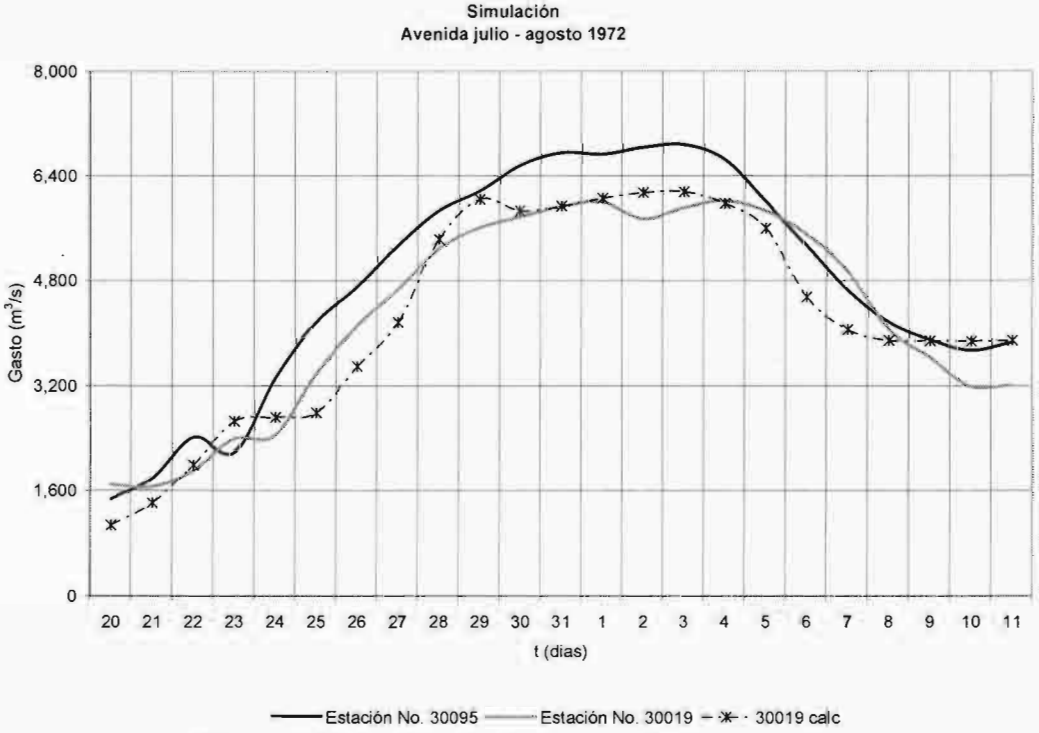


Figura 4.34 *Avenida simulada por la RNA entrenada*

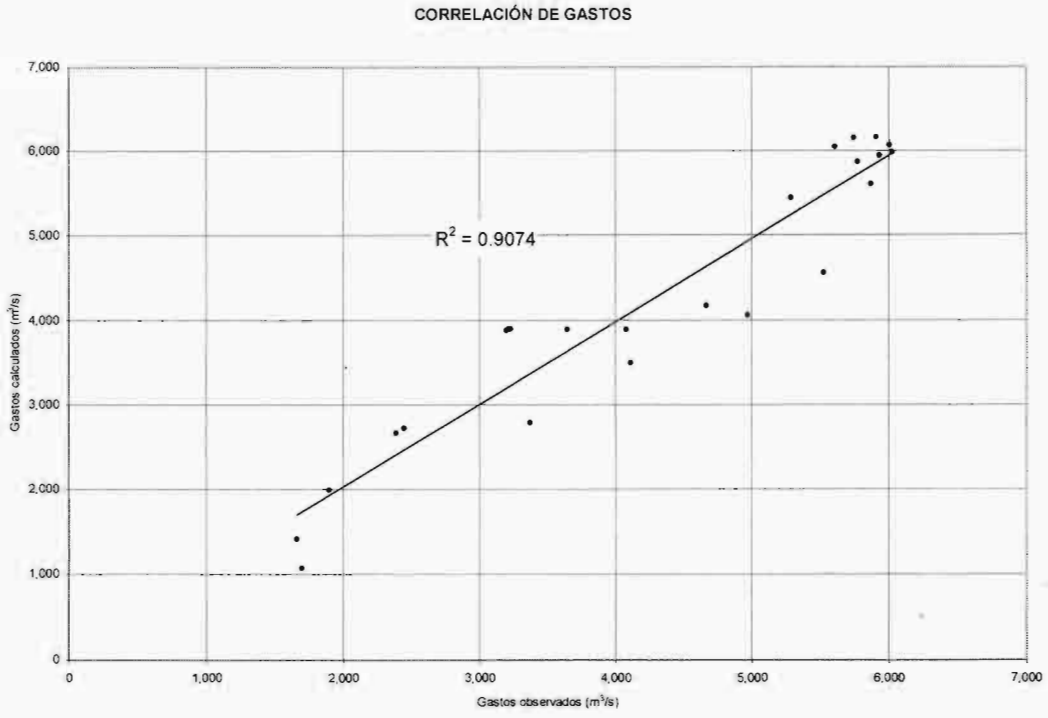


Figura 4.35 *Comparación de gastos calculados contra observados*

Para llevar a cabo el entrenamiento de la RNA propuesta se tomaron 5 avenidas en las que existió flujo lateral. Es importante comentar que el rango de gastos cubiertos en el entrenamiento de la red va desde los 1,300 hasta los 6,900 m^3/s y que las formas de las avenidas tomadas en cuenta son muy diferentes entre sí, lo que permite afirmar que es posible abarcar avenidas comprendidas en este rango de gastos que presenten diversas formas.

En la figura 4.35 se puede apreciar la correlación existente entre los gastos observados en la estación hidrométrica El Tigre (30019) y los calculados con la red neuronal entrenada. Se observa que existe buena correlación (R^2 = 0.9074) y buen comportamiento para la avenida simulada, la cual es notablemente diferente a las avenidas utilizadas en el entrenamiento.

4.5.3 Caso 3 – Tren de avenidas en la región hidrológica número 30 Grijalva – Usumacinta, en la parte baja del río Usumacinta.

Para el desarrollo del caso tercer caso se analizará la aplicación de una RNA a trenes de avenidas de las estaciones hidrométricas Boca del Cerro (30095) y El Tigre (30019) en la parte baja del río Usumacinta, Figura 4.27.

Después de revisar la información histórica se puede apreciar que los trenes de avenidas se presentan principalmente en los meses de junio, julio y agosto, por lo que se utilizarán 8 años para realizar el entrenamiento de la red en esos meses con arquitectura 3 – 6 – 3. Cada neurona de entrada y salida presentará 30 datos por cada año utilizado en el entrenamiento es decir, 240 datos por neurona.

Los hidrogramas de entrada y salida seleccionados para realizar el entrenamiento corresponden a los años 1964 a 1971 y posteriormente se simularán los años de 1972 a 1976.

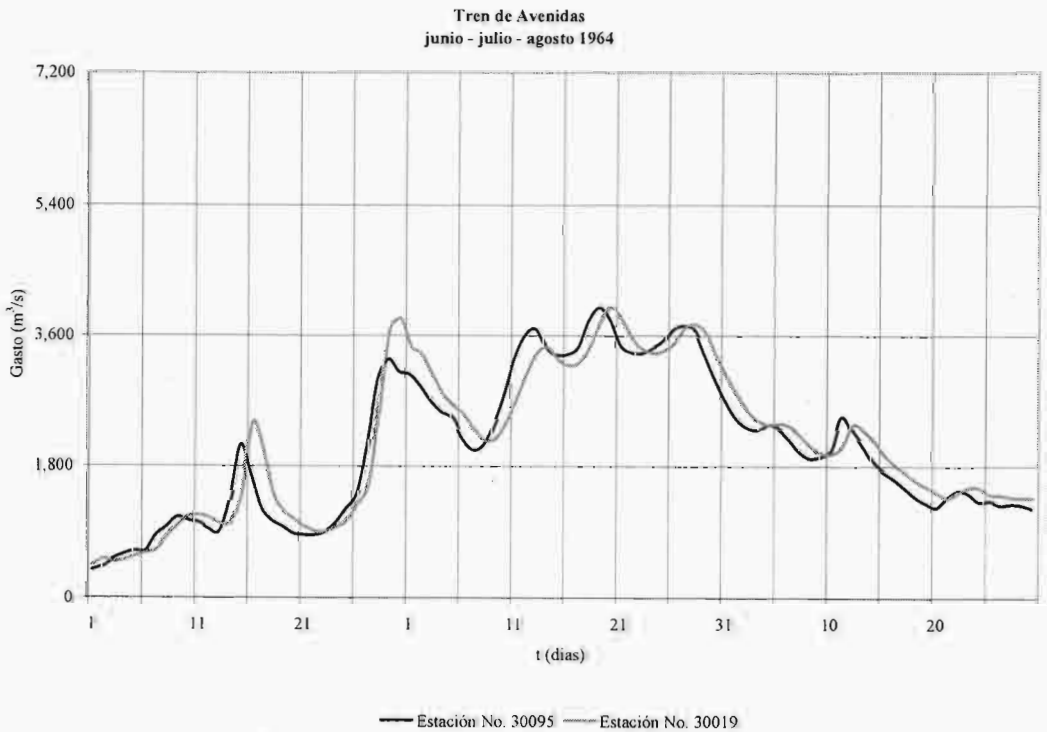


Figura 4.36 *Primer tren de avenidas para el entrenamiento de la RNA*

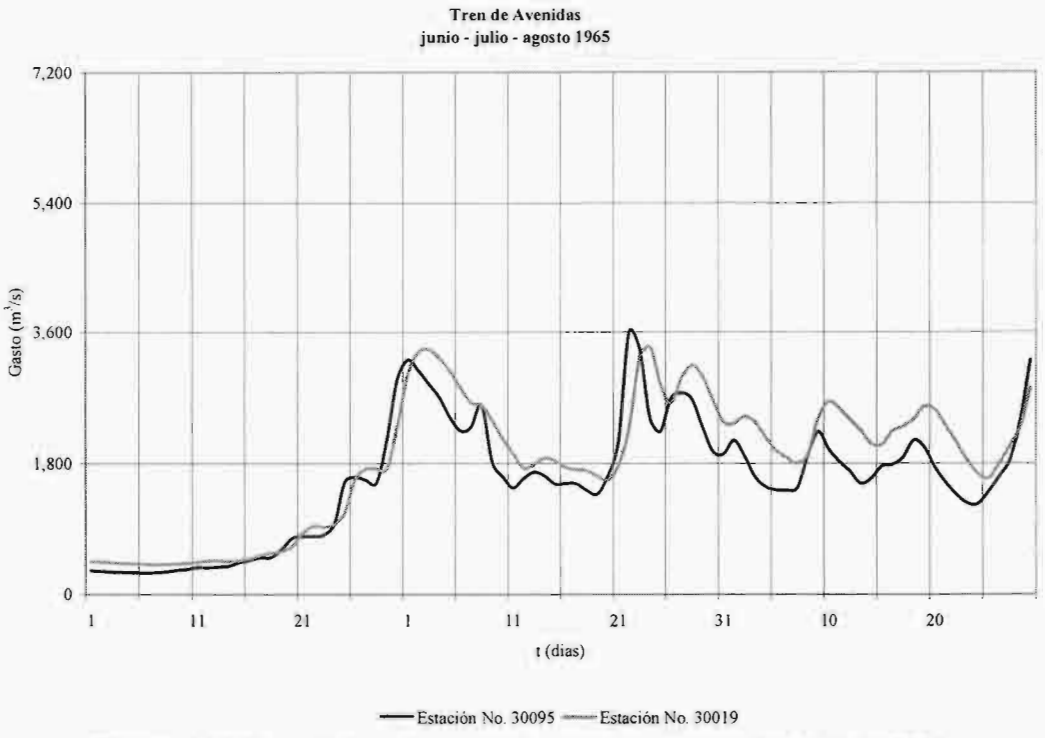


Figura 4.37 *Segundo tren de avenidas para el entrenamiento de la RNA*

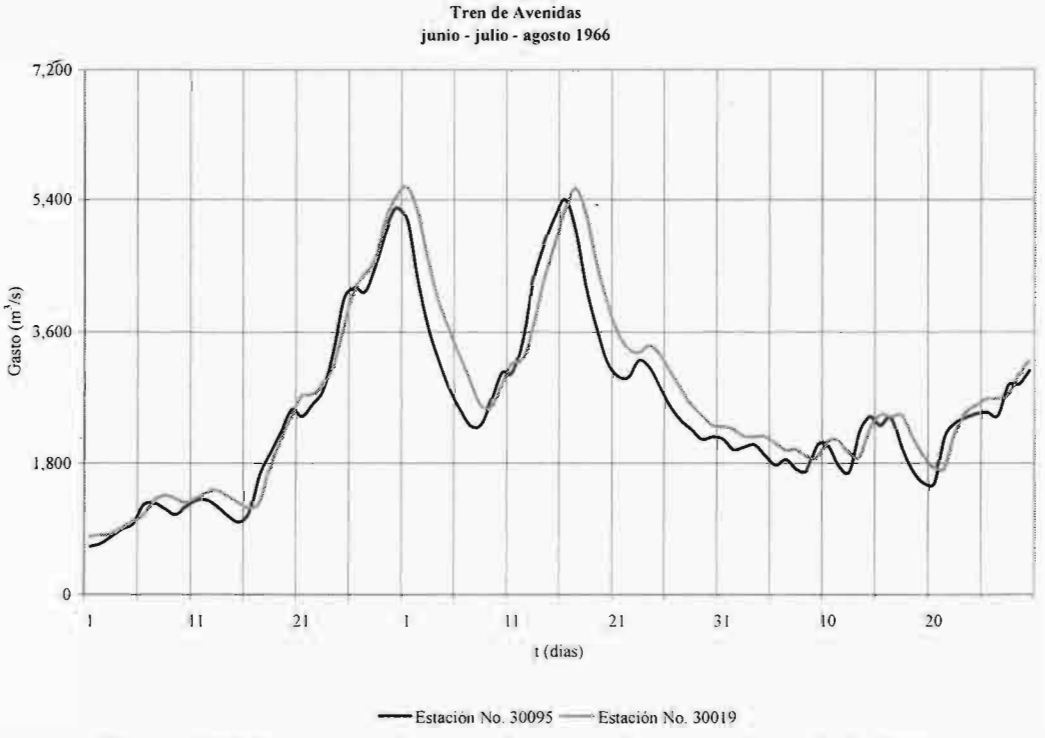


Figura 4.38 *Tercer tren de avenidas para el entrenamiento de la RNA*

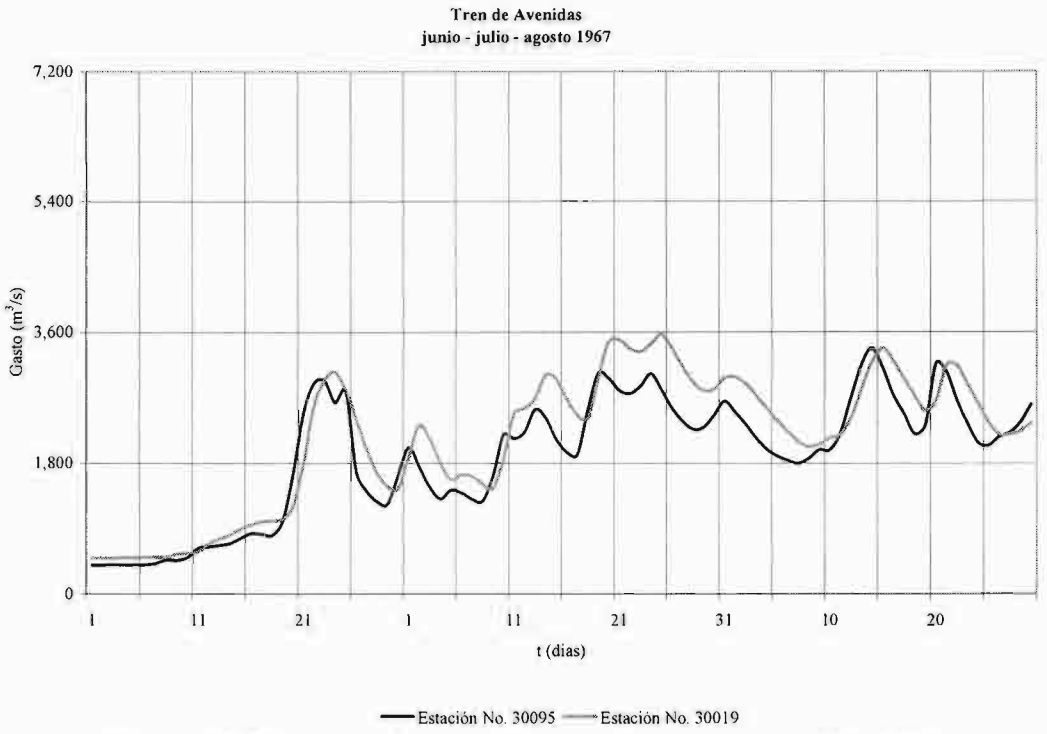


Figura 4.39 *Cuarto tren de avenidas para el entrenamiento de la RNA*

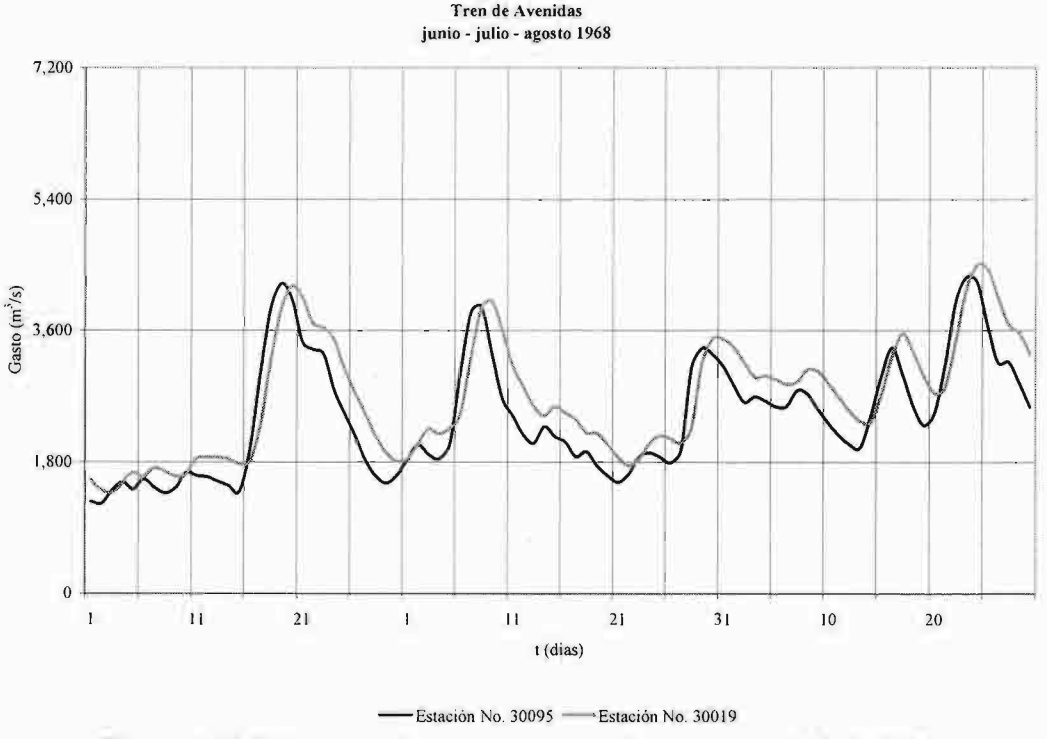


Figura 4.40 *Quinto tren de avenidas para el entrenamiento de la RNA*

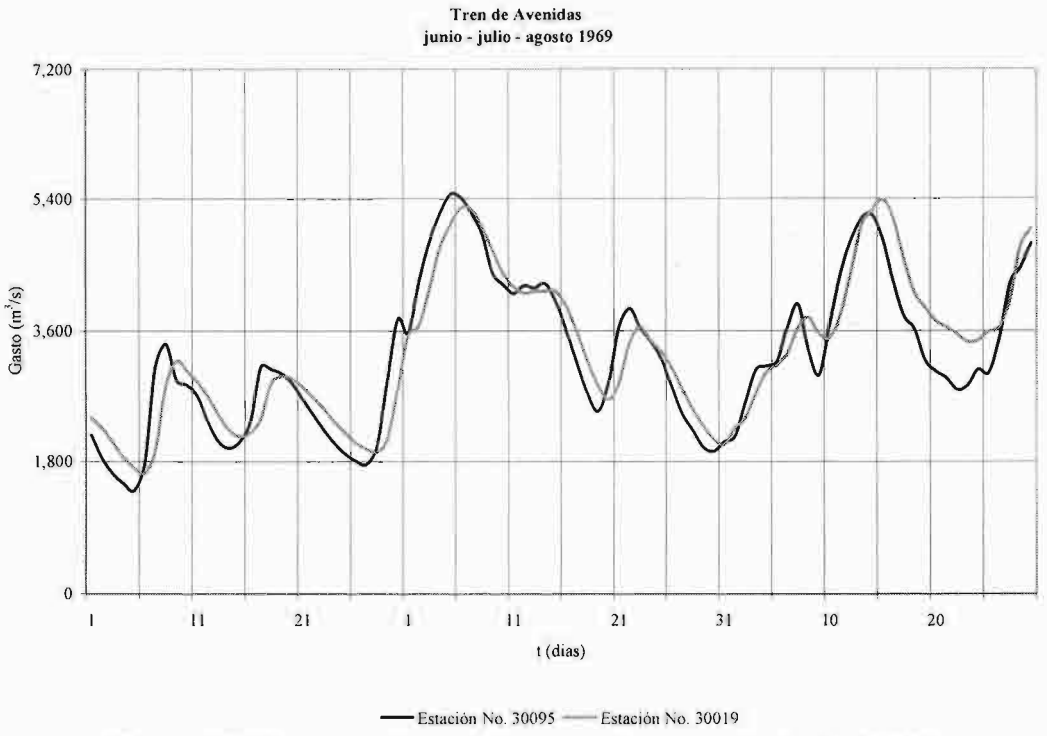


Figura 4.41 *Sexto tren de avenidas para el entrenamiento de la RNA*

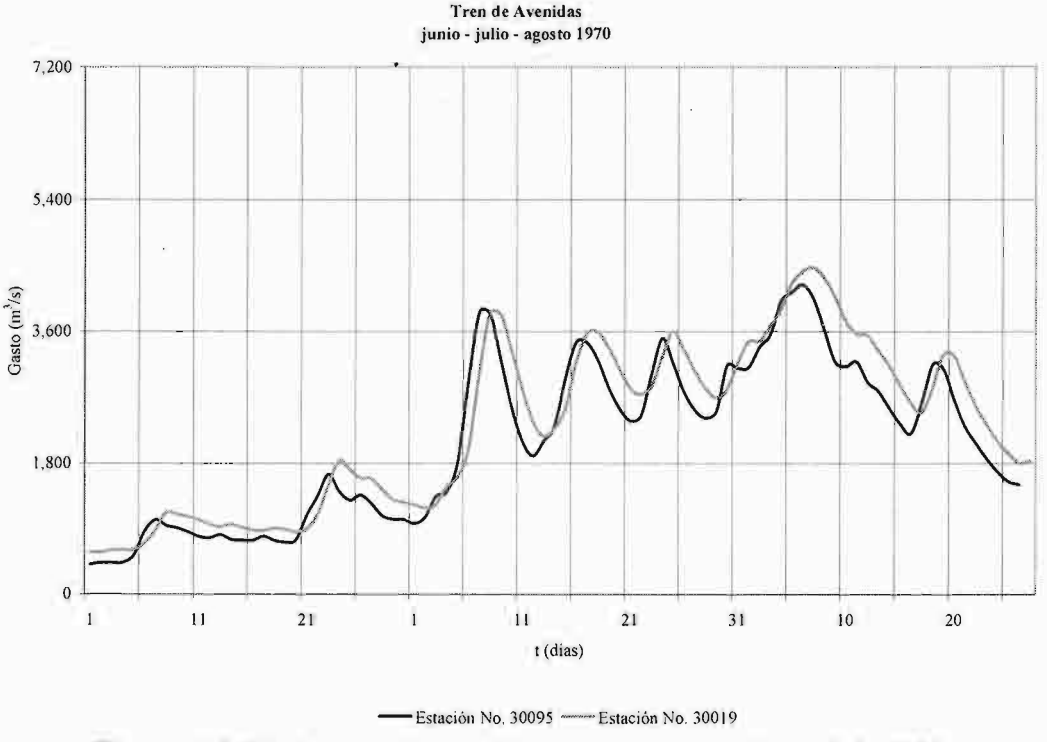


Figura 4.42 *Séptimo tren de avenidas para el entrenamiento de la RNA*

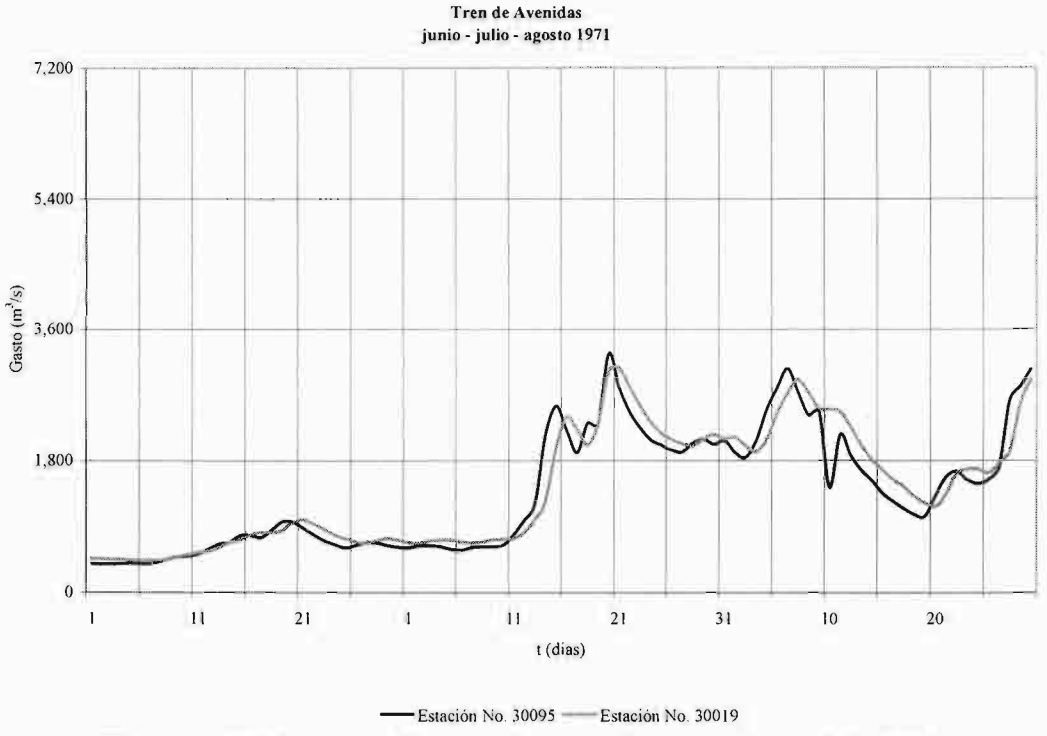


Figura 4.43 *Octavo tren de avenidas para el entrenamiento de la RNA*

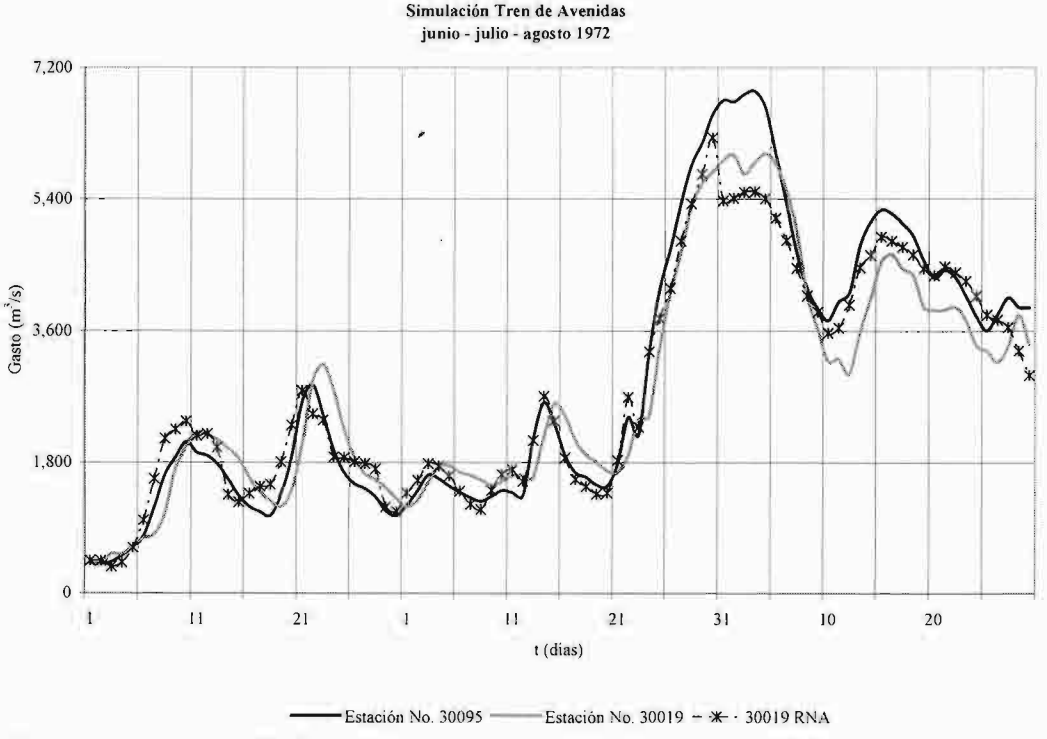


Figura 4.44 *Resultado de la simulación del año 1972 para la red neuronal entrenada*

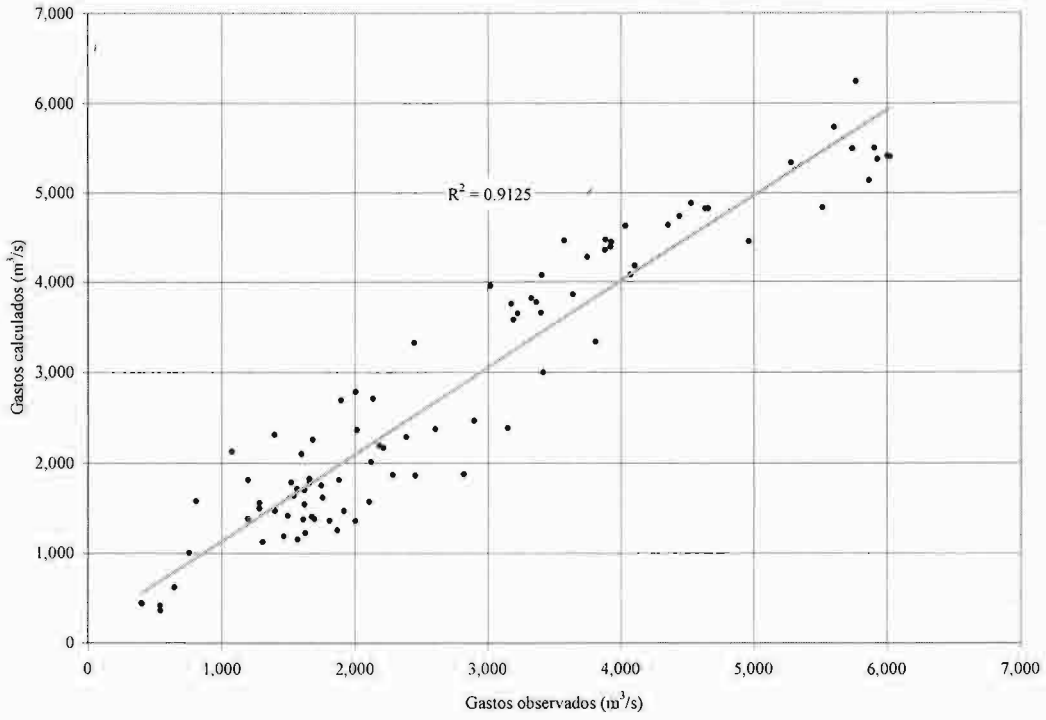


Figura 4.45 *Comparación de gastos calculados contra observados de la simulación de 1972*

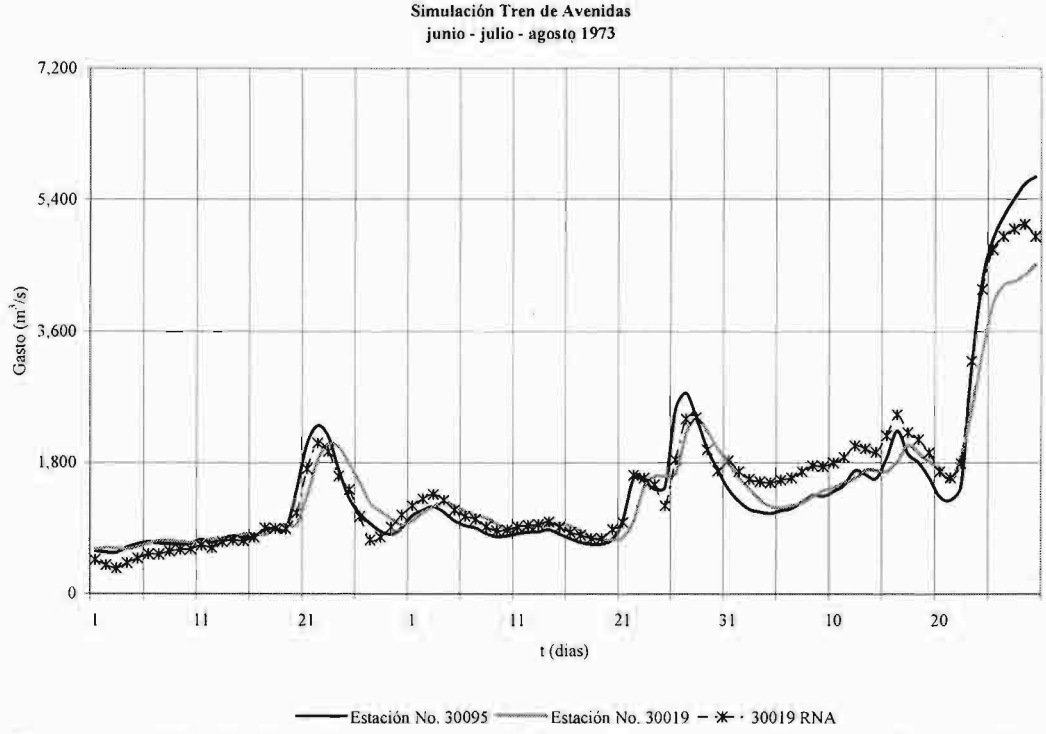


Figura 4.46 *Resultado de la simulación del año 1973 para la red neuronal entrenada*

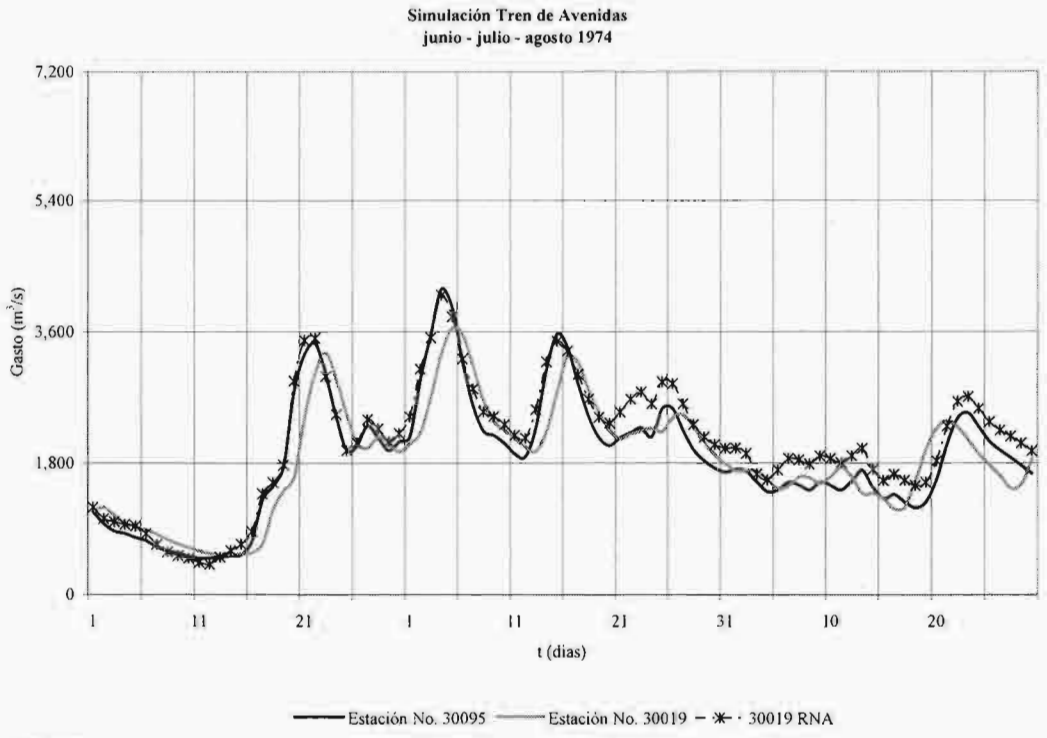


Figura 4.47 *Resultado de la simulación del año 1974 para la red neuronal entrenada*

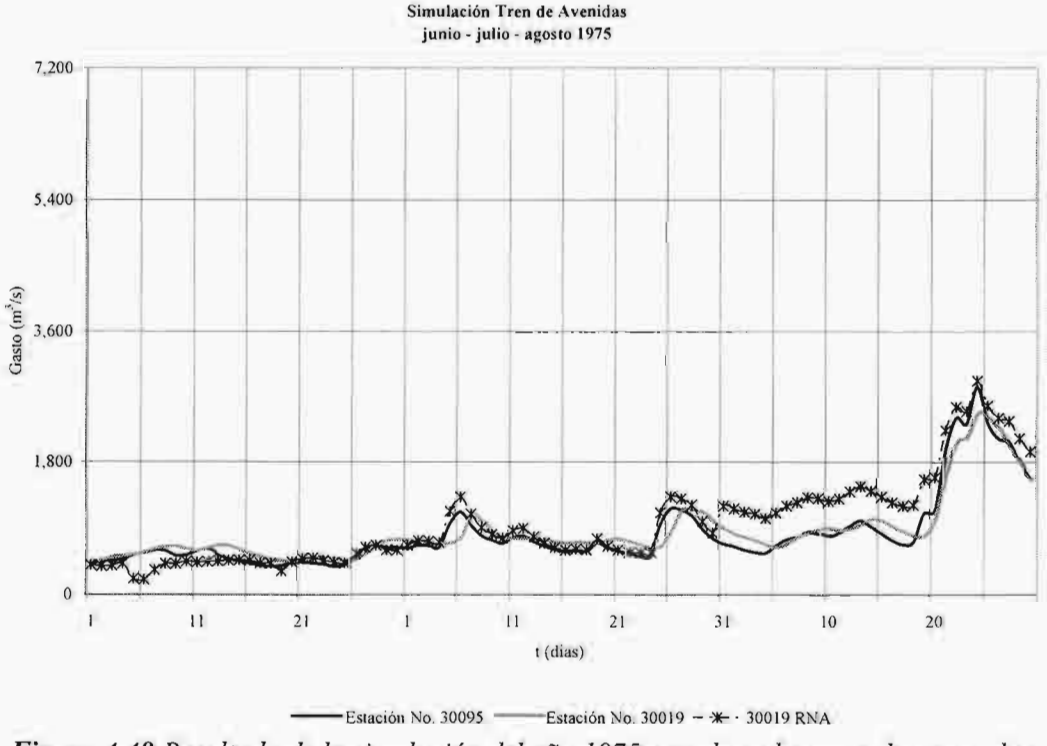


Figura 4.48 *Resultado de la simulación del año 1975 para la red neuronal entrenada*

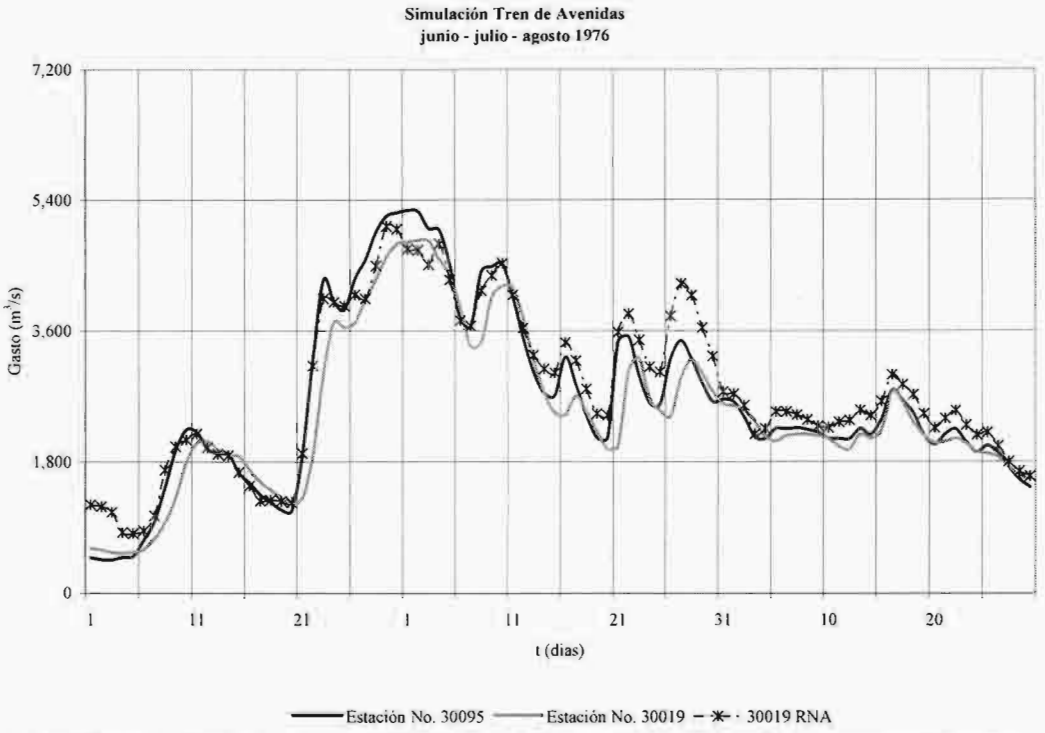
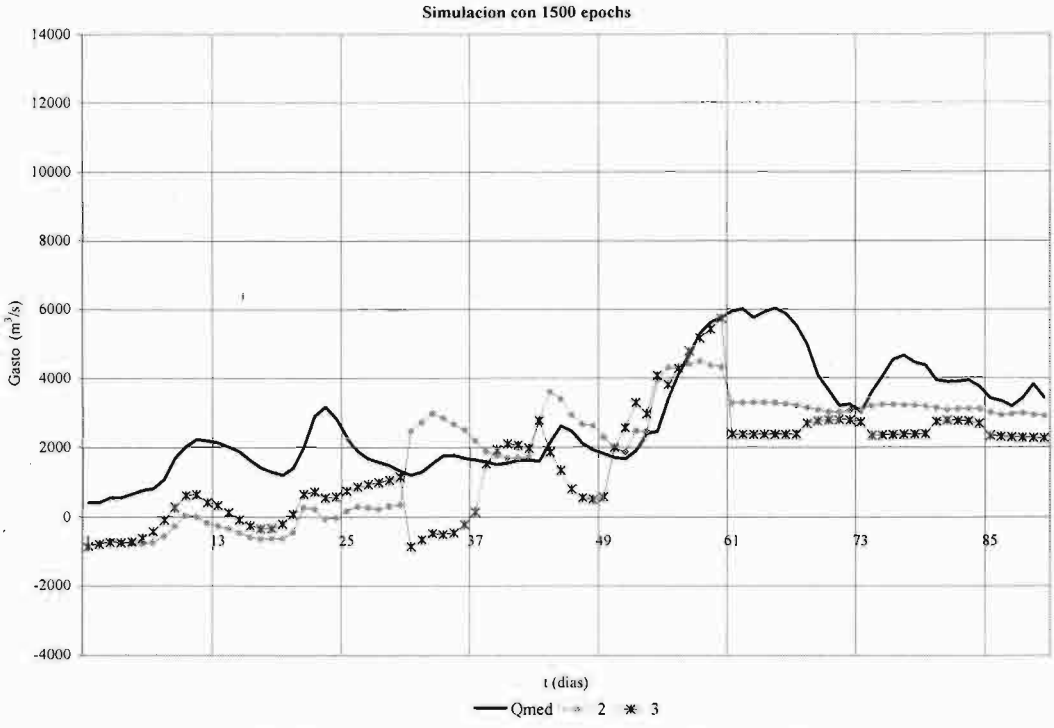


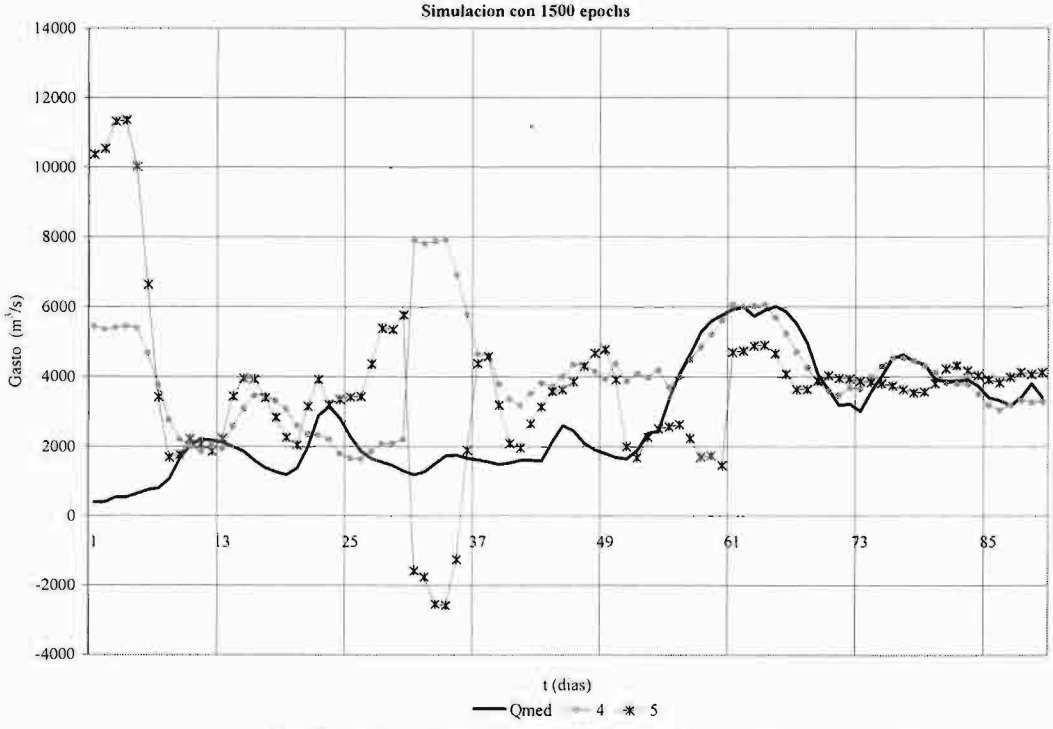
Figura 4.49 *Resultado de la simulación del año 1976 para la red neuronal entrenada*

En las figuras 4.50 (a, b, c) se puede observar la diferencia existente en la simulación por parte de la RNA 3 – 6 – 3 para el desarrollo del caso 2. Si se varía la cantidad de ejemplos del aprendizaje, se puede observar que a medida que el aprendizaje de la red se lleva a cabo con más información, mejor es su capacidad de generalización, y que el valor de mse aumenta según la cantidad de neuronas ocultas para llevar a cabo el aprendizaje, por lo que dicho valor es punto de comparación siempre y cuando se esté tomando en una red la misma cantidad de información. En la figura 4.50 (a) con 2 avenidas para el aprendizaje la red trabaja con 180 datos en la capa de entrada y 90 en la de salida, con 3 avenidas lo hará con 270 en la capa de entrada y 90 en la de salida, y así de forma sucesiva. Se aprecia claramente la ventaja de trabajar con un registro amplio de datos en la fase del entrenamiento.

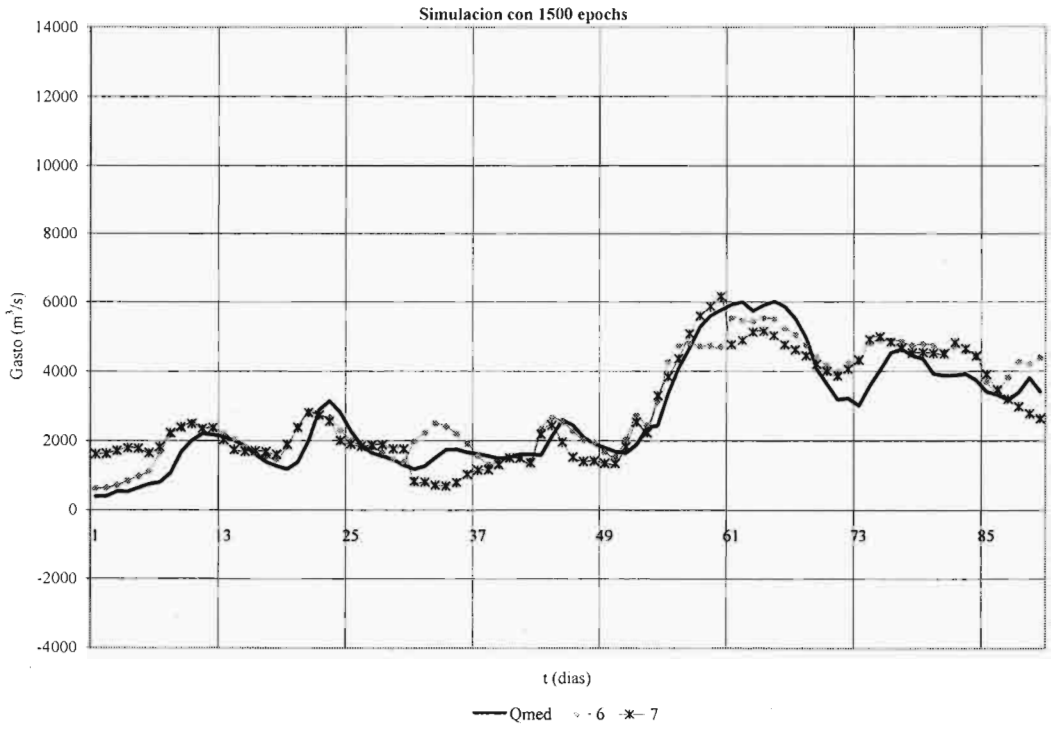
Durante el desarrollo comparativo del número de trenes de avenidas utilizado para llevar a cabo el entrenamiento de la RNA propuesta, se puede apreciar la importancia de tener para ello información suficiente, pues de acuerdo con las figuras 4.50 (a) dos y tres trenes avenidas utilizadas, 4.50 (b) cuatro y cinco avenidas utilizadas y 4.46 (c) seis y siete avenidas utilizadas se observa como el resultado de la simulación hecha para el tren de avenidas de entrada de junio - julio - agosto de 1972 registrada en la estación hidrométrica Boca de Cerro (30095) va ajustándose mejor al tren de avenidas de salida en el cauce registrado en ese mismo año en la estación hidrométrica El Tigre (30019) de acuerdo con el número de ejemplos utilizados para el aprendizaje.



a) *Dos y tres trenes de avenidas en el aprendizaje*

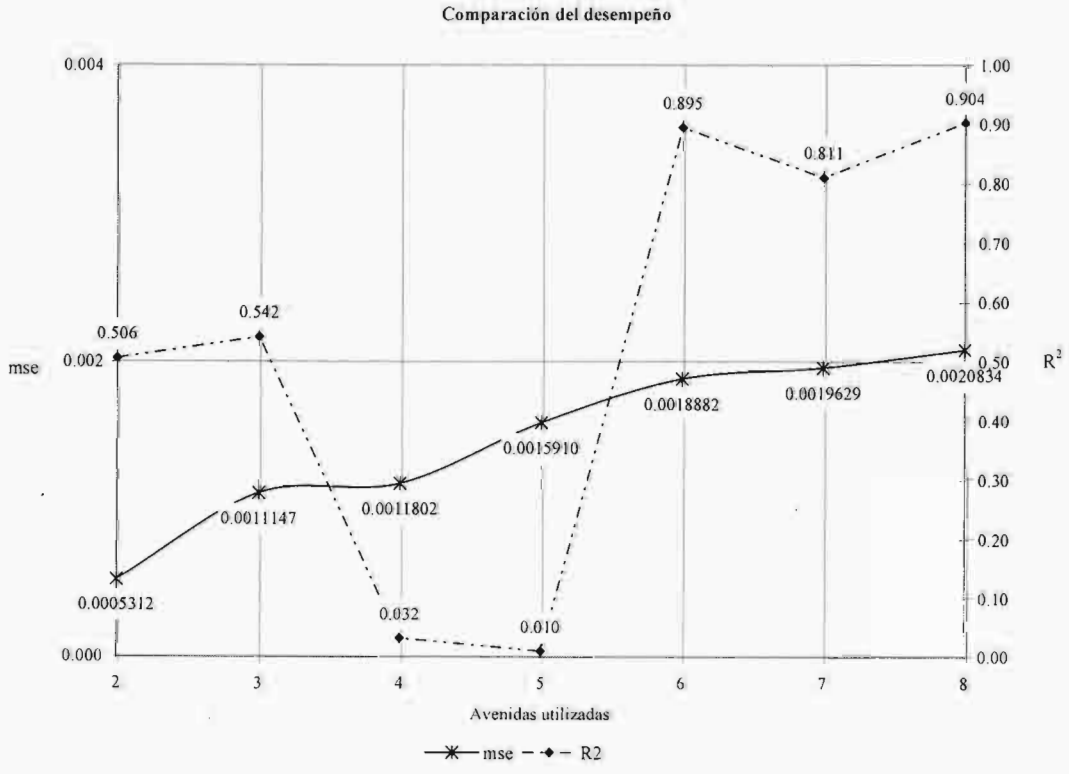


b) *Cuatro y cinco trenes de avenidas en el aprendizaje*



c) *Seis y siete trenes de avenidas en el aprendizaje*

Figuras 4.50 *a, b, c. Comparación del desempeño de la RNA utilizando diferente número de trenes de avenidas durante el aprendizaje*



Figuras 4.51 *Valor de mse y R^2 de la RNA con distinto numero de avenidas en el entrenamiento*

En la figura 4.51 se aprecian los valores tanto de mse como de R^2 obtenidos en el entrenamiento de la RNA con arquitectura 3 – 6 – 3 con distinto número de trenes de avenidas. Se puede apreciar que conforme aumentan los ejemplos de entrenamiento (aumento de información) el valor de mse aumentará de tal manera que no es un parámetro de comparación para determinar la cantidad de ejemplos necesarios para llevar a cabo el entrenamiento. Con respecto al valor de R^2 se observa que la RNA entrenada no presentará una tendencia de comportamiento conforme se aumente la cantidad de ejemplos para llevar a cabo el entrenamiento, razón por la cual es importante graficar los resultados para poder definir con que número de ejemplos es adecuado el entrenamiento para el caso de estudio

Finalmente la simulación mostrada en la figura 4.44 realizada para la RNA entrenada con 8 trenes de avenidas nos muestra ya un comportamiento adecuado con respecto al tren de avenidas de salida para el año de 1972 y como se grafica en la figura 4.45 su coeficiente de correlación $R^2 = 0.9125$ nos muestra la bondad en la exactitud de los valores calculados mediante la red.

Recordemos que lo importante es conseguir el mejor ajuste posible pero teniendo en cuenta que la arquitectura de la red propuesta deberá ser lo más sencilla posible, si el caso lo amerita y se debe tener un mejor ajuste entonces deberá modificarse el número de neuronas dentro de la capa oculta. Posteriormente se mostrará la importancia de utilizar un número adecuado de neuronas en la capa oculta para lograr un entrenamiento correcto.

En las figuras 4.46, 4.47, 4.48 y 4.49 se aprecia que la red entrenada con 8 trenes de avenidas genera hidrogramas de salida con buena aproximación en dichos años con respecto de los registrados en la estación aguas abajo (Boca del Cerro 30019), observándose la capacidad de generalización de las RNA.

4.5.4 Caso 4 – Hidrogramas de avenidas en confluencia en la región hidrológica número 28 Papaloapan, Río Tesechoacán

El cuarto caso evalúa la posibilidad de utilizar una RNA para simular avenidas registradas en la estación hidrométrica Azueta (28013) en el río Tesechoacán producto de la confluencia de las avenidas aforadas en los ríos Manso y Cajones en los cuales se encuentran ubicadas las estaciones hidrométricas Zapote (28075) y Monte Rosa (28017) respectivamente.

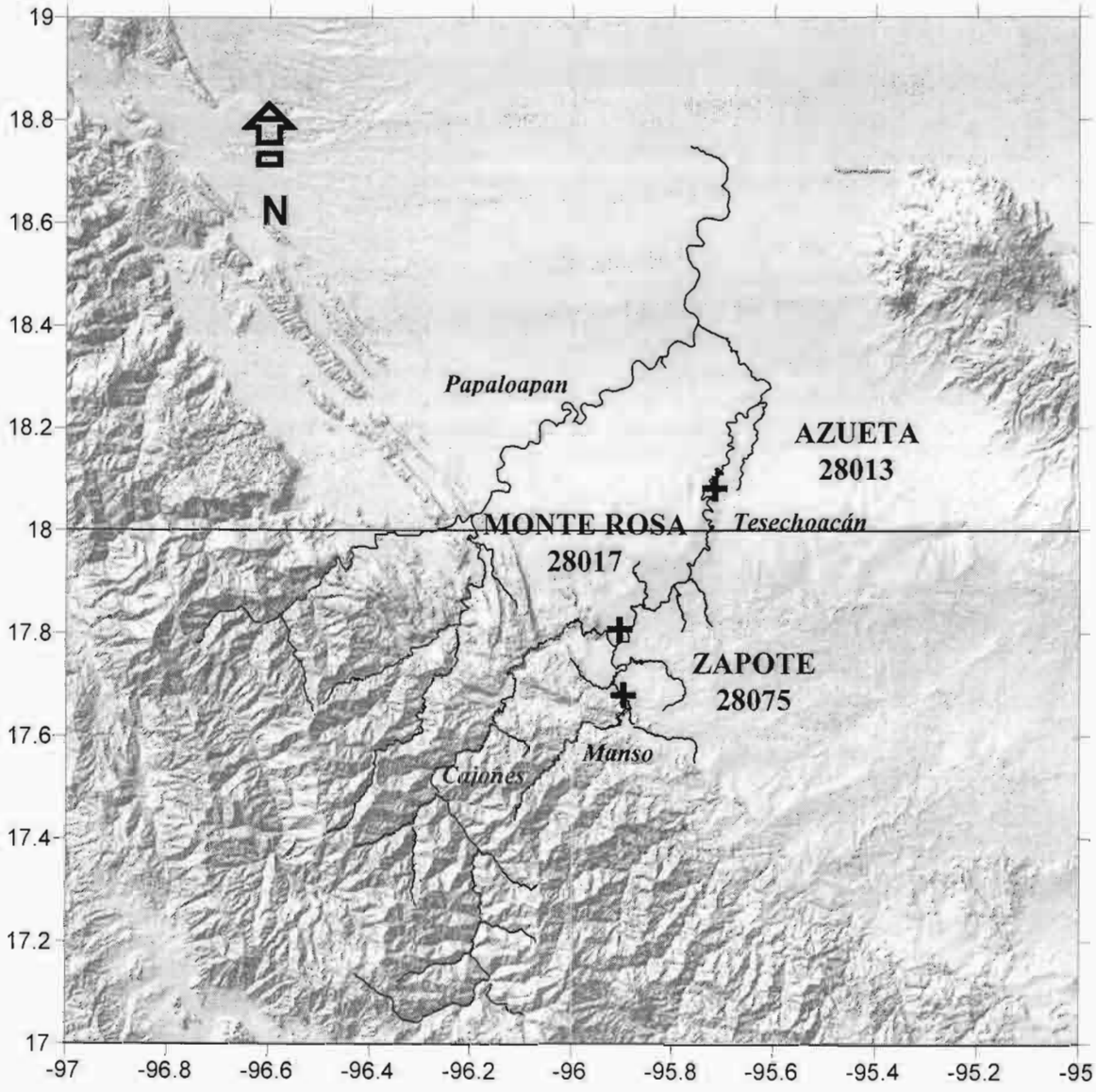


Figura 4.52 *Ubicación de la estaciones de estudio dentro de la Región Hidrológica 28*

El río Tesechoacán se forma por la unión de los ríos Cajones, afluente izquierda, y Manso, afluente derecho, unos 15 Km. aguas arriba de la población de Playa Vicente o Venustiano Carranza y desemboca en el río Papaloapan, 20 km aguas arriba de Tlacotalpan. La dirección general del cauce principal es SW – NE y la cuenca es de forma alargada, con su parte más ancha en el sur y terminando en el norte donde su eje longitudinal es al igual que el cauce principal de dirección SW – NE. La estación 28013 registra los gastos de una cuenca de 4,655.70 km^2 y se ubica en las inmediaciones de la población de Villa Azueta, Veracruz a 60 km de Ciudad Alemán, Veracruz. Parte del área drenada pertenece a la zona noreste del estado de Oaxaca.

La subcuenca del río Cajones ocupa la parte SW de la cuenca del río Tesechoacán y tiene una superficie aproximada de 2,298 km^2 hasta los llanos de Ozumacín donde el cauce principal desde su nacimiento, aguas arriba de Ayutla hasta su entrada a los llanos de Ozumacín, cae desde los 2800 msnm hasta los 100 msnm, aproximadamente, tiene una longitud de 132 km y una pendiente topográfica de alrededor del 2.49%. Hasta el sitio de la estación hidrométrica 28017 la cuenca drenada es de 2850 km^2. Pertenecen a esta subcuenca y son formadores del río Cajones, los ríos de Ayutla, Yacoche, Santo Domingo, Roayaga, Yatzona, Rincón o Grande y algunos más de menor importancia.

La subcuenca del río Manso ocupa la porción oriental media de la cuenca del río Tesechoacán cae desde los 1900 hasta los 85 msnm, tiene una longitud de 73 km. y una pendiente media aproximada del 2.49%, se forma con las aportaciones de su curso superior que nace en la zona alta del distrito de Choapan, del río Chiquito y de los arroyos Lodo, Carrizo y Tomates, Tiene un área de 805 km^2 hasta la estación hidrométrica 28075 situada en San José de Río Manso.

Para el desarrollo de la red neuronal se utilizará la arquitectura mostrada a continuación:

```
Tipo de red: Unidireccional con retropropagación
Rango de Entradas: [0,1]
Función de entrenamiento: trainlm
Función de aprendizaje: learngdm
Función de desempeño: mse
Número de capas: 3 (1 de entrada, 1 oculta, 1 salida)
Número de neuronas capa oculta: n = 6
Número de neuronas capa oculta: n = 12
Número de neuronas capa oculta: n = 3
Función de transferencia: Logsig
```

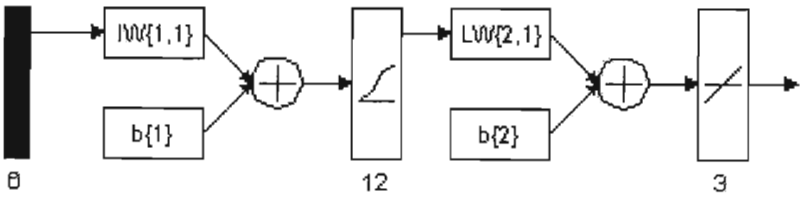


Figura 4.53 *Esquema de la RNA para el caso 4*

Dado que este caso presenta una mayor dificultad que los estudiados anteriormente debido a que se manejará más información pues se involucran ahora tres estaciones correspondiendo dos a los cauces aguas arriba de la unión (Monte Rosa y Zapote) y la restante al cauce aguas abajo de la unión (Azueta) y se trata de simular registros anuales de escurrimientos producto de las avenidas presentadas en la cuenca del río Tesechoacán.

Por tanto se propone utilizar una RNA con estructura 6 – 12 – 3, asignando tres neuronas por cauce que presente la información de manera cuatrimestral para el número de años tomados para llevar a cabo el entrenamiento y la posterior simulación; así pues, cada neurona manejará 122 datos por m años seleccionados. Las neuronas de la capa oculta se han propuesto observando que de los casos previos las redes neuronales presentan un buen funcionamiento a partir de que las neuronas de la capa oculta son el doble de las de la capa de salida.

Para realizar el estudio del funcionamiento de las RNA en la unión de dos ríos se tomarán los registros históricos correspondientes a los años 1975, 1976 y 1977, en al caso de las neuronas de entrada se destinarán los dos primeros al entrenamiento de la red propuesta y el último para la simulación mientras que en las neuronas de salida los dos primeros años serán los valores objetivos de la red y el último se utilizará simplemente para llevar a cabo la comparación con lo valores simulados en el año de 1977. En las figuras 4.54, 4.55 y 4.56 se muestran los registros históricos seleccionados para relizar el entrenamiento y la simulación de las estaciones Monte Rosa, Zapote y Azueta respectivamente.

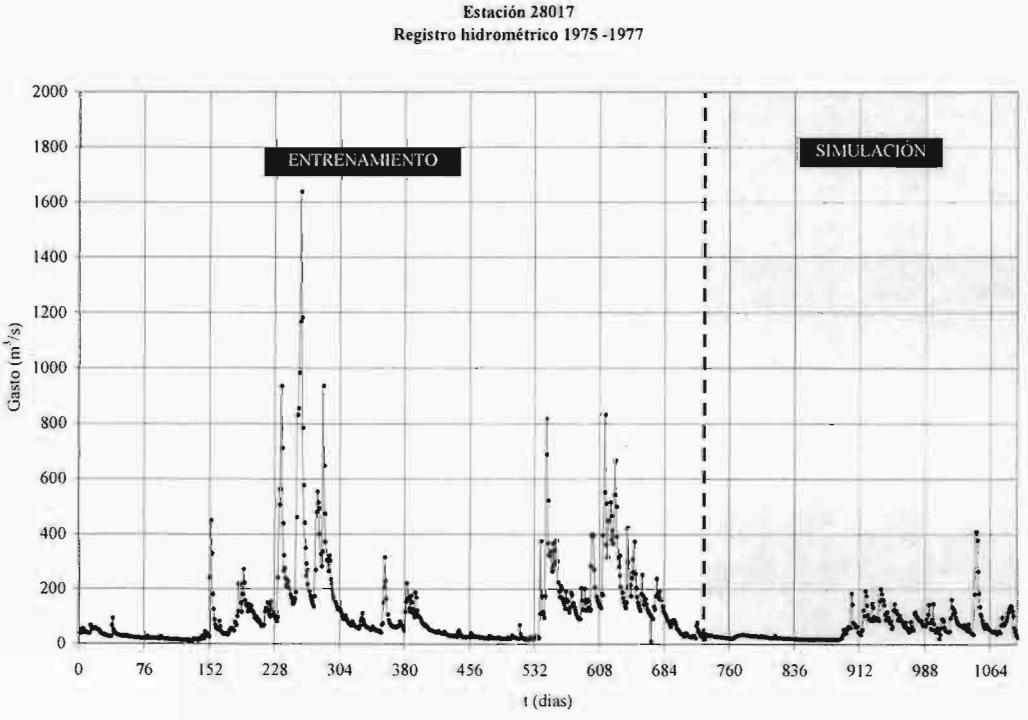


Figura 4.54 *Registro hidrométrico 1975 – 1977 de la estación 28017 "Monte Rosa".*

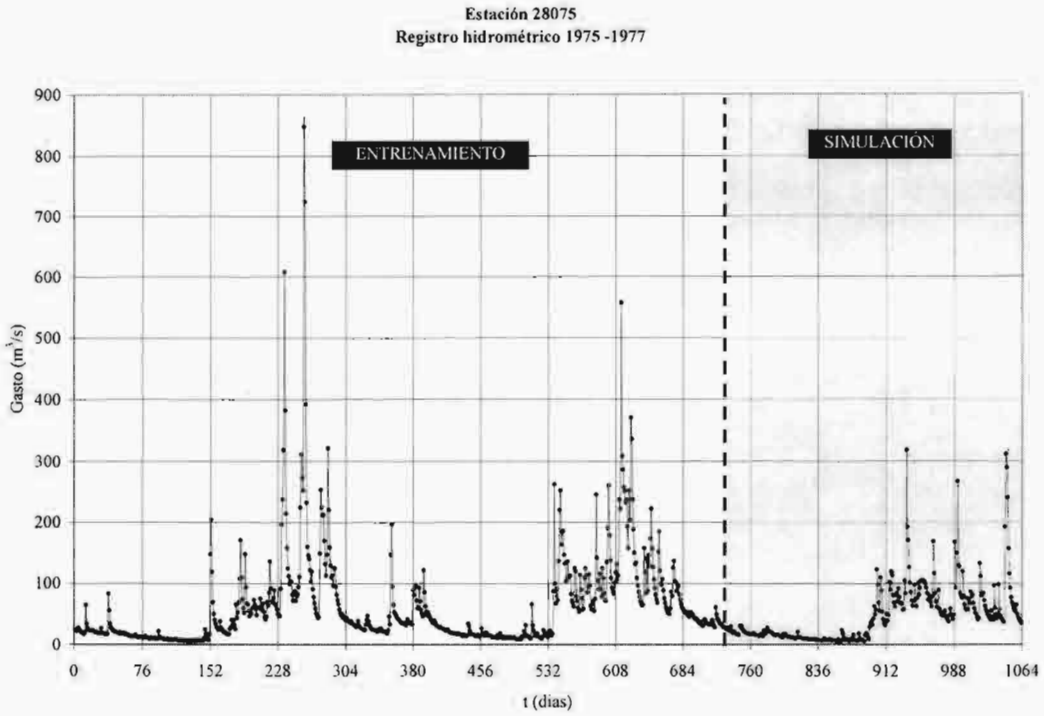


Figura 4.55 *Registro hidrométrico 1975 - 1977de la estación 28075 "Zapote".*

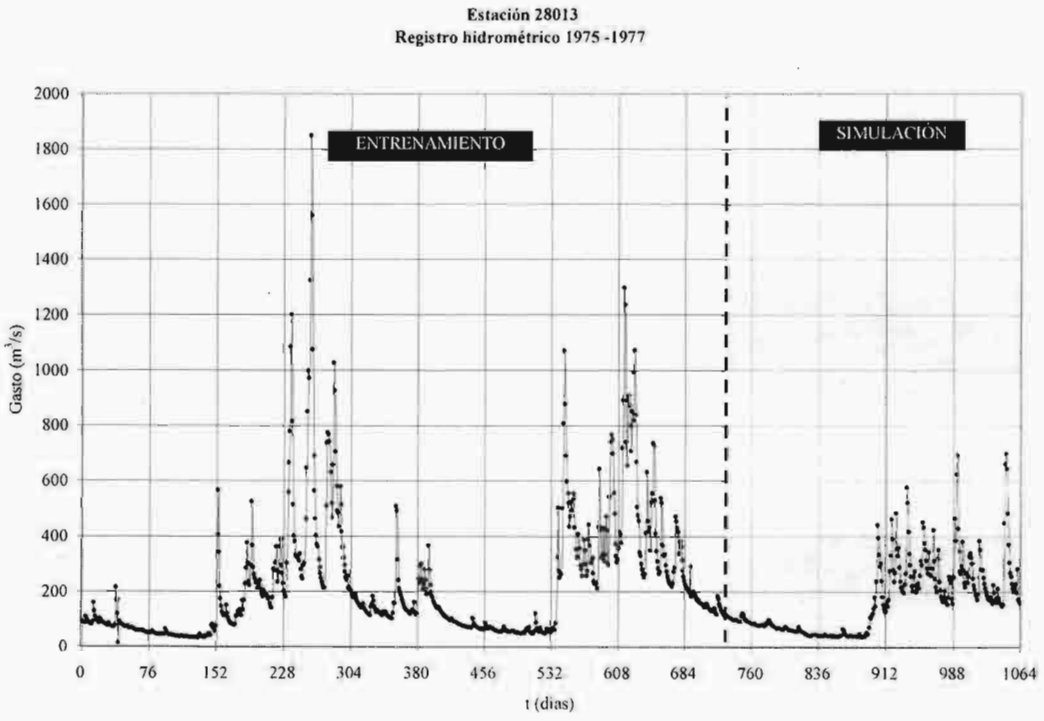


Figura 4.56 *Registro hidrométrico 1975 -1977de la estación 28013 "Azueta".*

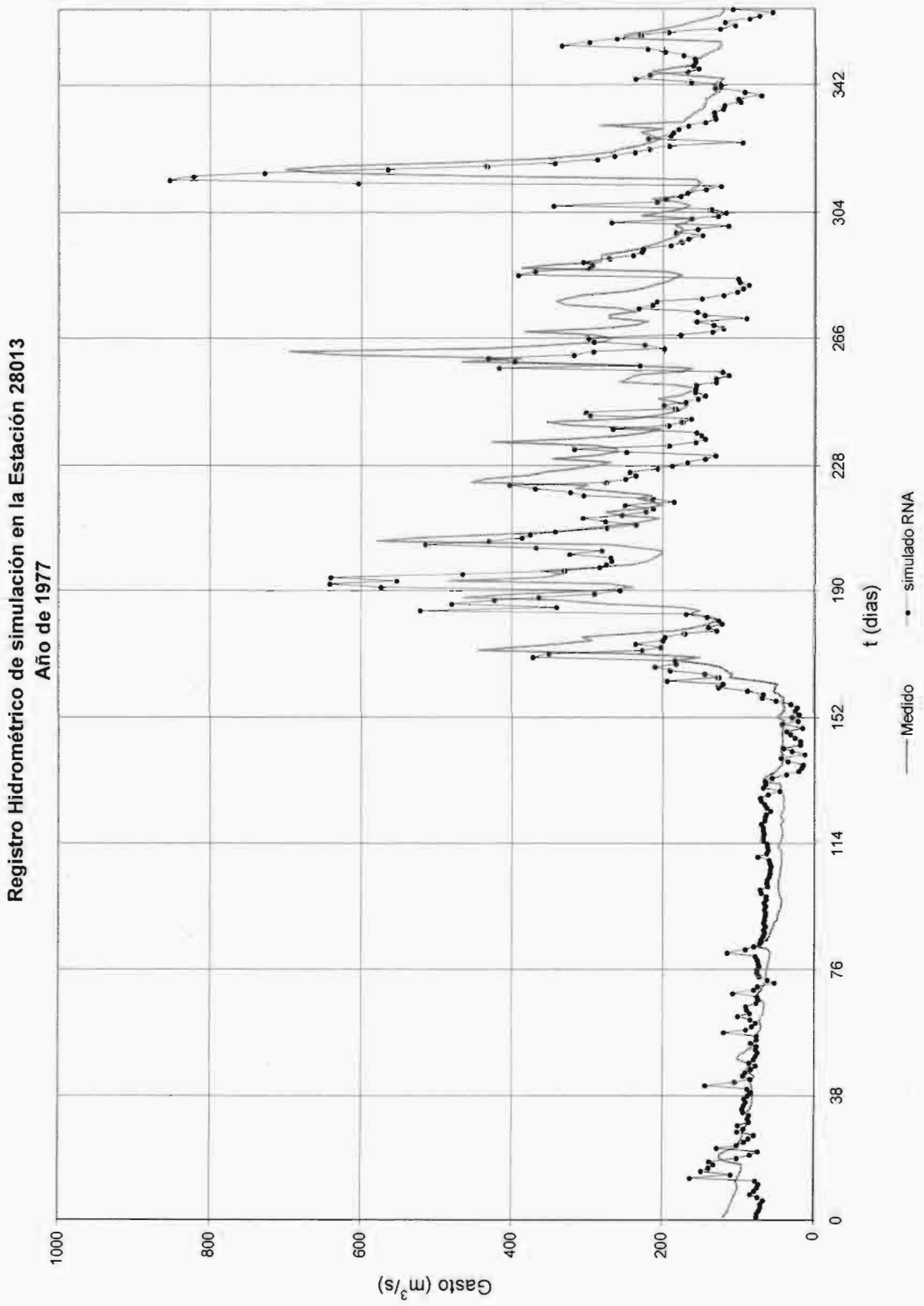


Figura 4.57 *Resultado de la simulación del registro hidrométrico de 1977 de la estación 28013 "Azueta" mediante la RNA entrenada.*

En la figura 4.57 se aprecia el resultado de la simulación hecha para el año de 1977 una vez que se llevó a cabo el entrenamiento de la red. Es importante señalar que dada la dificultad que representa el tránsito de avenidas en uniones de cauces, con la presencia de flujos laterales y para un amplio registro de datos, el comportamiento presentado es bueno en general y en algunas zonas pueden existir ciertas diferencias con respecto al valor observado.

Adicionalmente se llevó a cabo el entrenamiento de una RNA de mayor tamaño con arquitectura 12 – 20 – 6, para observar si una RNA de mayor dimensión genera una mejor generalización. Se observa en la figura 4.58 que efectivamente el manejo de una red lo más sencilla posible nos genera resultados adecuados y que no necesariamente una red de mayores dimensiones en su arquitectura garantiza un aprendizaje y posterior generalización correctas.

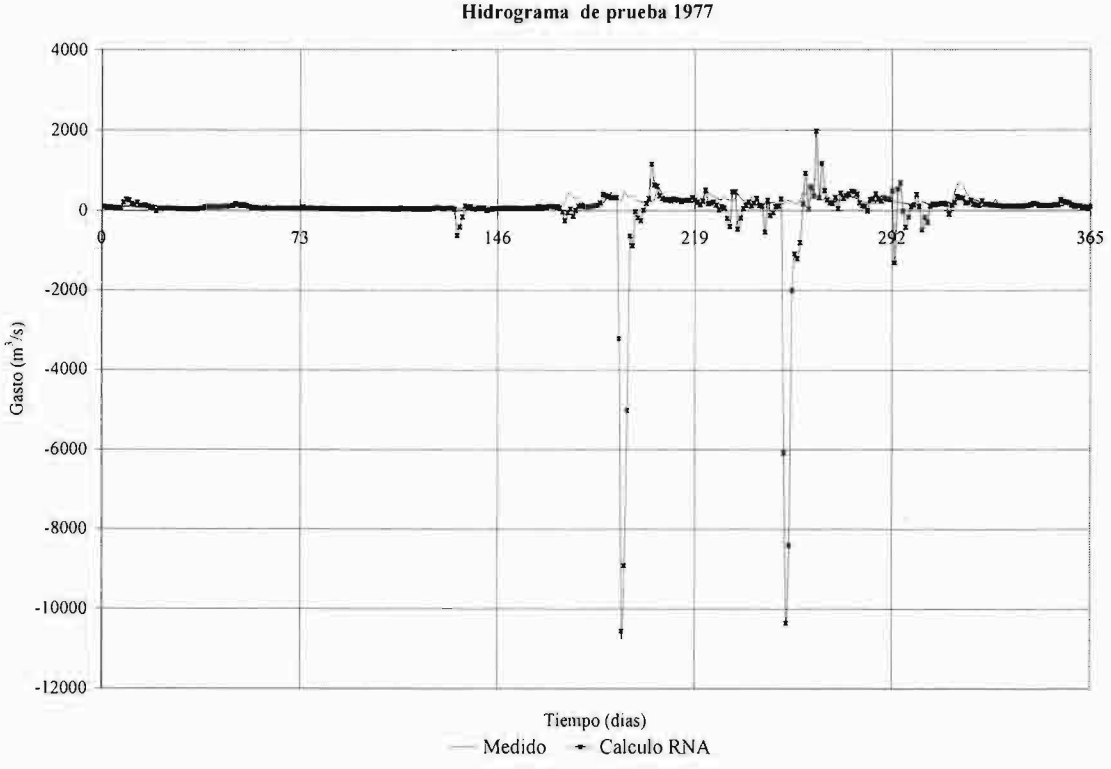


Figura 4.58 *Resultado de la simulación del registro hidrométrico de 1977 de la estación 28013 "Azueta" mediante una RNA12 – 20 – 6 entrenada.*

En la Tabla 4.5 y la figura 4.60 se muestran los valores de los coeficientes de correlación y R^2 que se dan para las distintas propuestas de arquitectura con fines ilustrativos, todas ellas con los mismos parámetros a excepción del número de neuronas en la capa oculta. Se observa para los casos 6 – 6 – 3 y 6 – 12 – 3 que los valores de estos coeficientes son prácticamente los mismos, sin embargo, en la figura 4.61 se graficaron los valores calculados en la simulación de ambas arquitecturas y se observa la diferencia del entrenamiento de cada una de ellas, presentando la RNA 6 – 12 – 3 un mejor desempeño no sólo en el valor de dichos coeficientes sino en representar más acordemente el hidrograma de salida registrado en la estación 28013 en 1977

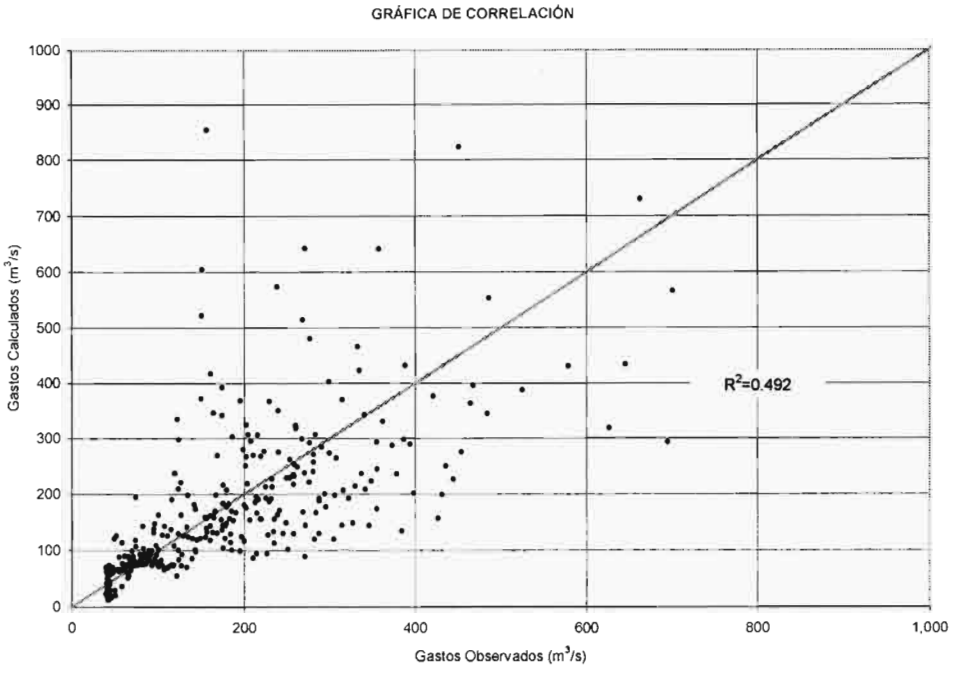


Figura 4.59 *Comparación de gastos calculados contra observados en la simulación de 1977en Azueta para la RNA 6 – 12 – 3 entrenada.*

Coeficiente	Arquitectura					
	6 - 6 - 3	6 - 8 – 3	6 - 10 - 3	6 - 12 - 3	6 - 14 - 3	6 - 16 – 3
Correlación	0.699	0.483	0.451	0.702	0.431	0.684
R^2	0.489	0.233	0.203	0.492	0.186	0.467

Tabla 4.5 *Coeficientes de correlación y R^2 para distintas arquitecturas de RNA*

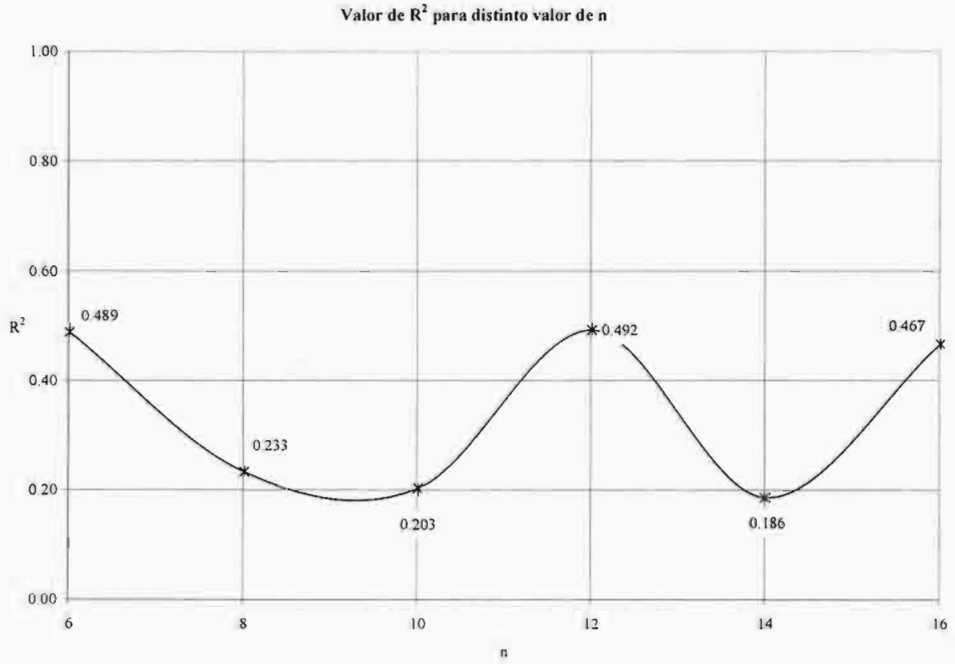


Figura 4.60 *Valores de R^2 para distinto número de neuronas en la capa oculta*

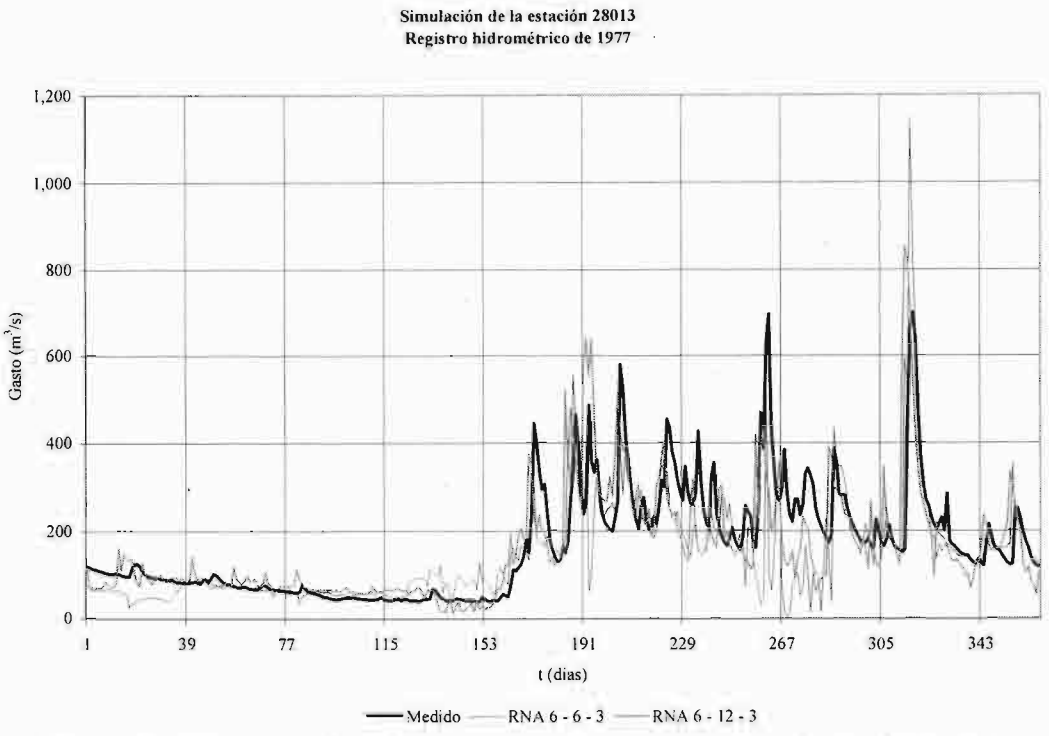


Figura 4.61 *Comparación del comportamiento y desempeño en la simulación del registro histórico de 1977 para las RNA 6 – 6 – 3 y 6 – 12 – 3*

En la figura 4.62 se han graficado 6 series de valores simulados con una RNA de arquitectura 6 – 6 < n < 16 - 3contra los valores registrados en la estación hidrométrica Azueta. Se aprecian en los valores calculados que para gastos de hasta 300 m^3/s las redes tienen un desempeño aceptable pero a medida que el valor del gasto aumenta se marca una clara diferencia entre las distintas arquitecturas (número de neuronas en la capa oculta).

Se aprecia para n = 6 que existe una serie de gastos en el rango de 200 a 400 m^3/s calculados que prácticamente son cero por lo que la red no tuvo suficiente grado de libertad para aprender dado que es una arquitectura insuficiente (tamaño de la capa oculta) para aprender los ejemplos presentados en el entrenamiento.

En el caso de n = 8 aparecen gastos negativos lo que físicamente no es posible y se debe a que aún la arquitectura de la red no es capaz de aprender los ejemplos de entrenamiento de igual forma valores calculados en un rango de 200 a 400 m^3/s permanecen en su mayoría por debajo de la línea de correlación.

Para el resto de las arquitecturas se puede aprecia que para gastos menores a 200 m^3/s la correlación es en general buena, en el rango de 200 a 400 m^3/s se comienza a marcar una diferencia clara del desempeño de los diferentes tamaños de la capa oculta y finalmente para valores mayores a 400 m3/s el mejor desempeño en la simulación es hecho por la arquitectura de n = 12.

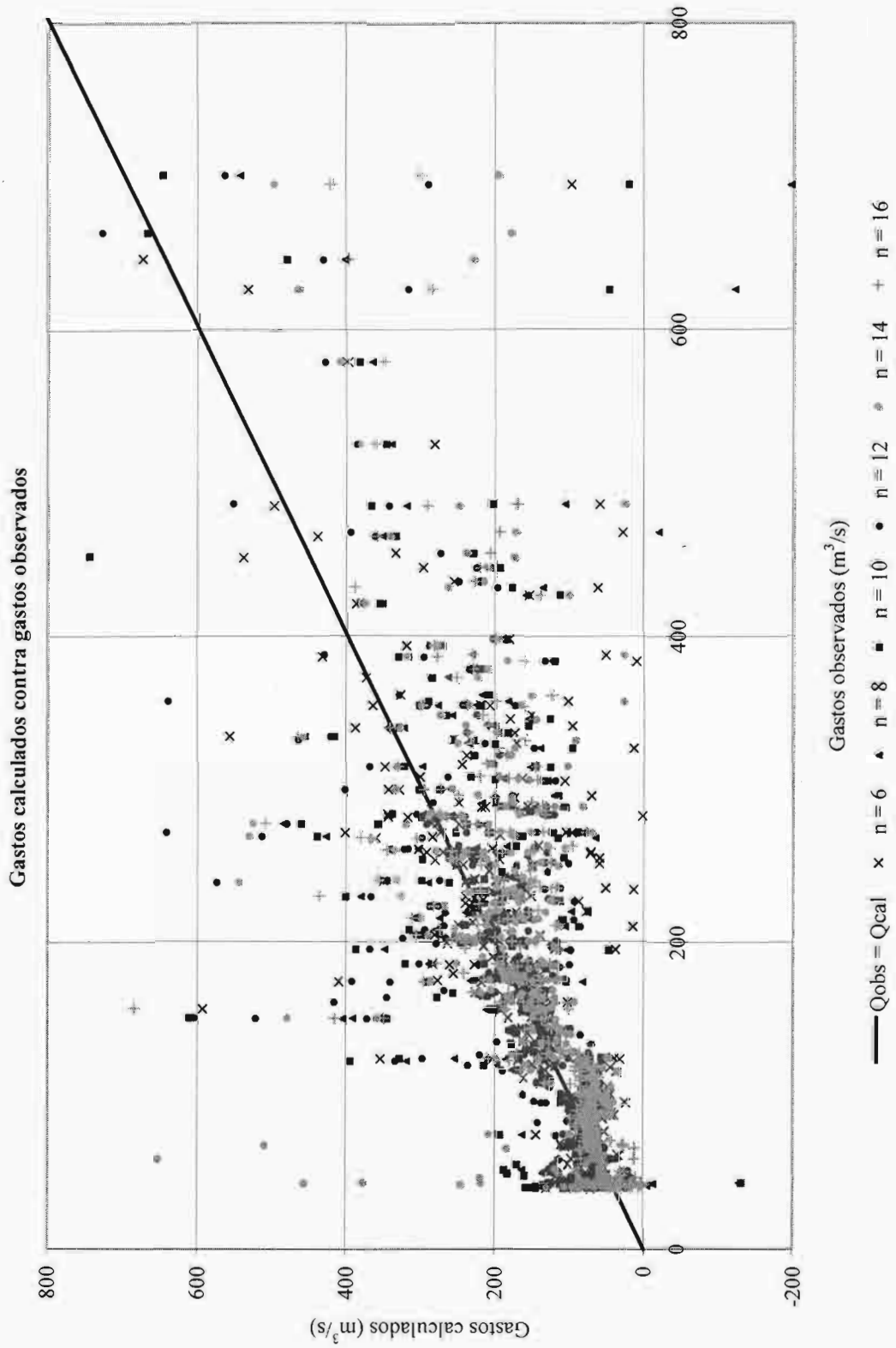


Figura 4.62 *Tendencia del desempeño respecto al número de neuronas utilizada en la capa oculta de la RNA con arquitectura 6 – n – 3*

5. Conclusiones y Recomendaciones

CONCLUSIONES

El flujo en un río o canal es un proceso complejo influenciado por factores como la topografía, la cobertura vegetal, tipo de suelos, características del cauce, presencia de acuíferos, distribución de precipitación y área urbanizada entre muchos otros. Los proyectos de ingeniería y análisis de impacto ambiental frecuentemente requieren estimaciones del escurrimiento en un ámbito amplio de problemas relacionados con el diseño y gestión de los recursos hídricos o sus propiedades características como el gasto pico y tiempo al pico en lugares donde no se tienen datos de gastos. Una gran variedad de métodos han sido desarrollados para este propósito, incluidos los modelos paramétricos o los modelos empíricos. Las redes neuronales unidireccionales con retropropagación están siendo ampliamente utilizadas en este tipo de aplicaciones.

Las RNA es una técnica con estructura matemática flexible capaz de identificar relaciones no lineales complejas entre los datos de entrada y salida sin intentar alcanzar una comprensión de la naturaleza del fenómeno. Trabaja como una caja negra simple para identificar una relación directa entre las entradas y salidas sin consideraciones detalladas de la estructura interna de los procesos físicos.

Las RNA proveen una solución nueva e interesante al problema de relacionar las variables de entrada y salida en sistemas complejos. Si hay pocas neuronas en las capas ocultas, la red puede ser incapaz de describir la función implícita a causa de la insuficiencia de parámetros (grados de libertad) para poder representar todos los puntos en los datos de entrenamiento. Por el contrario, si hay demasiadas neuronas, la red tiene demasiados parámetros libres y pueden provocar lo que se llama sobre entrenamiento perdiendo la habilidad de generalizar. Además, un excesivo número de neuronas ocultas puede retardar el proceso de entrenamiento a tal grado que la red necesite una cantidad considerable de tiempo para aprender.

Dado que las propiedades de los fenómenos naturales pueden ser descritas mediante relaciones entre observaciones medidas, el factor más significativo de la descripción física de los fenómenos naturales es que una propiedad desconocida puede ser predicha cuantitativamente a partir de otra si se conoce la relación existente entre las variables correspondientes.

Se concluye que mediante el uso de las RNA se pueden tomar en cuenta las dificultades que representa el tránsito de avenidas (avenidas simples, trenes de avenidas y uniones en cauces), como la influencia de la topografía en zonas altas de las cuencas donde las entradas laterales son significativas dado que el área tributaria en el tramo inicial es prácticamente inexistente y por lo tanto las entradas laterales llegan a representar casi la totalidad del escurrimiento en el cauce, grandes registros de información, gran cantidad de formas de los hidrogramas, amplios rangos del valor del escurrimiento, influencia de las temporadas de estiaje y lluvia registrados, etc. y tener un buen resultado en cuanto al desempeño y la representación de escenarios futuros.

En el caso del tránsito de trenes de avenidas entradas laterales en el tramo de cauce en estudio, tanto el método de Muskingum como el Muskingum – Cunge tienen serias limitaciones. Además se puede apreciar que las RNA son una buena herramienta para tratar el caso de confluencia de ríos que representan gran dificultad para los métodos tradicionales, mediante el aprendizaje de la información registrada por una arquitectura adecuada.

A lo largo del desarrollo del este trabajo se observa que entrenar una red neuronal con un amplio registro de información garantiza que la red entrenada tendrá buena capacidad de generalización, por lo que será capaz de predecir con buena exactitud avenidas futuras o reproducir las variaciones que sufren las avenidas.

En el desarrollo del caso 1 se entrenó una RNA unidireccional con retropropagación de arquitectura 3 – 6 – 3 que mostró un mejor desempeño que los métodos hidrológicos tradicionales y difirió con el desempeño del método hidráulico (para el cual se conocía la información necesaria para su aplicación). Sin embargo, la RNA entrenada presenta buenos resultados.

En el caso 2 se puso a prueba la arquitectura de la RNA del caso 1 con el aprendizaje de 5 avenidas históricas registradas en la región hidrológica 30 apreciándose que de igual forma tiene un desempeño adecuado en la generalización de avenidas de entrada, dada la gran diferencia de forma existente en el desarrollo de las avenidas utilizadas para el entrenamiento, situación muy común en la realidad. Por tanto, las RNA son una buena herramienta y da buenos resultados.

En el caso de trenes de avenidas se utilizó de nueva cuenta la misma arquitectura para la RNA, siendo la diferencia con los casos previos el tamaño del vector de entrada presentado a la red neuronal para el aprendizaje. No obstante que se trata de una estructura simple para tratar la dificultad de este problema, la RNA genera buenos resultados debido a que se consideraron los ejemplos suficientes (8 trenes de avenidas)

para lograr un buen entrenamiento. Se observó que para problemas de mayor dificultad, la RNA necesita más información que en los casos "sencillos" con la finalidad de tener un aprendizaje adecuado y buena capacidad de generalización.

Finalmente en el tratamiento de la confluencia de dos ríos dado que se utilizan registros anuales, sé manejó una estructura 6 – 12 – 3 para la red unidireccional con retropropagación, se observa la importancia de tener las neuronas suficientes en la capa oculta que contrarresta la falta de ejemplos de entrenamientos suficientes (en este caso solamente se tomaron dos registros anuales), no obstante la red entrenada presenta buena capacidad de generalización dada la dificultad que presenta el registro de avenidas utilizado para llevarla a cabo.

Se puede observar claramente entonces que es importante tener información suficiente para realizar el aprendizaje de la red y básicamente en función de dicha información se hace la propuesta de diseño de la arquitectura de la red (neuronas en la capa oculta).

RECOMENDACIONES

Es importante hacer notar que de acuerdo con lo observado en los resultados reportados en este trabajo, se deberá de procurar trabajar con arquitecturas lo más sencillas posibles, cuidando que se garantice un aprendizaje adecuado. Es importante hacer notar que la arquitectura de la red por utilizar cambiará de acuerdo con la naturaleza del caso de estudio, por lo que no hay nada definitivo en cuanto a la topología de las redes.

De acuerdo con los resultados de este trabajo y con la literatura consultada para el mismo (Kavzoglu, 2003), se recomienda utilizar en la(s) capa(s) oculta(s) como primer propuesta de arquitectura para el número de neuronas ocultas el doble de las utilizadas en la capa de entrada, para con esto dar libertad a la red de aprender más rápidamente y con mayor efectividad.

Una vez diseñada la arquitectura de la red, es importante asignar a la misma correctamente los parámetros necesarios para que realice el aprendizaje mediante el algoritmo seleccionado. Uno de los más importantes es el número de ciclos (`epochs`) asignado para que se realice la modificación de los pesos mediante el algoritmo de retropropagación; si se asigna un número alto de ciclos la red se sobreentrena y los resultados están lejos de los deseados, por el contrario un número pequeño no garantiza un aprendizaje adecuado, por tal razón y de acuerdo a los resultados de los casos estudiados se recomienda un rango de 1500 a 3000 epochs durante el aprendizaje según la dificultad del caso en estudio (figura 4.50) y por último detener el aprendizaje de la red en caso de observar que el número asignado de ciclos ocasiona sobre entrenamiento, lo que puede apreciarse en la gráfica de disminución del error al momento que ésta comience a volverse asintótica a un valor cualquiera.

De igual forma otro parámetro importante es la tolerancia del error al cual la red neuronal tratará de acercarse durante el aprendizaje. En este sentido se recomienda, utilizar un error del orden de 0.001, dado de que si se da a la red un valor de cero, la red se sobre entrena aún cuando el número de ciclos asignado previamente sea pequeño.

De acuerdo con lo apreciado en el desarrollo del caso 4 y observando los registros hidrométricos se puede apreciar claramente la presencia de la temporada de estiaje y la temporada de avenidas donde se tiene una fuerte influencia de una serie de gasto del mismo orden de magnitud contra una serie de gastos de un rango de valor mucho más amplio, el coeficiente de correlación resultó pequeño (R^2 = 0.4924). Dada esta circunstancia, se recomienda que en el caso de abordar este tipo de problemas se proponga la creación de dos RNA donde cada una de ellas trate una serie por separado, dándose con ello una mejor simulación y un valor mayor del factor de correlación.

Bibliografía

Aguilar E., *"Método hidrológico con base física para el tránsito de avenidas en cauces"*. Universidad Nacional Autónoma de México, División de Estudios de Posgrado de la Facultad de Ingeniería, Tesis Maestría, 1995.

Aldama, R. A. A., *"Least-Squares Parameter Estimation for Muskingum Flood Routing"*, Journal of Hydraulic Engineering, vol. 116, n° 4, April, p.580 – 586, 1990.

Aparicio J., *"Fundamentos de Hidrología de Superficie"*, Limusa Noriega ed., México, 2001.

Ballini, R., *"Redes neuronales para la previsión lluvia escurrimiento"*. Sao Carlos, 105. Instituto de Ciencias Matemáticas, Universidad de Sao Paulo, 1996.

Ballini, R., Franca, E., Kadowaki, M., Soares, S., Andrade, M., *"Modelos de redes neuronales y Box – Jenkins para la previsión de escurrimientos medios mensuales"*. Simposium Brasileño de Recursos Hídricos, 12ª, CD – Rom, 1997.

Berezowsky, V. M., *"Escurrimiento a Superficie Libre"*, Cap. A.2.9. del Manual de Diseño de Obras Civiles, Comisión Federal de Electricidad, México, 1980.

Bishop, C.M., *"Neural networks for pattern recognition"*. New York, Oxford University Press Inc., 1995.

Brahm, Alfredo, Varas, Eduardo, *"Disminución de los tiempos de entrenamiento en redes neuronales artificiales aplicadas a hidrología"*, Ingeniería hidráulica en México vol. XVIII, núm. 2, p. 69 – 82, abril – junio de 2003.

Campolo, M., Soldati, A., Andreussi, P., *Artificial neural network approach to flood forecasting in the River Arno"*. Hydrological Sciences Journal 48(3), p. 381 – 398, June 2003.

Campos Aranda D. F., *"Tránsito hidrológico de crecientes en ríos con flujo lateral"*. Agrociencia 34: p. 271 – 281, 2000.

Campos Aranda D. F., *"Procesos del ciclo Hidrológico, vol. 1 Tomo 2"* Primera Reimpresión. Universidad Autónoma de San Luis Potosí.

Carvalho, A. C. P.L.F., Braga, A. P., Ludemir, T.B. *"Fundamentos de redes neuronales artificiales"*. 11ª Escuela de computación, Rio de Janeiro, p. 246, 1998.

Carvalho Filho, E. C. B., Marar, J. F., *"O computador neural: uma máquina biologicamente inspirada"*, In: Simp. Sobre o Cérebro, Recife, 1993. – Anais, Recife, p. 73 – 84, 1993.

CFE, *"Cuencas del Grijalva y Usumacinta"*, México, 1977.

Chang, Fi-John, Chang Li-Chiu, Huang Hau-Lung. *"Real – Time recurrent learning neural network for stream – flow forecasting"*. Hydrological Processes 16 (3), p. 2577 – 2588, Sept. 2002.

Chibanga, R., Berlamont, J., Vandewalle, J., *"Modelling and forecasting of hydrological variables using artificial neural networks: The Kafue River sub – basin"*. Hydrological Sciences Journal 48(3), p. 363 – 380, June 2003.

Chow, V. T., Maidment D. R., Mays L. W., *"Hidrología Aplicada"*, McGraw Hill, Colombia, 1994.

Chow, V. T., *"Open Channel Hydraulics"*, McGraw Hill, New York, 1959.

Cigizoglu, H. K., *"Estimation, forecasting and extrapolation of river flows by artificial neural networks"*, Hydrological Sciences Journal, 48 (3), June 2003.

Collado, J., Wagner A. I., *"Estimación de parámetros para el método de Muskingum mediante selección de hidrogramas"*. Memorias 11° Congreso Nacional de Hidráulica, p. 296 – 307, 1990.

Collado, J., Wagner A. I., *"Tránsito aproximado de avenidas mediante transformadas de Fourier"*. Memorias 14° Congreso Latinoamericano de Hidráulica, p. 887 – 898, 1990.

Comisión internacional de limites y aguas México Guatemala, *"Cuenca del río Usumacinta"*, México, 1988.

Coulibaly. P., Bobée, B., Anctil, F., *"Improving extreme hydrologic events forecasting using a new criterion for artificial neural network selection"*. Hydrological Processes, 15, p.1533 – 1536, 2001.

Crespo, J.L., Mora, E., *"Drought estimation with neural networks"*. Advances in engineering software, n° 18, p. 167 – 170, 1993.

Dawson, C. W., Wilby, R., *"An artificial neural network approach to rainfall – runoff modeling"*. Hidrological Sciences Journal 43 (1), p. 47 – 66, February 1998.

Demuth, H., Beale, M., *"Neural Network Toolbox for use with Matlab"*. Math Works Inc. 1998.

Dolling, O. R., Varas, E. A., *"Artificial neural networks for streamflow prediction"*. Journal of Hydraulic Research, n° 5, vol. 40, p. 547 – 554, 2002.

Duan, Q., Sorooshian, S., Gupta, V. K., *"Effective and efficient global optimization for conceptual rainfall – runoff models*". Water Resources Research, 28(4), p. 1015 – 1031, 1992.

Figueroa, H. A., *"Plan de desarrollo de la margen derecha del Papaloapan, Región Tesechoacán – San Juan: 1ª parte diagnostico"*, México, 1988.

Flood, I., Kartman, N., *"Neural network in civil engineering. I: principles and understanding"*, Journal of Computing in Civil Engineering, vol. 8, n° 2, p. 131 – 148, 1994.

Flood, I., Kartman, N., *"Neural network in civil engineering. II: systems and application"*, Journal of computing in Civil Engineering, vol. 8, n°. 2, p. 149 – 162, 1994.

French, M. N., Krajewski, W. F., Cuykendall, R. R., *"Rainfall forecasting in space and time using a neural network*". Journal of Hydrology, n° 137, p. 1 – 31, 1992.

Galvão, C. O., Valença, M. J. S., *"Sistemas inteligentes: aplicaçoes a recursos hídricos e ciéncias ambientais"*, 1ª ed. Porto Alegre, Editora da universidade/UFRGS/ABRH, p. 246, 1999. (coleçao ABRH de Recursos Hídricos, vol. 7)

Gill, M. A., *"Flood routing by the Muskingum method*". Journal of Hydrology 36, p. 353 – 363, 1978.

Guyton, A., Hall, J. E., *"Tratamento de fisiologia médica"*, 9ª ed., Río de Janeiro, Editora Guanabara Koogan S. A., 1997. cap. 45, p. 510 – 537.

Haykin, S., *"Neural networks: a comprehensive foundation*". Ontario, Prentice Hall International, Inc., p. 696, 1994.

Hebb, D. *"The organization of behavior: a neuropsychological theory"*. New York, Wiley, 1949.

Hopfield, J. J., *"Neural networks and physical systems with emergent collective computational abilities"*. Proceeding of the National Academy of Sciences of the U.S.A. n° 79, p. 2554 – 2558, 1982.

Hornik, K., Stinchcombe, M. and White, H., *"Multilayer feedforward networks are universal approximators"*, Neural Networks, 2, p. 395 – 403, 1989.

Hsu, K., Gupta, H. V., Sorooshian, S., *"Artificial neural network modeling of the rainfall – runoff process"*. Water Resources Research, vol. 31, n° 10, p. 2517 – 2530, 1995.

Hsu, K., Gupta, H. V., Sorooshian, S., *"A superior training strategy for three – layer feedforward artificial neural networks"*. Tucson, University of Arizona, (Technique report, HWR n° 96 – 030, Department of Hydrology and Water resource), 1996.

Hydrologic Modeling System, *"HEC – HMS Technical Reference Manual"*.

Imrie, C. E., Durucan, S., Korre, A., *"River flow prediction using artificial neural networks: generalization beyond the calibration range"*. Journal of Hydrology 233 (2000) p. 138 – 153.

Instituto Mexicano de Tecnología del Agua (IMTA), *"Sistema de Información de Aguas Superficiales SIAS 1.0"* (BANDAS). 1997

Kadowaki, M., Soares, S., Andrade, M. G., *"Previsão de vazões mensais utilizando redes neurais multicamadas com algoritmo backpropagation"*. In: 4° Simpósio Brasileiro de Redes Neurais, Goiânia, 1997.

Karunanithi, N., Grenney, J.W., Whitley, D., Bovee, K., *"Neural networks for river flow prediction"*. Journal of Computing in Civil Engineering. vol. 8, n° 2, 1994.

Kavzoglu, t., Mather, P. M., "The use of backpropagating artificial neural networks in land cover classification". International Journal of Remote Sensing, vol. 24, n° 23, p. 4907 – 4938, 2003.

Kerem, C. H., *"Estimation, forecasting and extrapolation of river flows by artificial neural networks"*. Hydrological Sciences Journal 48 (3), p. 349 – 361, June 2003.

Kuligowski R. J., Barros, A. P., *"Using Artificial neural networks to estimate missing rainfall data"*. Journal of the American Water Resources Association. vol. 34, n° 6, p. 1437 – 1447, December 1998.

Larrañaga, P., Iñaki, I., *"Tema 8. Redes Neuronales"*, Departamento de ciencias de la computación e Inteligencia artificial. Universidad del país vasco – Euskal Erico Unibertsitatea

Lanna, A. E., Schwarzbach , M., *"MODHAC – modelo hidrológico auto-calibrável"*. Florianópolis, IPH – Instituto de Pesquisas Hidráulicas de UFRGS, 1988.

Linsley R. K. Jr., Kohler A. M., Paulhus J. L. H., *"Hidrología para ingenieros"*. Segunda edición. McGraw Hill, México, 1992.

Matlab 6.5 The mathworks, Inc. Matlab, 2002.

Matlab 6.5, neural network toolbox, User's Guide Version 4.0, 2002.

McCulloch, W.S., Pitts, W. *"A logical calculus of the ideas immanent in nervous activity"*. Bulletin of Mathematical Biophysics, 5: p. 115 – 133, 1943.

Minns, A. W., Hall, M. J., *"Artificial neural networks as rainfall – runoff models"*, Hydrologic Sciences 41 (3), 1996.

Minsky, M. L., Papera, S. A. *"Perceptrons: an introduction to computational geometry"*. Massachusetts, MIT Press, 1969.

Mockus, V. & W. Styner, *"National Engineering Handbook Part 630, Chapter 17"*. 100 p. National Resources Conservation Service, 1972.
http://www.wcc.nrcs.usda.gov/hydro-techref-neh-630.html

NERC (Natural Environment Research Council), *"Flood Studies Report. vol. III: Flood Routing Studies"*, London, England, p. 76, 1975.

Newton, D. W., Vinyard, J. W., *" Computer – determined unit hydrograph from floods"*, Journal Hydraulics Div. 93, p. 219 – 235, 1967.

Nielsen, R. H., *"Neurocomputing"*. USA, Addison – Wesley Publishing Company, 1991.

Ochoa-Rivera, J. C., Garcia-Bartual, R., Andreu, J., *"Artificial neural networks modeling approach applied to the probabilistic management of water resources system"*, Departamento de Hidráulica e Ingeniería ambiental, Universidad Politécnica de Valencia

O´Donnell, T., *"A direct three-parameter Muskingum procedure incorporating lateral inflow"*, Journal of Hydrological Sciences, vol. 30, n° 4, Dec., p. 479 – 496, 1985.

Rajurkar, M. P., Kothyari, U. C., Chaube, U. C., *"Artificial neural networks for daily rainfall – runoff modeling"*. Hydrological Sciences Journal 47(6), p. 865 – 877, December 2002.

Rao Govindaraju, *"Artificial neural networks in Hydrology I: Preliminar concepts"*, ASCE task committee on application of artificial neural networks in Hydrology, Journal of hydrologic Engineering 5(2), p. 115 – 123, 2000.

Rao Govindaraju, *"Artificial neural networks in Hydrology II: Hydrologic applications"*, ASCE task committee on application of artificial neural networks in Hydrology, Journal of hydrologic Engineering 5(2), 124 – 137, 2000.

Rochester, N., Holland, J. N., Haibt, L. H., Duda, W. L., *"Test on cell assembly theory of action of the brain, using a large digital computer"*, IRE transactions of information theory IT – 2, p. 80 – 93, 1956.

Rosenblatt, F., *"Principles of neurodynamics: perceptrons and the theory of brain mechanisms"*. New York, Spartan Books, 1962.

Rosenblatt, F., *"The perceptron: a probabilistic model for information storage and organization in the brain"*. Psychological Review, n° 65, p. 386 – 408, 1958.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., *"Learning representations by back – propagating errors"*. Nature, 323, p. 533 – 536, 1986.

Salas, J. D., Markus, M. and Tokar, A. S., *"Streamflow forecasting based on artificial neural networks"*, Artificial neural networks in hydrology, R. S. Govindaraju and A. Ramachandra Rao (eds.), Kluwer Academic Publiser, Dordrecht, 23 – 51, 2000.

Sanchez – Fernandez, L. F., Haie, N., *"Modelo de red neuronal para simulación de hidrogramas en cuencas hidrográficas"*. Ingeniería Hidráulica en México. vol. XIX, n° 2, p. 17 – 29, abril – junio 2004.

Sarmento, F. J., *"Modelagem de séries hidrológicas através de redes de neuônios"*, Revista Brasileira de Recursos Hidricos – RBRH, vol. 1, n° 2, p. 19 – 31, 1996.

Secretaria de Recursos Hidráulicos, "Reconocimiento de la cuenca del río Tesechoacán", México, 1958.

Secretaria de Recursos Hidráulicos, *"Regiones hidrológicas num. 30 (Grijalva – Usumacinta) y num. 31 (Yucatan oeste)"*, México, 1969.

Scalero, R. S., Tepedelenliogiu, N., *"A fast new algorithm for training feedforward neural networks"*. IEEE trans. Signal Process, 40(1), p. 202 – 210, 1992.

Shamseldin, A. Y., O´Connor, K. M., Liang, G.C., *"Methods for combining the outputs of different rainfall – runoff models"*. Journal of Hydrology, n° 197, p. 203 – 229, 1996.

Shamseldin, A. Y., *"Application of a neural network technique to rainfall – runoff modeling"*. Journal of Hidrology, n° 199, p. 272 – 294, 1997.

Singh, V. P, *"Elementary Hydrology"*. Prentice Hall, 1992.

Solomatine, D. P., Dulal K. N., *"Model tress as an alternative to neural networks in rainfall – runoff modeling"*. Hydrological Sciences Journal 48(3), p. 399 – 411, June 2003.

Sotelo, A. G., *"Apuntes de Hidráulica General II"*, Facultad de Ingeniería, UNAM, México, 1989.

Souza, M. S., Machado, D., Martino, M. B., *"Determinação de vazões utilizando redes neurais", In: 11° simpósio Brasileiro de Recursos Hídricos, vol. I, p. 63 – 68, 1995.*

Stendinger, J. R. and Taylor, M. R., *"Synthetic streamflow generation I. Model verification and validation"*, Water Resources Research, 18 (4), 909 – 918, 1982.

Tafner, M. A., Xerez, M., Rodrigues Filho, I. W., *"Redes neurais artificiais: introdução e príncipios de neurocomputação"*, Blumenau, EKO / Editora de FURB, p. 199, 1996.

Tang, Z., Almeida, C., Fishwick, P. A., *"Times series forecasting using neural networks vs. Box – Jenkins methodology"*, In simulations, p. 301 – 310. Simulations Councils, 1991.

Tayfur, G., *"Artificial neural network for sheet sediment transport"*. Hydrological Sciences Journal 47(6), p. 879 – 892, December 2002.

Thirumalaiah, K., Deo, M.C., *"River stage forecasting using artificial neural networks"*. Journal of Hydrologic Engineering, vol. 3, n° 1, 1998.

Tutorial de Matlab.
http://www.mathworks.com/access/helpdesk/help/pdf doc/nnet/nnet.pdf

US Army Corps of Engineers (1994), *"Flood Runoff Analysis, Chapter 9"*, 24 p.

Valencia, M. J. S., *"Aplicação de redes neurais na área de recursos hídricos"*. In simpósio Brasileiro de Recursos Hídricos, 120, CD – ROM, 1997.

Viessman, W. & G. L. Lewis, *"Introduction to Hydrology"*, Harper Collins, 4ª ed. 1995.

Wanielista, M., *"Hydrology and Water Quality Control"*, 2ª edición Ed. Wiley, 1997.

Widrow, B., Hoff, M. E. *"Adaptative switching circuits"*, Instituto of Radio Engineers, Western Electronic Show and Convention, p. 96 – 104, 1960.

Xiao, R., Chandrasekar, V., *"Multiparameter radar rainfall estimation using neural network technique"*, In: 27ª Conference on Radar Meteorology, Colorado, p. 199 – 201, 1995.

Yang, C. C., Chang, L. C., Chen, C. S., *"Comparison of integrated artificial neural network with time series modeling for flood forecast"*, Journal of Hydroscience and Hydraulic Engineering, vol. 17, N° 2, 1999.

Yao, X., *"Evolving artificial neural networks"*. Proceeding of the IEEE, Vol. 87, n° 9, p. 1423 – 1439, September 1999.

Yitian Li, Gu Roy R., *"Modeling flow and sediment transport in a river system using an Artificial Neural network"*. Environmental Management Vol. 31, n° 1, p. 122 – 134 Ene. 2003.

Zhang, B., Govindaraju, R. S., *"Geomorpholgy – based artificial neural networks (GANNs) for estimation of direct runoff over watersheds"*, Journal of Hydrology n° 273, p. 18 – 34, 2003.

Zealand, C. M., Burn, D.H. and Simonovic, S. P., *"Short term streamflow forecasting using artificial neural networks"*, Journal of Hydrology 214, p. 32 – 48, 1999.

Anexo A. Comparación de algoritmos de entrenamiento

Algoritmo Levenberg – Marquardt (trainlm)
Fuente: Matlab 6.5, neural network toolbox, User´s guide Version 4.0, 2002.

Similar a los métodos Cuasi – Newton, el algoritmo de Levenberg – Marquardt fue diseñado para acercarse a la rapidez del entrenamiento de segundo orden sin tener que calcular la matriz Hessiana. Cuando la función de desempeño (`performance function`) tiene la forma de una suma de cuadrados (como es típica en un entrenamiento de redes feedforward), entonces la matriz Hessiana puede aproximarse como:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}$$

Y el gradiente puede ser calculado como

$$\mathbf{g} = \mathbf{J}^T \mathbf{e}$$

Donde ***J*** es la matriz Jacobiana que contiene las primeras derivadas de los errores de la red con respecto a los pesos (`reights`) y los umbrales (`biases`), y ***e*** es el vector de errores de la red. La matriz Jacobiana puede ser calculada mediante una técnica estándar de retropropagación que es mucho menos complejo que calcular la matriz Hessiana.

El algoritmo Levenberg – Marquardt utiliza esta aproximación a la matriz Hessiana en la siguiente actualización del método de Newton:

$$\mathbf{X}_{k+1} = \mathbf{X}_k - [\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e}$$

Cuando el escalar μ es cero, se trata solamente del método de Newton, utilizando la aproximación de la matriz Hessiana. Cuando μ es grande, se convierte en el gradiente descendiente con paso pequeño. El método de Newton es más rápido y más cercano a la tolerancia del error, entonces el objetivo es cambiar al método de Newton tan pronto sea posible. Así μ es disminuido en cada paso exitoso (reducción en la función de desempeño) y es incrementado únicamente cuando un paso tentativo incrementará la función de desempeño. De esta forma, la función de desempeño siempre será reducida en cada iteración del algoritmo.

Este algoritmo aparece para ser el método más rápido para entrenar redes neuronales con conexión hacia adelante de tamaño moderado (arriba de cientos de pesos).

Algoritmo Levenberg – Marquardt reducción de memoria

La principal desventaja del algoritmo Levenberg – Marquardt es que requiere el almacenamiento de algunas matrices que puede ser muy grandes para ciertos problemas. El tamaño de la matriz jacobiana es $\boldsymbol{Q}$ x $\boldsymbol{n}$, donde $\boldsymbol{Q}$ es número de casos para entrenamiento y $\boldsymbol{n}$ es el número de pesos y umbrales en la red. Esto obliga a que la matriz no pueda ser calculada y guardada como una sola. Por ejemplo, si dividimos el jacobiano en dos submatrices iguales podemos calcular la aproximación de la matriz Hessiana como sigue:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} = \mathbf{J}_1^T \mathbf{J}_1 + \mathbf{J}_2^T \mathbf{J}_2$$

Por consiguiente, el jacobiano completo no existe en ningún momento. La aproximación hessiana puede ser calculada por la suma de una serie de subterminos. Una vez que un subtermino ha sido calculado, la correspondiente submatriz del jacobiano puede ser aclarada.

Cuando usamos la función de entrenamiento `trainlm`, el parámetro `mem_reduc` es utilizado para determinar cuantas filas del jacobiano son calculadas en cada submatriz. Si `mem_reduc` es colocado igual a 1, entonces el Jacobiano completo es calculado, y la reducción de memoria no es lograda. Si `mem_reduc` es colocado igual a 2, entonces solamente la mitad del jacobiano será calculado en una sola vez. Esto ahorra la mitad de la memoria usada para el cálculo del jacobiano completo.

Existe una desventaja utilizando reducción de memoria. Una significativa sobrecarga computacional es asociada con el cálculo del jacobiano en submatrices. Si se tiene suficiente memoria disponible, entonces es mejor colocar `mem_reduc` en 1 y calcular el jacobiano completo. Si se tiene una serie grande para el entrenamiento, y se esta fuera de la capacidad de la memoria, entonces debe de colocarse `mem_reduc` en 2, y tratar de nuevo. Si aun no existe memoria suficiente, continuar incrementando el valor de `mem_reduc`.

Siempre que se use reducción de memoria, el algoritmo Levenberg – Marquardt siempre calculará la aproximación de la matriz hessiana, cuyas dimensiones son $\boldsymbol{n}$ x $\boldsymbol{n}$. Si la red es muy grande, entonces no habrá espacio de memoria disponible. Si ese es el caso, entonces tratar con `trainscg`, `trainrp` o uno de los algoritmos de gradiente conjugado.

Comparación en velocidad y uso de memoria de los algoritmos

Es muy complicado saber cuál algoritmo de entrenamiento será el más rápido para un cierto problema. Dependerá de muchos factores, incluyendo la complejidad del problema, la cantidad de información en la colección de datos para entrenamiento, el número de pesos y umbrales en la red, el error deseable, y si la red esta siendo utilizada para reconocimiento de patrones (análisis discriminante) o función de aproximación (regresión).

La siguiente tabla enlista los algoritmos que son probados y los acrónimos utilizados para su identificación.

Acrónimo	Algoritmo
LM	`trainlm` - Levenberg-Marquardt
BFG	`Trainbfg` - BFGS Quasi-Newton
RP	`trainrp` - Resilient Backpropagation
SCG	`Trainscg` - Scaled Conjugate Gradient
CGB	`Traincgb` - Conjugate Gradient with Powell / Beale Restarts
CGF	`Traincgf` - Fletcher-Powell Conjugate Gradient
CGP	`Traincgp` - Polak-Ribiére Conjugate Gradient
OSS	`Trainoss` - One-Step Secant
GDX	`Traingdx` - Variable Learning Rate Backpropagation

Tabla A1. *Acrónimos de algoritmos de entrenamiento*

Ejemplo de comparación – Conjunto de información

Se trata de una función de aproximación simple para la función seno. Una red 1 – 5 – 1, con funciones de transferencia `tansig` en la capa oculta y lineal en la capa de salida, es usada para aproximar un periodo simple de la función seno. La siguiente tabla muestra los resultados del entrenamiento de la red usando nueve algoritmos de entrenamiento diferentes.

Cada registro en la tabla representa 30 diferentes ensayos. Donde se utilizaron diferentes valores aleatorios iniciales en cada ensayo. En cada caso, la red fue entrenada hasta que la suma del error al cuadrado fue menor a 0.002.

El algoritmo mas rápido para este problema es el Levenberg – Marquardt. En promedio, es cuatro veces más rápido que el siguiente más rápido algoritmo. Estos son los tipos de problemas para los cuales el algoritmo LM es más conveniente. Una función de aproximación donde la red tenga menos de cien pesos y la aproximación resulta ser muy precisa.

Algoritmo	Tiempo promedio (s)	Razón	Tiempo mínimo (s)	Tiempo máximo (s)	Std. (s)
LM	1.14	1.00	0.65	1.83	0.38
BFG	5.22	4.58	3.17	14.38	2.08
RP	5.67	4.97	2.66	17.24	3.72
SCG	6.09	5.34	3.18	23.64	3.81
CGB	6.61	5.80	2.99	23.65	3.67
CGF	7.86	6.89	3.57	31.23	4.76
CGP	8.24	7.23	4.07	32.32	5.03
OSS	9.64	8.46	3.97	59.63	9.79
GDX	27.69	24.29	17.21	258.15	43.65

Tabla A2. *Tiempo utilizado por los algoritmos de entrenamiento*

El desempeño de varios algoritmos puede ser afectado por la exactitud requerida para la aproximación. Esto se demuestra en la figura A1, que grafica el error cuadrático promedio contra el tiempo de ejecución (promedio de los 30 ensayos) para algunos algoritmos representativos. Se puede apreciar que el error en el algoritmo LM decrece mucho más rápido con el tiempo que con los otros algoritmos mostrados.

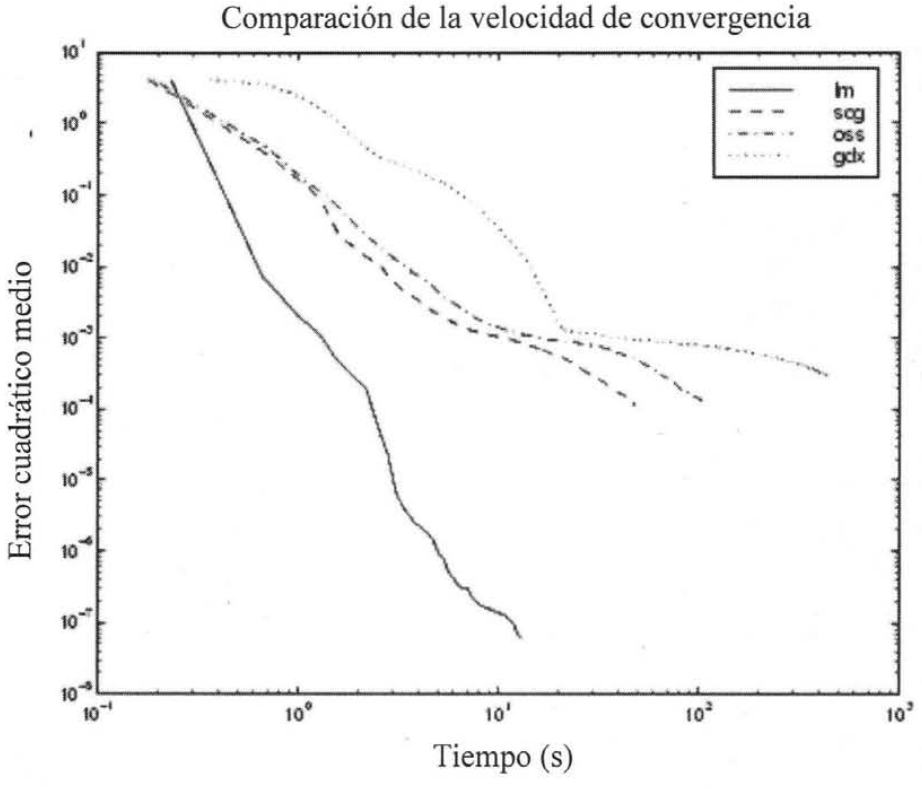


Figura A1. *Tiempo utilizado por los algoritmos de entrenamiento*

La relación entre los algoritmos es ilustrado en la figura A2, la cual muestra el tiempo requerido para converger contra el error cuadrático medio de convergencia al objetivo.

Se puede observar como el error al objetivo es reducido, el mejoramiento provisto por el algoritmo LM es más pronunciado. Algunos algoritmos desempeñan mejor como el error al objetivo es reducido (LM y BFG), y otros degradan como el error al objetivo es reducido (OSS y GDX)

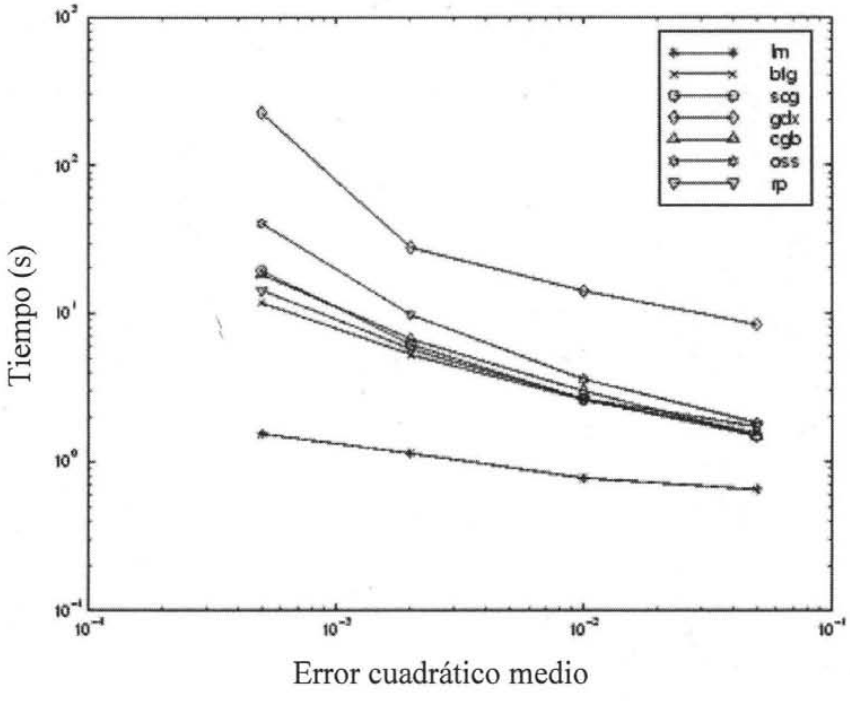


Figura A2. *Tiempo utilizado por los algoritmos de entrenamiento*

Resumen

En general, en problemas de aproximación de funciones, y para lo cual no se requiera de muchos pesos, el algoritmo Levenberg – Marquardt tendrá la más rápida convergencia.

Esta ventaja es especialmente notable si re requiere de un entrenamiento muy preciso. En muchos casos `trainlm` es capaz de obtener errores cuadráticos promedio más pequeños que cualquier otro algoritmo probado. Sin embargo, conforme el número de pesos en la red aumenta, la ventaja de `trainlm` va decreciendo.

Anexo B. Algoritmo para red Unidireccional con retropropagación

A continuación se presenta el código para la creación de una red unidireccional con retropropagación del error en Matlab 6.5 mediante aprendizaje supervisado, si no se cuenta con la herramienta NNtool.

Neural0a.m

```
% CÓDIGO BACKPROPAGATION
% DEFINIENDO PATRONES DE ENTRADA Y SALIDA
% MENÚ PRINCIPAL DE RNA CON BACKPROPAGATION
%
option = menu ( 'REDES NEURONALES MEDIANTE BACKPROPAGATION. Seleccione una
opción:',...
                ' Preparación de los datos',...
                ' Escalamiento de los datos',...
                ' Entrenamiento de la red neuronal',...
                ' Prueba del modelo de la red neuronal',...
                ' Mantenimiento de archivos',...
                ' Ninguna opción [Default]');
disp('')
```

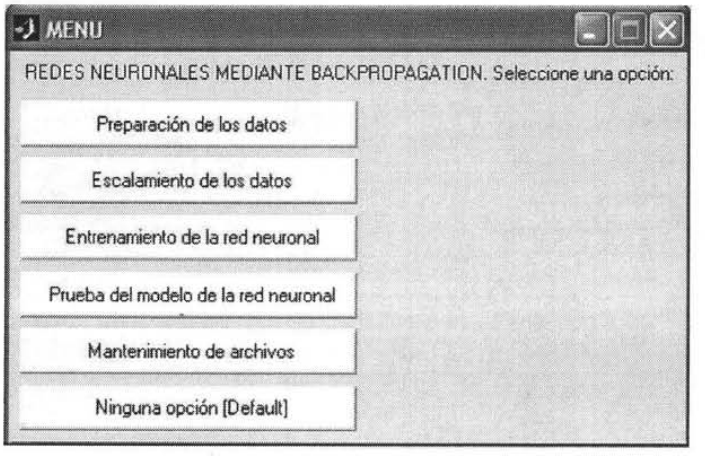


Figura B.1 *Menú principal del programa para creación de RNA´s.*

```
%
        if option == 1 neural0a;
                elseif option == 2
                        neural0b;
                elseif option == 3
                        neural3;
                elseif option == 4
                        neural4;
                elseif option == 5
                        neural5;
                else
        clc;
end
```

Neural0a.m

```
% MENÚ PRINCIPAL PARA LA PREPARACIÓN DE DATOS
%
option = menu ( ' La preparación de los datos es ',...
                ' Para la construcción del modelo',...
                ' Para la prueba del modelo',...
               ' Ninguna opción [Default]');
disp('');
```

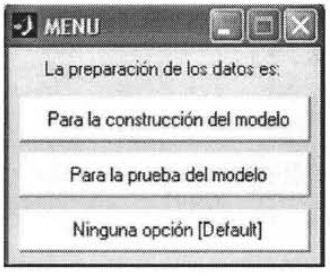


Figura B.2 *Submenú principal de preparación de datos*

```
%
        if option == 1
                neural1a;
        elseif option == 2
                neural1b;
        else
                neural0;
        clc;
end
```

Neural0b.m

```
% MENÚ PRINCIPAL DE ESCALAMIENTO DE DATOS
%
option = menu ( ' El escalamiento es ',...
                ' Para los datos de entrenamiento',...
                ' Para los datos de prueba del modelo',...
                ' Ninguna opción [Default]');
disp('');
```

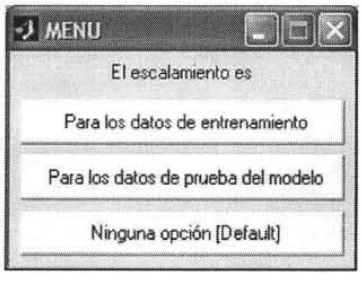


Figura B.3 *Submenú principal de escalamiento de datos*

```
%
        if option == 1
                neural2a;
        elseif option == 2
                neural2b;
        else
                neural0;
        clc;
```

```
end
```

Neural1.m

```
% CÓDIGO BACKPROPAGATION. (neural1)
% DEFINIENDO PATRONES DE ENTRADA Y SALIDA
% CREAR LOS ARCHIVOS EN UN FORMATO ESTÁNDAR
% CARGAR LA MATRIZ DE DATOS > EN LA PRÓXIMA LÍNEA INCLUYA EL NOMBRE DEL
ARCHIVO
% DONDE ESTA LA MATRIZ  (EL NOMBRE DE LA MATRIZ ES P)
% PREPARACIÓN DE LA MATRIZ P
%
clear all;
        echo off;
        final_row = 0;
        flag = 0;
        flag3 = 0;
clc
        flag4 = menu ( ' Los datos originales son:',...
                        ' Un archivo en formato MAT',...
                        ' Un archivo en formato ASCII',...
                        ' Ninguna opción (Default)');
disp('');
clc;
        fprintf ( ' \ n \ n PROCEDIMIENTO BACKPROPAGATION \ n \ n ' )
        fprintf ( ' PREPARACIÓN DE LOS DATOS \ n \ n' )
        disp ( ' SELECCIONANDO UN ARCHIVO ' );
        if flag4 == 1
        flag3 = 1;
        IN = input ( ' Indique el nombre del archivo MAT: (entre apóstrofes) ' );
clc
        disp ( ' Archivos MAT. Los datos deben estar en formato MAT ' );
        disp ('')
        disp ( ' Presione cualquier tecla para continuar ' );
        pause
        load(IN);
        disp ( ' Desde la siguiente lista, seleccione el nombre del archivo ' );
        whos
        TEMPO = input ( ' Indique el nombre seleccionado: ' );
 clc
        disp ( ' Cargando archivo MAT en TEMPO ' );
        disp ( ' Presione cualquier tecla al estar listo ' );
        pause
        echo off
        elseif flag4 == 2
                flag = 1;
        while ( flag==1 )
                disp ( ' Indique unidad y nombre del archivo ASCII: ' );
                IN = input ( ' (entre apóstrofes. Ej:c: \ mydata.dat):  ' );
                file = fopen ( IN, ' rw ' );
                if file == -1
                fprintf ( ' \ n \ n Error: Este archivo no existe. Trate de nuevo... ' )
        flag = 1;
        else
                flag = 0;
        end
end
        fprintf ( ' \ n \ n Cargando el archivo... \ n \ n ' )
```

```
        data = fscanf ( file, ' %f ' );
        variables = input ( ' Indique el total de variables ( patrones ):  ' );
        rows = length(data) / variables;
        data = reshape (data, rows, variables);
        TEMPO = data;
end
        if flag3 == 1
                flag = input ( ' Otro archivo ( 0 si / 1 no): ' );
                fclose ( ' all ' );
                flag3 = 1;
        end
end
%
% DEFINIENDO Y GUARDANDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
        if flag3 == 1
clc
        type_data = menu ( ' Clase de datos a guardar:',...
                           ' Ejemplos de salida para la red',...
                           ' Ejemplos de entrada para la red');
        disp('')
                if type_data == 1
                Out = TEMPO;
                save Output.mat Out
                clear TEMPO;
        else
                Inp = TEMPO;
                save Input.mat Inp
                clear TEMPO;
        end
end
```

Figura B.4 *Submenú para cargar patrones de entrada y salida*

Neural1a.m

```
% CÓDIGO BACKPROPAGATION. (neural1a)
% DEFINIENDO PATRONES DE ENTRADA Y SALIDA
% CREAR LOS ARCHIVOS EN UN FORMATO ESTÁNDAR
% CARGAR LA MATRIZ DE DATOS > EN LA PRÓXIMA LÍNEA INCLUYA EL NOMBRE DEL
ARCHIVO
% DONDE ESTA LA MATRIZ (EL NOMBRE DEL LA MATRIZ ES P)
% PREPARACIÓN DE LA MATRIZ P
%
clear all;
        echo off;
        final_row = 0;
        flag = 0;
        flag3 = 0;
clc
```

```
        flag4 = menú ( ' Los datos originales son:',...
                        ' Un archivo en formato MAT',...
                        ' Un archivo en formato ASCII',...
                        ' Ninguna opción (Default)');
        disp('');
clc;
        fprintf ( ' \ n \ n PROCEDIMIENTO BACKPROPAGATION \ n \ n ' )
        fprintf ( ' Preparación de los datos \ n \ n ' )
        disp ( ' SELECCIONANDO UN ARCHIVO ' );
        if flag4 == 1
        flag3 = 1;
                IN = input ( ' Indique la ruta y archivo MAT: (entre apóstrofes) ' );
        clc
                disp ( 'Archivos MAT. Los datos deben estar en formato MAT ' );
                disp ('')
                disp ( ' Presione cualquier tecla para continuar ' );
                pause
                load (IN);
                disp ( ' Desde la siguiente lista, seleccione el nombre del archivo ' );
                whos
                TEMPO = input ( ' Indique el nombre seleccionado: ' );
        clc
                disp ( ' Cargando archivo MAT en TEMPO. ' );
                disp ( ' Presione cualquier tecla al estar listo. ' );
                pause
                echo off
        elseif flag4 == 2
        flag = 1;
                while (flag == 1)
                disp ( 'Indique la ruta y archivo ASCII:');
                IN = input ( ' ( entre apóstrofes. Ej:c:\mydata.dat):  ' );
                file = fopen (IN, ' rw ' );
                if file == -1
                fprintf ( ' \ n \ n Error: Este archivo no existe. Trate de nuevo... ' )
        flag = 1;
                else
        flag = 0;
        end
end
        fprintf ( ' \ n \ n Cargando el archivo... \ n \ n ' )
        data = fscanf ( file, ' %f ' );
        variables = input ( ' Indique el total de variables (patrones): ');
        rows = length(data) / variables;
        data = reshape (data, rows, variables);
        TEMPO = data;
else
        neural0a;
end
        if flag3 == 1
                flag = input ( ' Otro archivo( 0 si / 1 no): ' );
                fclose ( ' all ' );
        flag3 = 1;
end
%
% DEFINIENDO Y GUARDANDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
```

```
if flag3 == 1
        clc
        type_data = menU ( ' Clase de datos a guardar:',...
                           ' Ejemplos de salida para la red',...
                           ' Ejemplos de entrada para la red');
        disp (")
if type_data == 1
        Out=TEMPO;
        figure;
        plot (Out);
        save Output.mat Out
        clear TEMPO;
else
        Inp =TEMPO;
        save Input.mat Inp
        clear TEMPO;
end
        neural0a;
end
```

Figura B.5 *Submenú para guardar patrones de entrada y salida*

Neural1b.m

```
% CÓDIGO BACKPROPAGATION. (neural1b)
% DEFINIENDO PATRONES DE ENTRADA Y SALIDA
% CREAR LOS ARCHIVOS EN UN FORMATO ESTÁNDAR
% CARGAR LA MATRIZ DE DATOS > EN LA PRÓXIMA LÍNEA INCLUYA EL NOMBRE DEL
ARCHIVO
% DONDE ESTA LA MATRIZ (EL NOMBRE DE LA MATRIZ ES P)
% PREPARACIÓN DE LA MATRIZ P
%
clear all;
        echo off;
        final_row = 0;
        flag = 0;
        flag3 = 0;
clc
        flag4 = menu ( ' Los datos originales son:',...
                       ' Un archivo en formato MAT',...
                       ' Un archivo en formato ASCII',...
                       ' Ninguna opción (Default)');
        disp (");
clc;
        fprintf ( ' \ n \ n PROCEDIMIENTO BACKPROPAGATION \ n \ n ' )
        fprintf ( ' Preparación de los datos \ n \ n ' )
        disp ( ' SELECCIONANDO UN ARCHIVO ' );
        if flag4 == 1
        flag3 = 1;
        IN = input ( ' Indique la ruta y archivo MAT: (entre apóstrofes) ' );
```

```
clc
        disp ( ' Archivos MAT. Los datos deben estar en formato MAT ' );
        disp (")
        disp ( ' Presione cualquier tecla para continuar ' );
        pause
        load (IN);
        disp ( ' Desde la siguiente lista, seleccione el nombre del archivo ' );
        whos
        TEMPO = input ( ' Indique el nombre seleccionado: ' );
clc
        disp ( ' Cargando archivo MAT en TEMPO.' );
        disp ( ' Presione cualquier tecla al estar listo.' );
        pause
        echo off
        elseif flag4 == 2
                flag = 1;
        while (flag == 1)
                disp ( ' Indique la ruta y archivo ASCII: ' );
                IN = input( ' ( entre apóstrofes. Ej: c: \ mydata.dat ):  ' );
                file = fopen ( IN,' rw ');
                if file == -1
                fprintf ( ' \ n \ n Error: Este archivo no existe. Trate de nuevo... ')
        flag = 1;
                else
        flag = 0;
                end
end
        fprintf  ( ' \ n \ n Cargando el archivo... \ n \ n ' )
        data = fscanf (file, ' %f ' );
        variables = input ( ' Indique el total de variables (patrones): ' );
        rows = length (data) / variables;
        data = reshape(data, rows, variables);
        TEMPO = data;
end
        if flag3 == 1
                flag = input( ' Otro archivo( 0 si / 1 no): ' );
                fclose ( ' all ' );
                flag3 = 1;
end
%
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
if flag3 == 1
clc
        type_data = menu ( ' Clase de datos a guardar: ',...
                                ' Ejemplos de salida para la red ',...
                                ' Ejemplos de entrada para la red ' );
disp(")
        if type_data == 1
                Out = TEMPO;
                figure;
                plot(Out)
                save TestOut.mat Out
        clear TEMPO;
else
                Inp = TEMPO;
```

```
            save TestInp.mat Inp
        clear TEMPO;
end
        neural0a;
end
```

***Figura B.6** Submenú para guardar patrones de entrada y salida*

Neural2.m

```
% CÓDIGO BACKPROPAGATION. (neural2)
% ESCALAMIENTO DE EJEMPLOS DE SALIDA Y/O ENTRADA
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
clc
        fprintf('Escalamiento de los datos\n\n')
        ype_data = menu ( ' Clase de datos a ser escalados:',...
                          ' Ejemplos de salida de la red',...
                          ' Ejemplos de entrada de la red',...
                          ' Ninguna opción [Default]');
disp('')
        if type_data == 1
            load Output;
            TEMPO = Out;
        elseif type_data == 2
            load Input;
            TEMPO = Inp;
        else
            return
end
        flag1 = input ( ' Quiere hacer escalamiento( 0 si / 1 no): ' );
        if flag1 == 0
            fprintf ( ' \ n \ n Rango de escalamiento...' );
            tmin = input( ' \ n Valor mínimo en la escala: ' );
            tmax = input ( ' Valor máximo en la escala: ' );
% P3 = escalamiento ( TEMPO, tmin, tmax );
            fprintf( ' \ n \ n Escalando.... \ n \ n ' );
            A = TEMPO;
            [M,N] = size(A);
            V = [min(A); max(A)];
            Units = ones(M, N);
            P1 = A – units (1:M, 1:1) * V(1:1, 1:N);
        for l = 1:M;
        for k = 1:N;
            if V(2,k)-V(1,k) ~= 0;
            P1 (l, k) = (P1 (l, k)) / ( V (2, k) – V (1, k) );
            P1 (l, k) = tmin + P1 (l, k) * (tmax – tmin );
        end;
end;
```

```
end;
        clear A;
        clear TEMPO;
        TEMPO = P1';
        clear P1;
        if type_data == 1
                Out = TEMPO;
                save Output.mat Out
        else
                Inp = TEMPO;
                save Input.mat Inp
        end
end
```

***Figura B.7** Submenú para escalamiento de ejemplos de entrada y/o salida*

Neural2a.m

```
% CÓDIGO BACKPROPAGATION. (neural2)
% ESCALAMIENTO DE EJEMPLOS DE SALIDA Y/O ENTRADA
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
clc
        fprintf ( ' Escalamiento de los datos \ n \ n ' )
        type_data = menu ( ' Clase de datos a ser escalados:',...
                           ' Ejemplos de salida de la red',...
                           ' Ejemplos de entrada de la red',...
                           ' Ninguna opción [Default]');
disp('')
        if type_data == 1
                load Output;
                TEMPO = Out;
        elseif type_data == 2
                load Input;
                TEMPO= Inp;
        else
                return
end
        flag1 = input ( ' Quiere hacer escalamiento( 0 si / 1 no): ' );
        if flag1== 0
                fprintf ( ' \ n \ n Rango de escalamiento...' );
                tmin = input ( ' \ n Valor mínimo en la escala: ' );
                tmax = input ( ' Valor máximo en la escala: ' );
% P3 = escalamiento (TEMPO, tmin, tmax);
                fprintf ( ' \ n \ n Escalando.... \ n \ n ' );
                A = TEMPO;
                [M, N] = size(A);
                V = [min(A); max(A)];
                Units = ones(M, N);
```

```
            P1 = A – units (1:M, 1:1) * V(1:1, 1:N);
        for l = 1:M;
        for k = 1:N;
            if V(2,k) - V(1,k) ~= 0;
            P1 (l, k) = (P1 (l, k)) / ( V(2, k) – V(1, k) );
            P1 (l, k) = tmin + P1(l, k) * (tmax – tmin);
        end;
end;
end;
        clear A;
        clear TEMPO;
        TEMPO = P1;
        clear P1;
        if type_data == 1
            Out = TEMPO;
            save Output.mat Out
        else
            Inp = TEMPO;
            save Input.mat Inp
        end
            neural0b;
end
```

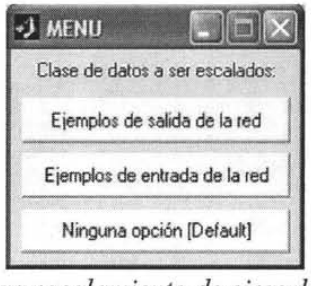


Figura B.8 *Submenú para escalamiento de ejemplos de entrada y/o salida*

Neural2b.m

```
% CÓDIGO BACKPROPAGATION. (neural2)
% ESCALAMIENTO DE EJEMPLOS DE SALIDA Y/O ENTRADA
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
clc
        fprintf( ' Escalamiento de los datos \ n \ n ' )
        type_data = menú ( ' Clase de datos a ser escalados: ' ,...
                           ' Ejemplos de salida de la red',...
                           ' Ejemplos de entrada de la red',...
                           ' Ninguna opción [Default]');
disp('')
        if type_data == 1
            load TestOut;
            TEMPO = Out;
        elseif type_data == 2
            load TestInp;
            TEMPO = Inp;
        else
            return
end
        flag1 = input ( ' Quiere hacer escalamiento( 0 si / 1 no): ' );
        if flag1 == 0
            fprintf (' \ n \ n Rango de escalamiento...' );
```

```
                tmin = input ( ' \ n Valor mínimo en la escala: ' );
                tmax = input ( ' Valor máximo en la escala: ' );
% P3 = escalamiento ( TEMPO, tmin, tmax);
                fprintf ( ' \ n \ n Escalando.... \ n \ n ' );
                A = TEMPO;
                [M, N] = size(A);
                V = [min(A); max(A)];
                Units = ones (M,N);
                P1 = A – units(1:M, 1:1) * V(1:1, 1:N);
        for l = 1:M;
        for k = 1:N;
                if V(2, k) – V(1, k) ~= 0;
                P1 (l, k) = (P1 (l, k)) / (V(2, k) - V(1, k) );
                P1 (l, k) = tmin + P1(l,k) * (tmax - tmin);
        end;
end;
end;
        clear A;
        clear TEMPO;
        TEMPO = P1;
        clear P1;
        if type_data == 1
                Out = TEMPO;
                save TestOut.mat Out
        else
                Inp = TEMPO;
                save TestInp.mat Inp
        end
                neural0b;
end
```

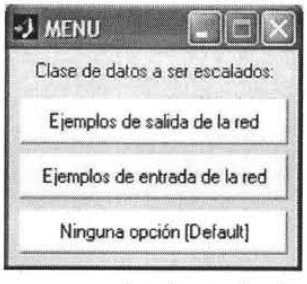


Figura B.9 *Submenú para escalamiento de ejemplos de entrada y/o salida*

Neural3.m

```
% CÓDIGO BACKPROPAGATION
% MATRIZ PARA BACKPROPAGATION
clc;
        fprintf ( ' \ n \ n ENTRENAMIENTO mediante Backpropagation \ n \ n ')
        fprintf ( ' \ n \ n Modelo de red neuronal basado en tres capas \ n \ n ')
        load Output.mat;
        load Input.mat;
        T1 = ' logsig ';
        T2 = ' tansig ';
%
% DEFINING FINAL INPUT PATTERNS AND OUTPUTS
%
        [M, N] = size ( Inp ' );
        P =Inp ';
        T = Out ';
```

```
%
% PATRÓN DE ENTRADA Y SALIDA
%
% M ES EL NÚMERO DE COMPONENTES DE CADA PATRÓN DE ENTRADA
number_inputs = M;
number_patterns = N;
%
% INICIANDO LA ARQUITECTURA DE LA RED NEURONAL
%
% COLOCANDO EL TAMAÑO DEL VECTOR DE ENTRADA R, TAMAÑO DE LAS CAPAS
S1 & S2, TAMAÑO DE LA PARTIDA Q.
fprintf ( ' \ n Inicialización. Espere.....' )
[R, C] = size ( P );
[R1, C1] = size ( T );
% S1: numero de neuronas ocultas (Este valor puede ser cambiado hasta lograr un entrenamiento adecuado)
S1 = fix ( ( R + R1 ) / 2 );
S2 = R1;
IN =menu ( 'Seleccione el proceso: ',...
            ' Iniciando el proceso de la red',...
            ' Reprocesando la red');
disp('')
```

Figura B.9 *Submenú para selección de proceso de la red*

```
fprintf ( ' \ n Cargando los pesos..... \ n ' );
        if IN == 1
                % RANDS Generador simétrico aleatorio
                % W1 = rands (S1, R);
                % B1 = rands (S1, 1);
                % W2 = rands (S2, S1);
                % B2 = rands (S2, 1);
                % NWLOG  Nguyen-Widrow Generador aleatorio para neuronas tipo LOGSIG.
                [W1, B1] = nwlog (S1, R);
                [W2, B2] = nwlog (S2, S1);
        else
                load whgts.mat
% La siguiente línea es utilizada cuando se quiere usar los pesos anteriores
                W1 = NW1; B1 = NB1; W2 = NW2; B2 =NB2;
end
%
% ENTRENAMIENTO DE LA RED
%
% PARÁMETROS DE ENTRENAMIENTO
% PARÁMETROS:
% ERROR DESEABLE: error_goal
% NÚMERO MÁXIMO DE ITERACIONES: max_epochs
% TAZA DE APRENDIZAJE: lr
% LOS SIGUIENTES PARÁMETROS PUEDEN SER CAMBIADOS
disp_freq =10;
max_epochs = 200;
```

```
error_goal = 0.003;
lr = 0.01;
lr_inc = 1.05;
lr_dec = 0.7;
mom_const = 0.95;
err_ratio = 1.04;
F1 =' logsig ';
F2 = ' logsig ';
% MOSTRANDO LOS PARÁMETROS UTILIZADOS POR DEFINICIÓN
fprintf( ' \ n PARAMETROS PRESELECCIONADOS: \ n ' )
fprintf ( ' Frecuencia de resultados:  %d \ n Número máximo de pasos (epochs): % d \ n' , disp_freq,
max_epochs )
fprintf ( ' Error permitido: %f \n Tasa de Aprendizaje: %f\n',error_goal,lr)
fprintf ( ' Incremento del aprendizaje: %f \ n Momento: % f \ n', lr_inc, mom_const )
fprintf ( ' Función de activación para la capa oculta: %s \ n' ,F1 )
fprintf ( ' Función de activación para la capa de salida: %s \ n', F2 )
fprintf ( ' El número de nodos en la capa oculta sugerido es: %d \ n', S1 )
par = menu ( ' Selección de los parámetros de entonación de la red ',...
            ' Fijar nuevos parámetros ',...
            ' Establecer los parámetros sugeridos');
disp('');
        if par == 1
        % INTRODUCIR LOS NUEVOS PARÁMETROS
        S1 = input (' Número de nodos ocultos ');
        f size (S1) == [0,0]
                S1 = fix((R+R1)/2);
        end
disp_freq = input ( ' Frecuencia de la presentación de resultados: ' );
        if size ( disp_freq ) == [0,0]
                disp_freq = 10;
        end
max_epochs = input ( ' Máximo número de pasos(epochs): ' );
        if size (max_epochs) == [0,0]
                max_epochs = 200;
        end
error_goal = input ( ' Error permitido de convergencia: ' );
        if size (error_goal) == [0,0]
                error_goal = 0.003;
        end
lr = input ( ' Tasa de aprendizaje: ' );
        if size (lr) == [0,0]
                lr = 0.01;
        end
lr_inc = input ( ' Incremento de la tasa de aprendizaje: ' );
        if size (lr_inc) == [0,0]
                lr_inc = 1.05;
        end
mom_const = input ( ' Nuevo momento: ' );
        if size (mom_const) == [0,0]
                mom_const = 0.95;
        end
fprintf ( ' Tipo de función de activación: tansig (-1, 1), logsig (0, 1) \ n ' );
        F1 = input ( ' Función de activación para la capa oculta: ' , ' s ' );
                if F1 ~= T1 | F1 ~= T2
                F1 = ' logsig ';
        end
```

```
        F2 = input ( ' Función de activación para la capa de salida: ' , ' s ' );
                if F2 ~= T1 | F2 ~= T2
                F2 = ' logsig ';
        end
end
%
% NOTA: EL RESTO DEL CÓDIGO DE ENTRENAMIENTO PUEDE SER REMPLAZADO
MEDIANTE LA FUNCIÓN:
% [ W1, B1, W2, B2, epoch, TR ] = trainbp (W10, B10, 'tansig', ...W20, B20, ' purelin ', P, T, TP );
% FASE DE APRENDIZAJE
fprintf ( ' \ n \ n ENTRENADO LA RED. Espere.....\ n \ n ' )
figure;
TP = [disp_freq max_epochs error_goal lr lr_inc lr_dec mom_const err_ratio];
[ NW1, NB1, NW2, NB2, epoch,error ] = ...
        trainbpx (W1, B1, F1, W2, B2, F2, P, T, TP );
        Actual = logsig ( W2 * logsig ( W1 * P, B1 ), B2 );
%
% GRÁFICA DEL COMPORTAMIENTO DE LA CURVA DE ERROR
%
figure;
        ploterr(error);
        pause
        fprintf( ' \ n \ n Convergence. Wait.....\ n \ n ' )
        save whgts.mat NW1 NB1 NW2 NB2;
        neural0;
```

Neural4.m

```
% CÓDIGO BACKPROPAGATION. (neural4)
% USANDO LAS MATRICES ouput.mat E input.mat PARA PROBAR LA RED
%
        fprintf ( ' \ n \ n ENTRENAMIENTO mediante Backpropagation \ n \ n ' )
        fprintf ( ' \ n \ n Prueba del modelo de red neuronal \ n \ n ' )
        load TestOut;
        load TestInp;
%
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
        [ M, N ] = size( Inp' );
%
% PATRONES DE ENTRADA Y SALIDA
% M ES EL NUMERO DE COMPONENTES EN CADA PATRÓN DE ENTRADA
%
        number_inputs = M;
        number_patterns = N;
%
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
        P = Inp' ;
        T = Out' ;
        clear Inp; clear Out;
%
% PATRONES DE ENTRADA Y SALIDA
%
        fprintf ( ' \ n \ n Testing the network. Wait..... \ n \ n ' )
        fprintf ( ' Desired Actual Squared \ n ' )
```

```
        fprintf ( ' Output Output Error \ n ' )
%
% INICIANDO LA ARQUITECTURA DE LA RED
% LA SIGUIENTE LÍNEA ES UTILIZADA CUANDO SE QUIEREN USAR LOS PESOS ANTERIORES
%
        load whgts.mat
        W1 = NW1; B1 = NB1; W2 = NW2; B2 = NB2;
%
% PROBANDO LA RED
%
        SSE = sumsqr ( T - logsig (W2 * logsig ( W1 * P, B1 ), B2 ) );
        Actual = logsig ( W2 * logsig ( W1 * P, B1 ), B2 );
        fprintf ( ' %10.5f %10.5f %10.5f \ n ', T, Actual, SSE)
        fprintf ( ' \ n \ n End Testing..... \ n \ n ' )
        figure;
                plot(T), hold, plot(Actual, 'r' );
        fprintf ( ' \ n \ n Saving Results ( Target output, Actual output, Sum Squared Error ) ' );
        save results.mat T Actual SSE;
```

Neural5.m

```
% CÓDIGO BACKPROPAGATION. (neural5)
% GUARDANDO EJEMPLOS DE SALIDA Y ENTRADA COMO INFORMACIÓN PARA
% ENTRENAMIENTO O PRUEBA DE LA RED
% DEFINIENDO LOS PATRONES FINALES DE ENTRADA Y SALIDA
%
clc
        fprintf ( ' Mantenimiento de Datos \ n \ n ' )
        type_data1 = menu ( ' Tipo de datos a respaldar',...
                            ' Datos de entrenamiento',...
                            ' Datos de prueba',...
                            ' Modelo de la red neural entrenada',...
                            ' Ninguna opción [Default] ');
disp('')
        if type_data1 == 1
                type_data = menu ( ' Datos de entrenamiento a respaldar ',...
                                   ' Ejemplos de salida ',...
                                   ' Ejemplos de entrada ',...
                                   ' Ambos ',...
                                   ' Ninguna opción [ Default ] ' );
        elseif type_data1 == 2
                type_data = menú ( ' Datos de prueba a respaldar ',...
                                   ' Ejemplos de salida ',...
                                   ' Ejemplos de entrada ',...
                                   ' Ambos ',...
                                   ' Ninguna opción [ Default ] ' );
        elseif type_data1 == 3
                fprintf ( ' La red neuronal respaldada en net.mat contiene la matriz \ n ')
                fprintf ( ' de pesos sinápticos de las capas oculta y de salida \ n ')
                load whgts.mat
% LA SIGUIENTE LÍNEA ES UTILIZADA CUANDO SE QUIEREN UTILIZAR LOS PESOS ANTERIORES
                save net.mat NW1 NB1 NW2 NB2;
                type_data = 4;
        neural0
```

```
end
disp('')
        if type_data == 1
        if type_data1 == 1
                load Output;
                save Train.mat Out
                neural0
        elseif type_data1 == 2
                load TestOut;
                save Test.mat Out
                neural0
end
        elseif type_data == 2
        if type_data1 == 1
                load Input;
                save Train.mat Inp
                neural0
        elseif type_data1 == 2
                load TestInp;
                save Test.mat Out Inp
                neural0
end
        elseif type_data == 3
        if type_data1 == 1
                load Input;
                load Output;
                save Train.mat Out Inp
                neural0
        elseif type_data1 == 2
                load TestInp;
                load TestOut;
                save Test.mat Out Inp
                neural0
end
else
        neural0;
end;
```