DETECCIÓN AUTOMÁTICA DE REGISTROS SÍSMICOS ASOCIADOS AL COMPORTAMIENTO DEL VOLCÁN GALERAS HACIENDO USO DE REDES NEURONALES ARTIFICIALES

DARÍO FERNANDO ARCOS GUERRERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE FÍSICA
PROGRAMA DE FÍSICA
SAN JUAN DE PASTO
2008

DETECCIÓN AUTOMÁTICA DE REGISTROS SÍSMICOS ASOCIADOS AL COMPORTAMIENTO DEL VOLCÁN GALERAS HACIENDO USO DE REDES NEURONALES ARTIFICIALES

DARÍO FERNANDO ARCOS GUERRERO

Trabajo presentado como requisito previo
Para optar por el Titulo de Físico

Director
Físico OSCAR ERNESTO CADENA IBARRA

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE FÍSICA
PROGRAMA DE FÍSICA
SAN JUAN DE PASTO
2008

“Las ideas y conclusiones aportadas en este trabajo de grado, son responsabilidad exclusiva de los autores”.

Artículo Primero del Acuerdo No. 324 de Octubre 11 de 1966 del Honorable Consejo Directivo de La Universidad de Nariño.

Nota de aceptación

_____
_____
_____
_____

_____
Director

_____
Jurado

_____
Jurado

San Juan de Pasto, Octubre de 2008.

DEDICATORIA

Con mucho cariño a mis padres, que siempre me apoyaron. Es un gran logro para ellos el verme graduado como profesional.
A mi hermano que siempre me decía “si se puede”, ayudándome y dándome fuerzas para continuar.

AGRADECIMIENTOS

De manera muy sincera expreso mis agradecimientos a:

Oscar E. Cadena, primero como un gran y buen amigo y compañero de trabajo y además quien me dirigió por el camino para graduarme e invirtió tiempo colaborándome como mi asesor de trabajo, agradezco de todo corazón, gracias amigo.

Ingeniero Diego Mauricio Gómez Martínez, Coordinador del OVSP y a toda la gran familia que conforma el OVSP, quienes siempre me brindaron total colaboración, desde el inicio hasta el final y me acogieron como parte de su grupo de trabajo, a todos ellos gracias por su interés, constante apoyo y valiosas orientaciones.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

Emulando algunas de las capacidades fundamentales de la neurona biológica, tales como, su capacidad de aprender a partir de ejemplos y proporcionar una respuesta de lo aprendido en un entorno, como apoyo a las labores de monitoreo volcánico se aplica una herramienta basada en inteligencia computacional utilizando Redes Neuronales Artificiales (RNA), particularmente el modelo de red tipo Perceptron Multicapa. Esta aplicación permitirá la detección automática de eventos sísmicos mediante la diferenciación entre patrones asociados a episodios sísmicos y ruido para las estaciones Anganoy (ANGV) y Cráter-2 (CR2R), estaciones base de clasificación de la sismicidad originada en Galeras. Empleando la técnica ensayo-error se determina una arquitectura de red neuronal artificial tanto para ANGV como para CR2R que optimice la cantidad de aciertos en su labor de detección. La topología de la RNA que mostró muy buenos resultados fue de tres capas: una de entrada, una oculta y una de salida, cuyo conexionado neuronal fue: 12-3-1 para ANGV y 12-3-1 para CR2R. Los entrenamientos o fase de aprendizaje y prueba se llevaron a cabo con información seleccionada de la base de datos del Observatorio Vulcanológico y Sismológico de Pasto (OVSP). Además se hizo uso de los Coeficientes de Predicción Lineal (LPC) como técnica de reducción de dimensionalidad, debido a que el tamaño de los registros contienen un elevado número de muestras, por tanto, con la ayuda de 12 LPC se representa 300 muestras equivalentes a 3 segundos de registro. Se seleccionaron 500 eventos y 500 registros de ruido tanto para ANGV como para CR2R. Las rutinas para el tratamiento de los registros y manipulación de los mismos se desarrollaron en el programa MATLAB, además de la simulación de las RNA.

ABSTRACT

Emulating some of the fundamental capacities of the biological neuron, such as, its capacity to learn from examples and to provide an answer according to the acquire knowledge in an environment, to support the works of volcanic monitoring a tool is applied based on Artificial Intelligence using Artificial Neuronal Network (ANN), particularly the pattern of net type multi-layer Perceptron. This application will allow the automatic detection of seismic events using the deferens between patterns associated to seismic episodes and noise for the stations Anganoy (ANGV) and Crater-2 (CR2R), stations base of classification to the GALERAS seismic. Using the technical rehearsal-error an architecture of Artificial Neuronal Network is determined as much for ANGV as for CR2R to optimise the quantity of successes in their detection work. Very good results were showed by the three layers model: one of entrance, a hidden one, and one of exit, whose neuronal connection was: 12-3-1 for ANGV and 12-3-1 for CR2R. The trainings or learning phase and it proves they were carried out with selected information from the database of the Observatorio Vulcanológico y Sismológico de Pasto (OVSP). Also the LINEAR PREDICTION COEFFICIENTS (LPC) were used as technique of dimensional reduction, because the size of the records contains a high number of samples, therefore, with the help of 12 LPC it is represented 300 equivalent samples to 3 seconds of record. 500 events and 500 helicorders of noise were selected as much for ANGV as for CR2R. The routines for the treatment of the records and theirs manipulation developed in MATLAB program, and also the simulation of the ANN.

INTRODUCCIÓN

El problema que se aborda en el presente trabajo es distinguir manifestaciones de fuentes de sismicidad asociadas con procesos físicos en el interior del sistema volcánico, de señales aleatorias en el tiempo (ruido) no asociadas al origen de la sismicidad, este aporte a la solución del problema contribuye en la generación de modelos de comportamiento volcánico desde el punto de vista de la dinámica cambiante del sistema.

El cerebro, cumbre de la evolución biológica, es un gran procesador de información; entre algunas de sus características se destaca, que es capaz de procesar a gran velocidad grandes cantidades de información procedentes de un entorno, combinarla o compararla con información almacenada y dar respuestas. Además su capacidad para aprender a representar la información necesaria para desarrollar tales habilidades, sin instrucciones explícitas para ello. Se han desarrollado algunos modelos matemáticos que tratan de simular su comportamiento. Estos modelos se han basado sobre las características esenciales de las neuronas y sus conexiones.

Nuestro elemento esencial para el desarrollo del presente trabajo es la neurona biológica, emulando su capacidad de aprender y asociar patrones similares se pretende afrontar un problema de difícil solución como es la detección automática de eventos sísmicos en una traza. Cada neurona se representa como una unidad de proceso y control, las cuales se organizan en capas y estas a su vez constituyen la red neuronal. Las neuronas y capas se conectan entre sí a través de sinapsis, siendo este conexionado de tipo paralelo. Los Sistemas Neuronales Artificiales imitan la estructura esquemática del sistema nervioso, con la intención de construir sistemas de procesamiento de información paralelos, distribuidos y adaptativos; paralelismo de cálculo, memoria distribuida y adaptabilidad al entorno junto con interfaces de entrada y salida.

Con el objeto de apoyar las labores de monitoreo volcánico se aplica una herramienta computacional encargada de la detección automática de registros sísmicos asociados a la actividad y comportamiento del volcán Galeras haciendo uso de Redes Neuronales Artificiales. Se realiza un reconocimiento y diferenciación entre patrones que representan señales de ruido y registros sísmicos (eventos) que tienen como fuente el volcán Galeras. Haciendo uso del programa MATLAB simulamos la estructura funcional de la red neuronal biológica reproduciendo algunas de sus capacidades. Para problemas de reconocimiento de patrones, el modelo de red neuronal artificial que arroja mejores resultados se conoce como Perceptron Multicapa (MLP) que se entrena con ayuda del algoritmo

de aprendizaje Backpropagation BP o alguna de sus variantes. Se observó que los mejores resultados se presentan cuando la arquitectura del modelo se ejecuta con una sola capa oculta. Se utilizó la técnica ensayo-error para determinar una arquitectura de red que se ajuste a la relación entrada-salida con base en la minimización de una función de coste, es decir, se busca un conexionado de red que responda con el mayor número de aciertos cuando se evalué dicha arquitectura de red.

Para el entrenamiento se hace uso de información obtenida por el OVSP, particularmente señales registradas por los sismómetros de las estaciones Anganoy (ANGV) y Cráter-2 (CR2R). Debido a que cada registro cuenta con un alto número de muestras (cien muestras por segundo) utilizamos los Coeficientes de Predicción Lineal (LPC) como técnica de reducción de dimensionalidad con la que se representan ventanas de 3 segundos de duración (300 muestras) con 12 coeficientes, sin mayor pérdida de información.

Cabe destacar que el sistema se puede re-entrenar con nuevos ejemplos asociados a un periodo de tiempo en el que las características del ruido son diferentes o entrenar una red neuronal artificial que logre diferenciar señales particulares y así realizar digamos, un filtro, por ejemplo para discriminar señales de interferencia o radio, no asociadas a la sismicidad; para determinar el orden de arribo de un paquete de ondas o búsqueda de señales; además podríamos evaluar periodos de tiempo, una evaluación cuantitativa de eventos y así comparar con información del mismo tipo con otros periodos de tiempo, evidenciando la capacidad adaptativa al entorno de la red neuronal.

ESTADO DEL ARTE

Buscando aplicaciones de Redes Neuronales Artificiales enfocadas al área de Geofísica, y entorno a la temática destacaremos las siguientes:

En Caracas, Venezuela, en septiembre de año 2000; el trabajo realizado por Carlos Barraéz, encargado de la instrumentación de la Red Sismológica de los Andes Venezolanos llamado [11] “DETECTOR DE EVENTOS SÍSMICOS EN TIEMPO REAL UTILIZANDO REDES NEURONALES ARTIFICIALES” adoptando el algoritmo backpropagation, concluyó que la detección no depende del nivel de ruido, utilizándolo como herramienta para detección de eventos falsos, relacionados con señales de radio e interferencias, con efectividad de casi 95 % y detección de microsismos.

Aplicando este tipo de tecnología se efectúo un trabajo en el departamento de Geología de la Universidad de Wisconsin-Madison por Haijiang Zhang para detectar automáticamente los arribos de las ondas P alcanzando un 90 % de efectividad.

El trabajo titulado [13] “EXTRACTION OF INFORMATION FROM SEISMOGRAMS BY NEURAL NETWORKS”, desarrollado por Janusz NIEWIADOMSKI, Institute of Geophysics, Polish Academy of Sciences Warszawa, Poland, en el año 2004, mediante el proceso de detección automática aplicada a los sismogramas extrae sismos regionales capturando tan solo, lo que ellos llaman el momento sísmico y primero calculan la magnitud del evento sísmico y luego mediante el orden de arribo en las estaciones, su localización.

En el estudio de los volcanes Italianos se utiliza para la correcta identificación de eventos sísmicos obteniendo resultados del 100 %, trabajo titulado “AUTOMATIC ANALYSIS OF SEISMIC DATA BY USING NEURAL NETWORKS: APPLICATIONS TO ITALIAN VOLCANOES”, efectuado en el año 2007, trabajando con señales de los volcanes Monte Vesubio y Stromboli.

Además según Journal of Intelligent Material Systems and Structures, en uno de sus artículos [15] publicados en el año 2002, comenta que se utiliza para la detección de eventos falsos asociados a interferencias y señales de radio.

[14] El trabajo desarrollado por Jim Wang y Ta-liang Teng, Department of Earth Sciences University of Southern California, Los Angeles, se basa en la identificación y selección automática de fases tipo S, para eventos locales y regionales, mostrando resultados de 86 % de efectividad en la identificación y 74

% en la selección de ondas S usando redes neuronales. En general se adoptó para la detección de microsismos.

[12] “DETECCIÓN AUTOMÁTICA DE ARRIBOS DE ONDAS P Y S”, trabajo realizado en el año 1995 por Hengchang Dai y Colin MacBeth, Department of Geology and Geophysics, University of Edinburgh, entrena la red con nueve grupos de arribos P, S y ruido; y comparando con el análisis manual obtuvieron identificación correcta de ondas P 84 % y de ondas S 63 %.

En The Institute of Seismology, University of Helsinki, Finland, Timo Tiira en junio de año 2000, usando el tipo de red neuronal Perceptron Multicapa, con una capa oculta desarrollo un detector automático de telesismos entrenando la red con 193 eventos telesismos, encontraron 25% más eventos que Reviewed Event Bulletins (REB) of the International Data Center in Finland.

A nivel de Colombia Carlos Álzate, Germán Castellanos, Grupo de Control y Procesamiento Digital de Señales Universidad Nacional de Colombia Sede Manizales, desarrollaron un detector en línea del tiempo de arribo de ondas P sobre registros electrónicos de tres componentes de eventos tectónicos. Los registros procesados se obtuvieron de la base de datos del Instituto IRIS (Incorporated Research Institutions for Seismology) y de Ingeominas (Instituto Colombiano de Geología y Minería) y su análisis dan como resultado un mejor desempeño del sistema en la detección del arribo, aunque no presenta una exactitud aceptable en la estimación del tiempo de arribo de la onda P.

En Nariño, el trabajo desarrollado por Oscar E. Cadena, Universidad de Nariño, Departamento de Física, en noviembre de año 2006, llamado [9] “DISCRIMINACIÓN ENTRE REGISTROS SÍSMICOS TIPO LPS Y VTA PRODUCIDOS POR EL VOLCÁN GALERAS UTILIZANDO REDES NEURONALES”, realiza una clasificación entre estos dos tipo de eventos sísmicos obteniendo resultados para la estación ANGV (Anganoy) de 97.67 % de aciertos; haciendo uso del modelo de red neuronal artificial Perceptron Multicapa cuya configuración es 99-3-1.

1. OBJETIVOS

1.1. OBJETIVOS GENERALES

Haciendo uso de una Red Neuronal Artificial, modelo Perceptron Multicapa, realizar la diferenciación automática entre registros sísmicos (eventos) y ruido de las estaciones de corto periodo Anganoy y Cráter-2 las cuales hacen parte de la red de estaciones que monitorean constantemente el volcán Galeras.

1.2. OBJETIVOS ESPECÍFICOS

a) Reducir la dimensionalidad de los registros sísmicos y ruido, representándolos con ayuda de la técnica de Coeficientes de Predicción Lineal (LPC).

b) Mediante la realización de entrenamientos, determinar una arquitectura del modelo de red neuronal artificial Perceptron Multicapa que optimice la cantidad de aciertos en la detección automática de eventos sísmicos para la estación Anganoy.

c) Mediante la realización de entrenamientos, determinar una arquitectura del modelo de red neuronal artificial Perceptron Multicapa que optimice la cantidad de aciertos en la detección automática de eventos sísmicos para la estación Cráter-2.

2. MARCO CONCEPTUAL

2.1. COEFICIENTES DE PREDICCIÓN LINEAL (LPC). [5] Su fundamento se basa en establecer un modelo de filtro del tipo todo polo para la señal fuente. Un sistema de predicción lineal es un procedimiento que, dada una señal, permite definir la función de transferencia del filtro que la ha generado. [10] Los polos hacen referencia a los valores que hacen nulo el denominador de la función de transferencia Hz. Por tanto un filtro tipo todo polo será aquel donde los ceros del filtro se encuentran en el denominador de la función de transferencia.

El modelo construye una aproximación a la estructura espectral de la señal. El método recibe este nombre porque pretende extrapolar el valor de la siguiente muestra de seña $\tilde{s}(n)$ como la suma ponderada de muestras pasadas s(n-1); s(n-2); … s(n-p):

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k S(n-k), \tag{2.1}$$

Donde { α_K } son los coeficientes de predicción lineal y el número p indica el número de ejemplos pasados considerados y también el orden de la predicción lineal. El error generado por esta estimación es:

(2($$s(n) - \tilde{s}(n) = e(n)$$ (2.2)

Donde s(n) es la señal real y e(n) el error de predicción lineal. Así:

$$e(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \tag{2.3}$$

Tomando transformada Z (TZ) tendremos:

$$E(z) = TZ\left[s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)\right] = \tag{2.4}$$

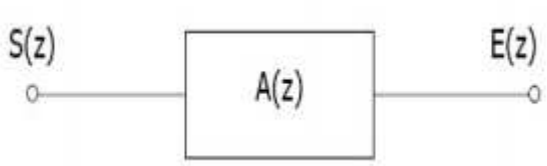
$$= S(z) - \sum_{k=1}^{p} \alpha_k S(z).z^{-k} = \tag{2.5}$$

$$= S(z).\left[1 - \sum_{k=1}^{p} \alpha_k.z^{-k}\right] = \tag{2.6}$$

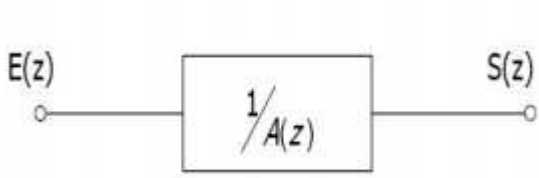
Si denominamos A(z) a la expresión:

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} \alpha_k.z^{-k} \tag{2.7}$$

Es decir, A(z), al que denominaremos filtro inverso, será la función de transferencia del sistema:



Además el sistema inverso al anterior será:



Excitando con el error de predicción un sistema cuya función de transferencia sea 1/A(z) obtendremos a la salida la señal s(z).

Sea H(z) la forma de un modelo de filtro todo polo, con q polos definido por:

$$H(z) = \frac{1}{1 - \sum_{k=1}^{q} a_k . z^{-k}} \tag{2.8}$$

El número de polos del modelo es igual al orden de predicción lineal, p = q, tendremos:

$$\{a_k\} = \{\alpha_k\} \tag{2.9}$$

$$H(z) = \frac{G}{A(z)} \tag{2.10}$$

Donde G es la ganancia del filtro.

El siguiente paso es determinar los coeficientes del filtro:

$H(z):\{a_K\} \quad 1 \leq k \leq p,$ coeficientes de predicción a partir de la señal original, de modo que el filtro equivalente H(z) se ajuste a la envolvente espectral en cada ventana de análisis. Por consiguiente, para encontrar un conjunto de coeficientes {ak} que minimicen el error de predicción cuadrático medio en cada trama de análisis n procedemos:

Definido el error cuadrático medio como:

$$E_n = \sum_m e_n^2(m) \tag{2.11}$$

Donde m representa la longitud de la ventana de análisis. Por consiguiente, desarrollando el error de predicción lineal tenemos:

$$E_n = \sum_m (s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k))^2 \tag{2.12}$$

Con el objeto de minimizar el error de predicción respecto al conjunto {ak}, tendremos que derivar E_n parcialmente respecto a cada coeficiente a_k e igualar a

cero, esto es:

$$\frac{\partial E_n}{\partial a_k} = 0 \tag{2.13}$$

para k = 1; 2…p, tenemos p ecuaciones con p variables desconocidas,
Resultando así:

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a_k} \sum_m s_n(m-i)s_n(m-k), \tag{2.14}$$

Donde los $\{\hat{a_k}\}$ serán los coeficientes de predicción óptimos y sea:

$$\Phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k), \tag{2.15}$$

el sistema de ecuaciones lineales a resolver para $1 \leq i \leq p, 0 \leq k \leq p$.

2.1.1. Método de autocorrelación. [10] Para la resolución del sistema de ecuaciones lineales planteado emplearemos el método de autocorrelación, la función de autocorrelación resulta de gran utilidad para encontrar patrones repetitivos dentro de una señal, como por ejemplo, la periodicidad o para identificar la frecuencia. En el procesado de señales, la función de autocorrelación se define como la correlación continua cruzada de la señal consigo mismo tras un desfase, siendo la correlación cruzada una medida de la similitud entre dos señales, frecuentemente usada para encontrar características relevantes en una señal desconocida por medio de la comparación con otra que sí se conoce, puesto que el fin es encontrar la información más representante de la señal. Formalmente, la autocorrelación R con un desfase j para una señal x_n es:

$$R(j) = \sum_n (x_n - d)(x_{n-j} - d), \tag{2.16}$$

donde d es el valor esperado de x_n.
Frecuentemente las autocorrelaciones se calculan para señales centradas

alrededor del cero, es decir con un valor principal de cero. En ese caso la definicion de la autocorrelación viene dada por:

$$R(j) = \sum_n x_n x_{n-j}, \tag{2.17}$$

[3] Antes de calcular los LPC se debe dividir la señal $s_n(m)$ en pequeños segmentos sucesivos que llamaremos ventanas. Con el fin de evitar que cortes bruscos de inicio y final de la señal introduzcan componentes de alta frecuencia en el espectro, se somete a los segmentos al proceso de ventaneo, que consiste en multiplicar el segmento por una función de ventana w(m) de longitud fija. La ventana más simple es la ventana rectangular de largo Nw :

$$w(m)=\begin{cases} 1, & 0 \le m \le N_w - 1, \\ 0, & \text{para los demàs} \end{cases}$$

La ventana tipo rectangular causa cambios abruptos en el dominio del tiempo además irregularidades en el dominio de la frecuencia y para prevenir estas distorsiones generalmente se usa la ventana de Hamming, definida por:

$$w(m)=\begin{cases} 0{,}54 - 0{,}46cos(\frac{2\pi n}{N_w - 1}), & 0 \le m \le N_w - 1, \\ 0, & \text{para los demàs} \end{cases}$$

[5] Supongamos que la señal $s_n(m)$ no se anula en el intervalo $0 \le m \le N_w - 1$ esto equivale a asumir que la señal $s_n(m)$ ha sido multiplicada en el tiempo por una ventana w(m) que vale cero fuera del intervalo $0 \le m \le N_w - 1$. De este modo sí $s_n(m) = s(n + m)$, podemos expresar la señal como:

$$s(m)=\begin{cases} s(n+m).w(m), & 0 \le m \le N_w - 1, \\ 0, & \text{resto.} \end{cases}$$

Así, el error cuadrático medio será distinto de cero en el intervalo $0 \le m \le N_w$ -1+p, por lo que podremos expresarlo como:

$$E_n = \sum_{m=0}^{N_w-1+p} e_n^2(m) \tag{2.18}$$

De acuerdo a lo anterior la ecuación (15) la podemos escribir:

$$\Phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \quad \begin{cases} 1 \le i \le p \\ 0 \le k \le p \end{cases} \tag{2.19}$$

O también:

$$\Phi_n(i,k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k), \quad \begin{cases} 1 \le i \le p \\ 0 \le k \le p \end{cases} \tag{2.20}$$

De este modo la expresión anterior la podemos reducir a la expresión de la función de autocorrelación:

$$\Phi_n(i,k) = r_n(i-k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k), \quad \begin{cases} 1 \le i \le p \\ 0 \le k \le p \end{cases} \tag{2.21}$$

Y, puesto que la función de autocorrelación es simétrica, es decir $r_n(- k) = r_n(k)$, las ecuaciones se pueden expresar como $r_n(l\ i- k\ l)$, así la ecuación (14) la podemos escribir como:

$$\sum_{k=1}^{p} r_n(|i-k|).\widehat{a_k} = r_n(i), \quad 1 \le i \le p \tag{2.22}$$

expresión que en forma matricial resulta:

$$\begin{pmatrix} r_n(0) & r_n(1) & r_n(2) & \cdots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \cdots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \cdots & r_n(p-3) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \cdots & r(0) \end{pmatrix} \cdot \begin{pmatrix} \widehat{a_1} \\ \widehat{a_2} \\ \widehat{a_3} \\ \vdots \\ \widehat{a_p} \end{pmatrix} = \begin{pmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{pmatrix} \quad (2.23)$$

La matriz de autocorrelaciones, de dimensión p x p, es una matriz tipo Toeplitz o matriz de diagonales constantes (simétrica, con diagonales principal y secundarias de elementos iguales).

2.1.2. Solución recursiva: algoritmo de Levinson-Durbin. Debido al tipo de matriz resultante, este conjunto de ecuaciones se puede resolver de forma recursiva, mediante el método de Levinson-Durbin, a saber:

Inicializamos a:

$$E_0 = r_n(0), \quad a_0(i) = 0 \quad \forall i \quad (2.24)$$

Luego, de forma recursiva, para m = 1, 2,… p, tenemos:

$$k_m = \frac{r(m) - \sum_{i=1}^{m-1} a_{m-1}(i).r_n(m-i)}{E_{m-1}} \quad (2.25)$$

$$a_m(m) = k_m \quad (2.26)$$

$$a_m(j) = a_{m-1}(j) - k_m.a_{m-1}(m-j), \quad 1 \leq j \leq m-1 \quad (2.27)$$

Las soluciones parciales, para m ≤ p permitirán calcular los coeficientes óptimos del filtro H(z) de orden m. La solución final buscada, para m = p, dará como resultado los coeficientes óptimos del filtro de orden p, esto es:

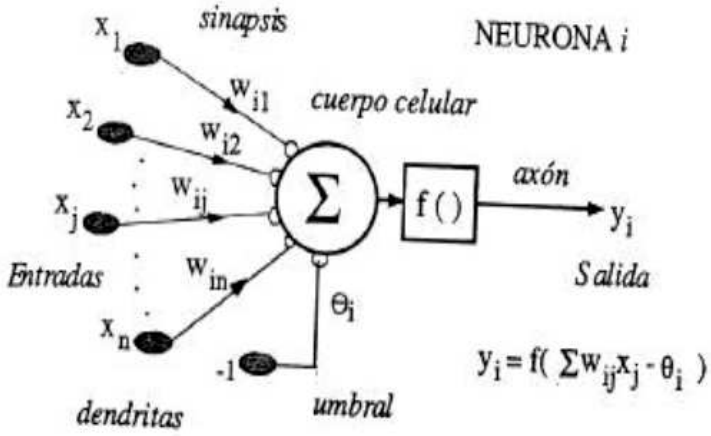
$$\widehat{a_i} = a_p(i), \quad 1 \leq i \leq p \tag{2.28}$$

[3] El orden de predicción p controla el número de polos con el que modelamos la envolvente espectral.
Como En (error cuadrático medio) nunca es negativo, esta condición garantiza que las raíces de A(z) estarán siempre dentro del círculo unidad. De esta manera el filtro H(z) = 1/A(z), será estable.

2.2. REDES NEURONALES ARTIFICIALES (RNA)

2.2.1. La neurona artificial. [1] Denominaremos procesador elemental o neurona a la unidad fundamental de red, como un dispositivo simple de cálculo, el cual a partir de un vector de entrada o de otras neuronas proporciona una única respuesta o salida. Los elementos que constituyen la neurona i son los siguientes:

Figura 2.1: Modelo de Neurona estándar.



● Conjunto de entradas x_j (t).

● Pesos sinápticos: de la neurona i,w_{ij} que representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i.

● Regla de propagación: σ(w_{ij}; x_j (t)), que proporciona el valor del potencial postsináptico h_i(t) = σ(w_{ij} ; x_j (t)) de la neurona i en función de sus pesos y entradas.

● Función de activación: $f_i(a_i\,(t\text{-}1), h_i\,(t))$, que proporciona el estado de activación actual $a_i\,(t)= f_i(a_i\,(t\text{-}1), h_i\,(t))$ de la neurona i, en función de su estado anterior $a_i\,(t\text{-}1)$ y de su potencial postsináptico actual.

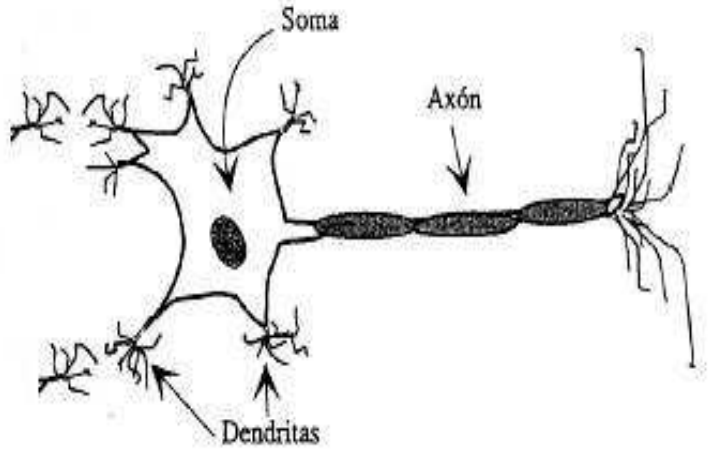
● Función de salida: $F_i\,(a_i\,(t))$, que proporciona la salida actual $y_i\,(t)= F_i\,(a_i\,(t))$, de la neurona i en función de su estado de activación.

La operación de la neurona i puede expresarse como:

$$y_i(t) = F_i(f_i[a_i(t-1), \sigma_i(w_{ij}, x_j(t))]) \qquad (2.29)$$

Este modelo general de neurona, se inspira en la operación de la biológica, en el sentido de integrar una serie de entradas y proporcionar una respuesta que se propaga por el axón. Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas), dependiendo del modelo y aplicación.

Figura 2.2: Estructura de una neurona biológica.



La regla de propagación permite obtener a partir de las entradas y los pesos, el valor del potencial postsináptico h_i de la neurona, la función más habitual es de tipo lineal, y se basa en la suma ponderada de las entradas con los pesos sinápticos:

$$h_i(t) = \sum_j w_{ij} x_j \qquad (2.30)$$

Formalmente interpretado como producto escalar de los vectores de entrada y pesos. La función de activación suele considerarse en la mayor parte de los modelos creciente y continua como se observa habitualmente en las neuronas biológicas. Algunas funciones de activación son de tipo escalón, identidad, lineal a tramos, sigmoidea, gaussiana, sinusoidal dependiendo de la respuesta que se desee. Para nuestro caso se requiere que la función de activación cumpla la condición de ser derivable, siendo las más empleadas en este sentido las de tipo sigmoideo, definida:

$$y = f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = tanh(x), \text{con } y \in [-1, 1] \quad (2.31)$$

2.2.2. Redes Neuronales Artificiales. Definiciones Básicas. Denominaremos arquitectura a la topología o estructura de conexionado de una red neuronal. En un sistema neuronal artificial los nodos se conectan por medio de sinapsis, esta estructura de conexiones determina el comportamiento de la red. Las conexiones son direccionales, propagándose la información en un único sentido desde la neurona presináptica a la postsináptica. Las neuronas suelen agruparse en unidades estructurales denominadas capas. Las neuronas de una capa pueden agruparse, a su vez, formando grupos neuronales. Así el conjunto de una o más capas constituye la red neuronal.

Distinguiéndose tres tipos de capas: de entrada, de salida y ocultas. Una capa de entrada o sensorial compuesta por neuronas que reciben datos o señales procedentes del entorno. Una capa de salida cuyas neuronas proporcionan la respuesta de la red. Capas ocultas, que no tienen conexión directa con el entorno proporcionando una mayor riqueza computacional. Las redes monocapa son aquellas compuestas por una única capa, las redes multicapa (layered networks) son aquellas cuyas neuronas se organizan en varias capas. Una definicion interesante de red neuronal hace uso del concepto matemático de grafo, objeto consistente en un conjunto de nodos (o vértices), más un conjunto de conexiones establecidas entre ellos. El grafo describe la arquitectura del sistema.

Una red neuronal es un grafo dirigido, es decir las conexiones tienen asignado un sentido; con las siguientes propiedades:

1.- A cada nodo i se asocia una variable de estado x_i.

2.- A cada conexión (i; j) de los nodos i y j se asocia un peso $w_{ij} \in R$.

3.- A cada nodo i se asocia un umbral θi.

4.- Para cada nodo i se define una función $f_i(x_i; w_{ij} ; \theta i)$, que depende de los pesos de sus conexiones, del umbral y de los estados de los nodos j a él conectados. Esta función proporciona el nuevo estado del nodo.

En otros términos los nodos son las neuronas y las conexiones son las sinapsis.

Modos de operación: recuerdo y aprendizaje. En los sistemas neuronales se distinguen dos modos de operación: el modo recuerdo o ejecución, y el modo aprendizaje o entrenamiento. Siendo el último de particular interés, ya que una característica fundamental de los sistemas neuronales es que se trata de sistemas entrenables, capaces de realizar un determinado procesamiento o cómputo aprendiéndolo a partir de un conjunto de patrones de aprendizaje o ejemplos.
Fase de aprendizaje. Aprendizaje o entrenamiento, proceso por el que se produce el ajuste de los parámetros libres de la red a partir de un proceso de estimulación por el entorno que rodea la red. Determinando un conjunto de pesos sinápticos que permita a la red realizar correctamente el procesamiento basado en los entrenamientos.

Cuando se construye un sistema neuronal, se parte de un cierto modelo de neurona y de una determinada arquitectura de red, estableciéndose los pesos sinápticos iniciales aleatorios. El entrenamiento o aprendizaje se lleva a cabo bajo el modelo de las sinapsis, que consiste en modificar los pesos sinápticos a partir de la optimización, midiendo la eficacia de la operación de la red. Sea w_{ij} el peso que conecta la neurona presináptica j con la postsináptica i en la iteración t, el algoritmo de aprendizaje, en función de las señales que en el instante t llegan procedentes del entorno, proporcionara el valor Δw_{ij} (t) que da la modificación que se debe incorporar en dicho peso, el cual quedará actualizado de la forma:

$$\triangle w_{ij}(t+1) = w_{ij}(t) + \triangle w_{ij}(t) \qquad (2.32)$$

El proceso de aprendizaje es iterativo, actualizándose los pesos una y otra vez, hasta que la red alcanza el rendimiento deseado.

Los dos tipos básicos de aprendizaje son el supervisado y el no supervisado, cuya distinción proviene en el campo del reconocimiento de patrones. Ambos pretenden estimar funciones entrada/salida, pero en el aprendizaje supervisado se proporciona cierta información sobre estas funciones (etiquetas de los patrones de entrada o salidas asociadas a cada patrón) y en el no supervisado o autoorganizado no se proporciona información alguna. Para nuestro caso nos centraremos en el aprendizaje supervisado por ser más exactos sus resultados, el cual puede definirse: Sea E [W] un funcional que representa el error esperado en

función de sus pesos sinápticos. En este tipo de aprendizaje se pretende estimar una cierta función desconocida: f: $R^n \rightarrow R^m$ (la que representa la red neuronal) a partir de muestras (x; y) (x Є R^n; y Є R^m) tomadas aleatoriamente, por medio de la minimización iterativa de E[W] mediante aproximación estocástica (estimando valores esperados a partir de cantidades aleatorias observadas).

En el aprendizaje supervisado se presenta a la red un conjunto de patrones, junto con la salida deseada u objetivo, e iterativamente ésta ajusta sus pesos hasta que su salida tiende a ser la deseada.

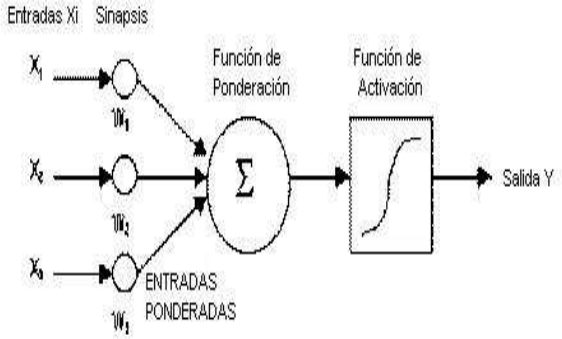
Fase de recuerdo o ejecución. Una vez que el sistema ha sido entrenado, se elige la mejor fase de aprendizaje y se desconecta, por lo que los pesos y la estructura quedan fijos, estando la red dispuesta para procesar datos.

En el proceso de entrenamiento es importante distinguir entre el nivel de error alcanzado al final de la fase de aprendizaje para el conjunto de datos de entrenamiento, y el error que la red ya entrenada comete ante patrones no utilizados en el aprendizaje, lo cual mide la capacidad de generalización de la red.

La neurona artificial como símil de la neurona biológica. [7] Las neuronas artificiales son modelos que tratan de simular el comportamiento de las neuronas biológicas. Cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, la red neuronal.

En la ilustración 2.3, dicha unidad de proceso consta de una serie de Entradas X_i, que equivalen a las dendritas de donde reciben la estimulación, ponderadas por unos pesos W_i, que representan como los impulsos entrantes son evaluados y se combinan con la función de red que nos dará el nivel de potencial de la neurona. La salida de la función de red es evaluada en la función de activación que da lugar a la salida de la unidad de proceso.

Figura 2.3: La neurona artificial se comporta como la neurona biológica.



Por las entradas Xi llegan unos valores que pueden ser enteros, reales o binarios. Estos valores equivalen a las señales que enviarían otras neuronas a la nuestra a través de las dendritas. Los pesos que hay en las sinapsis W_i, equivaldrían en la neurona biológica a los mecanismos que existen en las sinapsis para transmitir la señal. De forma que la unión de estos valores (X_i y W_i) equivalen a las señales químicas inhibitorias y excitadoras que se dan en las sinapsis y que inducen a la neurona a cambiar su comportamiento.
Estos valores son la entrada de la función de ponderación o red que convierte estos valores en uno solo llamado el potencial que en la neurona biológica equivaldría al total de las señales que le llegan a la neurona por sus dendritas. La función de ponderación suele ser una, la suma ponderada de las entradas y los pesos sinápticos.

La salida de función de ponderación llega a la función de activación que transforma este valor en otro en el dominio que trabajen las salidas de las neuronas. Suele ser una función no lineal como la función paso o sigmoidea aunque también se usa funciones lineales. El valor de salida cumpliría la función de la tasa de disparo en las neuronas biológicas.

Activación → tasa de disparo-función de activación → Salida

Dependiendo del modelo de RNA, de la arquitectura o topología de conexión y del algoritmo de aprendizaje, surgen varios modelos, que para nuestro caso el que mejor se adapta al objetivo es el Perceptron Multicapa por tener una alta aplicabilidad en problemas de reconocimiento de patrones.

2.2.3. El Perceptron Multicapa o MLP (Multi-Layer Perceptron). [1] El Perceptron Multicapa dentro del grupo de modelos de redes neuronales, se clasifica como tipo unidireccional, supervisado cuya arquitectura consta de una capa de entrada, una o más capas ocultas y una capa de salida. Gracias a las capas ocultas convierte las funciones linealmente no independientes en linealmente independientes. Esta arquitectura suele entrenarse mediante el algoritmo de aprendizaje denominado retropropagación de errores o BP (backpropagation), por lo general a este conjunto se le llama red de retropropagación. En el entrenamiento, se presentan a la red el patrón de entrada y el patrón de salida deseado, y automática y aleatoriamente se ajustan los pesos de la red para minimizar el error entre la salida real y el resultado deseado. El método de aprendizaje BP, dado un patrón de entrada compara el resultado obtenido en las unidades de salida con la respuesta que se desea obtener. A continuación reajusta los pesos de la red de manera que, la siguiente vez que se presente el mismo patrón de entrada, la red produzca un resultado más cercano al deseado, es decir que el error disminuya. Las actualizaciones de los pesos en la capa oculta dependen de todos los términos de errores de la capa de salida. Esto es a lo que se refiere con el nombre

de propagación hacia atrás. Los términos de error de las unidades ocultas, se calculan antes de que hayan sido modificados los pesos de conexiones con las unidades de la capa de salida. Al minimizar el error los planos que se generan se alinean tan cerca de los patrones de entrenamiento como sea posible.

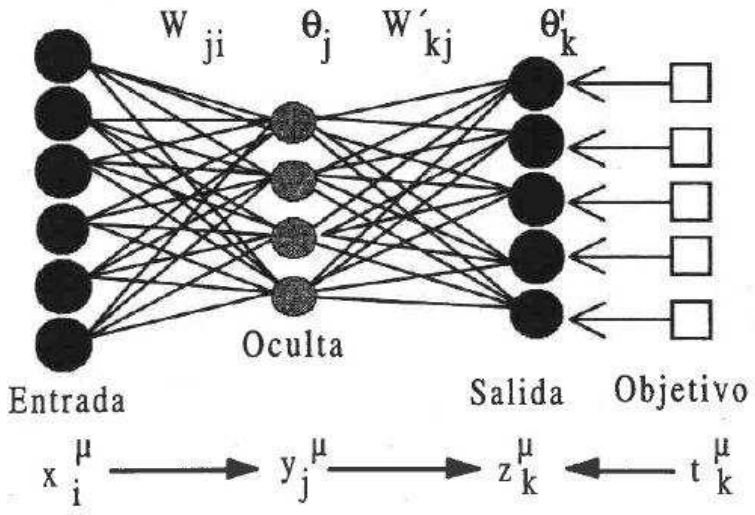
Para describir la estructura del MLP, denominaremos X_i a las entradas de la red, Y_j a las salidas de la capa oculta y Z_k a las de la capa final; t_k serán las salidas objetivo (target). Además W_{ij} son los pesos de la capa oculta y θ_j sus umbrales, $W`_{kj}$ los pesos de la capa de salida y $\theta`_k$ sus umbrales.

La operación de un MLP con una capa oculta y neuronas de salida lineal se expresa:

$$z_i = \sum_j w'_{kj} y_j - \theta'_i = \sum_j w'_{kj} f\left(\sum_i w_{ji} x_i - \theta_j\right) - \theta'_i \quad (2.33)$$

Siendo f (.) de tipo sigmoideo, que para nuestro caso tanh(x), función derivable.

Figura 2.4: Arquitectura del MLP.



3. METODOLOGÍA

A continuación se presenta la metodología aplicada en el desarrollo de este trabajo, así como también el proceso de selección de los registros que involucran eventos y ruido, esta elección se realizó con datos que se extraen de las bases del Observatorio Vulcanológico y Sismológico de Pasto (OVSP), tanto para la estación Anganoy como para Cráter-2, estaciones usadas como referencia para la clasificacion de sismicidad. Para el procesamiento de las señales y determinación de una arquitectura de red neuronal artificial (RNA) tipo Perceptron Multicapa se realizaron rutinas en el programa MATLAB, permitiendo almacenar, representar y manipular las señales en pro de realizar los entrenamientos y pruebas. A continuación se describe de forma detallada la selección de datos, la representación de los mismos y el proceso de la detección automática de eventos sísmicos.

3.1. SELECCIÓN DE DATOS. Gracias a medios telemétricos en la sede del OVSP se almacena continuamente las señales obtenidas por la red de estaciones sísmicas instaladas en puntos estratégicos alrededor del volcán Galeras. Para este propósito se cuenta con sismómetros tipo Corto Periodo y Banda Ancha. La señal recibida es discretizada con una rata de muestreo de 100 Hz, las señales muestreadas se almacenan en ventanas con una duración aproximada de 120 segundos, por tanto cada registro posee 12000 muestras, estos archivos tienen formato WVG (Wave Volcano Galeras).

Para el desarrollo del trabajo se emplean registros adquiridos por las estaciones de Corto Periodo Anganoy (ANGV, localizada a 0.8 km al E del cráter principal) y Cráter-2 (CR2R, ubicada a 1.48 km al SE respecto al cráter principal).

Para entrenar la RNA se toman registros de las bases del OVSP clasificados como LP y VT de diferentes años: 2007, 2006 y 2005, eventos asociados a movimiento de fluidos y fractura de material cortical respectivamente; donde los criterios para la elección del conjunto de entrenamiento se caracterizan por: nos interesa un registro donde se logre diferenciar claramente el evento, del ruido, es decir, determinar el inicio, coda y terminación del evento, sin importar la duración del mismo. La elección del registro de ruido se realizó de manera aleatoria tomando lapsos donde el nivel de amplitud máximo tuviera valores diferentes, ya que las condiciones del ruido varían constantemente en el tiempo, por agentes externos tales como las condiciones climatológicas entre otras.

Se seleccionaron 500 registros de eventos tanto para la estación ANGV como CR2R, así mismo se seleccionaron 500 registros de ruido para ANGV y CR2R.

Es decir, un total de 2000 registros. Los archivos seleccionados de formato WVG se convierten a formato ASCII (Americam Standard Code for Information Interchange) mediante el programa Sudasc, utilizado por Ingeominas para tal propósito, lo anterior es necesario para manipular los archivos en el programa MATLAB. El archivo en formato ASCII contiene la información de una traza completa (12000 muestras) arreglada en columnas, cada columna contiene los datos adquiridos por cada una de las componentes de las estaciones de la red sísmica de vigilancia.

3.2. REPRESENTACIÓN DE LOS REGISTROS SELECCIONADOS. Para extraer el evento de la traza continua, fue necesario elaborar una rutina que consiga tal fin. La rutina denominada delimit.m visualiza la traza y permite escoger el arribo y el final del evento, finalmente muestra en una ventana el tramo seleccionado. De manera similar se eligen los tramos de ruido con ayuda de la misma rutina. A continuación se muestra la rutina bajo [4] MATLAB denominada delimit.m:

```
1.-function y=delimit(x)
2.-m=mean(x);
3.-xm=x-m;
4.-plot(xm);
5.-pause
6.-[x1,y1]=ginput(2);
7.-xmin=x1(1,1);
8.-xmax=x1(2,1);
9.-tramo=xm(xmin:xmax);
10.-plot(tramo)
11.-y=tramo;
```

Descripción de la rutina:

1.-Propósito: programación.
2.-Propósito: valor medio de los elementos en el vector x.
3.-Propósito: trasladar el vector x a un sistema de coordenadas referenciado en el origen, llamado xm.
4.-Propósito: graficar el vector xm.
5.-Propósito: detiene el proceso de graficacion.
6.-Propósito: permite seleccionar dos puntos de la ventana xm utilizando el ratón.
7.-Propósito: primer punto seleccionado - inicio del evento.
8.-Propósito: segundo punto seleccionado - final del evento.
9.-Propósito: matriz compuesta por los vectores comprendidos entre los puntos seleccionados llamada tramo.mat.
10.-Propósito: grafica la selección (tramo).
11.-Propósito: la solución de la función, seleccionar o delimitar parte de la traza,

bien sea un evento o un tramo de ruido.

Gracias al método de codificación de predicción lineal (LPC) podemos extraer ciertas características espectrales del registro que queremos identificar. Antes de calcular los LPC del registro se debe dividir la señal en pequeños segmentos que llamaremos ventanas. Con el fin de evitar que cortes bruscos de inicio y final de la señal introduzcan componentes de alta frecuencia en el espectro se somete a los segmentos al proceso de ventaneo, que consiste en multiplicar el segmento por una función de ventana w(n). En este caso se utilizó la ventana de Hamming, cuya función en el dominio continuo tiene la siguiente forma:

$$w(n)=\begin{cases} 0{,}54 - 0{,}46cos(\frac{2\pi n}{N_w - 1}), & 0 \leq n \leq N_w - 1, \\ 0, & \text{para los demás.} \end{cases}$$

Luego se crea una rutina llamada partir.m, que realiza el proceso de ventaneo, particionando el tramo en ventanas de 3 segundos y calcula a cada uno de estos tramos 12 coeficientes de predicción lineal (LPC). Es decir, 300 muestras son representadas con 12 LPC sin mayor pérdida de información, excluyendo en todos los casos el primer coeficiente el cual es igual a 1.

Figura 3.1: Segmento sin ventaneo de Hamming.

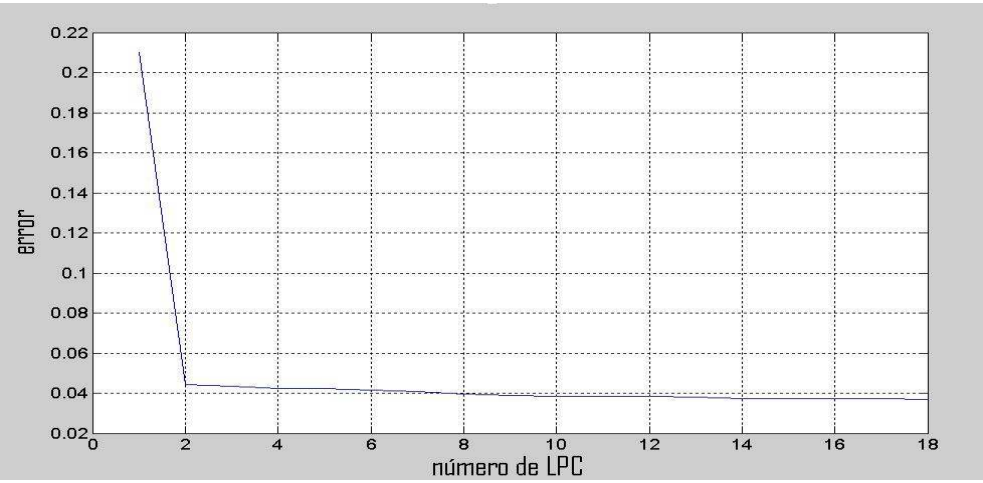
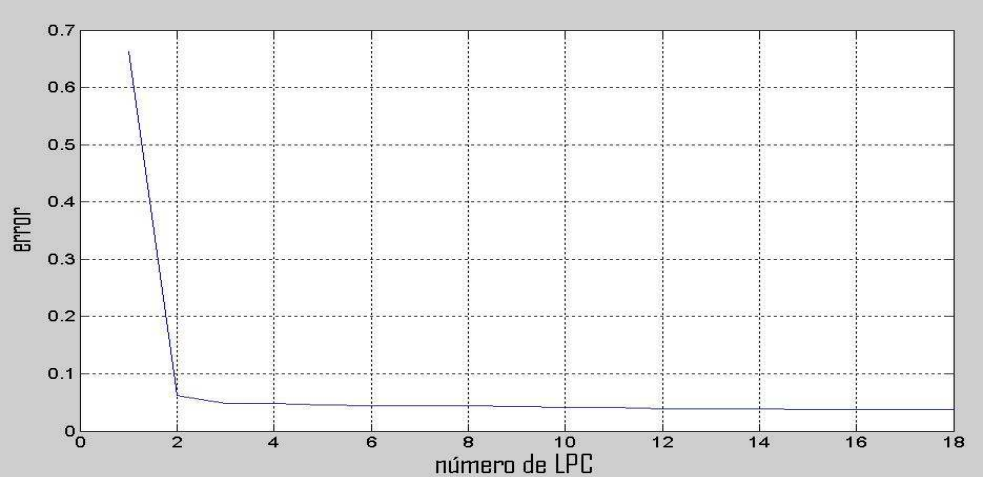


Figura 3.2: Segmento con ventaneo de Hamming.



En la figuras 3.1, 3.2 se ilustra la dependencia del error cometido al representar la serie de tiempo dada respecto al número de LPC para un tramo, se observa un comportamiento asintótico horizontal del error cuando el número de coeficientes es mayor de 7 en el segmento sin ventaneo y mayor que 5 en el segmento ventaneado. Una mayor calidad de representación de los segmentos de la señal se lograría con un número elevado de LPC, pero esto aumentaría el tiempo de computo en el entrenamiento de la red neuronal artificial, por tanto, la dimensión de los vectores de entrada debe ser moderada; de acuerdo a lo anterior se opta por representar cada segmento de señal con 12 LPC. La rutina desarrollada denominada partir.m es la siguiente:

```
12.-function y=partir(x)
13.-t=300;
14.-[m,n]=size(x);
15.-mn=mean(x);
16.-x=x-mn;
17.-m1=floor(m/t);
18.-x1=x(1:300*m1,1);
19.-x2=reshape(x1,t,m1);
20.-w=hamming(300); % ventana de Hamming de 3.0 seg.(300 muestras )
21.-for i=1:m1
22.- x3(:,i)=x2(:,i).*w;
23.-end
24.-x4=lpc(x3,12);
25.-x5=x4.';
```

26.-y=x5(2:13,:);

Descripción de la rutina:

12.-Propósito: programación.
13.-Propósito: definir la variable t.
14.-Propósito: la función size devuelve un vector fila cuyo primer elemento es el número de filas y cuyo segundo elemento es el número de columnas del vector x.
15.-Propósito: valor medio de los elementos en el vector x.
16.-Propósito: trasladar el vector x a un sistema de coordenadas referenciado en el origen, llamado xm.
17.-Propósito: la función floor redondea los elementos del cociente m/ t, donde m es el número de filas y t, un número.
18.-Propósito: crea una matriz, tomando elementos de x, de longitud un múltiplo de 300.
19.-Propósito: el comando reshape devuelve una matriz t x m1 cuyos elementos se toman de x1.
20.-Propósito: definir la ventana de Hamming, para 300 muestras.
21,22,23.-Propósito: crear un bucle o grupo de órdenes que se repetirán hasta el fin del mismo, cuyo fin es multiplicar la matriz obtenida en x2 de 300 filas y m1 columnas por W , ventana de Hamming.
24.-Propósito: calcular 12 LPC para cada fila de 300 muestras que se obtiene en x3.
25.-Propósito: calcular la transpuesta de la matriz x4.
26.-Propósito: la solución de la función, muestra ordenados por columnas de 12 filas cada 300 muestras de registro.

El segmento de evento o ruido seleccionado debe ser dividido en vectores de 300 muestras cada uno, a cada uno de estos vectores se les extrae los 12 LPC que representaran este intervalo de tres segundos de la serie de tiempo. El procedimiento descrito anteriormente se ejecuta con ayuda de la rutina llamada Union_eventos.m creada para tal fin.

% Delimitación de evento, calculo de LPC para ANGV Y CR2R %(500 eventos)
27.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
28.-if filename = 0
29.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^n]',' delimiter',';','headerlines',1);
30.-EAN001=delimit(AN);
31.-ECR001=delimit(CR);
32.-CEAN001=partir(EAN001);
33.-CECR001=partir(ECR001);
34.-clear AN CR EAN001 ECR001

35.-else
36.-errordlg('No ingreso ningún archivo!', 'Error', 'modal');
37.-break
38.-end
39.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
40.-if filename = 0
41.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^n]',' delimiter',';','headerlines',1);
42.-EAN002=delimit(AN);
43.-ECR002=delimit(CR);
44.-CEAN002=partir(EAN002);
45.-CECR002=partir(ECR002);
46.-clear AN CR EAN002 ECR002
47.-else
48.-warndlg('Usted ha ingresado 1 archivo!', 'Numero de archivos', 'modal');
49.-break
50.-end
51.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
52.-if filename = 0
53.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^n]',' delimiter',';','headerlines',1);
54.-EAN003=delimit(AN);
55.-ECR003=delimit(CR);
56.-CEAN003=partir(EAN003);
57.-CECR003=partir(ECR003);
58.-clear AN CR EAN003 ECR003
59.-else
60.-warndlg('Usted ha ingresado 2 archivos!', 'Numero de archivos', 'modal');
61.-break
62.-end

Descripción de la rutina:

27.-Propósito: permite acceder a los nombres de los archivos en formato ASCII ubicados, en este caso en el workspace y elegir uno.
28,35,36,37,38.-Propósito: en el caso de que no se elija ningún nombre de archivo, aparezca el mensaje No ingreso ningún archivo y finalice la función.
29.-Propósito: en el caso de elegir un nombre de archivo, tome de este la información contenida en las columnas 3 y 4, información de las estaciones ANGV y CR2R respectivamente.
30.-Propósito: aplicar la función delimit.m a la columna 3, y como resultado de esta aplicación genera la matriz EAN001.mat.
31.-Propósito: aplicar la función delimit.m a la columna 4, y como resultado de esta

aplicación genera la matriz ECR001.mat.
32.-Propósito: aplicar la función partir.m a la matriz generada en 30. y luego este resultado se guardará en la matriz CEAN001.mat.
33.-Propósito: aplicar la función partir.m a la matriz generada en 31. y luego este resultado se guardará en la matriz CECR001.mat.
34.-Propósito: al terminar el proceso de aplicación de las funciones delimit.m y partir.m, borre las matrices AN.mat, CR.mat, EAN001.mat, ECR001.mat y cuando termine el proceso solo visualice las matrices CEAN001.mat y CECR001.mat.
39-50.-Propósito: permite realizar los procesos descritos anteriormente, solo que en este caso luego de ingresar el primer archivo, si no se selecciona algún otro (40.), el arrojará un mensaje diciendo Usted ha ingresado 1 archivo.
51-62.-Propósito: permite realizar los procesos descritos anteriormente, solo que en este caso luego de ingresar el segundo archivo, si no se selecciona algún otro (52.), el arrojará un mensaje diciendo Usted ha ingresado 2 archivos.

La rutina se repite igualmente hasta completar un seleccionado de 500 archivos.

La rutina Union_eventos.m extrae las trazas correspondientes a las estaciones Anganoy (ANGV) y Cráter-2 (CR2R) desde el archivo en formato ASCII. Posteriormente la rutina aplica la función delimit.m a los vectores correspondientes a las trazas de ANGV y CR2R, en este momento se selecciona manualmente el inicio y el fin del tramo. Luego se aplica la función partir.m al tramo escogido, dividiendo el vector en subvectores de 300 muestras a las que se ejecuta el proceso de ventaneo y finalmente la extracción de los LPC. La información procesada por las rutinas se almacena en una matriz cuya nomenclatura es la siguiente: la primera letra (C) indica que se trata del cálculo de los LPC, la segunda (E) indican que dichos coeficientes corresponden al registro de un evento sísmico, si se tratase de un registro de ruido esta letra será una R; las siguientes dos letras indicaran a que estación pertenecen los registros AN cuando se trate de Anganoy y CR en el caso de Cráter-2, a estas letras le siguen tres números entre el 001 y 500 que diferenciaran cada matriz del mismo tipo. La rutina descrita anteriormente se aplicó a 500 eventos y 500 segmentos de ruido correspondientes a la estación Anganoy, y a un número igual de eventos y segmentos de ruido para la estación Cráter-2. Finalmente se almacenaron 2000 matrices en total. Para los segmentos de ruido se tiene la siguiente rutina llamada Union_ruido.m:

%Delimitación de ruido, calculo de LPC para ANGV Y CR2R (500 tramos)
64.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
65.-if filename = 0
66.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^ n]',' delimiter',';','headerlines',1);
67.-RAN001=delimit(AN);
68.-RCR001=delimit(CR);

```
69.-CRAN001=partir(RAN001);
70.-CRCR001=partir(RCR001);
71.-clear AN CR RAN001 RCR001
72.-else
73.-errordlg('No ingreso ningún archivo!', 'Error', 'modal');
74.-break
75.-end
76.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
77.-if filename = 0
78.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^ n]',' delimiter',';','headerlines',1);
79.-RAN002=delimit(AN);
80.-RCR002=delimit(CR);
81.-CRAN002=partir(RAN002);
82.-CRCR002=partir(RCR002);
83.-clear AN CR RAN002 RCR002
84.-else
85.-warndlg('Usted ha ingresado 1 archivo!', 'Numero de archivos', 'modal');
86.-break
87.-end
88.-[filename, pathname] = uigetfile('*.*', 'Load ASC File');
89.-if filename = 0
90.-[AN,CR]=textread(filename,' %*n %*n %n %n %*[^ n]',' delimiter',';','headerlines',1);
91.-RAN003=delimit(AN);
92.-RCR003=delimit(CR);
93.-CRAN003=partir(RAN003);
94.-CRCR003=partir(RCR003);
95.-clear AN CR RAN003 RCR003
96.-else
97.-warndlg('Usted ha ingresado 2 archivos!', 'Numero de archivos', 'modal');
98.-break
99.-end
```

Descripción de la rutina:

64-99.-Propósito: permite realizar lo descrito entre 27. y 62., en este caso cambia la nomenclatura de las matrices, haciendo referencia al tipo de archivos.

En la figura 3.3 se muestra un evento seleccionado y en la figura 3.4 se muestra el tramo seleccionado manualmente al que se aplican las rutinas descritas anteriormente.

Figura 3.3: Evento seleccionado.

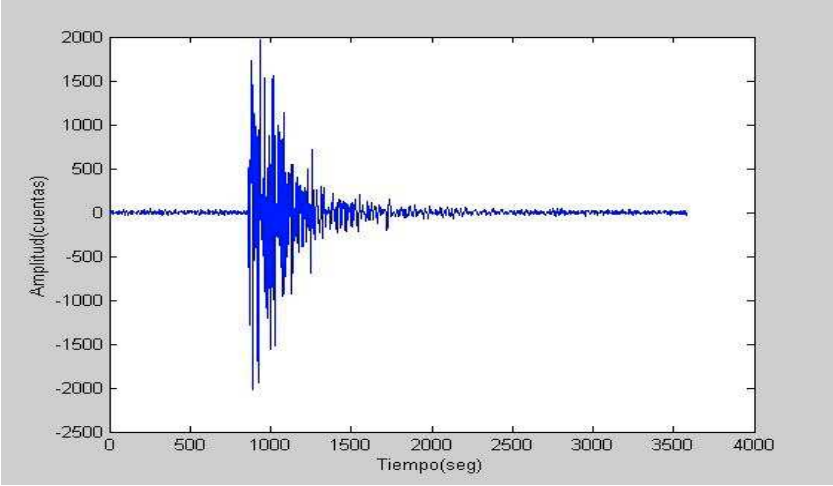
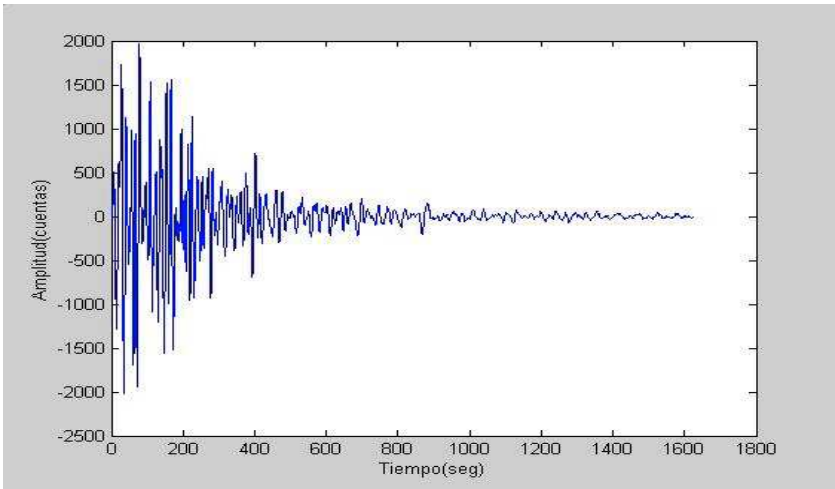


Figura 3.4: Evento seleccionado delimitado.



En las figuras 3.5 y 3.6 se ilustra la extracción de un segmento de un registro de ruido.

Figura 3.5: Registro de ruido seleccionado.

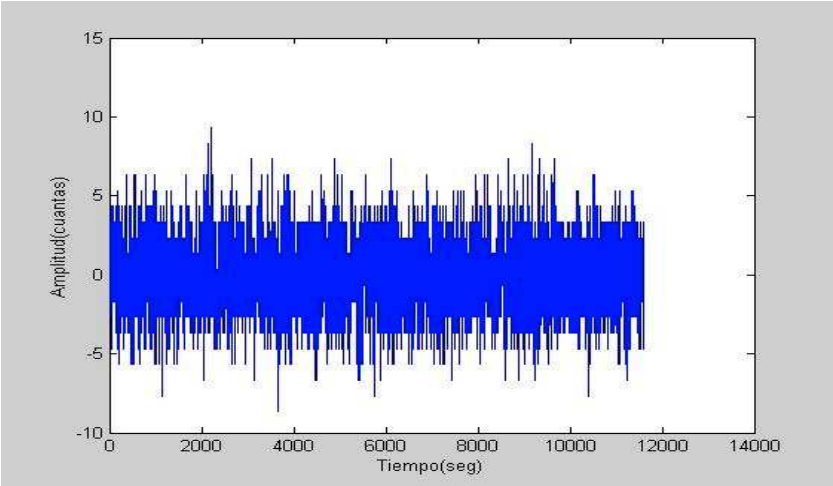
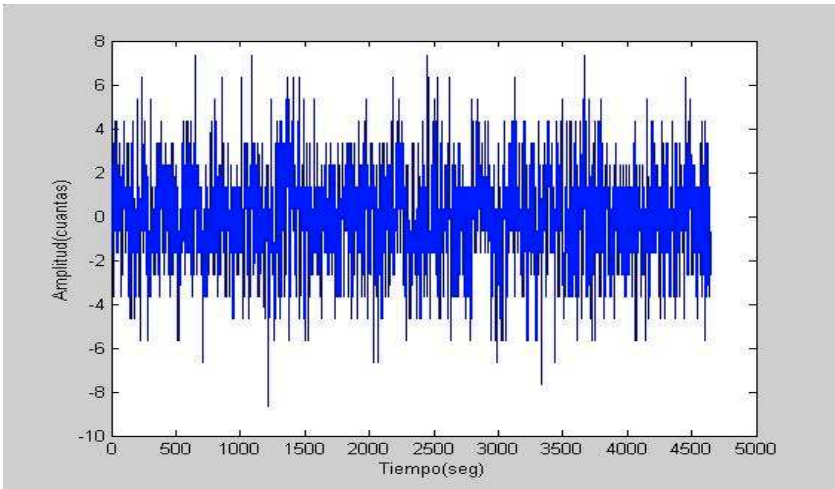


Figura 3.6: Registro de ruido seleccionado delimitado.



3.3. PROCESO DE DETECCIÓN AUTOMÁTICA. A continuación se procede a formar las matrices de entrenamiento y prueba que se presentaran a la red neuronal. En primer lugar ordenamos todos los vectores que contienen los LPC que representan cada segmento de los eventos seleccionados para la estación ANGV, dando lugar a una matriz de dimensiones 12 x 3136 donde cada columna contiene los 12 LPC que representan la fracción del evento. La matriz construida anteriormente se guarda con el nombre de matang.mat.

De manera similar se procede con los vectores que representan los segmentos de eventos que corresponden a la estación CR2R y los segmentos de ruido que pertenecen tanto a CR2R como a ANGV, obteniendo las siguientes matrices: para ANGV la matriz que representa cada segmento de ruido se llama matrang.mat de dimensiones 12 x 6794 y para CR2R la matriz llamada matrcr.mat de dimensiones 12 x 6593 y para CR2R la matriz matcr.mat de dimensiones 12 x 2758 me representan los vectores de eventos para esta estación. Luego, para realizar las secciones de entrenamiento y prueba en las estaciones elegidas es necesario preparar las matrices de la siguiente manera: fusionamos la matriz que contiene la información de los eventos con la matriz que representa ruido, de tal forma que a un vector columna que represente un tramo de evento le siga un vector que represente ruido, así la información queda intercalada. Para ANGV, de las matrices matang.mat y matrang.mat se han tomado las primeras 3000 columnas que harán parte de los lotes de entrenamiento y prueba; la matriz así construida se guarda con el nombre de matinang.mat de dimensiones 12 x 6000, de igual forma se procedió para la información de la estación CR2R con la diferencia de que tanto de su matriz que representa eventos como ruido se tomaron las primeras 2700 columnas y por tanto la matriz almacenada con el nombre matincr.mat tiene dimensiones 12 x 5400. Las rutinas que generan estas matrices son las siguientes:

La rutina que genera la matriz matinang.mat se denomina ordenang.m:

```
100.-[m,n]=size(matang);
101.-[p,q]=size(matrang);
102.-for i=0:2999
103.- matinang(:,2*i+1)=matang(:,i+1);
104.- matinang(:,2*i+2)=matrang(:,i+1);
105.-end
```

Descripción de la rutina:

100.-Propósito: la función size devuelve un vector fila cuyo primer elemento es el número de filas y cuyo segundo elemento es el número de columnas de la matriz matang.mat.
101.-Propósito: dimensiones de la matriz matrang.mat.
102,103,104,105.-Propósito: crear un bucle o grupo de órdenes que se repetirán hasta el fin del mismo, cuyo fin es intercalar información de la matriz de eventos con la matriz de ruido; evento (103.) tomando lugares impares y ruido (104.) pares.

Y la rutina que genera la matriz matincr.mat es ordencr.m:

```
106.-[m,n]=size(matcr);
107.-[p,q]=size(matrcr);
108.-for i=0:2699
109.-matincr(:,2*i+1)=matcr(:,i+1);
110.- matincr(:,2*i+2)=matrcr(:,i+1);
111.-end
```

Descripción de la rutina:

106-111.-Propósito: permite realizar lo descrito entre 100. y 105., en este caso cambia la nomenclatura y el tamaño de las matrices, haciendo referencia a la información de la estación CR2R.

Para efectuar el entrenamiento de la RNA es necesario construir las matrices que contengan los valores que se han de asociar con cada una de las columnas de las matrices de entrenamiento. La rutina unosan.m crea el vector fila taran.mat que consta de 5900 elementos cuyos valores intercalados entre sí son 1 y -1 que serán los objetivos asociados a evento y ruido respectivamente, por tanto se tiene:

```
112.-unos=[1 -1];
113.-for i=0:2949
114.- taran(:,2*i+1)=unos(:,1);
115.- taran(:,2*i+2)=unos(:,2);
116.-end
```

Descripción de la rutina:

112.-Propósito: genera un vector fila de dos columnas 1 y -1.
113-116.-Propósito: crear un bucle o grupo de órdenes que se repetirán hasta el fin del mismo, cuyo fin es intercalar información 1 y -1; donde 1 (114.) toma lugares impares y -1 (115.) pares.

Para la información de la estación CR2R este vector fila consta de 5300 elementos y lleva el nombre de tarcr.mat, construido bajo la rutina unoscr.m:

```
117-.unos=[1 -1];
118.-for i=0:2649
119.- tarcr(:,2*i+1)=unos(:,1);
120.- tarcr(:,2*i+2)=unos(:,2);
121.-end
```

Descripción de la rutina:

117-121.-Propósito: permite realizar lo descrito entre 112. y 116., en este caso cambia la nomenclatura y el tamaño de la matriz.

3.3.1. Lotes de datos y entrenamiento. Para la estación Anganoy tenemos la matriz matinang.mat de dimensiones 12 x 6000. Separamos esta matriz en diferentes lotes de entrenamiento, de tal manera que cada lote contenga 5900 columnas, las 100 restantes serán utilizadas para probar la efectividad de la red en su tarea de clasificación.

Se han seleccionado 10 lotes de entrenamiento que se listan en la tabla Nº 3.1. La matriz que contiene los datos de la estación Cráter-2, es de dimensiones 12 x 5400, de manera análoga a la matriz que contiene la información de la estación Anganoy se han separado 100 columnas para efectos de probar la red, los 5300 restantes formarán parte de los lotes de entrenamiento.

En la tabla Nº 3.2 se muestran los lotes de entrenamiento y prueba para la estación Cráter-2.

Tabla 3.1: Lotes de entrenamiento seleccionados para ANGV.

Lotes ANGV	Columnas	Test
1	101 a 6000	1a 100
2	1a 3000 U 3101 a 6000	3001a 3100
3	1a 5900	5901a 6000
4	1a 1000 U 1101 a 6000	1001a 1100
5	1a 2000 U 2101 a 6000	2001a 2100
6	1a 4000 U 4101 a 6000	4001a 4100
7	1a 5000 U 5101 a 6000	5001a 5100
8	1a 500 U 601 a 6000	501a 600
9	1a 5400 U 5501 a 6000	5401a 5500
10	1a 4400 U 4501 a 6000	4401a 4500

Tabla 3.2: Lotes de entrenamiento seleccionados para CR2R.

Lotes CR2R	Columnas	Test
1	101 a 5400	1a 100
2	1a 2700 U 2801 a 5400	2701a 2800
3	1a 5300	5301a 5400
4	1a 1000 U 1101 a 5400	1001a 1100
5	1a 2000 U 2101 a 5400	2001a 2100
6	1a 4000 U 4101 a 5400	4001a 4100
7	1a 3000 U 3101 a 5400	3001a 3100
8	1a 1500 U 1601 a 5400	1501a 1600
9	1a 5000 U 5101 a 5400	5001a 5100
10	1a 700 U 801 a 5400	701a 800

[8] El siguiente paso es desarrollar las rutinas que crean una arquitectura de red y permiten el entrenamiento de la misma. Inicialmente se cargan las matrices de entrenamiento y prueba en el workspace del programa MATLAB. Todos los vectores deben ser normalizados, requisito exigido por el algoritmo de aprendizaje utilizado. A continuación se propone un conexionado o arquitectura de red e inicia el entrenamiento. Luego se procede a la prueba de la RNA haciendo uso del lote de test y calculando los parámetros que determinan la efectividad del entrenamiento. Una típica rutina de entrenamiento se describe a continuación:

```
_ [lotean1n,amin,amax,tarann,tmin,tmax]=premnmx(lotean1,taran);
_testan1n=tramnmx(testan1,amin,amax);
_net1=newff(minmax(lotean1n),[3,1],'tansig' 'tansig','trainbr');
_net1.trainParam.epochs=100;
_net1.trainParam.show=25;
_net1=init(net1);
_tic,[net1,tr1]=train(net1,lotean1n,tarann),toc;
TRAINBR, Epoch 0/100, SSE 9010.9/0, SSW 13.7292,
Grad 5.29e+003/1.00e-010,  #Par 4.30e+001/43
 TRAINBR, Epoch 25/100, SSE 571.118/0, SSW 316.62,
 Grad 3.83e+001/1.00e-010,  #Par 3.73e+001/43
TRAINBR, Epoch 50/100, SSE 537.99/0, SSW 1799.48,
Grad 4.15e+000/1.00e-010, #Par 4.01e+001/43
TRAINBR, Epoch 75/100, SSE 535.801/0, SSW 2415.87,
Grad 5.98e+000/1.00e-010, #Par 4.12e+001/43
TRAINBR, Epoch 100/100, SSE 535.288/0, SSW 2709.09,
 Grad 3.90e-001/1.00e-010, #Par 4.12e+001/43
TRAINBR, Maximum epoch reached.
```

elapsed time = 24;4450

_resul=sim(net1,testan1n);
_ [m,b,r]=postreg(resul,tpos)

m = 0.9843

b = 0.0094

r = 0.9995

Descripción de la rutina:

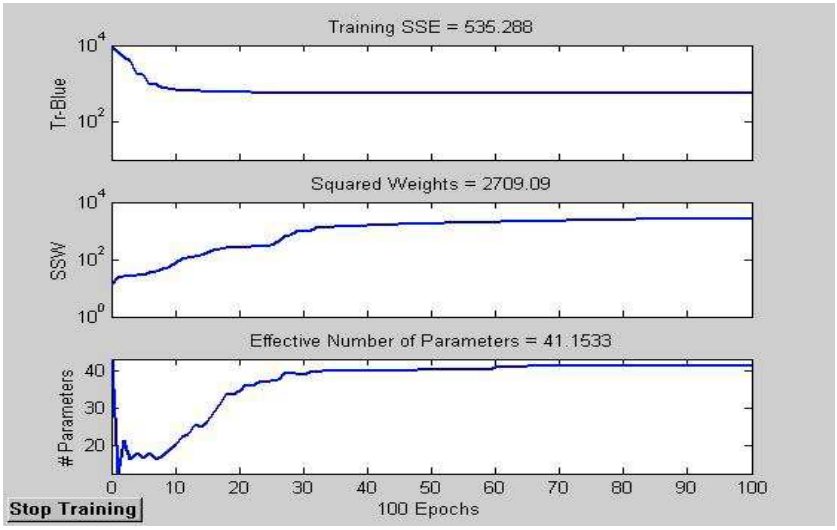
La rutina presentada anteriormente fue creada para entrenar con un lote que contiene las columnas 101 a 6000 y como lote de prueba se utilizó las columnas desde la 1 a 100. El comando premnmx normaliza los vectores de entrenamiento y el vector de objetivos taran.mat en el rango [-1,1] genera las matrices lotean1n.mat y tarann.mat normalizadas, donde amin y amax muestran los valores mínimo y máximo que toma la matriz lotean1n.mat, mientras que tmin y tmax muestran los valores mínimo y máximo que toma la matriz tarann.mat. El comando tramnmx normaliza los vectores de la matriz de prueba o test (testan1.mat) para lo cual utiliza los vectores amin y amax creando una nueva matriz llamada testan1n.mat.

Con ayuda del comando newff se propone una red neuronal artificial unidireccional que tiene una capa oculta con tres neuronas y una capa de salida con una neurona. La función de transferencia es de tipo sigmoideo (tansig). Se aplica una modificación del algoritmo de aprendizaje de retro-propagación mediante el comando trainbr. Con la orden net1.train Param.epochs se fija el número máximo de iteraciones a lo largo del entrenamiento, para este trabajo fue de 100, es decir el programa introducirá de manera iterativa 100 veces el lote de ANGV 1. La orden net1.train Param.show visualiza información del progreso del entrenamiento cada 25 iteraciones. El comando init inicializa los pesos sinápticos antes del entrenamiento con valores aleatorios; el comando train da comienzo al entrenamiento de la red neuronal artifical en función de el lote de entrenamiento lotean1n.mat y el lote de objetivos tarann.mat. Los vectores de entrenamiento ingresan de uno en uno a la red y cuando todos hayan pasado por la red esta calcula el valor de ajuste de los pesos con el fin de disminuir el error que la red comete al asociarlos con los objetivos. Cuando el proceso termina despliega tres gráficas con información del entrenamiento (figura 3.7), la primera grafica muestra la curva de la suma de los errores medios cuadráticos con respecto al número de iteraciones; la segunda curva muestra la suma de los cuadrados de los pesos sinápticos con respecto al número de iteraciones, la última representa el número de parámetros que fueron efectivamente utilizados durante el ajuste de los pesos. La nueva configuración de los pesos sinápticos es almacenada en net1. Con el

comando sim simulamos el funcionamiento de la RNA aplicado a la matriz de prueba, vale aclarar que la red no tiene ningún tipo de información acerca de los vectores de este lote. Se genera la matriz resul.mat, la cual almacena los resultados del reconocimiento entre evento y ruido.

El comando postreg aplica la técnica de regresión lineal para encontrar la mejor recta que ajuste los valores generados por la red y los valores esperados representados por la matriz tpos.mat donde m, b y r representan los valores de pendiente, intercepto y coeficiente de correlación respectivamente. Para el ejemplo descrito anteriormente r toma el valor 0.9995 (figura 3.8) evidenciando los buenos resultados del tipo de RNA y arquitectura elegida.

Figura 3.7: Gráficas de información acerca del proceso de entrenamiento.

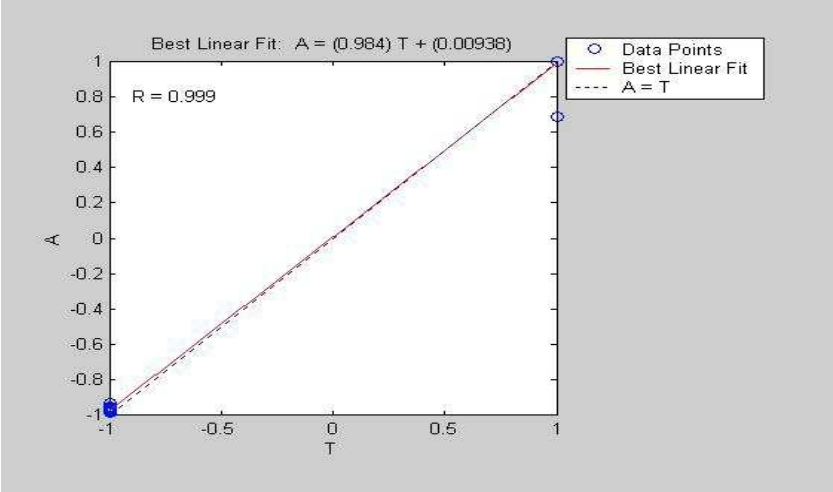


Si se aplica nuevamente el comando init los pesos son ajustados aleatoriamente en la RNA y podemos iniciar un nuevo entrenamiento, que lo llamaremos net2, con el mismo lote ANGV 1:

```
_net2=init(net1);
_tic,[net2,tr2]=train(net2,lotean1n,tarann),toc;
```

El comando tic-toc calcula el tiempo (en segundos) empleado en realizar el proceso de entrenamiento. Así sucesivamente con los 10 lotes de ANGV se realiza cinco entrenamientos (cinco inicializaciones de red) con cada lote e igualmente con los lotes de entrenamiento de CR2R.

Figura 3.8: Regresión Lineal.



4. RESULTADOS

Una vez aplicada la metodología descrita anteriormente, los resultados obtenidos sugieren que la mejor arquitectura de red o estructura de conexionado del Perceptron multicapa se compone de una capa oculta compuesta de tres neuronas y una neurona de salida. Recordemos que en el MLP las neuronas de la capa oculta y de salida tienen como función de activación la función sigmoidea. Por tanto, la arquitectura de red utilizada para la información de las estaciones Anganoy y Cráter2 fue de 12-3-1.

Los resultados que muestran cada una de las tablas que se listan a continuación contienen la información de los entrenamientos ejecutados con un mismo lote cuya diferencia radica en que se efectuaron varias inicializaciones con la misma arquitectura de red. Para cada una de estas inicializaciones (net1, net2, net3,...) se listan el porcentaje de efectividad, los parámetros de regresión lineal m, b y r y el tiempo que tardo el entrenamiento.

También mostraremos las gráficas de entrenamiento y regresión lineal para la mejor net con respecto al lote de entrenamiento, en medio magnético se anexan la graficas de entrenamiento y regresión lineal para todas las nets de cada lote de entrenamiento. La efectividad hace referencia a la cantidad de aciertos obtenidos al simular la red con el lote de prueba formado por 100 vectores.

4.1. RESULTADOS PARA LA ESTACIÓN ANGV.

A continuación se muestra los resultados obtenidos para ANGV:

Tabla 4.1: Resultados Lote de entrenamiento y prueba 01 ANGV.

Lote01- ANGV	net1	net2	net3	net4	net5
Efectividad	100%	99%	99%	100%	99%
m	0.9843	0.9913	0.9900	0.9856	0.9900
b	0.0094	-0.0028	-0.0055	0.0121	-0.0055
r	0.9995	0.9987	0.9977	0.9999	0.9977
time (s)	24.23	20.99	20.57	20.48	20.56

Tabla 4.2: Resultados Lote de entrenamiento y prueba 02 ANGV.

Lote02- ANGV	net1	net2	net3	net4	net5
Efectividad	99%	99%	99%	100%	100%
m	0.9528	0.9528	0.9527	0.9849	0.9849
b	0.0072	0.0075	0.0074	0.0148	0.0148
r	0.9777	0.9779	0.9778	0.9989	0.9989
time (s)	21.19	20.13	20.23	20.07	20.11

Tabla 4.3: Resultados Lote de entrenamiento y prueba 03 ANGV.

Lote03- ANGV	net1	net2	net3	net4	net5
Efectividad	100%	100%	100%	100%	100%
m	0.9820	0.9949	0.9949	0.9948	0.9821
b	0.0105	-0.0025	-0.0024	-0.0025	0.0103
r	0.9995	0.9994	0.9994	0.9994	0.9994
time (s)	20.11	20.01	20.04	19.81	20.04

Tabla 4.4: Resultados Lote de entrenamiento y prueba 04 ANGV.

Lote04- ANGV	net1	net2	net3	net4	net5
Efectividad	86%	89%	92%	92%	87%
m	0.7959	0.8499	0.8552	0.8551	0.7801
b	0.1675	0.1329	0.1367	0.1366	0.2100
r	0.8874	0.9327	0.9392	0.9392	0.8928
time (s)	22.09	19.87	19.46	19.49	19.48

Tabla 4.5: Resultados Lote de entrenamiento y prueba 05 ANGV.

Lote05- ANGV	net1	net2	net3	net4	net5
Efectividad	97%	99%	99%	100%	99%
m	0.9574	0.9633	0.9627	0.9813	0.9723
b	0.0290	0.0355	0.0373	0.0104	0.0174
r	0.9942	0.9956	0.9965	0.9985	0.9968
time (s)	19.84	19.44	19.41	19.22	19.26

Tabla 4.6: Resultados Lote de entrenamiento y prueba 06 ANGV.

Lote06- ANGV	net1	net2	net3	net4	net5
Efectividad	80%	78%	87%	79%	79%
m	0.6516	0.6635	0.7490	0.6601	0.6559
b	0.3435	0.3088	0.2510	0.3370	0.3404
r	0.7585	0.7708	0.8069	0.7720	0.7686
time (s)	19.52	19.81	20.42	20.14	20.14

Tabla 4.7: Resultados Lote de entrenamiento y prueba 07 ANGV.

Lote07- ANGV	net1	net2	net3	net4	net5
Efectividad	93%	93%	93%	94%	94%
m	0.8792	0.8800	0.8800	0.9072	0.8920
b	-0.0432	-0.0425	-0.0425	-0.0696	-0.0819
r	0.9303	0.9305	0.9305	0.9396	0.9305
time (s)	20.16	19.94	20.00	19.97	19.99

Tabla 4.8: Resultados Lote de entrenamiento y prueba 08 ANGV.

Lote08- ANGV	net1	net2	net3	net4	net5
Efectividad	90%	91%	91%	91%	91%
m	0.8523	0.8384	0.8384	0.8392	0.8390
b	-0.1471	-0.1336	-0.1336	-0.1336	-0.1342
r	0.9025	0.8899	0.8898	0.8908	0.8909
time (s)	19.75	20.00	19.94	20.00	19.89

Tabla 4.9: Resultados Lote de entrenamiento y prueba 09 ANGV.

Lote09- ANGV	net1	net2	net3	net4	net5	net6
Efectividad	93%	96%	96%	97%	93%	97%
m	0.9314	0.9446	0.9406	0.9337	0.9315	0.9451
b	-5.5074e-004	-0.0143	-0.0113	0.0091	-4.7093e-004	-0.0029
r	0.9724	0.9848	0.9720	0.9770	0.9725	0.9856
time (s)	19.86	19.86	19.86	20.10	19.84	20.12

Tabla 4.10: Resultados Lote de entrenamiento y prueba 10 ANGV.

Lote10- ANGV	net1	net2	net3	net4	net5	net6
Efectividad	94%	94%	95%	94%	95%	98%
m	0.9032	0.9032	0.9215	0.9032	0.9211	0.9349
b	-0.0065	-0.0065	-0.0118	-0.0065	-0.0113	-0.0090
r	0.9434	0.9434	0.9527	0.9434	0.9524	0.9701
time (s)	19.77	19.97	19.94	20.05	19.88	20.10

A continuación se presentan las gráficas de entrenamiento y las rectas obtenidas mediante la aplicación del método de regresión lineal de la mejor net de cada lote para ANGV, teniendo en cuenta la efectividad, el coeficiente de correlación.

Figura 4.1: Información de entrenamiento para net4 lote01-ANGV.

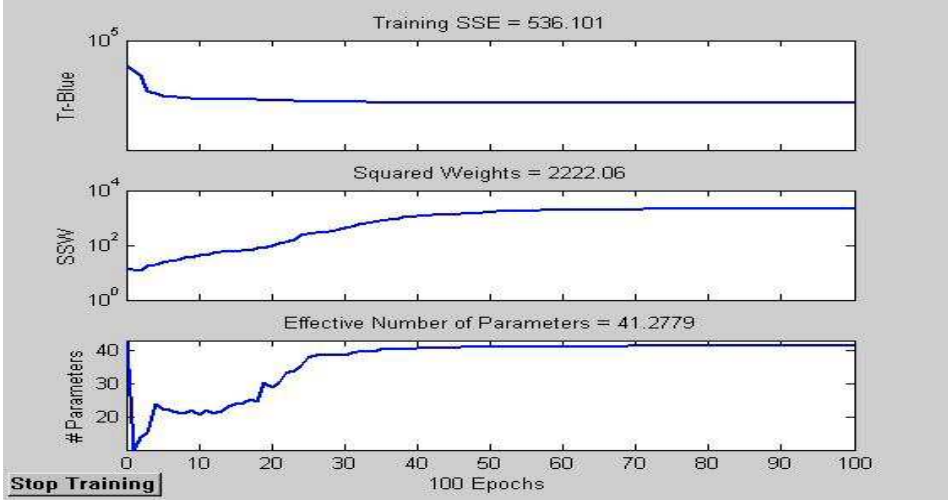


Figura 4.2: Información de regresión lineal para net4 lote01-ANGV.

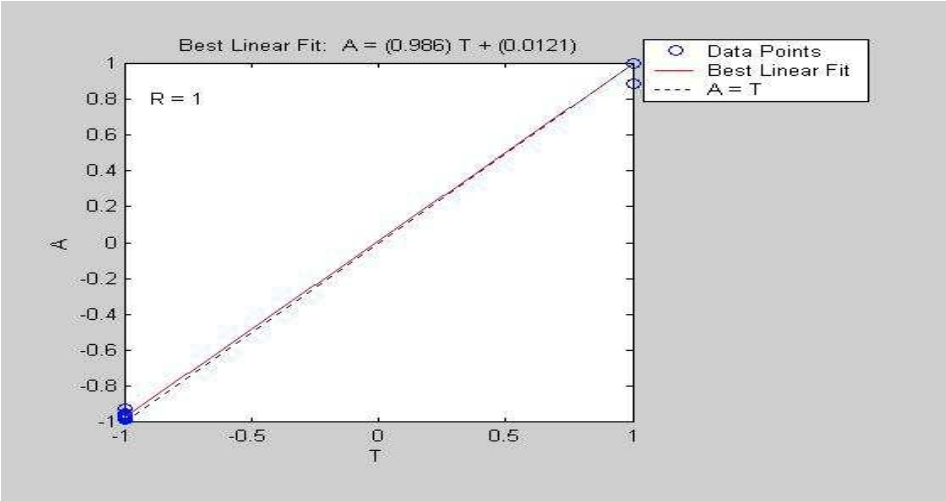


Figura 4.3: Información de entrenamiento para net4 lote02-ANGV.

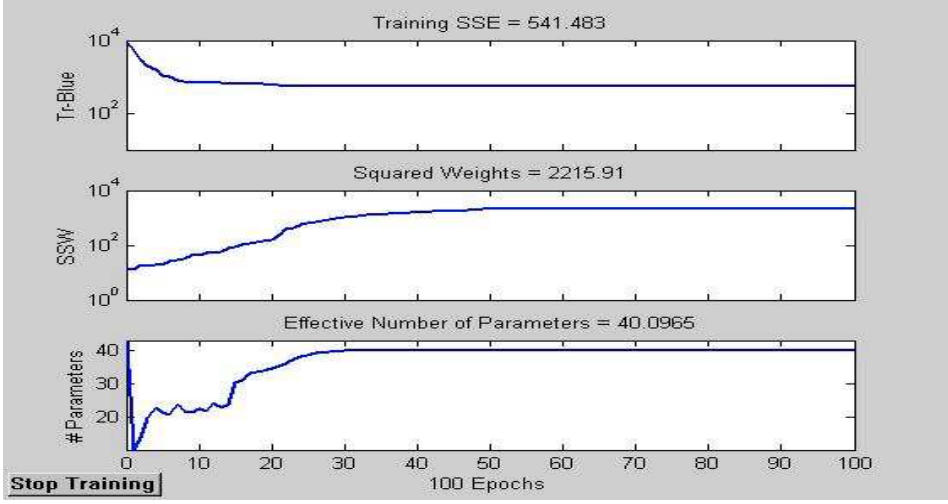


Figura 4.4: Información de regresión lineal para net4 lote02-ANGV.

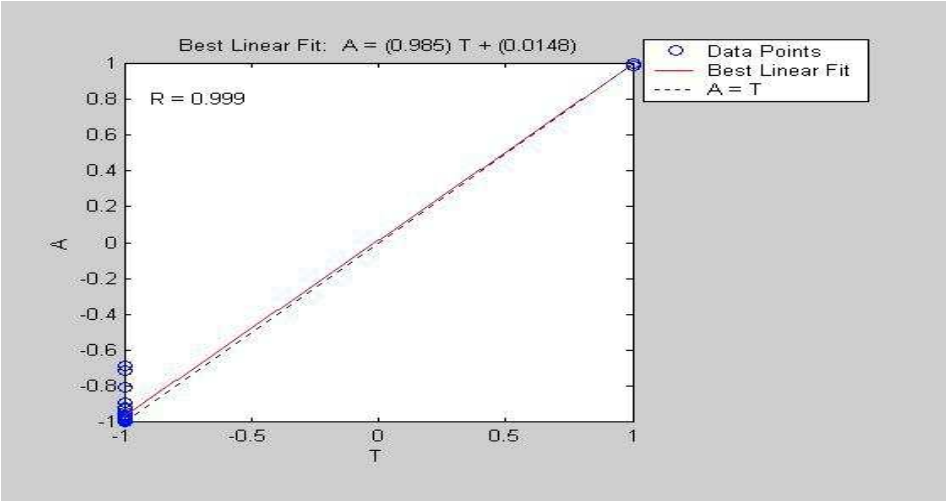


Figura 4.5: Información de entrenamiento para net1 lote03-ANGV.

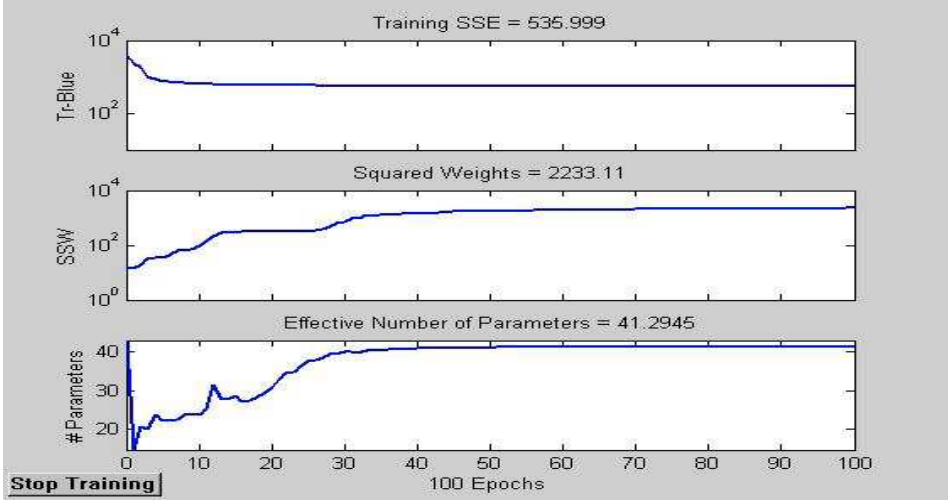


Figura 4.6: Información de regresión lineal para net1 lote03-ANGV.

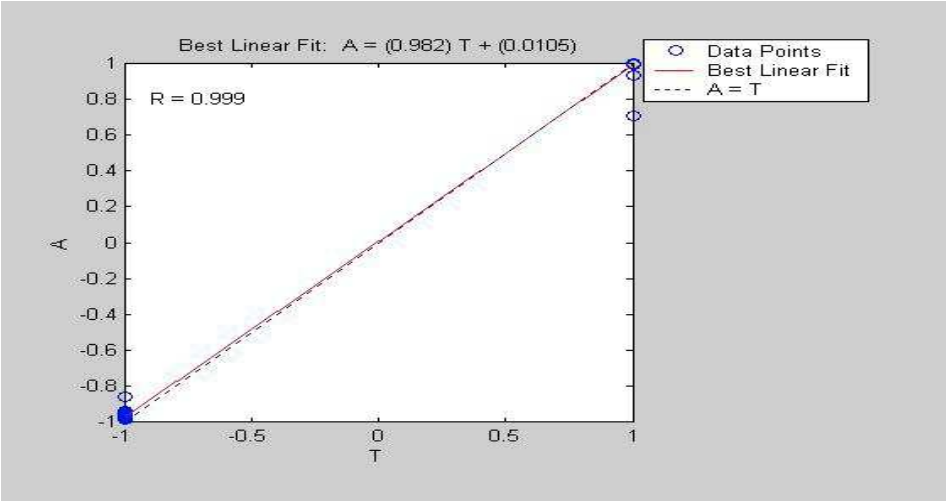


Figura 4.7: Información de entrenamiento para net4 lote04-ANGV.

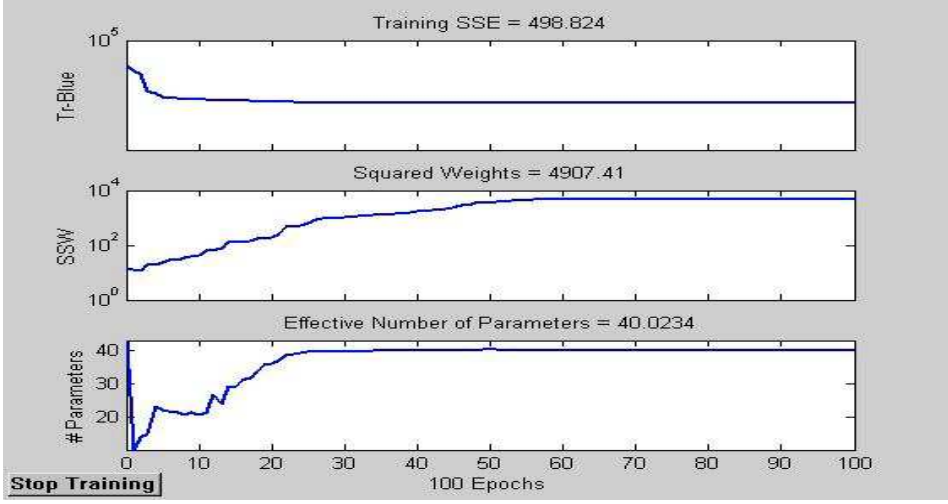


Figura 4.8: Información de regresión lineal para net4 lote04-ANGV.

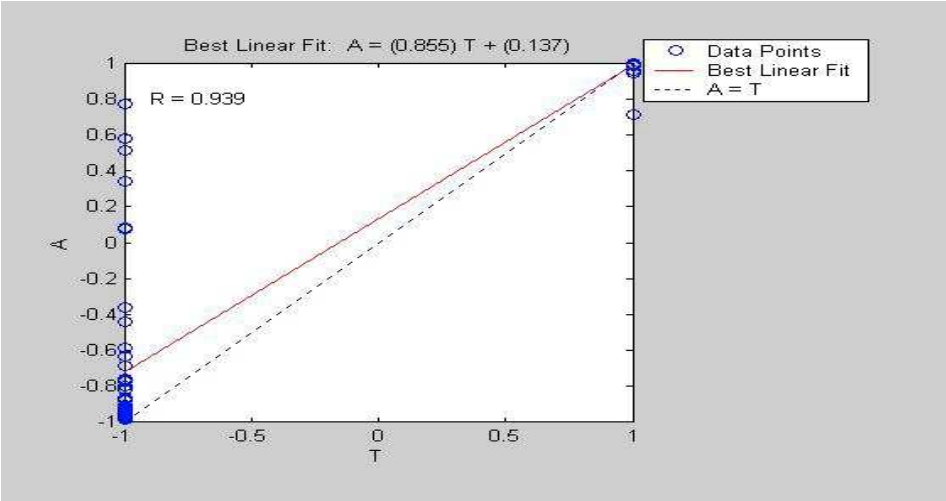


Figura 4.9: Información de entrenamiento para net4 lote05-ANGV.

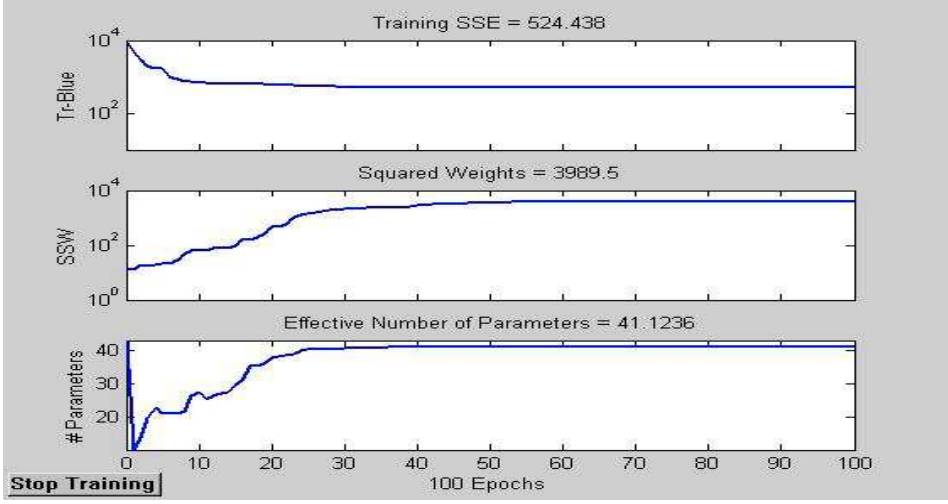


Figura 4.10: Información de regresión lineal para net4 lote05-ANGV.

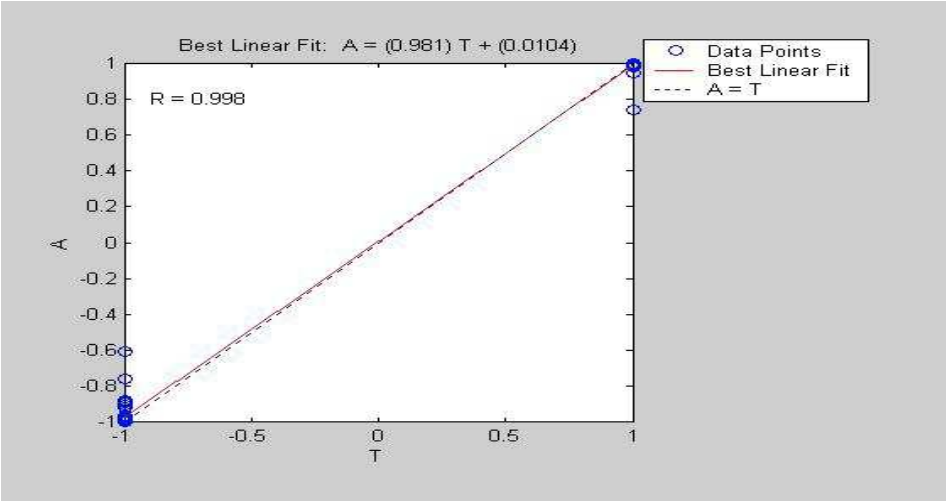


Figura 4.11: Información de entrenamiento para net3 lote06-ANGV.

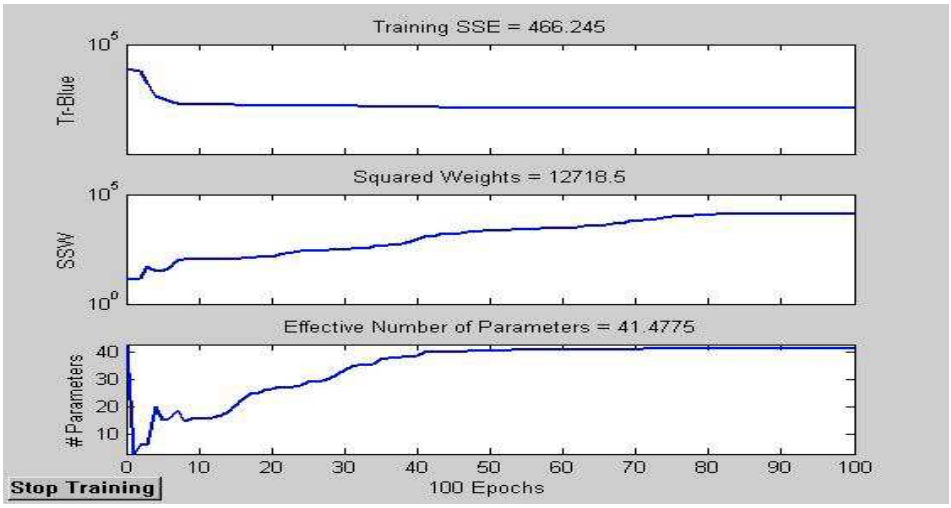


Figura 4.12: Información de regresión lineal para net3 lote06-ANGV.

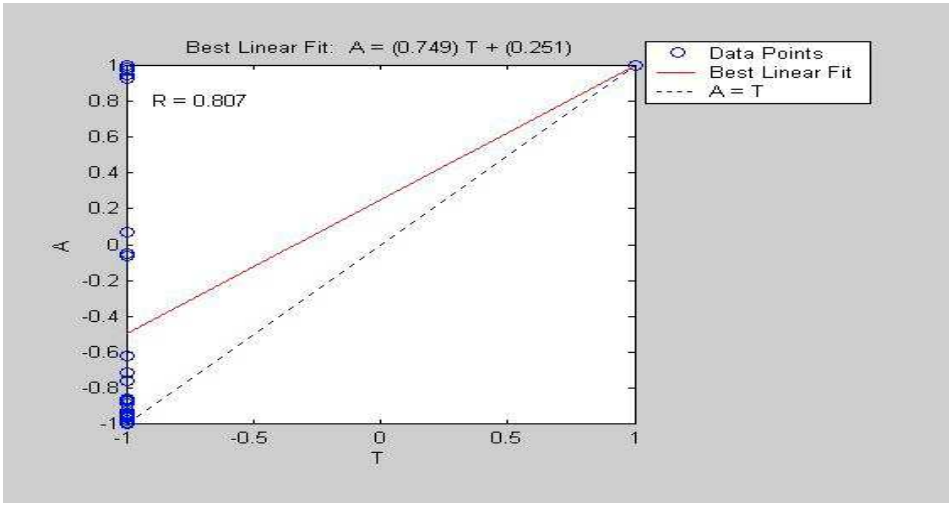


Figura 4.13: Información de entrenamiento para net4 lote07-ANGV.

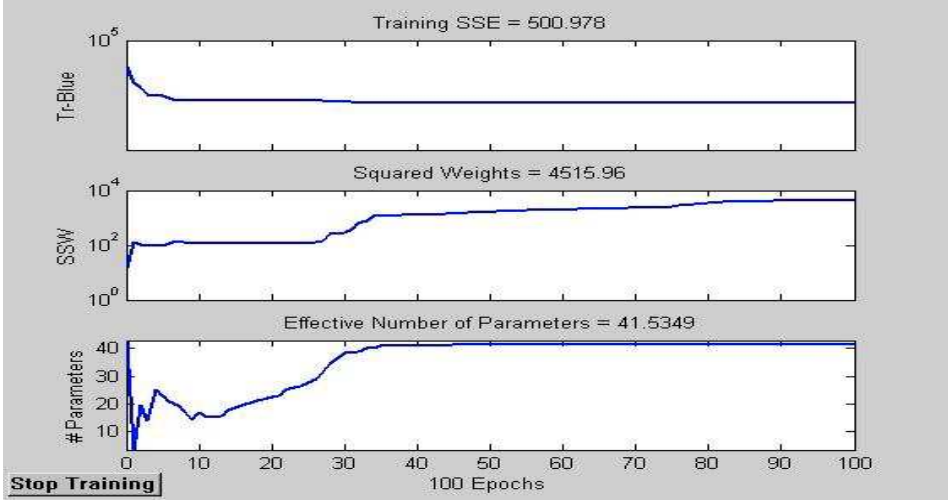


Figura 4.14: Información de regresión lineal para net4 lote07-ANGV.

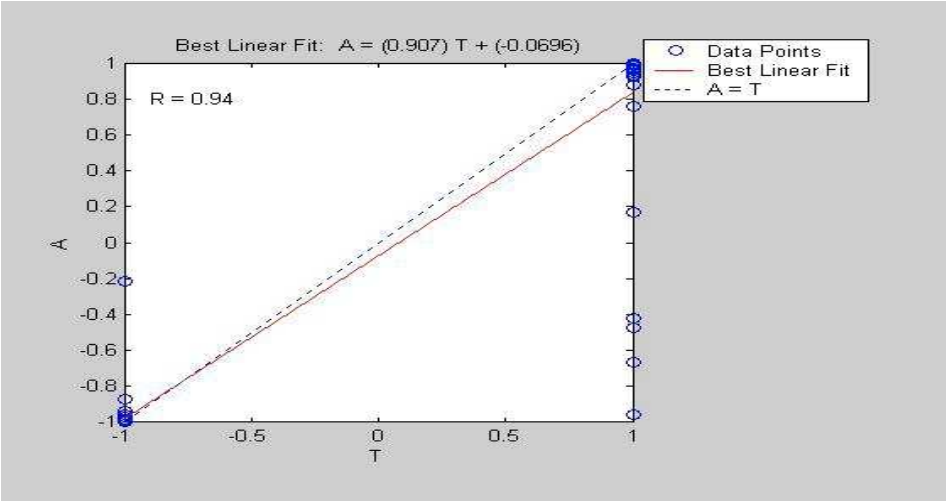


Figura 4.15: Información de entrenamiento para net5 lote08-ANGV.

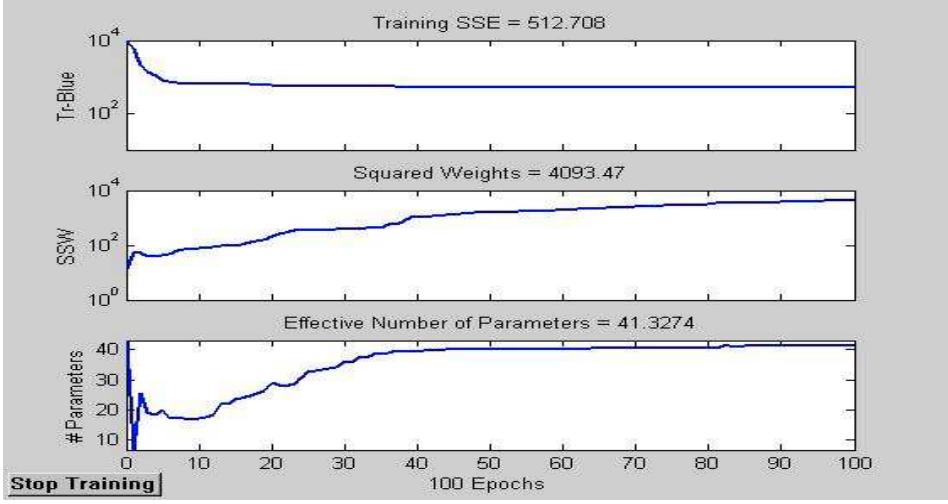


Figura 4.16: Información de regresión lineal para net5 lote08-ANGV.

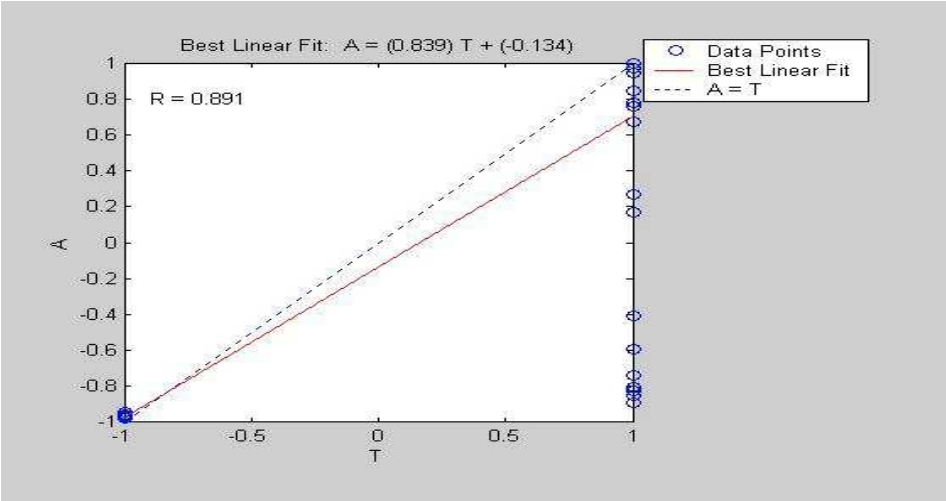


Figura 4.17: Información de entrenamiento para net6 lote09-ANGV.

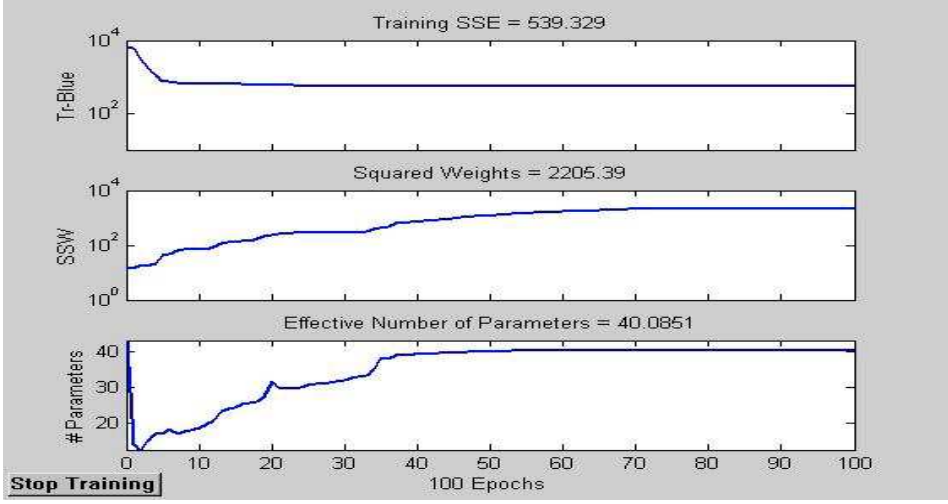


Figura 4.18: Información de regresión lineal para net6 lote09-ANGV.

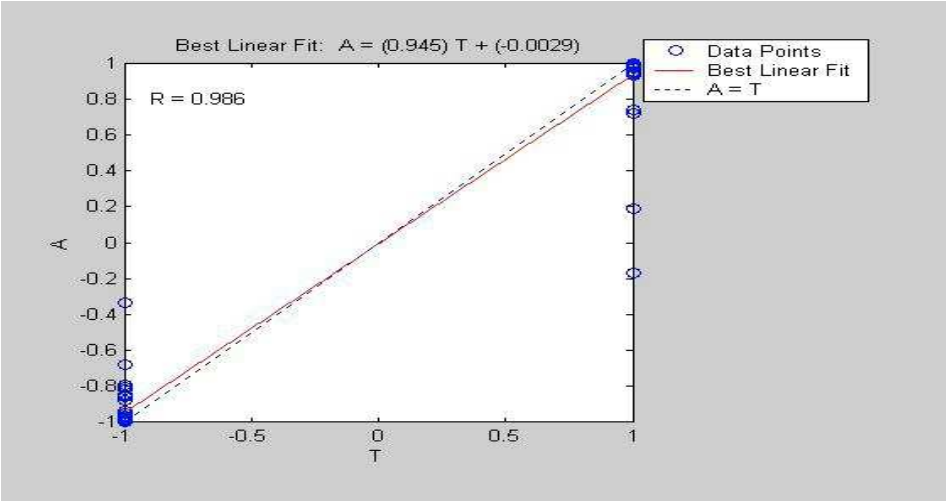


Figura 4.19: Información de entrenamiento para net4 lote10-ANGV.

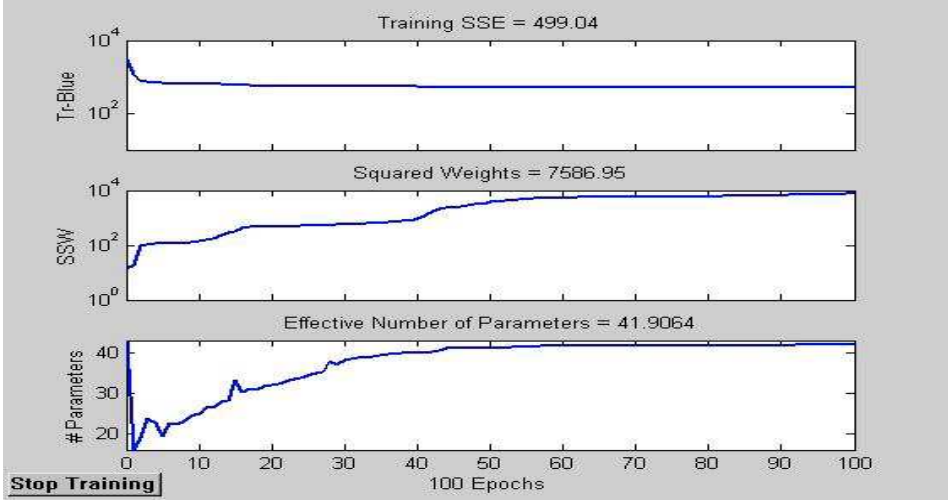
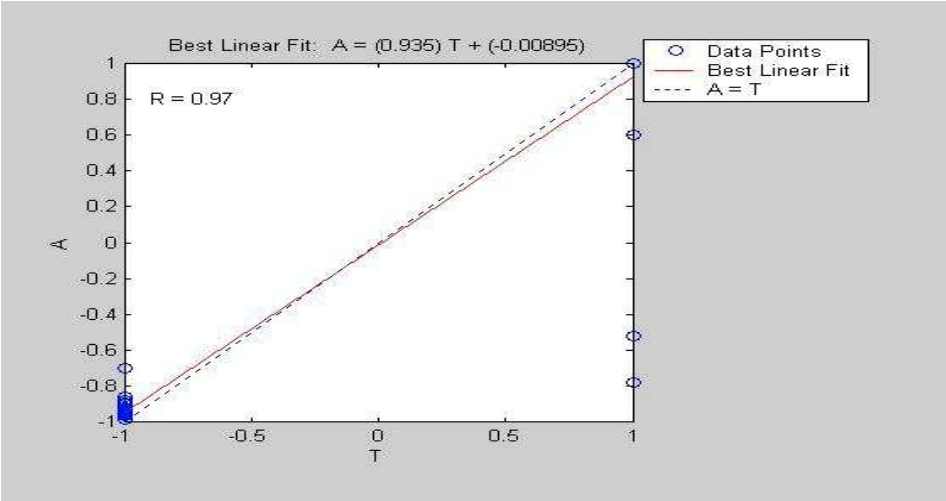


Figura 4.20: Información de regresión lineal para net4 lote10-ANGV.



4.2. RESULTADOS PARA LA ESTACIÓN CR2R. De igual forma se adopta la misma metodología, mostrando los siguientes resultados para CR2R:

Tabla 4.11: Resultados Lote de entrenamiento y prueba 01 CR2R.

Lote01- CR2R	net1	net2	net3	net4	net5	net6
Efectividad	91%	86%	84%	55%	85%	93%
m	0.7900	0.7441	0.7479	0.3093	0.7852	0.8919
b	-0.0536	0.0222	0.0127	0.3307	0.0040	-0.0389
r	0.8614	0.8213	0.8203	0.3960	0.8530	0.9195
time (s)	17.77	18.27	17.94	17.92	17.80	17.98

Tabla 4.12: Resultados Lote de entrenamiento y prueba 02 CR2R.

Lote02- CR2R	net1	net2	net3	net4	net5	net6
Efectividad	98%	98%	97%	97%	98%	97%
m	0.9208	0.9208	0.9288	0.9288	0.9208	0.9288
b	0.0763	0.0763	0.0626	0.0626	0.0763	0.0626
r	0.9952	0.9952	0.9872	0.9872	0.9952	0.9872
time (s)	17.93	17.94	17.92	18.02	18.05	17.71

Tabla 4.13: Resultados Lote de entrenamiento y prueba 03 CR2R.

Lote03- CR2R	net1	net2	net3	net4	net5
Efectividad	80%	88%	80%	80%	80%
m	0.7428	0.8266	0.7429	0.7428	0.7428
b	-0.0249	0.0811	-0.0249	-0.0249	-0.0249
r	0.8595	0.8960	0.8595	0.8595	0.8595
time (s)	17.50	17.45	17.51	17.57	17.71

Tabla 4.14: Resultados Lote de entrenamiento y prueba 04 CR2R.

Lote04- CR2R	net1	net2	net3	net4	net5
Efectividad	67%	68%	68%	69%	66%
m	0.5506	0.5629	0.5594	0.5663	0.5442
b	-0.1237	-0.1281	-0.1295	-0.1305	-0.1270
r	0.6518	0.6648	0.6610	0.6684	0.6404
time (s)	19.44	19.81	18.61	18.54	18.54

Tabla 4.15: Resultados Lote de entrenamiento y prueba 05 CR2R.

Lote05- CR2R	net1	net2	net3	net4	net5
Efectividad	92%	92%	92%	89%	92%
m	0.8912	0.8911	0.8911	0.8470	0.8911
b	0.1087	0.1088	0.1089	0.1529	0.1089
r	0.9672	0.9671	0.9670	0.9180	0.9671
time (s)	19.12	18.66	18.89	18.59	18.81

Tabla 4.16: Resultados Lote de entrenamiento y prueba 06 CR2R.

Lote06- CR2R	net1	net2	net3	net4	net5
Efectividad	85%	87%	85%	85%	87%
m	0.7714	0.7927	0.7714	0.7714	0.7927
b	0.0968	0.0775	0.0968	0.0968	0.0775
r	0.8782	0.9056	0.8782	0.8782	0.9056
time (s)	18.78	18.50	18.48	18.38	18.26

Tabla 4.17: Resultados Lote de entrenamiento y prueba 07 CR2R.

Lote07- CR2R	net1	net2	net3	net4	net5
Efectividad	92%	92%	90%	92%	92%
m	0.8579	0.8579	0.8394	0.8579	0.8579
b	0.0428	0.0428	0.0777	0.0428	0.0428
r	0.9435	0.9435	0.9409	0.9435	0.9435
time (s)	18.31	18.36	18.19	18.36	18.26

Tabla 4.18: Resultados Lote de entrenamiento y prueba 08 CR2R.

Lote08- CR2R	net1	net2	net3	net4	net5	net6
Efectividad	86%	86%	81%	86%	86%	86%
m	0.7785	0.7785	0.7282	0.7785	0.7954	0.7954
b	-0.0693	-0.0693	-0.0371	-0.0693	0.0143	0.0144
r	0.8519	0.8519	0.8220	0.8519	0.9129	0.9130
time (s)	18.22	18.37	18.28	18.17	17.98	18.38

Tabla 4.19: Resultados Lote de entrenamiento y prueba 09 CR2R.

Lote09- CR2R	net1	net2	net3	net4	net5
Efectividad	87%	90%	88%	88%	86%
m	0.7861	0.8176	0.7959	0.7957	0.7625
b	-0.1390	-0.1490	-0.1242	-0.1244	-0.1891
r	0.8789	0.8871	0.8883	0.8881	0.8472
time (s)	18.41	18.51	18.50	18.60	18.47

Tabla 4.20: Resultados Lote de entrenamiento y prueba 10 CR2R.

Lote10- CR2R	net1	net2	net3	net4	net5
Efectividad	71%	71%	71%	71%	71%
m	0.5765	0.5765	0.5765	0.5765	0.5765
b	-0.2509	-0.2509	-0.2509	-0.2509	-0.2509
r	0.7491	0.7491	0.7491	0.7491	0.7491
time (s)	18.43	18.49	18.44	18.58	18.52

Igualmente teniendo en cuenta la mejor net para cada lote, se presentan las siguientes gráficas de entrenamiento y regresión lineal.

Figura 4.21: Información de entrenamiento para net6 lote01-CR2R.

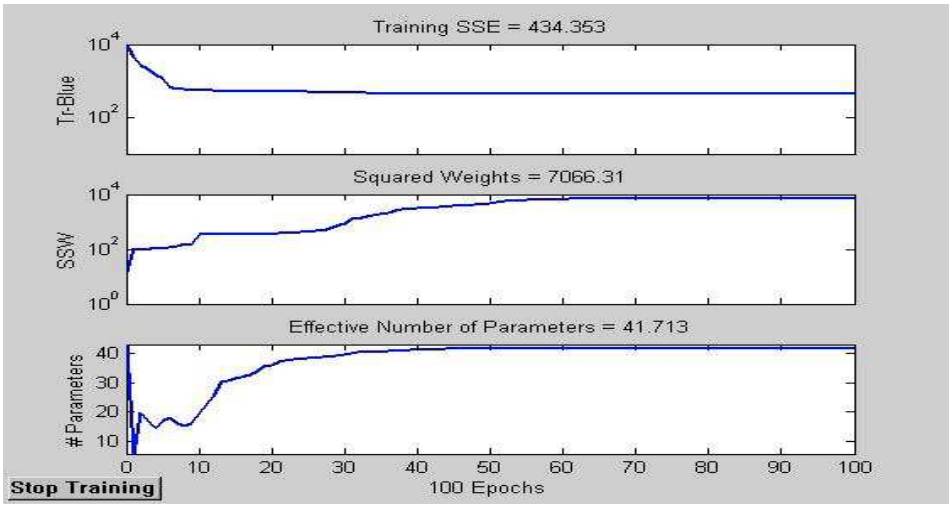


Figura 4.22: Información de regresión lineal para net6 lote01-CR2R.

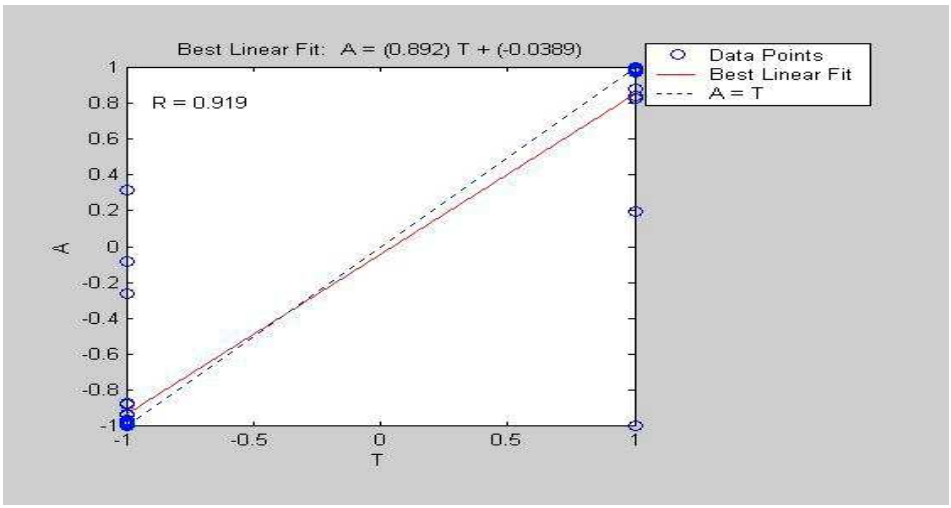


Figura 4.23: Información de entrenamiento para net1 lote02-CR2R.

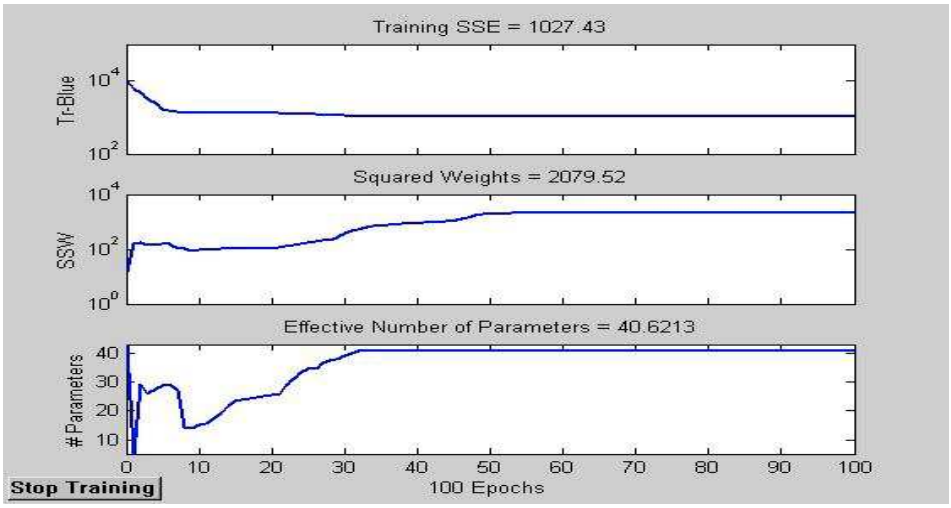


Figura 4.24: Información de regresión lineal para net1 lote02-CR2R.

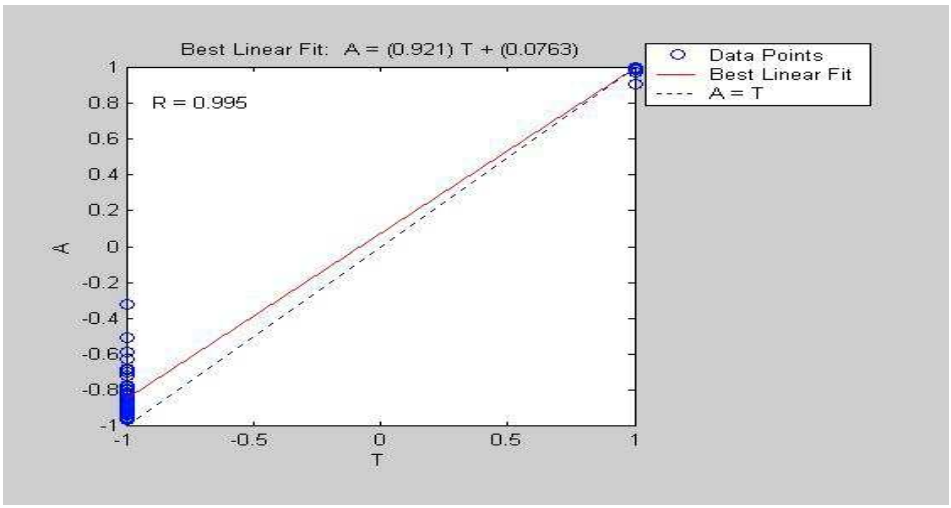


Figura 4.25: Información de entrenamiento para net2 lote03-CR2R.

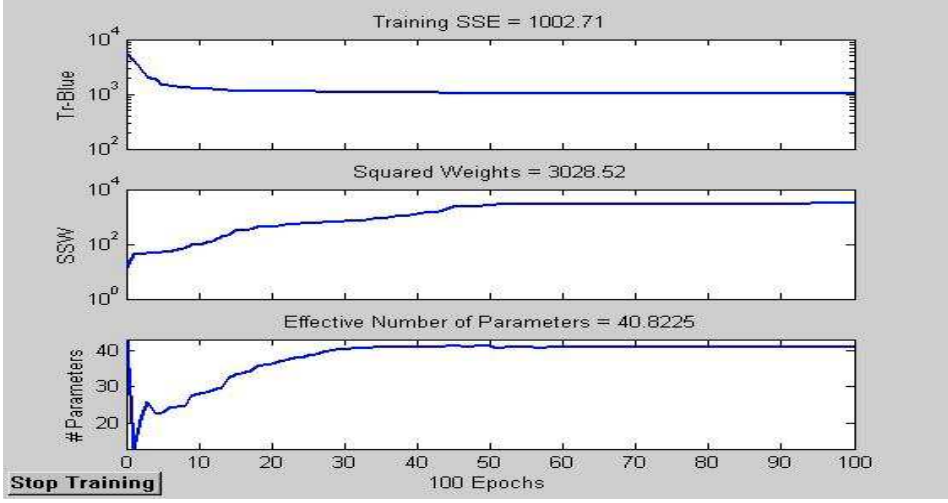


Figura 4.26: Información de regresión lineal para net2 lote03-CR2R.

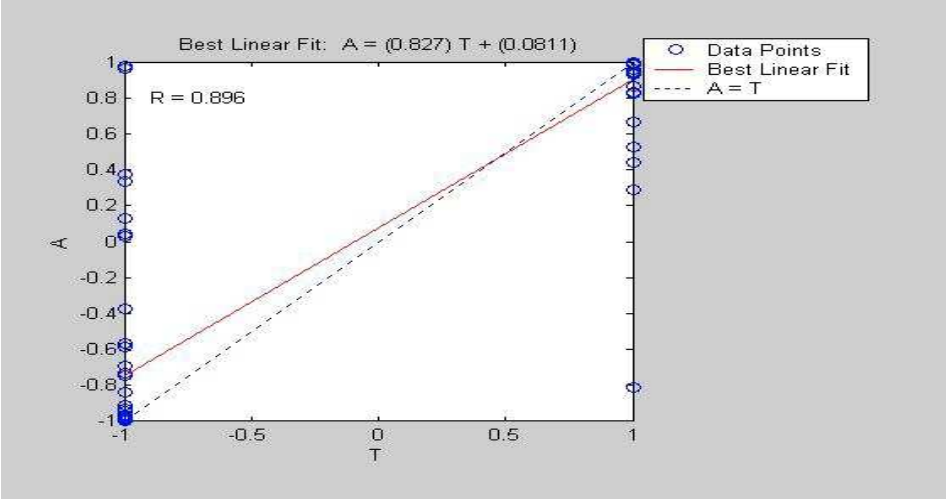


Figura 4.27: Información de entrenamiento para net4 lote04-CR2R.

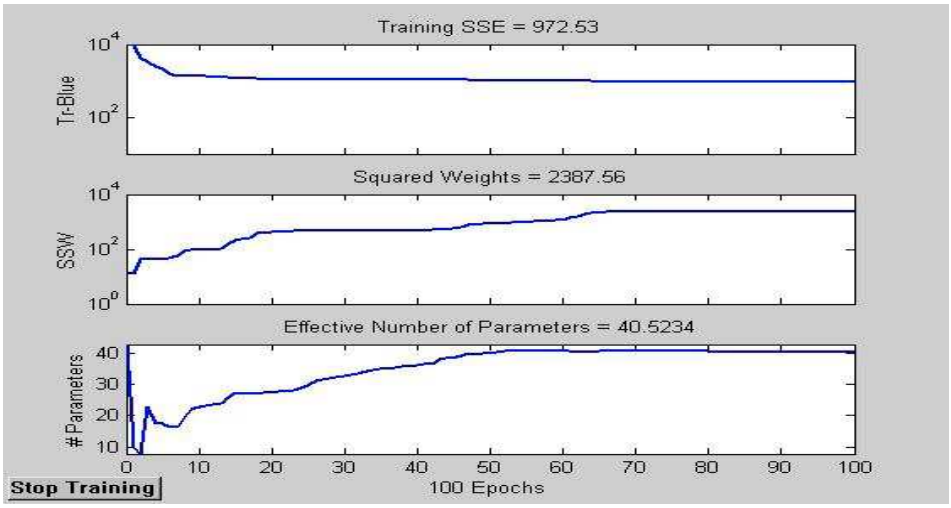


Figura 4.28: Información de regresión lineal para net4 lote04-CR2R.

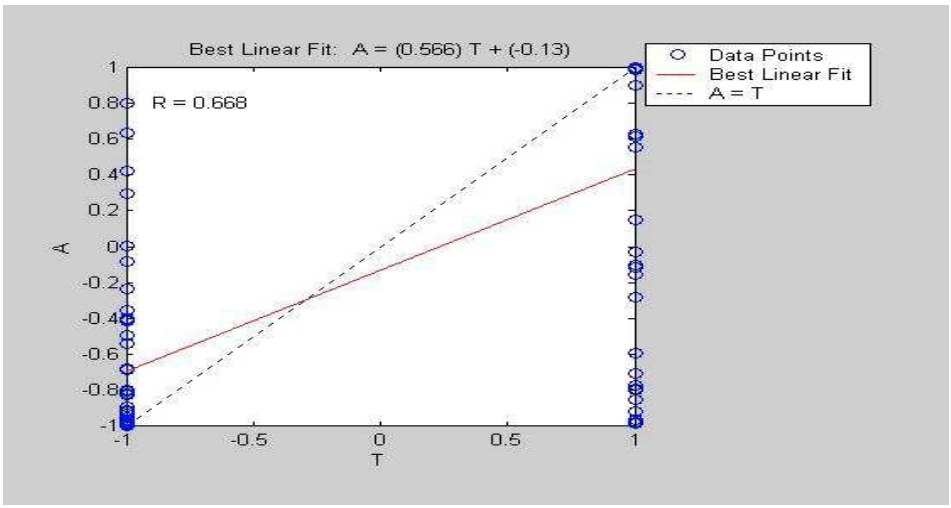


Figura 4.29: Información de entrenamiento para net1 lote05-CR2R.

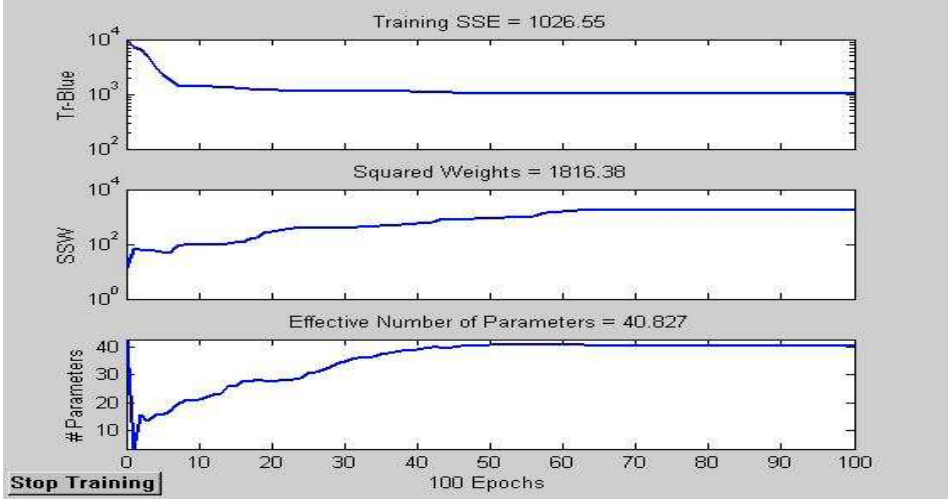


Figura 4.30: Información de regresión lineal para net1 lote05-CR2R.

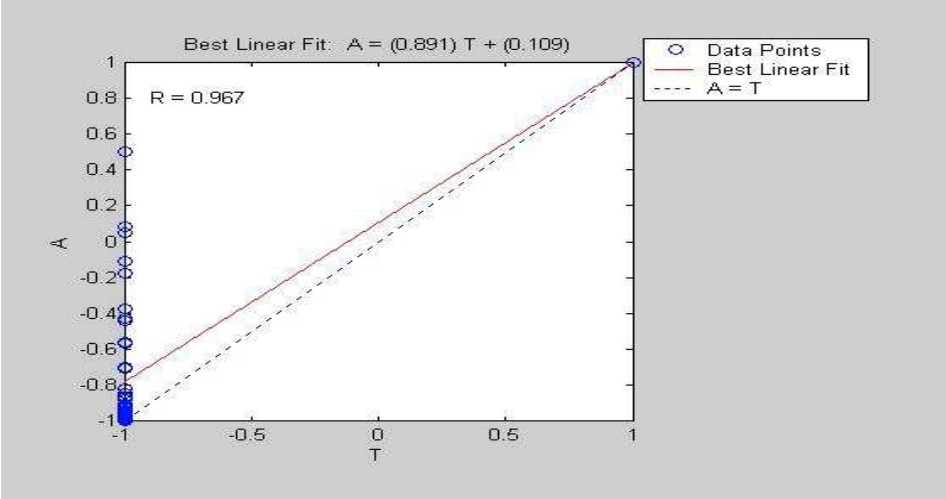


Figura 4.31: Información de entrenamiento para net5 lote06-CR2R.

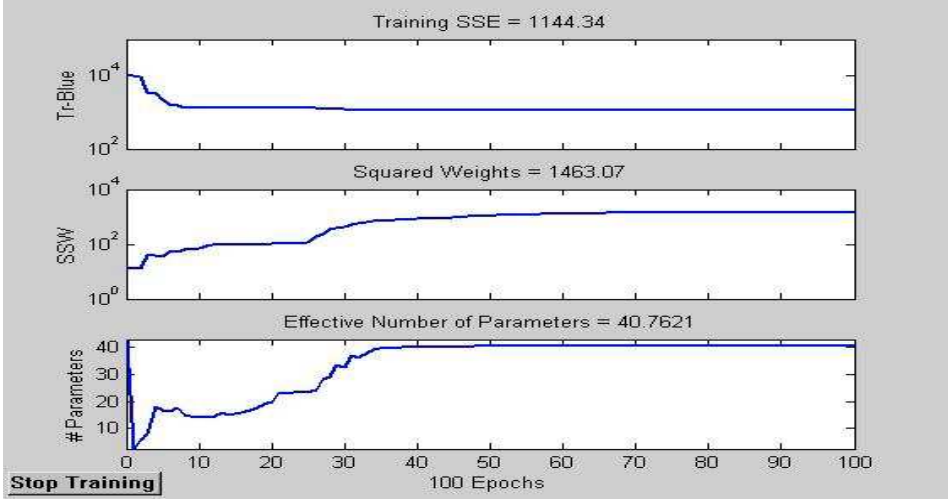


Figura 4.32: Información de regresión lineal para net5 lote06-CR2R.

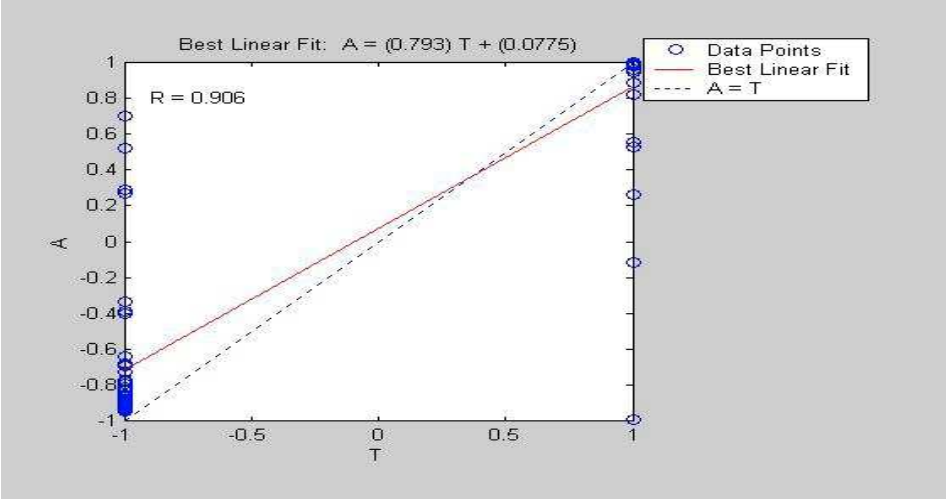


Figura 4.33: Información de entrenamiento para net5 lote07-CR2R.

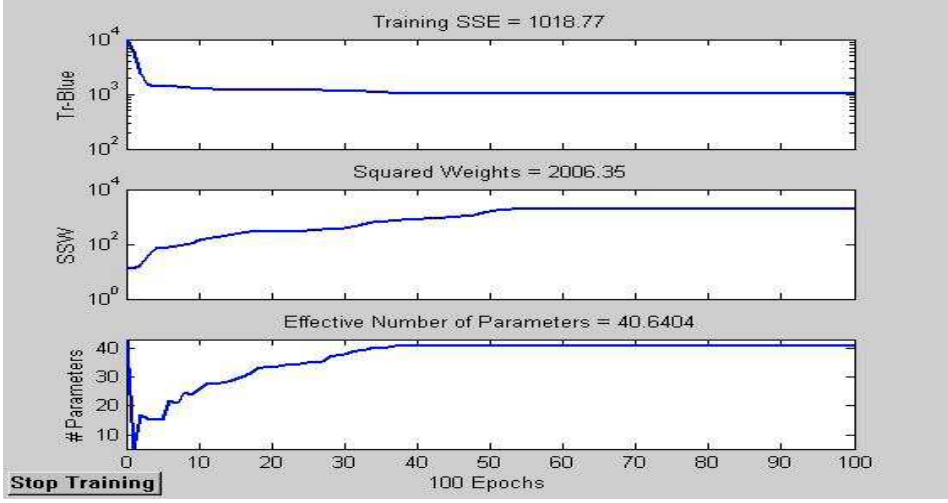


Figura 4.34: Información de regresión lineal para net5 lote07-CR2R.

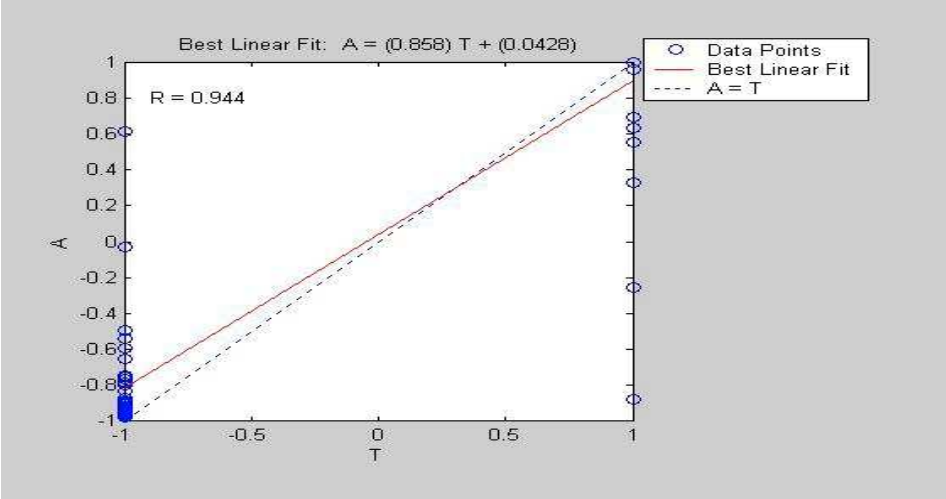


Figura 4.35: Información de entrenamiento para net6 lote08-CR2R.

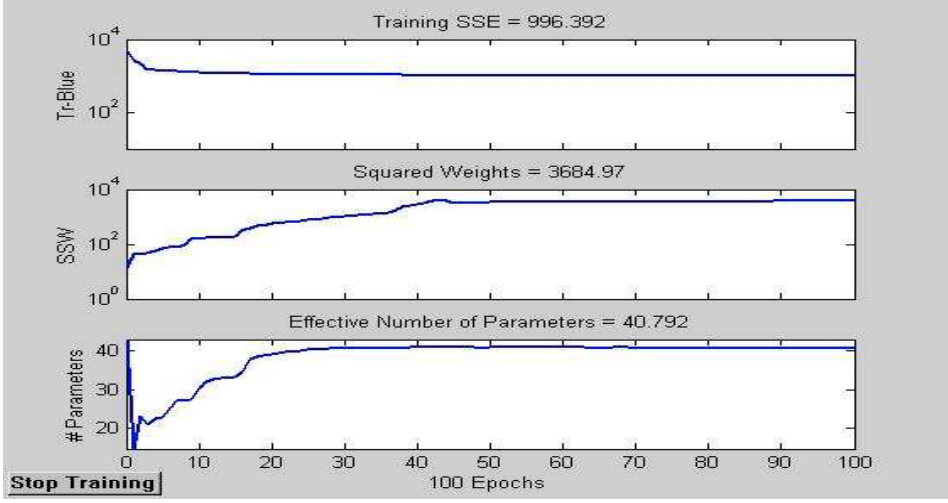


Figura 4.36: Información de regresión lineal para net6 lote08-CR2R.

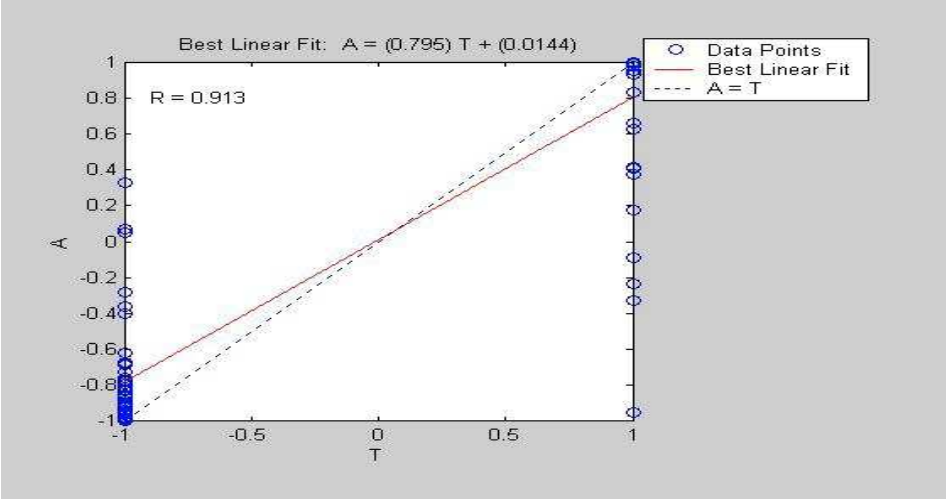


Figura 4.37: Información de entrenamiento para net2 lote09-CR2R.

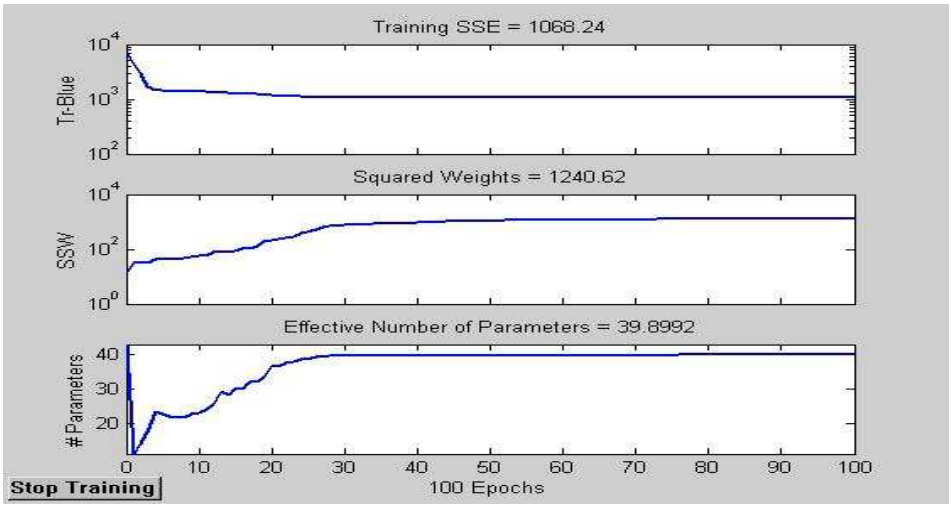


Figura 4.38: Información de regresión lineal para net2 lote09-CR2R.

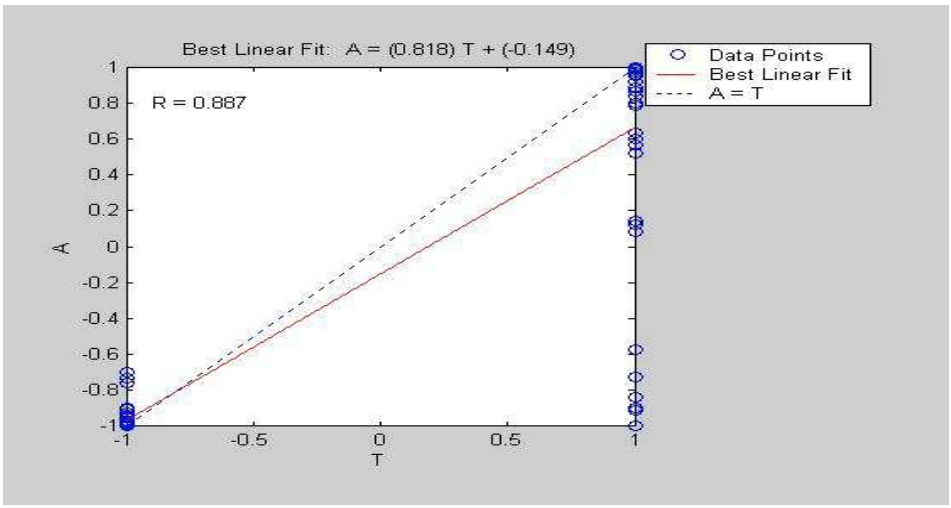


Figura 4.39: Información de entrenamiento para net1 lote10-CR2R.

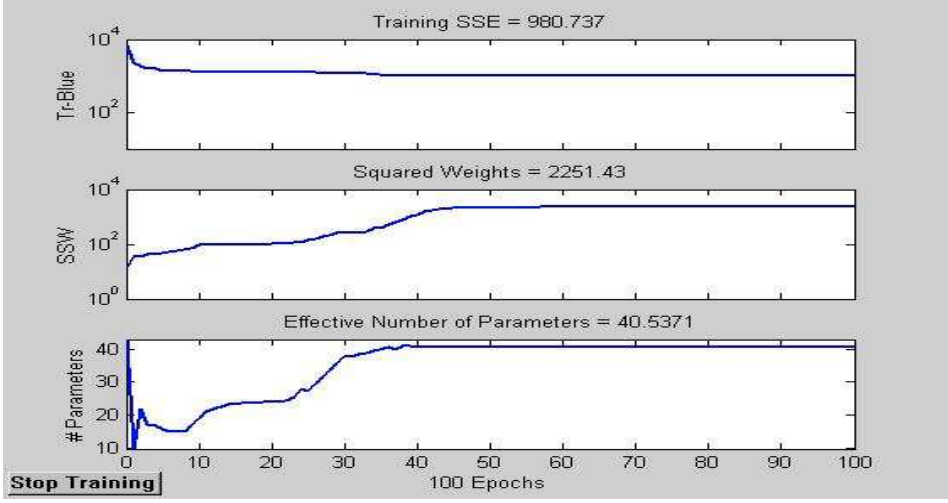
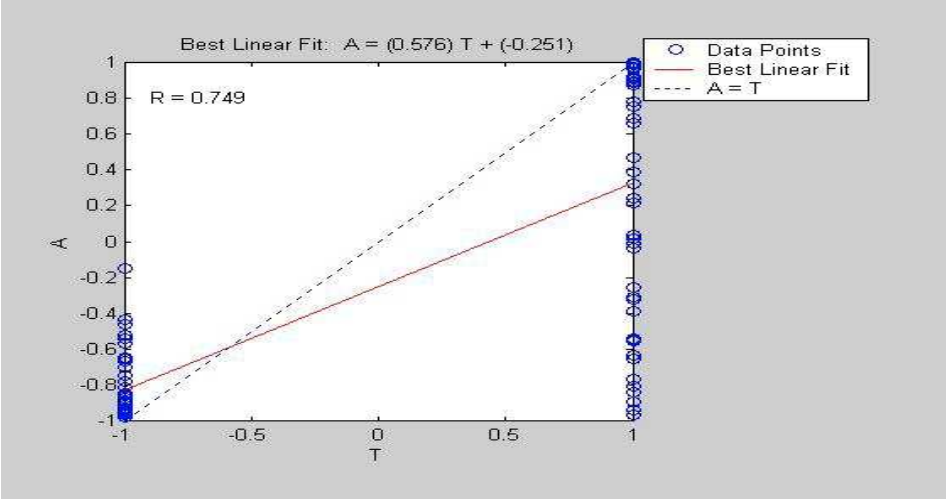


Figura 4.40: Información de regresión lineal para net1 lote10-CR2R.



CONCLUSIONES

Se logró representar 300 muestras equivalentes a 3 segundos de registro en 12 coeficientes de predicción lineal (LPC) lo cual es muy aprovechable para la minimización del tiempo de cómputo en la fase de entrenamiento, esto tanto para la información de la estación ANGV como para CR2R.

La metodología tratada en el desarrollo del trabajo arrojó muy buenos resultados y se logra la meta propuesta, evidenciando la capacidad enorme de las Redes Neuronales Artificiales adoptando el modelo Perceptron Multicapa para solventar este tipo de propuestas.

La topología de conexionado de las neuronas se ejecuto con 3 capas, una de entrada, una oculta y una de salida, mostrando mejores resultados la configuración 12-3-1 para ANGV y para CR2R 12-3-1, es decir, con 3 neuronas en la capa oculta.

Los mejores resultados en promedio de cantidad de aciertos para ANGV se obtuvieron con el lote de entrenamiento-03 siendo del 100 % y para CR2R el lote de entrenamiento-02 de 97.5%.

BIBLIOGRAFÍA

CADENA IBARRA, Oscar Ernesto. [9] DISCRIMINACION DE REGISTROS SISMICOS TIPO LPS, VTA Y HBD ORIGINADOS POR EL VOLCAN GALERAS UTILIZANDO REDES NEURONALES ARTIFICIALES. [Tesis de Grado en Física]. San Juan de Pasto: Universidad de Nariño. Facultad de Ciencias Naturales y Matemáticas; Departamento de Física; 2006.

DAI, H. C.,MACBETH, C. [12] AUTOMATIC PICKING OF SEISMIC ARRIVALS IN LOCAL EARTHGUAKE DATA USING AN ARTIFICIAL NEURAL NETWORK, 1995, Geophysics. J. Int., 120, pp.758-774.

DEL BRIO, Bonifacio. SANZ, Alfredo. [1] REDES NEURONALES Y SISTEMAS DIFUSOS, Universidad de Zaragoza. 2002.

DEMUTH, Howard. BEALE, Mark. [8] NEURONAL NETWORK TOOLBOX, MATLAB 6.0. 2002.

GUADA BARRAÉZ, Carlos. [11] DETECTOR DE EVENTOS SISMICOS EN TIEMPO REAL UTILIZANDO REDES NEURONALES. Interciencia, Revista de Ciencia y Tecnología de América, 2000; vol. 25, No. 006: pp.293-298.

HAYKIN, Simon. [6] NEURONAL NETWORKS, McMaster University. 1994.

ISLAM, Tamaña. [3] INTERPOLATION OF LINEAR PREDICTION COEFICIENTS FOR SPEECH CODING, McGill University. Montreal Canada. 2000.

LINEAR PREDICTION COEFICIENTS. [5] [Sitio en Internet] http://arantxa.ii.uam.es. Tema3 TAPS.

LINEAR PREDICTIVE CODING (LPC). [10] [Sitio en Internet] http://en.wikipedia.org/wiki/Linear predictive coding.

NEURAL NETWORKS FRAMEWORK. [7] [Sitio en Internet] http://www.redesneuronales.netfirms.com.

NIEWIADOMSKI, Janusz. [13] EXTRACTION OF INFORMATION FROM SEIS-MOGRAMS BY NEURAL NETWORKS. Institute of Geophysics, Polish Academy of Sciences,Warszawa, Poland. 2004; ACTA GEOPHYSICA POLONICA; Vol. 52, No. 2.

PROAKIS, John. MANOLAKIS, Dimitris. [2] TRATAMIENTO DIGITAL DE SEÑALES. 1998.
STRUCTURAL DAMAGE DETECTION USING AN INTERACTIVE NEURAL NETWORK JOURNAL OF INTELLIGENT MATERIAL SYSTEMS AND STRUCTURES, [15] 2000; Vol. 11, No. 1, pp. 32-42.

THE MATHWORK, Inc. MATLAB, edición de estudiante. [4] Guía de usuario. Versión 4. 1995.

WANG, Jim. TENG, Ta-ling. [14] ARTIFICIAL NEURAL NETWORK BASED SEISMIC DETECTOR. Bulletin of Seismological Society of America, 1995; vol. 85, pp.308-319.

ANEXOS

En el CD adjunto se incluye:

Anexo A. RUTINAS EN MATLAB.

Se consignan las rutinas empleadas, desarrolladas en MATLAB:

(a). delimit.m
(b). partir.m
(c). union_eventos.m
(d). union_ruido.m
(e). ordenang.m
(f). ordencr.m
(g). unosan.m
(h). unoscr.m

Anexo B. INFORMACIÓN DE ENTRENAMIENTOS Y REGRESIÓN LINEAL.

Las gráficas, información del entrenamiento y regresión lineal de la totalidad de entrenamientos tanto para ANGV como para CR2R.

Anexo C. RESULTADOS DE CADA ENTRENAMIENTO.

Tablas en formato Excel que contienen los resultados obtenidos al efectuar la respectiva prueba de la red en cada entrenamiento con cada lote de datos.

Anexo D. CONFIGURACIONES DE RNA.

Las respectivas configuraciones de RNA adquiridas durante los procesos de entrenamiento, llamadas net.