UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERIA
DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y COMPUTACION

DETERMINACION DE FASES SISMICAS PN USANDO REDES NEURONALES

FRANKLIN ALBERTO RENGIFO MORA

BOGOTA
2002

CONTENIDO

INTRODUCCIÓN

El cálculo del tiempo de arribo de fases sísmicas juega un papel importante en sismología, varias técnicas usando estos tiempos de arribo han sido desarrolladas con el objeto de crear modelos de la estructura de velocidades del manto terrestre (ref. 10.). La fase sísmica que se usa en este trabajo es la onda Pn (figura 1.). Debido a que las redes sismológicas se han expandido dramáticamente en los últimos años, y más datos deben ser procesados, el procedimiento de marcar el arribo de fases sísmicas ocupa cada vez más tiempo a los sismólogos. Por lo tanto un sistema eficiente y preciso para marcar las fases es un objetivo en la moderna sismología, el cual también permitiría la implementación de un sistema automático de localización de epicentros.

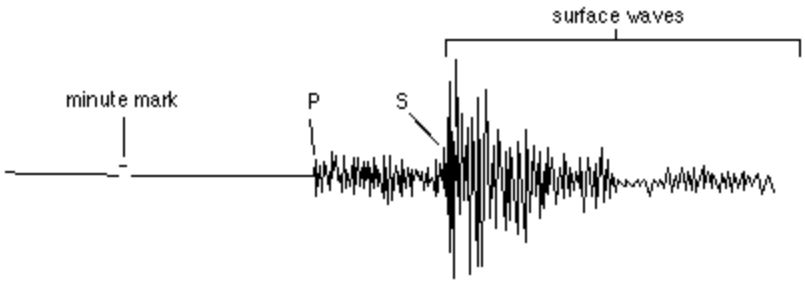

Figura 1

La tarea de marcar una fase sísmica se divide en dos partes: primero, determinar la existencia de la fase, y segundo, determinar el tiempo de arribo de esta. Existen varios métodos para detectar las fases sísmicas, los cuales se clasifican en cuatro categorías: en el dominio del tiempo, el dominio de la frecuencia, procesamiento del movimiento de la partícula, comparación con patrones. El método más usado es el algoritmo de STA/LTA en el dominio del tiempo. Este método usa para detectar la onda, la razón entre el promedio de una ventana pequeña de tiempo, y el promedio en una ventana grande de tiempo. Además trabaja bien para ondas con contenido frecuencial que se ajustan a las preseleccionadas ventanas de

tiempo, pero no muy bien para otras.

En este trabajo, se hace uso de redes neuronales para determinar el arribo de fases sísmicas Pn originadas en el Nido de Bucaramanga, las cuales son registradas por la estación sismológica ubicada en el municipio de Barichara (Santander), perteneciente a la RED SISMOLÓGICA NACIONAL DE COLOMBIA (anexo A). El número de registros sísmicos usados como conjunto de patrones de entrenamiento fue de: 848 patrones de onda Pn y 940 patrones de ruido, mientras que el número de patrones para la validación de la red neuronal fue en total de 840.

CAPITULO 1 MARCO TEÓRICO

1.1 RED NEURONAL

Las redes neuronales están en esencia inspiradas en el sistema nervioso, estas están conformadas principalmente por una red de elementos de simple procesamiento que están densamente interconectadas, y ejecutan de manera análoga las más elementales funciones de una neurona biológica.

Una breve historia del desarrollo de las redes neuronales y una introducción básica a la teoría de estas, es presentada en este capitulo. Además se discute temas relacionados con métodos para reducir la complejidad en la arquitectura de las redes neuronales.

1.2 PERCEPTRON

La idea de implementar un modelo de neurona emergió en 1940 con el trabajo de McCulloch y Pitts. El movimiento de la cibernética el cual intento combinar biología, phisicologia, ingeniería y matemáticas resulto en arquitecturas para redes neuronales, las cuales podían ejecutar gran variedad de tareas.

En 1950 la investigación continua inicialmente en el desarrollo de redes para ejecutar tareas especificas, pero luego el objetivo se encaminó en desarrollar maquinas que pudieran aprender. Posteriormente, cerca del final de esa década, no se realizaron desarrollos de gran importancia.

En la década de 1960 el interés fue revivido con la publicación de un libro realizado por Rosenblatt donde él define el concepto del PERCEPTRON, y desarrolla algunas teorías sobre ellos. En su más simple forma, estos elementos de procesamiento son conocidos como nodos o neuronas artificiales, los cuales

tienen la estructura ilustrada en la figura 2.

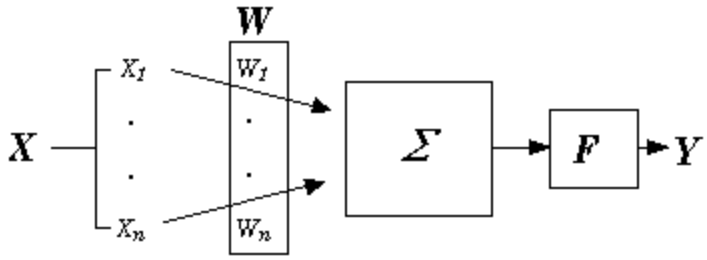

Figura 2

Un conjunto de entradas (X1 a Xn) es aplicado a cada nodo representando las entradas desde el mundo exterior, o también pueden ser salidas de otros nodos. Cada entrada es multiplicada por un peso (W1 a Wn) asociado con los nodos de entrada a los cuales esta conectado, luego las entradas ponderadas por los pesos son sumadas.

$$Y = F(\sum_{n=1}^{N} (X_n * W_n) + C)$$

Un valor (C) de umbral para cada nodo es adicionado a la suma ponderada de los pesos, y la suma resultante se pasa a través de una función de activación.

El perceptron efectivamente divide el patrón de entrada dentro de dos regiones distintas, una región siendo representada por un uno, y la otra por un cero. El algoritmo de entrenamiento de Rosenblatt para el perceptron podía converger si los patrones de entrada eran linealmente separables. El perceptron podía por lo tanto aproximar la frontera de decisión entre las dos clases de salidas.

Los perceptrones fueron exitosamente entrenados para ejecutar ciertas tareas, pero tuvieron errores que no podían ser solucionados. Minsky y Papert señalaron

serios defectos de los perceptrones, y el interés en esta área de nuevo declino.

La función exclusiva-or es la principal ilustración de la limitación de los perceptrones. Una salida de uno es generada si las entradas son (0,1) o (1,0), y una salida de cero si las entradas son (0,0) o (1,1). Esta no es una función linealmente separable así que el perceptron no puede aprenderla. Así que una más complicada superficie de decisión es requerida.

1.3 PERCEPTRONES MULTICAPA

Minsky y Papert propusieron una solución al problema encontrado en la función exclusiva-or. Ellos sugirieron que una capa extra de nodos con funciones de activación no lineal podría ser introducida. La salida podría ahora no ser una combinación lineal de las entradas, así que una más complicada superficie de decisión podría ser presentada. El problema en es momento, fue el no existir un algoritmo de entrenamiento para una red de perceptrones.

1.3.1 ALGORITMO BACK - PROPAGATION

Este algoritmo está basado en el gradiente de la superficie de error, donde los pesos y los bias, son ajustados para minimizar una función de costo igual al error cuadrático medio en la red.

Para una red neuronal de tres capas con N nodos de entrada y M nodos de salida, los pesos de la red son inicialmente inicializados a pequeños valores aleatorios. Un vector P de entrada Xp0, Xp1... XpN-1 con su respectivo vector de salida T0,T1,....TM-1, es presentado a la red.

De este vector de entrada, la red obtiene un vector de salida el cual puede ser comparado con el vector de salida correspondiente al vector patrón de entrada. Si

no hay diferencia entre el producido y el patrón del vector de salida, no hay aprendizaje. Si hay diferencia, los pesos son cambiados para reducir el error. Los pesos son adaptados usando un algoritmo recursivo, el cual empieza en los nodos de salida, y trabaja hacia atrás hasta la capa de nodos ocultos.

El error en la red, cuando el vector patrón de entrada y salida es presentado, se define como:

$$E_p = 1/2 \sum (t_{pj} - o_{pj})^2 \qquad (2.1)$$

donde:

- t_{pj} es el valor de salida esperado para el nodo de salida j esimo.
- O_{pj} es el valor obtenido por la red para el nodo de salida j esimo cuando se presenta el patrón a la red como vector de entrada.

Por lo tanto el error total es

$$E = \sum_p E_p \qquad (2.2)$$

La entrada al nodo j es

$$net_{pj} = \sum_i w_{ji} o_{pi} \qquad (2.3)$$

donde

- W_{ji} es el peso desde el nodo i de la capa anterior al jodo j en el tiempo t, cuando los patrones de entrada y salida son presentados a la red neuronal.
- O_{pi} es la salida del nodo i de la anterior capa.

Una función de activación no lineal es empleada en cada nodo tal que la salida del nodo j es:

$$o_{pj} = f_j(net_{pj}) \qquad (2.4)$$

Para implementar un gradiente descendiente en la superficie de error, el negativo de la derivada de Ep con respecto a wij debe ser proporcional al cambio en el peso wij. Por lo tanto,

$$\Delta_p w_{ji} \propto -\frac{\delta E_p}{\delta w_{ji}} \qquad (2.5)$$

Aplicando la regla de la cadena a (2.5) se obtiene

$$\frac{\delta E_p}{\delta w_{ji}} = \frac{\delta E_p}{\delta net_{pj}} \frac{\delta net_{pj}}{\delta w_{ji}} \qquad (2.6)$$

Desde (2.3) se obtiene

$$\frac{\delta net_{pj}}{\delta w_{ji}} = \frac{\delta}{\delta w_{ji}} \sum_i w_{ji} o_{pi} = o_{pi} \qquad (2.7)$$

Al aplicar la regla de la cadena a (2.6) se calcula

$$\frac{\delta E_p}{\delta net_{pj}} = \frac{\delta E_p}{\delta o_{pj}} \frac{\delta o_{pj}}{\delta net_{pj}} \qquad (2.8)$$

Desde (2.1) resulta que

$$\frac{\delta E_p}{\delta o_{pj}} = \frac{\delta}{\delta o_{pj}} 1/2 \sum (t_{pj} - o_{pj})^2 = -(t_{pj} - o_{pj}) = -\delta_{pj} \qquad (2.9)$$

Desde (2.4) se obtiene

$$\frac{\delta o_{pj}}{\delta net_{pj}} = f_j'(net_{pj}) \qquad (2.10)$$

Substituyendo (2.7), (2.9) y (2.10) dentro de (2.6) da como resultado

$$-\frac{\delta E_p}{\delta w_{ji}} = -(-\delta_{pj}) f_j'(net_{pj}) x_{pi} = \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.11)$$

Como se mencionó al comienzo, la derivada negativa de Ep con respecto al peso wij es proporcional al cambio en el peso Wij, por lo tanto

$$\Delta_p w_{ji} \propto \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.12)$$

Entonces

$$\Delta_p w_{ji} = \eta \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.13)$$

donde la constante de proporcionalidad es conocida como la tasa de aprendizaje.

La ecuación (2.13) es conocida como la regla estándar, y define como los pesos son cambiados después de la presentación de los patrones de entrada y salida a la red. La función de activación debe ser no lineal. Porque si no, la red neural podría ejecutar una transformación lineal entre cada capa, y por lo tanto se reduce a su equivalente de una simple capa, la efectividad de la capa extra de perceptrones es entonces perdida. La función de activación debe ser además diferenciable como requiere la ecuación (2.11). La función sigmoid es la más usada ya que esta satisface todos los anteriores requerimientos.

1.3.2 MOMENTUM

Algunos investigadores como D. E. Rumelhart, G. E. Hinton y R. J. Williams propusieron una modificación al algoritmo de propagación inversa (Ref 1.). La idea es adicionar una "inercia" a las formulas de actualización de los pesos, este término es proporcional a la dirección anterior de variación de los pesos. Este valor adicional hace que el cambio en los pesos sea más suave, y así filtra las variaciones de alta frecuencia de la superficie de error en el espacio de los pesos, y por lo tanto tendría una convergencia más rápida que si se hace uso del algoritmo básico de propagación inversa.

La actualización de los pesos seria entonces:

$$\Delta w_{ij}(t+1) = h * d_j * O_i + a * \Delta w_{ij}(t) \qquad 2.14$$

Donde α es el factor que determina la importancia del momentun en la actualización de los pesos.

1.3.3 CONEXIONES EN LA RED NEURONAL

La más común de las redes neuronales es la de tres capas totalmente conectada y alimentada hacia adelanta, un ejemplo es mostrado en la figura 3. Los nodos están ordenados en tres capas; una capa de entrada, una capa oculta, y una capa de salida con entradas fluyendo hacia adelante desde la capa de entrada hasta la capa de salida a través de la capa oculta, excepto en el entrenamiento como se menciono en la sección anterior. En este tipo de red, las entradas de cada nodo en la capa oculta son conectadas a las salidas de cada nodo en la capa de entrada, y las entradas de cada nodo en la capa de salida, son conectadas a las salidas de cada nodo en la capa oculta. Los nodos en la capa de entrada son usados para

introducir las señales externas dentro de la red neuronal, y los nodos en la capa de salida, son usados para hacer la decisión final, y transmitir la señal producida hacia fuera del mundo.

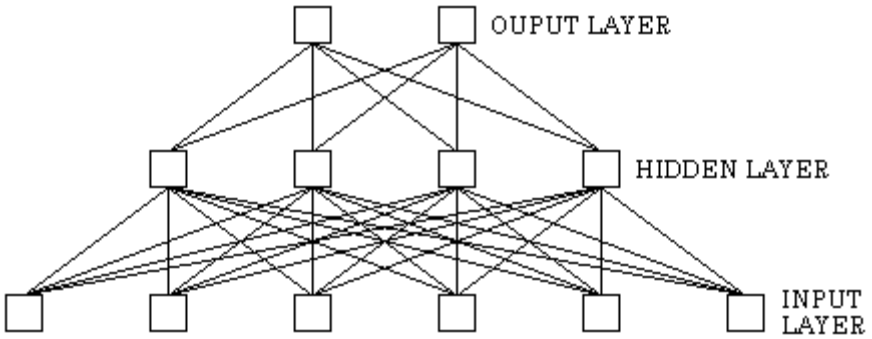

Figura 3

La capa oculta es donde existe el mayor problema ha resolver. Los valores de los pesos en los nodos de la capa oculta constituyen la "representación interna" de la red. Entre más nodos ocultos existan, es más compleja la curva de decisión. Sin embargo, si hay muchos nodos ocultos, la red poseerá mucho poder de modelamiento, lo cual induce en la red incapacidad para generalizar; y por lo tanto no podría ser capaz de reconocer nuevos datos no contenidos en el conjunto de entrenamiento. Algunas reglas empíricas son ofrecidas para determinar un buen número de unidades ocultas, pero pocas parecen estar basadas en alguna teoría sólida. Actualmente el método más común para determinar un número óptimo de unidades ocultas, es el de ensayo y error; sin embargo en el presente trabajo se eligió un método que se base en el valor absoluto del peso de la conexión, el cual será explicado brevemente en la siguiente sección.

1.4 PRUNING

El objetivo de los algoritmos de pruning es tratar de hacer la red más pequeña

cortando los links o nodos, esto se realiza por varias razones:

- Es posible de este modo encontrar una arquitectura de la red neuronal más adecuada.
- El costo de una red puede ser reducido (tiempo en ejecución, memoria y costo para una implementación en hardware)
- La generalización puede ser mejorada (no necesariamente)
- Innecesarias unidades de entrada pueden ser cortadas para evidenciar la relevancia de los valores de entrada.

Los algoritmos de pruning se dividen básicamente en dos partes:

¿Qué será eliminado? Se distingue aquí si son nodos o links, y hay especiales tipos de nodos a eliminar, los cuales son nodos de entrada y nodos ocultos.

¿Cómo se escogerá el nodo o link a eliminar? Actualmente existen dos maneras básicas de escoger, la primera son los algoritmos que “castigan” los pesos, y la segunda son los algoritmos sensitivos. A continuación explicaremos brevemente en que consiste cada uno.

Con respecto a los algoritmos de la primera clase, se tiene como principal ejemplo el algoritmo de Backpropagation con decaimiento de los pesos, el cual fue introducido por P. Werbos. El término que genera el decaimiento de los pesos, causa que estos converjan a un valor absoluto más pequeño de lo que tendrían normalmente. Valores grandes de pesos pueden perjudicar la generalización en dos diferentes maneras. Excesivamente grandes pesos permiten que los nodos ocultos puedan causar que la salida de la función sea bastante rugosa, posiblemente con discontinuidades (Ref 11). Además, excesivos valores en los pesos, permiten que las funciones de activación de los nodos de salida, puedan dar valores más allá del rango de los datos, si la función de activación de salida no

es acotada al mismo rango de los datos. O en otras palabras, grandes valores de pesos pueden causar excesiva variancia de los valores de salida. Y además, según Bartlett (1997), el tamaño de los pesos es más importante que el número de pesos para determinar la generalización (Ref 14).

Por otro lado, existen los algoritmos de sensitividad, estos ejecutan entrenamiento y eliminación de la red neuronal alternativamente, de acuerdo a los siguientes pasos (Ref 5):

1.Escoger una razonable arquitectura para la red.

2. Entrenar la red con backpropagation o algún algoritmo similar de aprendizaje en un mínimo de la red.

3.Computar la saliency el cual es una medida de la relevancia del nodo o link para la ejecución de la red.

4. Cortar el elemento con el más pequeño saliency.

5.Reentrenar nuevamente la red dentro de un mínimo.

6.Si el error en la red no es muy grande, repetir el procedimiento desde el paso 3.

7.Recrear el último elemento eliminado para lograr un más pequeño error de la red.

Existen métodos sofisticados como por ejemplo Optimal Brain Damage o Optimal Brain Surgeon que se basan en aproximar el cambio del error cuando se elimina un nodo, a través de una serie de Taylor hasta segundo orden, sin embargo en esta tesis se usó el método denominado Magnitude Based Pruning (Eliminación

basado en la magnitud del peso), el cual después de cada entrenamiento, el link con el más pequeño peso es removido. Así el valor de la *saliency* de un link es justo el valor absoluto de su peso. Aunque este método es muy simple, este raramente da peores resultados que los métodos más sofisticados (Ref 5.).

1.5 COUNTERPROPAGATION

Es una red de tres capas, la capa de entrada es una capa simple que presenta y distribuye el patrón de entrada a cada nodo en la capa intermedia. La capa intermedia es una capa de tipo Kohonen que usa un esquema de aprendizaje competitivo. Tal esquema asegura que la capa intermedia agrupe los patrones de entrada presentados, y por lo tanto modelara la distribución estadística de los patrones de entrada. La capa de los nodos de salida llamada capa de Grossberg, puede ser usada para asociar un estimulo de un simple nodo con un patrón de salida de arbitraria complejidad.

En operación, un patrón de entrada es presentado a la red y distribuido por la capa de entrada a la capa intermedia, ahí los nodos compiten para determinar que nodo con los pesos asociados, se aproxima más al patrón de entrada, este generará una salida de uno, mientras que los demás será de cero. Por otro lado, la capa de salida tiene nodos que han sido entrenados por clásico condicionamiento, esto con el fin de generar una específica salida, en respuesta a un específico estimulo desde la capa intermedia.

En esencia la red counterpropagation es bastante simple. La capa de Kohonen categorizará cada patrón de entrada, y la capa de salida reproducirá si el patrón de salida es aproximado a esta categoría.

Este tipo de red neuronal fue originalmente propuesto como una red para

búsqueda de patrones que toman ventaja de la arquitectura paralela de las redes neuronales. Pero además, la red Counterpropagation es útil en la realización de aplicaciones que tienen que redireccionar memorias asociativas.

Ahora se describirá de manera más formal el proceso de aprendizaje de este tipo de red neuronal. Cuando se presenta un patrón, la red clasifica este patrón usando un vector de referencia. Los nodos de la capa oculta juegan un papel importante en este proceso, ya que la capa oculta ejecuta una clasificación competitiva para agrupar los patrones. Este tipo de red trabaja bien en agrupar patrones en distintos grupos.

Dos tipos de capas son usados: La capa oculta es de tipo Kohonen con nodos competitivos que realizan un aprendizaje no supervisado; la capa de salida es una capa de tipo Grossberg, la cual es totalmente conectada con la capa oculta y no es competitiva.

Cuando se entrena la red, esta trabaja de la siguiente manera. Después de presentar un patrón en la capa de entrada, los nodos en la capa oculta suman sus entradas de acuerdo a

$$\text{Netj} = \sum_i w_{ij} * O_i$$

y entonces compiten para responder al patrón de entrada (los pesos toman valores entre menos uno y uno, así que para maximizar la multiplicación de los pesos con los patrones de entrada estos se normalizan). El nodo con él mas alto valor de entrada es el ganador y su activación tiene un valor de uno, mientras que todos los otros son cero. Después de la competencia, la capa de salida suma las salidas de la capa oculta con los respectivos pesos

$$a_k = Netk$$

Si c es el índice de la unidad de la capa oculta ganadora, y ya que Oc es el único elemento no-cero en la suma, el cual es igual a uno. Se tiene que

$$a_k = w_{ck}$$

Por lo tanto el nodo oculto ganador activa un patrón en la capa de salida. Durante el entrenamiento los pesos son adaptados de la siguiente manera:

1. Un ganador en la capa oculta es escogido en respuesta al patrón de entrada.

2. Los pesos entre la capa de entrada y el nodo ganador son ajustados de acuerdo a (los demás pesos no se modifican.):

$$w_{ic}(t+1) = w_{ic}(t) + \alpha * (o_i - w_{ic}(t)) \qquad 2.14$$

3.La salida de la red es computada, y comparada al patrón de entrenamiento

4. Los pesos entre el nodo ganador y la capa de salida son actualizados de acuerdo a (los demás pesos no se modifican.):

$$w_{ck}(t+1) = w_{ck}(t) + \beta * (o_k - w_{ck}(t)) \qquad 2.15$$

1.6 REDES NEURONALES BASADAS EN FUNCIONES DE BASE RADIAL

Inicialmente se explicara brevemente en que consisten las funciones de base radial RBF (radial basis functions). Este tipo de funciones son ampliamente usadas en la teoría de la aproximación funcional, la cual consiste en: Dado un conjunto de valores independientes X = (x1, x2, x3,... xm), y sus respectivos valores dependientes Y = (y1, y2, y3,...yn), se debe encontrar una relación funcional F suave entre X y Y, tal que el error sea mínimo. La función F se calcula a partir de un conjunto ortogonal y completo de funciones (Ref 15), de ahí el adjetivo de función base, estas funciones bases en particular dependen además de un valor T (vector de la misma dimensión de X) el cual se denomina el centro de la función. La relación formal entre F y las funciones de base radial es

$$F(X) = \sum_{i-1}^{K} C_i * g(\|X - T_i\|) = \tilde{Y} \qquad 2.16$$

donde g son la RBF, C_i son coeficientes, T_i el centro de la función g, $\tilde{Y}$ es el valor aproximado de Y.

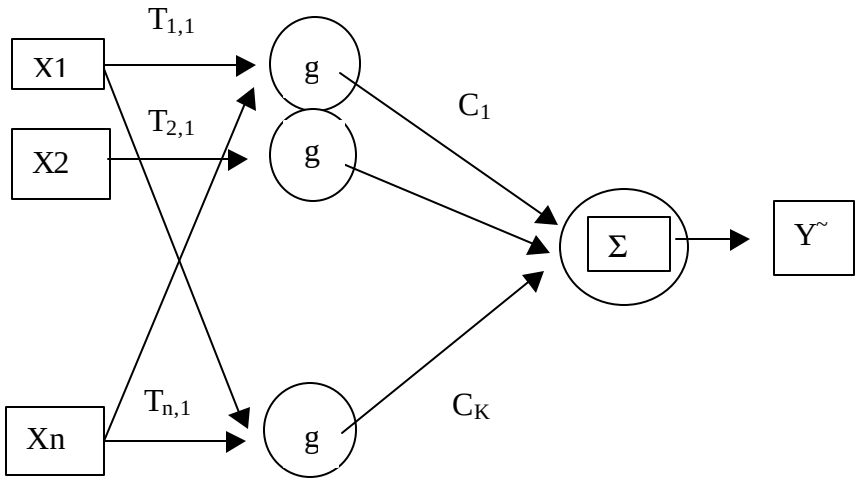

Figura 4

Las redes neuronales basadas en RBF, pueden fácilmente representar la ecuación

2.16 (Figura 4) por una red neuronal de tres capas feedforward. La capa de entrada consiste en n nodos los cuales representan los elementos del vector X. El número de nodos en la capa oculta representa el número de funciones de base radial, que en la ecuación 2.16 es K. Los pesos entre la capa de entrada y la oculta, modelan los valores de los K vectores T. Los nodos ocultos computan la norma entre X Y T, y luego calculan el valor de activación a partir de la función g. Si el vector Y tuviera solo una componente, existiría entonces solo un nodo de salida, el cual computa el valor de su activación, a partir de la suma ponderada de las activaciones de los nodos ocultos, el peso para cada uno de los enlaces entre la capa oculta y el nodo de salida representa el valor C_i en la ecuación 2.16.

En este trabajo se usa como RBF la función gaussiana definida como:

$$g(p,X,T)=e^{-p\|X-T\|} \qquad 2.17$$

donde p se define como el bias asociado al nodo de la capa oculta. Es útil usar este tipo de funciones como RBF, por que de esta manera es más fácil predecir los valores de salida del nodo; si X es lejano al vector T, el valor de la función g será pequeño, por el contrario, si el valor del vector X es próximo a T, el resultado será más grande (Ref 5). Es por esta razón que cuando se usan las redes neuronales RBF, como redes para clasificar, los vectores T son elementos que representan cada una de las clases; el cual es el punto de vista que se usa en este trabajo.

Como función de activación para el nodo de salida, se usa la conocida función *sigmoid*.

1.6.1 ENTRENAMIENTO DE LAS REDES NEURONALES RBF

Ahora explicaremos brevemente el entrenamiento de este tipo de redes

neuronales. La eficiencia de una red neuronal de este tipo depende del numero y valor de los vectores T, la función que se usa como RBF en los nodos ocultos, y el método usado para determinar los pesos de la red neuronal. Algunos investigadores han entrenado a las redes RBF, seleccionando los centros aleatoriamente del conjunto de patrones de entrenamiento; otros tienen procedimientos no supervisados para la selección de los centros, y algunos tienen procedimientos supervisados; por otro lado, también se es necesario escoger el método para optimizar, los valores de los pesos asociados, a los enlaces entre la capa oculta, y el nodo de salida, por lo general el procedimiento usado es el basado en el calculo del gradiente para descender por la curva de error.

Con base a lo anterior, el proceso de entrenamiento de una red neuronal RBF se divide en dos partes: la primera parte consiste en el proceso de determinar los valores T, que son los pesos de los enlaces entre la capa de entrada y de salida; la segunda parte trata de optimizar los pesos entre la capa de salida y la capa oculta, y los diversos bias. A primera vista este entrenamiento parece ser bastante costoso, sin embargo, una vez que se han seleccionado adecuadamente los valores de T, el proceso de optimización converge rápidamente. De hecho, una de las principales contribuciones de las redes RBF, fue su más rápido entrenamiento con respecto a redes entrenadas con el algoritmo Backpropagation (Ref 16).

En este trabajo se usó el método RBF_Weights, implementado en el software SNNS (Anexo C), como primera parte en el proceso de entrenamiento; este método consiste en los siguientes pasos: primero selecciona del conjunto de patrones de entrenamiento de una manera homogénea los centros T, y posteriormente asigna estos centros a los valores de los pesos entre la capa de entrada y la capa oculta, luego el valor del bias (parámetro P) se obtiene de un valor inicial dado por el usuario; por ultimo se calcula los valores de C (los pesos entre la capa oculta y la capa de salida) por medio de la siguiente ecuación (Ref 5, Ref 16):

$$C^{\rightarrow} = (G^T . G + l * G_o)^{-1} . G^T . O^{\rightarrow} \qquad 2.18$$

donde C es el vector de pesos (C_1 , C_2, ...C_k), K es el número de nodos ocultos (ver figura 4), O es el vector conformado por las salidas asociadas a cada uno de los patrones de entrenamiento, teniendo en cuenta que la salida de la red neuronal es la salida de un solo nodo, se tiene que O = (O_1, O_2, ..O_n), donde n es el número de patrones de entrenamiento. Según la ecuación 2.18 este paso es el más costoso en esta primera etapa.

Una vez realizada la primer parte en el proceso de entrenamiento, se prosigue con la etapa relacionada con la optimización de los pesos; para esto se usó el método del gradiente descendiente, el cual consiste en minimizar el error E por medio del gradiente (ecuación 2.2), esto se logra aplicando las siguientes ecuaciones que calculan los cambios de los pesos de la red RBF (Ref 5, Ref 16):

$$\Delta T_j = -h_1 * \partial E / \partial T_j \qquad 2.19$$

donde: 1<=j<= K, η_1 es la tasa de aprendizaje asociada a los centros de la RBF.

$$\Delta P_j = -h_2 * \partial E / \partial P_j \qquad 2.20$$

donde: 1<=j<= K, η_2 la tasa de aprendizaje asociada a los bias de los nodos ocultos.

$$\Delta C_j = -h_3 * \partial E / \partial C_j \qquad 2.21$$

donde: 1<=j<= K, η_3 es la tasa de aprendizaje asociada a los pesos de los enlaces entre los nodos de la capa oculta y el nodo de salida.

$$\Delta b = -h_4 * \partial E / \partial b \qquad 2.22$$

donde: η_4 es la tasa de aprendizaje asociada al nodo de salida que conforma la capa de salida.

Además, es posible mejorar el proceso de optimización, definiendo un valor de error mínimo, el cual es una cota inferior que evita el sobreentrenamiento, si para algún patrón se obtiene un error menor al error mínimo, este se establece a cero, y así los pesos no son modificados en el proceso de optimización.

Las tasas de aprendizaje η_1, η_2 y $\eta_{3,}$ deben ser escogidas de una manera muy cuidadosa, si los valores son muy grandes, existe la posibilidad de que el proceso de aprendizaje se vuelva inestable.

CAPITULO 2 CONCEPTOS BÁSICOS SOBRE SÍSMICA

2.1 PROCESO DE GENERACIÓN DE UN SISMO

La corteza terrestre esta dividida en grandes placas (figura 5) que están desplazándose lenta y constantemente, el desplazamiento relativo de las placas genera choques entre estas, lo cual provoca a su vez acumulación de esfuerzos, principalmente en sus contornos.

Figura 5

Actualmente la gran mayoría de científicos creen que este movimiento de las placas tectónicas se debe principalmente a las corrientes de convección que tiene el magma (Ref 10), este al estar en contacto con el interior del núcleo terrestre se calienta, y se vuelve menos denso, y por lo tanto más liviano que el magma más frío, por lo cual asciende hasta la superficie; por otro lado, el magma que esta en

contacto con la litosfera se enfría y desciende hasta el interior de la tierra, generándose grandes corrientes netas de convección (figura 6).

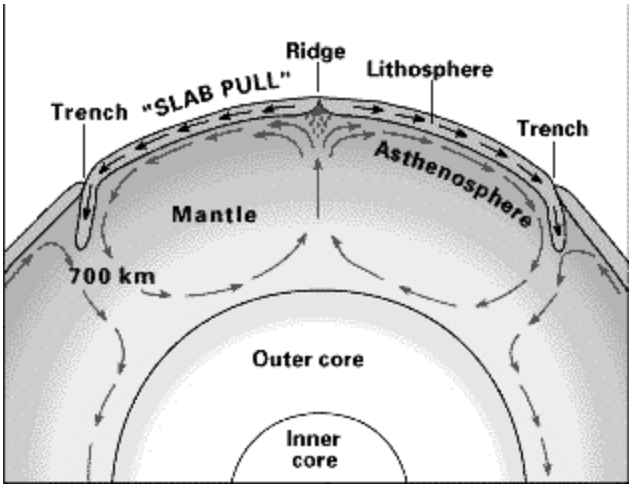

Figura 6

Los límites que hay entre las diversas placas que conforman la corteza terrestre se denominan fallas, y existen varios tipos según el desplazamiento relativo de las placas que se encuentran en contacto (figura 7).

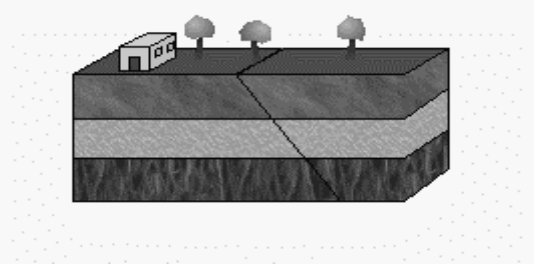

Figura 7

Cuando el esfuerzo que sé a acumulado entre las fallas sobrepasa él limite de resistencia del material, este se rompe bruscamente causando un movimiento repentino de la superficie terrestre, es entonces cuando ha ocurrido un sismo.

Esta perturbación de la corteza terrestre se transmite por el medio en forma de ondas (ya que la tierra es un medio elástico y en su más simple formalismo matemático esta puede obedecer a la ley de Hooke.). Existen varios tipos de ondas sísmicas generadas por la ocurrencia del sismo, esto se debe principalmente a condiciones de frontera del medio, a las propiedades de los materiales, a las reflexiones y refracciones que la onda original tiene, etc. Sin embargo en esta tesis el interés se centra en las ondas sísmicas Pn la cual explicaremos brevemente en la siguiente sección.

2.2 ONDAS SÍSMICAS Pn

Las ondas Pn son ondas de comprensión semejantes a las ondas que se producen en un resorte cuando este es comprimido, y luego se deja libre de presión (figura 8).

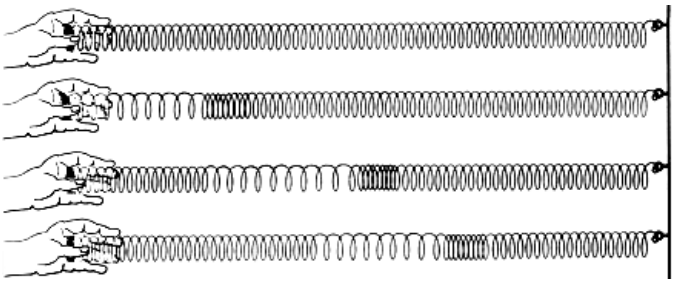

Figura 8

Este tipo de onda es de carácter longitudinal, ya que la dirección de perturbación es la misma dirección de propagación de la onda (figura 9), a diferencia de otros tipos de ondas como las de cizalla o transversales llamadas también ondas Sn, en donde su dirección de deformación es en sentido perpendicular a la dirección de propagación de la onda.

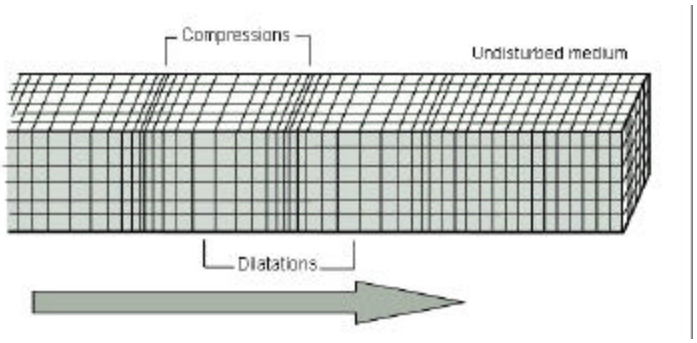

Figura 9

Como se mencionó anteriormente, existen muchos tipos de ondas sísmicas, entre ellas esta por ejemplo las ondas superficiales Love (figura 10), las cuales se producen al establecer las condiciones de frontera adecuadas en la ecuación de onda, para un medio en el cual existen dos interfaces, en este caso la corteza terrestre y el aire.

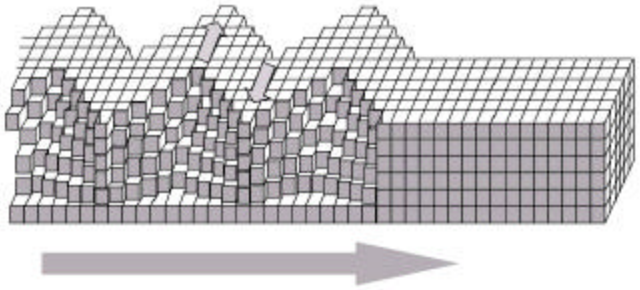

Figura 10

Actualmente se ha podido identificar en los diversos sismogramas varios tipos de ondas, como por ejemplo las de tipo K las cuales son ondas que atraviesan el núcleo terrestre (figura 11).

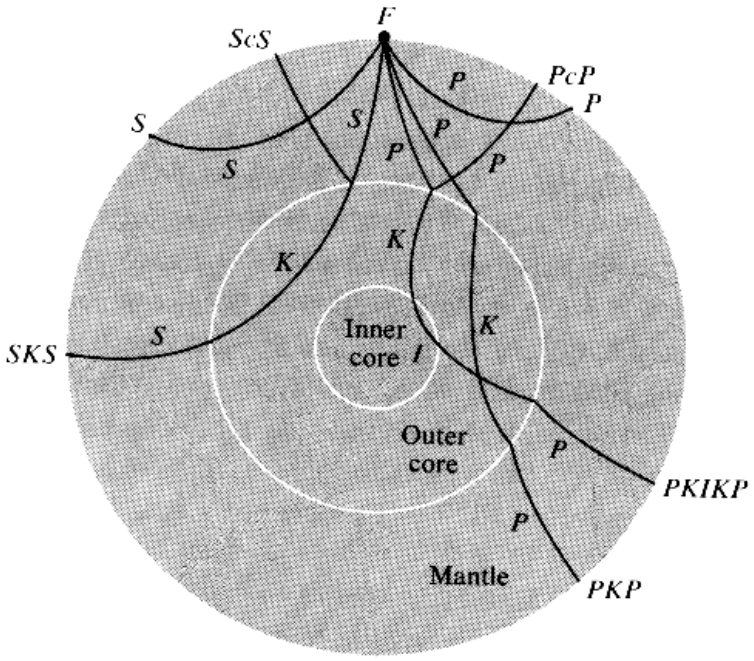

Figura 11

El interés de está tesis esta en identificar el tiempo de arribo de la fase Pn originada en una fuente sísmica bastante activa denominada Nido de Bucaramanga (anexo A), la cual es claramente identificada en el inicio del sismogra de la figura 12

Figura 12

CAPITULO 3 ASPECTOS IMPORTANTES ACERCA DE LOS MÉTODOS ACTUALES USADOS PARA LA DETECCIÓN DE FASES SÍSMICAS

3.1 ESTADO ACTUAL

Anteriormente el método que se usaba para determinar en tiempo real el arribo de una onda sísmica, consistía simplemente en comparar la amplitud de la señal con un valor umbral constante, si esta amplitud excedía el umbral, el sistema declaraba la ocurrencia de la onda. Este método no era muy confiable, ya que declaraba muchas falsas alarmas, o por el contrario, si el ruido sísmico aumentaba por algún tiempo, en la zona donde estaba ubicado el sensor (sismógrafo), se perdían los eventos sísmicos. Sin embargo se perfeccionó este método llegando a conocerse como algoritmo STA-LTA, el cual ha sido bastante usado en la última década.

En recientes años las redes neuronales han emergido como una tecnología práctica, la cual se ha aplicado con éxito en varios campos; en particular en el área de la sismología, se han realizado estudios para marcar el arribo de fases sísmicas, usando redes neuronales perceptrones con el algoritmo de entrenamiento backpropagation (Ref 6), como datos de entrada se ha usado: El espectro de frecuencias (Ref 3), las muestras en el dominio del tiempo, u otros discriminantes (Ref 8.). Además, se ha usado redes neuronales con la arquitectura *cascade-correlation*, para marcar el primer arribo de una fase sísmica, y también para la clasificación de señales de este tipo.

Otra tendencia que está surgiendo actualmente, se basa en el uso de la teoría de los *Wavelets*, la cual es bastante útil cuando se estudia señales no estacionarias, cuyo contenido frecuencial cambia en el tiempo (Ref 12.).

CAPITULO 4 PREPROCESAMIENTO

4.1 SELECCIÓN DE LAS SEÑALES SÍSMICAS

Como se mencionó en la introducción se escogieron en total 2634 (1788 de entrenamiento y 846 de validación) señales sísmicas provenientes de sismos en el denominado Nido de Bucaramanga (Anexo A.). A estos datos se les realizo el preprocesameinto que se describirá a continuación.

4.2 PREPROCESAMIENTO DE LOS DATOS

El preprocesamiento de los datos fue una fase fundamental en el desarrollo de este trabajo, por que el objetivo de este fue reducir la complejidad de los patrones sin perder tanto como sea posible información útil que pueda servir como discriminantes en la red neuronal (Anexo B.).

El primer paso fue determinar el ancho de la ventana de tiempo, que se usó como conjunto de datos para crear los patrones; las ventajas y desventajas de escoger un rango de ventanas de tiempo se describen en la tabla 1

VENTAJAS Y DESVENTAJAS DE UNA VENTANA DE TIEMPO PEQUEÑA	**VENTAJAS Y DESVENTAJAS DE UNA VENTANA DE TIEMPO GRANDE**
Se logra mayor precisión en la determinación del arribo de las onda Pn.	La precisión en la determinación del arribo de la onda Pn puede disminuir por parte de la red neuronal.
Debido a que el número de muestras que conforman la señal sísmica es proporcional al tamaño de la ventana, se logran patrones más pequeños en	El número de muestras aumenta al hacer más grande la ventana de tiempo, y por lo tanto los patrones aumentan en espacio físico.

espacio.	
Al tener patrones más pequeños su complejidad se puede reducir y por lo tanto la arquitectura de la red neuronal es al mismo tiempo menos compleja.	La arquitectura de la red neuronal puede aumentar con el aumento en la ventana de tiempo.
El número de falsas alarmas (identificación de una onda Pn por parte de la red neuronal cuando en realidad no existe) aumenta al tener ventanas de tiempo más pequeñas.	El número de falsas alarmas puede disminuir hasta cierto punto si se aumenta la ventana de tiempo.
El hecho de que la red neuronal sea menos compleja significa que el tiempo en el entrenamiento se reduce.	Puede ser necesario requerir de más tiempo para la etapa de entrenamiento de la red neuronal.

Tabla 1

Por otro lado hay que tener en cuenta además de las anteriores características; el tipo de onda sísmica que se desea detectar, ya que si la onda tiene un contenido frecuencial determinado, la ventana de tiempo debe ser lo suficiente para que alcance a visualizar varios periodos de la onda. Para esté trabajo se tuvo en cuenta la experiencia de los operarios de la Red Sismológica Nacional de Colombia, ya que ellos consideran en su labor, que el ancho de ventana apropiado que balancea las ventajas y desventajas que se mencionaron anteriormente es de 2 segundos, por lo tanto considerando que la velocidad de muestreo del digitalizador es de 60 muestras por segundo, se tiene que los patrones tienen 120 muestras.

El segundo paso consistió en normalizar la señal sísmica que corresponde al acho de ventana ya establecido, lo cual logra (Ref 2, Ref 6.) reducir la complejidad de los patrones de entrada, y por lo tanto también la arquitectura de la red neuronal.
El tercer paso fue revisar exhaustivamente el arribo de la onda Pn en los datos

sísmicos, evitando así que se infiltre en el conjunto de los patrones de entrenamiento y de validación, datos anómalos.

Luego, como cuarta etapa, se procedió a alinear la muestra que corresponde al arribo de la onda Pn en el centro de la ventana, es decir los patrones tendrán el arribo de la onda en la muestra 60.

En el uso de las redes neuronales multi-perceptron, con algoritmo de entrenamiento Backpropagation, se introdujeron los patrones resultantes del proceso anterior descrito, sin embargo para las redes neuronales tipo Counterpropagation fue necesario realizar decimación, lo cual se explicara en él capitulo 6.

En resumen se realizaron los siguientes pasos:

- Escoger las señales sísmicas de donde se obtendrán los patrones para las ondas Pn y el ruido.

- Escoger el acho de la ventana de tiempo apropiado

- Normalizar los datos

- Revisar en cada uno de los patrones que la determinación del arribo de onda Pn dada por el humano sea correcta.

- Alinear el arribo de la onda Pn en el centro de la ventana de tiempo.

- Para redes Counterpropagation realizar decimación

CAPITULO 5 REDES NEURONALES TIPO PERCEPTRON

5.1 RESULTADOS CON REDES NEURONALES ENTRENADAS CON EL ALGORITMO DE BACKPROPAGATION EN MODO BATCH

La metodología que se usó para buscar una buena arquitectura, consistió en los siguientes pasos:

1. El número de nodos en la capa de entrada son 120 ya que como se mencionó en el preprocesamiento, se escogió una ventana de tiempo de 2 segundos.

2. El número de nodos de salida es uno, siendo el valor de 1 para la ocurrencia de la fase Pn, y cero si no.

3. Se inició con 20 nodos en la capa oculta (posteriormente él numero de nodos y de conexiones se redujo utilizando un algoritmo de *pruning*.).

4. Se entrenó la red neuronal haciendo uso de los patrones de entrenamiento, y para evitar el sobreentrenamiento se usaron los patrones de validación.

5. El algoritmo de entrenamiento usado inicialmente fue el de Batch - Backpropagation, el cual se ha encontrado (Ref 4) da mejores resultados que el algoritmo estándar de Backpropagation (en el modo estándar los pesos se actualizan después de cada patrón, mientras que en modo Batch los pesos se actualiza después de cada época, es decir después de que ha pasado todo el conjunto de patrones de entrenamiento.).

6. Se realizó este entrenamiento con varios valores de la tasa de aprendizaje (ecuación 2.3), obteniendo los siguientes resultados (los círculos corresponden a los resultados con el conjunto de patrones de validación, mientras que los

cuadrados corresponden a los resultados con el conjunto de patrones de entrenamiento):

En primer lugar, se notó que el hecho de introducir los patrones de manera aleatoria induce una mejor convergencia en el entrenamiento. Por otro lado, con base en las figuras 13-a hasta 13-g, se deduce que a pesar de que la rapidez de la convergencia no cambia mucho, se obtiene un menor error MSE (ecuación 2.2) con η=0.7 en la época 20 (en el eje x hay que multiplicar por 10 para obtener el valor real de la época.).

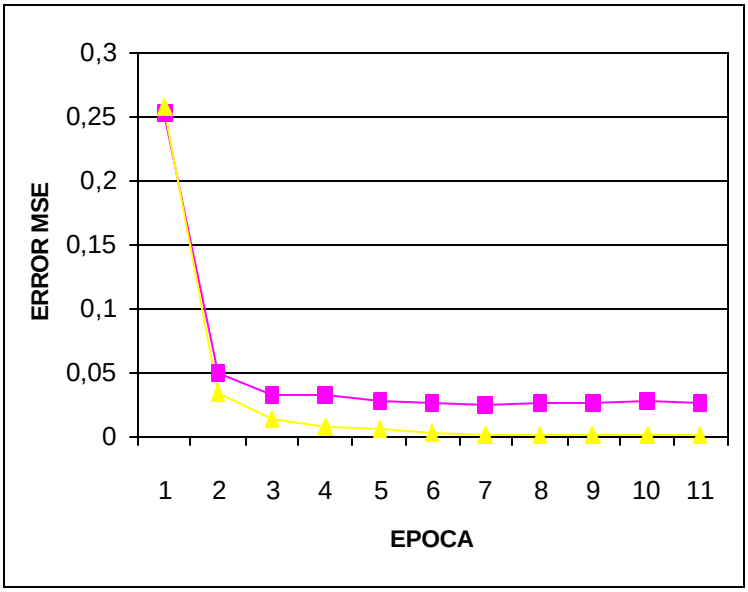

Figura 13-a η=0.1

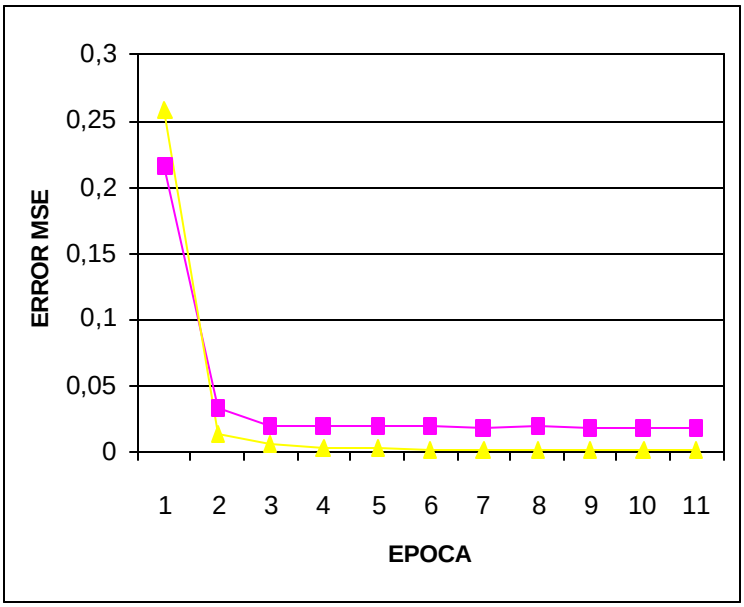

Figura 13-b η=0.2

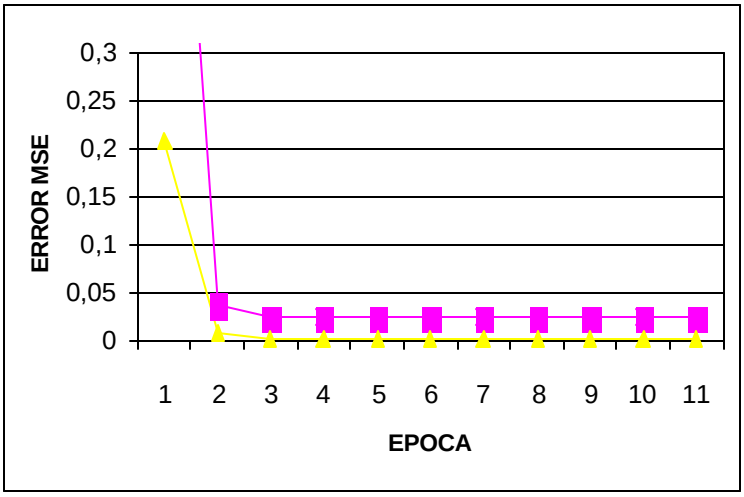

Figura 13-c η=0.3

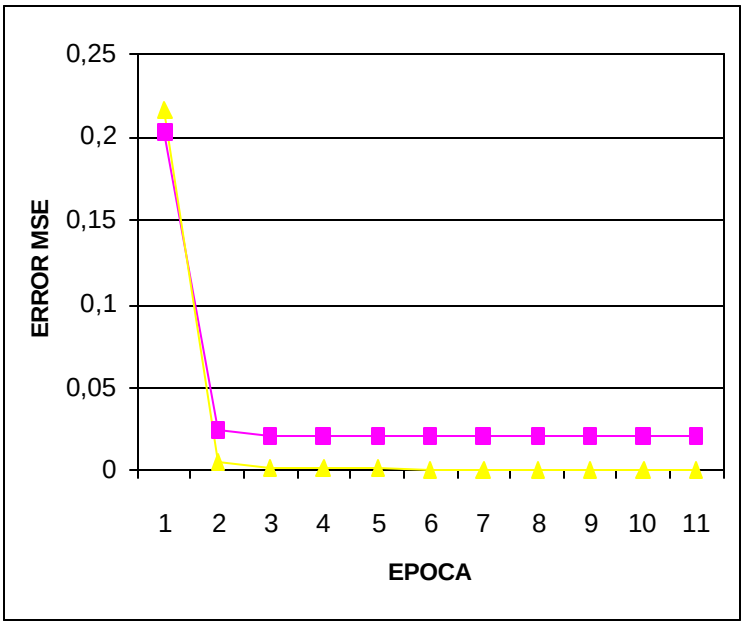

Figura 13-d $\eta = 0.4$

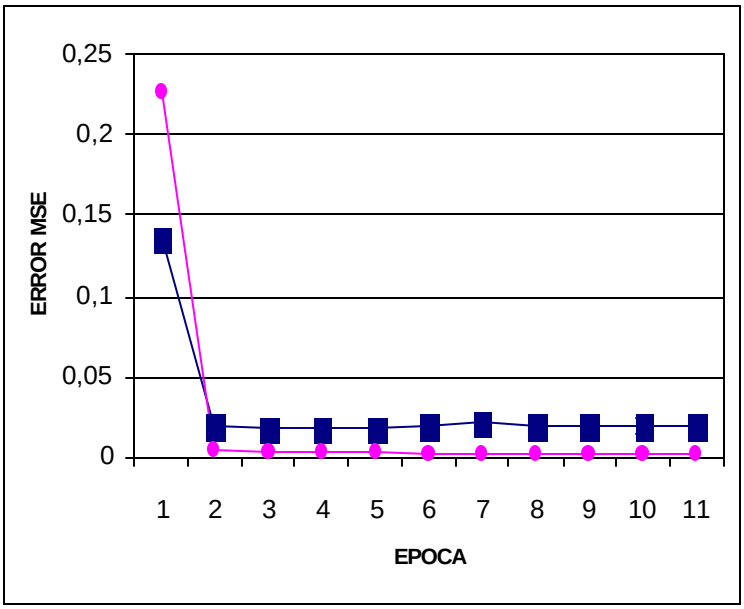

Figura 13-e $\eta = 0.5$

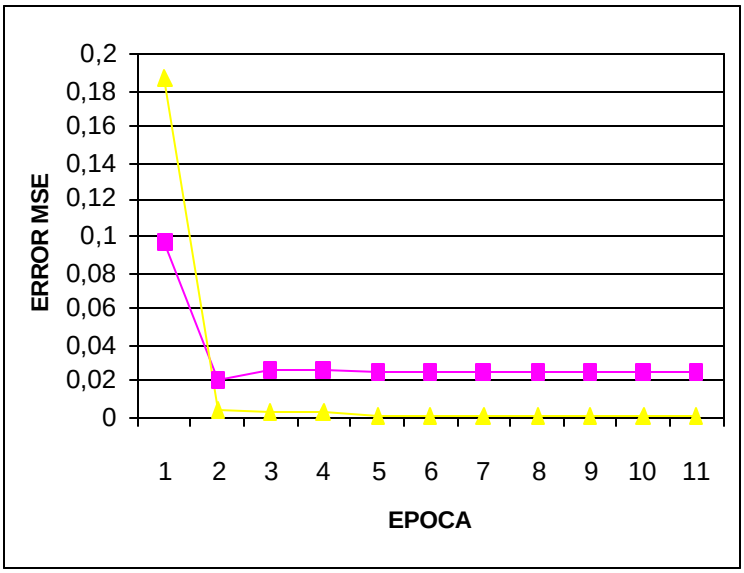

Figura 13-f η=0.7

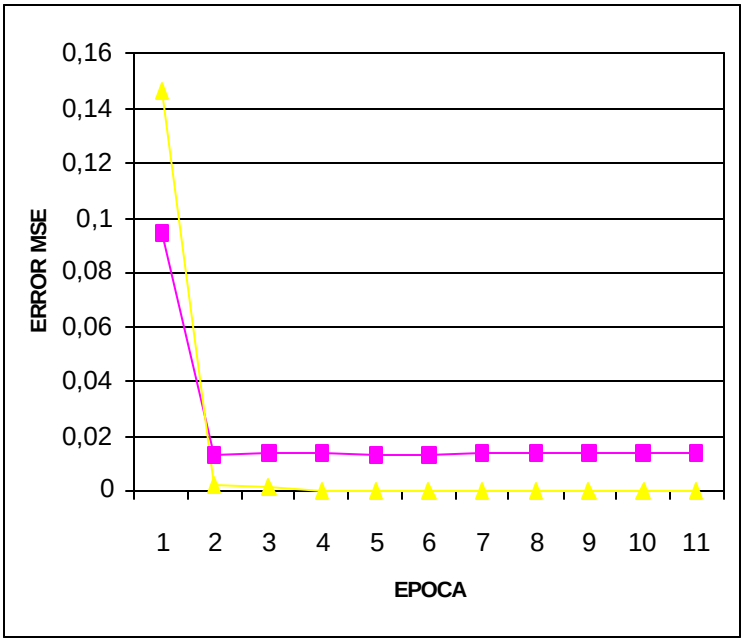

Figura 13-g η= 0.9

5.2 RESULTADOS CON EL ALGORITMO DE BACKPROPAGATION CON MOMENTUM EN MODO BATCH

Una vez que se determinó el mejor valor de la tasa de aprendizaje (η), se

prosiguió a buscar una más rápida convergencia, sin aumentar el error en el entrenamiento, y evitando además el sobre aprendizaje; este proceso se realizó siguiendo los siguientes pasos:

1. Se cambió el algoritmo de aprendizaje de la red neuronal por el de Backpropagation con momentun, el cual se basa en la ecuación 2.14
2. Se buscó el mejor valor de α, manteniendo constante la rata de aprendizaje η.

3. Se obtuvieron los siguientes resultados (los círculos corresponden a los resultados con el conjunto de patrones de validación, mientras que los cuadrados corresponden a los resultados con el conjunto de patrones de entrenamiento):

Basándose en las graficas 14-a hasta 14-h se encuentra que el momentun no tiene un efecto notable en la velocidad de la convergencia en el aprendizaje, sin embargo hay menor error MSE con α=0.7.

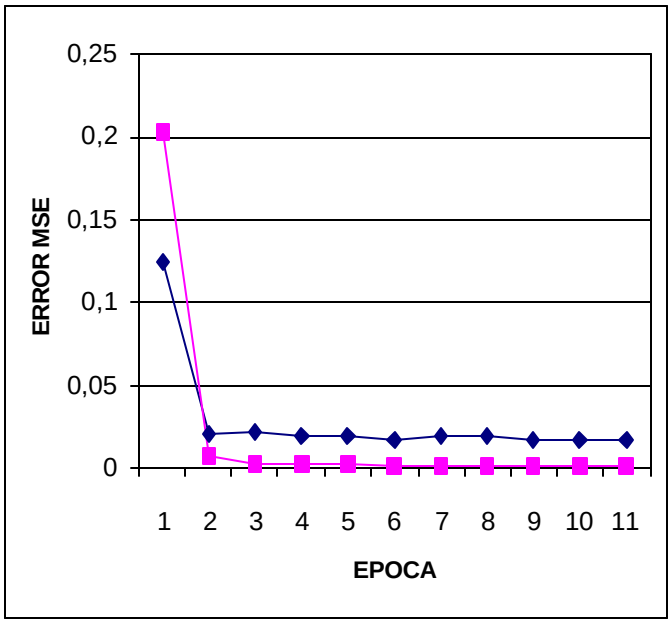

Figura 14-a α=0.1

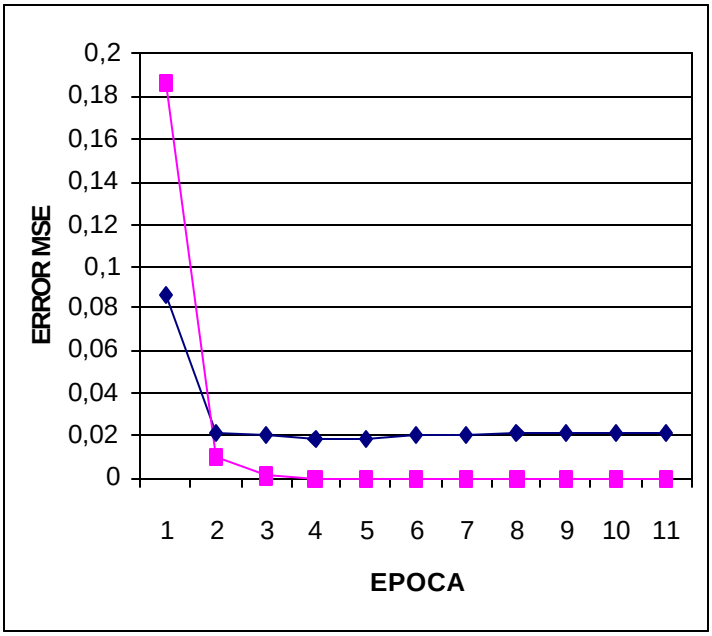

Figura 14-b α=0.15

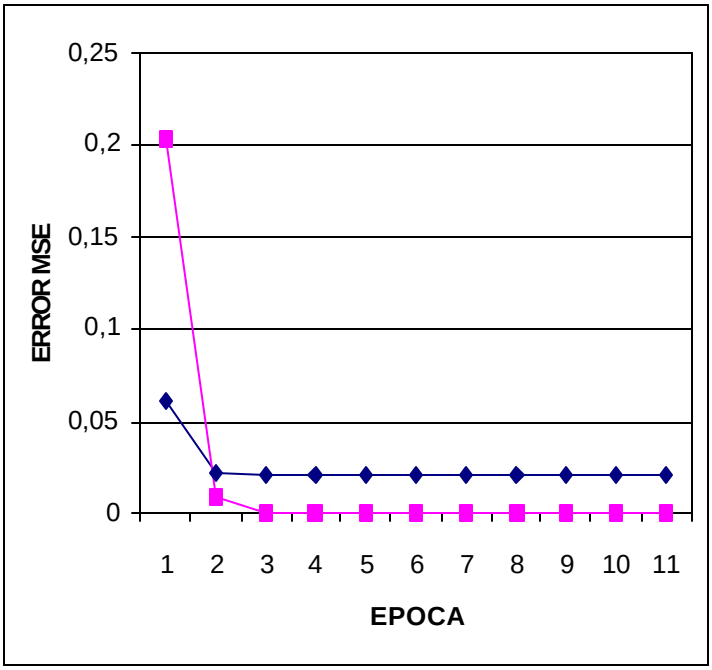

Figura 14-c α=0.3

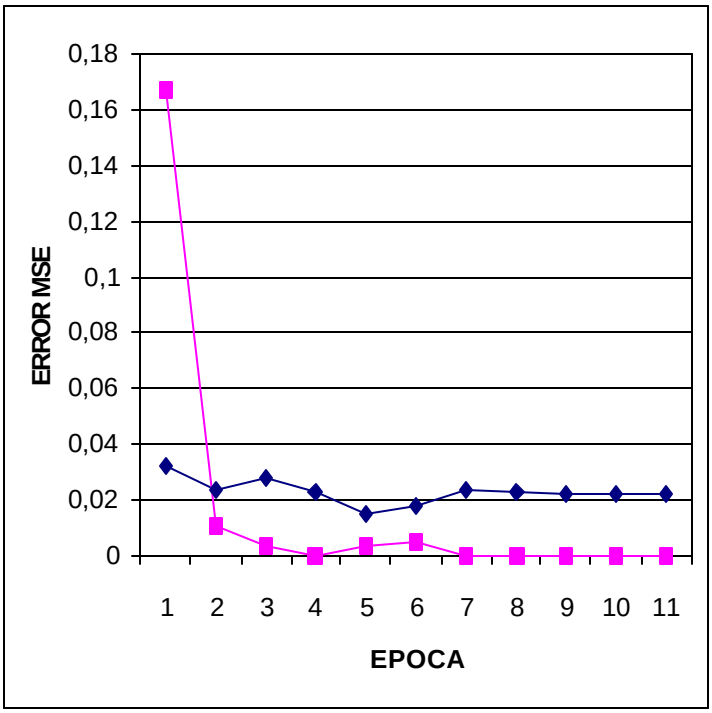

Figura 14-d α=0.5

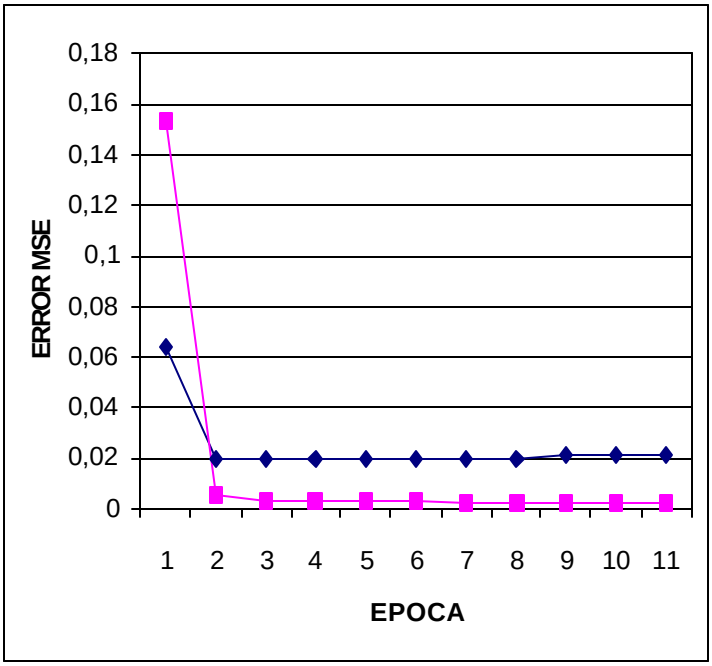

Figura 14-e α=0.6

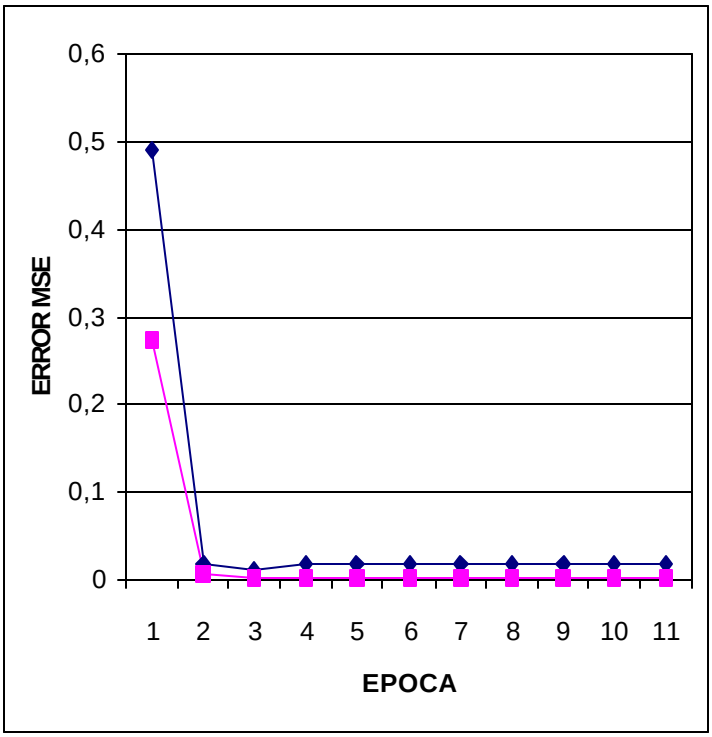

Figura 14-f α=0.7

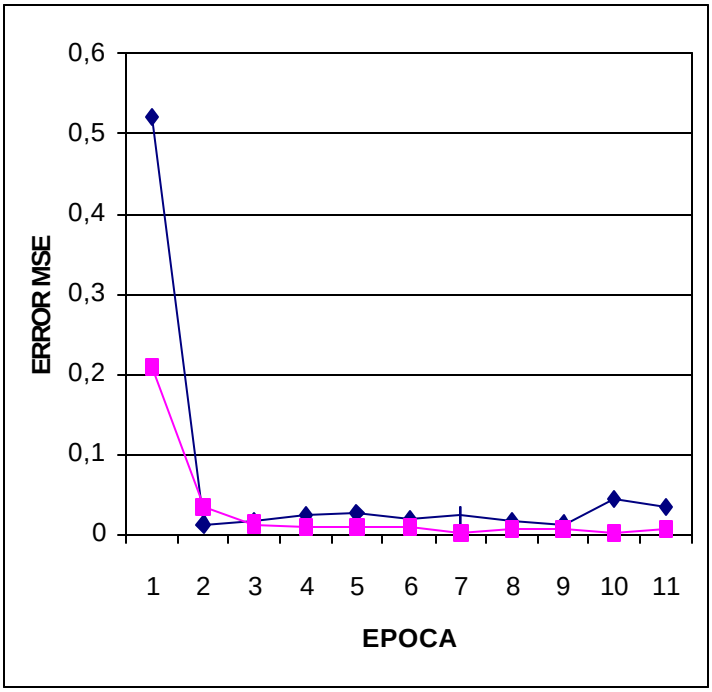

Figura 14-g α=0.8

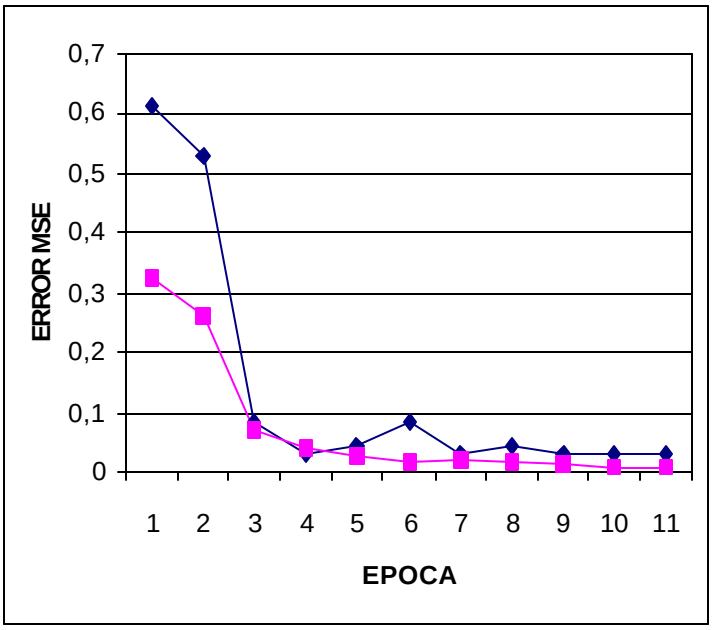

Figura 14-h α=0.9

5.3 RESULTADOS CON EL ALGORITMO DE PRUNING MAGNITUDE BASED PRUNING (VER SECCIÓN 1.4)

El objetivo en esta etapa consistió en reducir la complejidad de la arquitectura sin perder generalidad, y sin aumentar el error en el entrenamiento de la red neuronal. En esta etapa se usó la siguiente metodología:

1. Se eligió el método Magnitude Based Pruning, el cual se explica en la sección 1.4.

2. Se ejecutó el proceso de eliminación de nodos ocultos y conexiones, según lo explicado en la sección 1.4 (Ver también Ref 9.).

3. Para verificar el funcionamiento de la red neuronal se usó la nueva arquitectura resultante del algoritmo de pruning, sé reinicializaron los pesos de las

conexiones restantes, y se usaron los valores α y η, encontrados en las etapas anteriores.

4. Se obtuvieron los siguientes resultados (los círculos corresponden a los resultados con el conjunto de patrones de validación, mientras que los cuadrados corresponden a los resultados con el conjunto de patrones de entrenamiento):

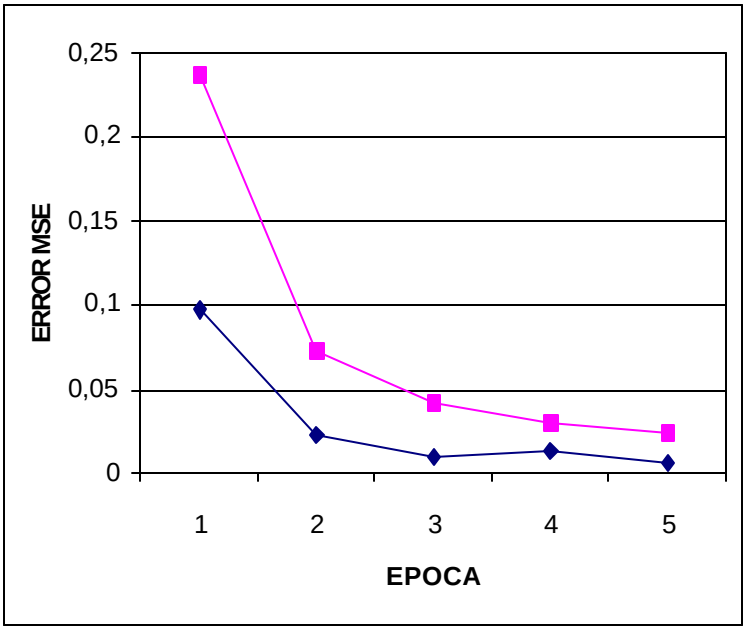

Figura 15 α=0.7, η=0.7 y nueva arquitectura

Inicialmente la arquitectura de la red neuronal estaba conformada por 120 nodos de entrada, 20 nodos ocultos, 1 nodo de salida, y en total 2420 conexiones ya que la red era totalmente conectada. Después del proceso de pruning se obtuvo una red con el mismo número de nodos de entrada y de salida, pero con 16 nodos ocultos y solo 51 conexiones. El entrenamiento con esta red se refleja en la figura 15, en donde se observa que la red a pesar de que ahora es menos compleja, necesita 10 épocas más (época 30) para llegar a un mínimo en el error, evitando además el sobreentrenamiento.

En la figura 16-a y 16-b, se observa la drástica disminución de conexiones; entre más rojo es el color indica que es más próximo a –1, el azul a cero, y el verde a 1.

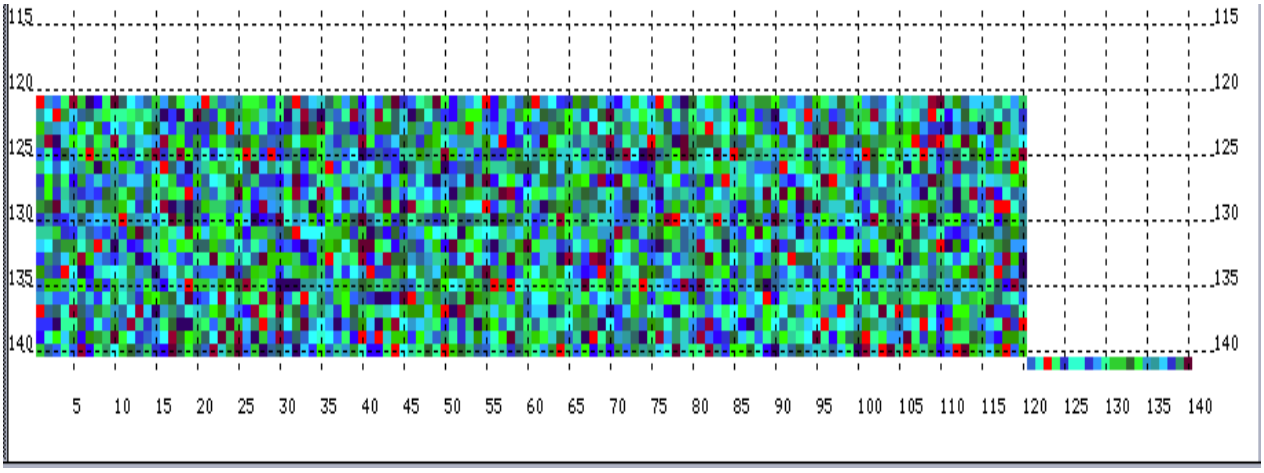

Figura 16-a Distribución de los pesos antes del proceso de pruning

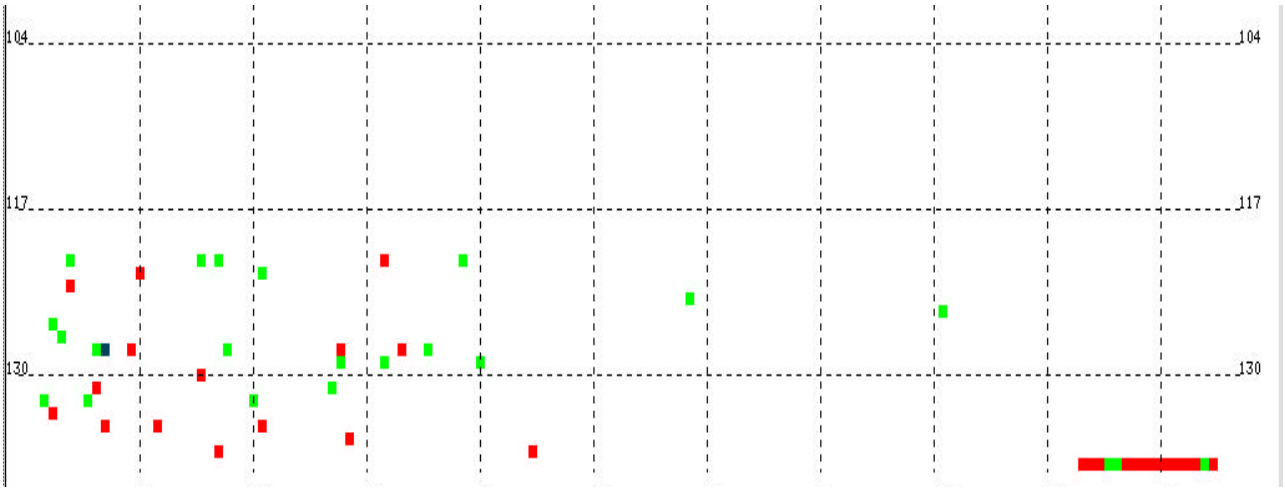

Figura 16-b Distribución de los pesos después del proceso de pruning

CAPITULO 6 REDES NEURONALES TIPO COUNTERPROPAGATION

6.1 RESULTADOS CON REDES NEURONALES ENTRENADAS CON EL ALGORITMO DE COUNTERPROPAGATION

Teniendo en cuenta que para determinar el arribo de la onda Pn es necesario identificar si la señal que se observa en una venta de tiempo dada es ruido o señal sísmica, nos encontramos con el problema de reconocimiento de clases o patrones; para nuestro caso solo hay dos clases posibles: ruido o señal sísmica. Por lo tanto, es interesante conocer si las redes neuronales asociativas, diseñadas especialmente para este tipo de problemas, tienen mejor rendimiento que las redes multicapa de perceptrones.

Inicialmente el trabajo se enfocó en redes neuronales asociativas de tipo Kohonen, pero no se obtuvo buenos resultados, porque manteniendo constante el numero de nodos de entrada (120 nodos), y variando el número de nodos de la capa competitiva entre 20 y 100, la red no lograba clasificar las dos clases de patrones a las que nos referimos.

Para solucionar el problema que se menciona en el párrafo anterior, se decidió por simplificar la complejidad de los patrones, que de hecho estaban normalizados, esto se hizo con la ayuda de la decimación de la señal sísmica; lógicamente esto ocasiona que la exactitud en la marca del arribo de la onda Pn disminuya.

Se ensayó con la reducción de 60 muestras/segundo a 12 y a 6 muestras /segundo, era de esperar que a pesar de que la complejidad y el tiempo de entrenamiento de la red disminuían al reducir el número de nodos de entrada, esta ofrecía menos calidad en los resultados; lo cual se evidencia en las figuras 17-a y 17-b (el eje X de las figuras 17-a y 17-b corresponden al verdadero valor de la

época).

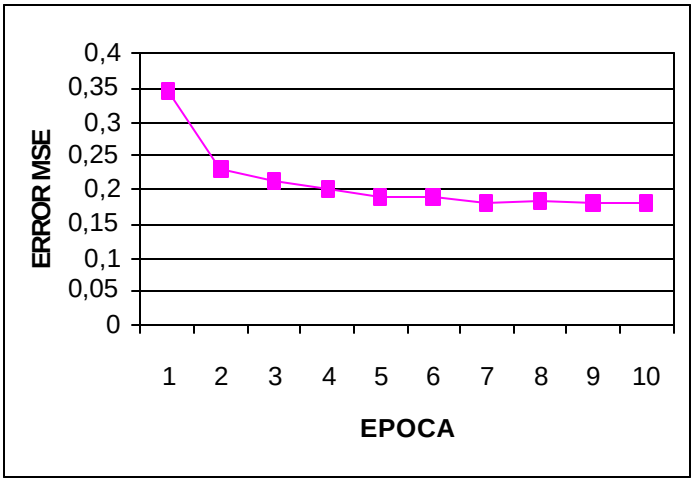

Figura 17-a Entrenamiento con 12 nodos de entrada

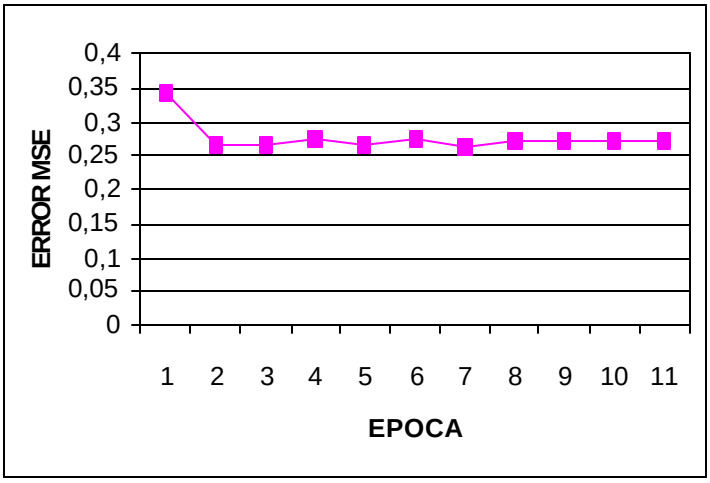

Figura 17-b Entrenamiento con 6 nodos de entrada

Cada red se entrenó variando los parámetros de aprendizaje entre cero y uno, obteniendo el mejor resultado en ambas redes neuronales con: α=0.09 y β=0.1 (ecuaciones 2.14 y 2.15).

El número de nodos ocultos de ambas redes fue de 20, pero mientras que la red neuronal para los patrones de 12 muestras tenía 1240 conexiones (totalmente

conectada), la red neuronal para los patrones de 6 muestras tenía solo 300 conexiones.

Por otro lado, para verificar que no hubo sobreentrenamiento en la red que ofrece mejores resultados (red con 12 nodos entrada), se probó la red con los patrones de validación y se obtuvo un error MSE = 0.13439.

CAPITULO 7 REDES NEURONALES RBF

7.1 RESULTADOS CON REDES NEURONALES RBF

Cuando se usan redes neuronales RBF el número de parámetros que se deben optimizar aumenta con respecto a las redes neuronales usadas en los capítulos anteriores; en este tipo de redes es necesario trabajar por lo menos en la etapa de inicialización con cinco parámetros: cota superior para el peso, cota inferior del peso, inicialización del bias para los nodos ocultos (este es, tal vez el parámetro más importante ya que afecta la forma de la función radial), el factor λ, que sirve para suavizar la curva, y por lo tanto, para evitar el sobreentrenamiento (ecuación 2.18), y el factor de perturbación de los vectores T, que también evita el sobreentrenamiento (Ref 5).

Además, en la etapa de optimización es necesario por lo menos trabajar con tres parámetros más: las tasas de aprendizaje para la actualización de los vectores T, de los bias de la RBF, de los pesos entre la capa oculta y la capa de salida, y el error mínimo (ver sección 1.6), que es útil para no sobré entrenar la red.

Y por ultimo, está el número de nodos ocultos; obteniendo un total de 9 parámetros, los cuales se deben escoger de la manera más adecuada posible; por lo tanto, el método usado para este tipo de redes es más laborioso.

Se usaron 10, 20, 30, 40, y 80 nodos en la capa oculta, para cada una de estas configuraciones, se usó el proceso que se explica a continuación.

Etapa de inicialización:

1. Basándose en la literatura (Ref 5), se encontró que las cotas máxima y mínima, aconsejadas para la inicialización de los pesos podían estar entre 4

y –4 respectivamente, así que estos fueron los valores seleccionados.

2. Al valor del bias para los nodos de la capa oculta, se le asigno un valor inicial de 1, luego se varió dependiendo del comportamiento de la red, es decir si al disminuir el valor inicial del bias se obtenía un error mínimo mayor entonces se cambiaba el bias por un valor mayor de 1.

3. Para los valores de λ y del factor de perturbación se usaron los valores más usados en este tipo de redes, 0.2 y 0.05 respectivamente (Ver ecuación 2.18).

Etapa de optimización:

Una vez que se ha inicializado la red neuronal, se tiene que optimizar los valores de los pesos de la red (sección 1.6.1), esto se hizo así:

1. Primero se entrenó la red actualizando únicamente los vectores T, que equivalen en la red, a los pesos entre la capa de entrada y la capa oculta. Esto se realizó variando la tasa de aprendizaje η_1 entre 0.01 y 0.9 (ecuación 2.19).

2. Cuando se encontró un valor adecuado para η_1, se prosiguió a optimizar el valor de los bias de los nodos ocultos, asignado un valor entre 0.005 y 0.2 a la tasa de aprendizaje η_2 (ecuación 2.20).

3. Luego se optimizó el valor de los pesos entre la capa oculta y la capa de entrada variando la tasa η_3 entre 0.005 y 0.2 (ecuación 2.21).

Vale la pena resaltar, que para las tres tasas de aprendizaje, la convergencia de la red neuronal se volvía inestable para valores mayores de 0.2. Por otro lado, las redes neuronales con 10, 20, 30 y 40 nodos ocultos, no funcionaron

correctamente, ya que el mínimo error que se obtuvo entre todas las configuraciones fue de 0.52 para la red neuronal con 40 nodos ocultos, también se encontró la tendencia de que el error disminuía a medida que se aumentaba el número de nodos en la capa oculta.

Para la red neuronal con 80 nodos en la capa oculta, los resultados fueron mejores, y es posible que el error siga disminuyendo para redes neuronales con más nodos ocultos; sin embargo, esto implica un mayor costo de tiempo en el proceso de inicialización y de optimización.

En resumen, los valores más adecuados encontrados para la red con 80 nodos en la capa oculta, fueron los siguientes: η_1=0.1, η_2=0.05, η_3=0.2, y una inicialización del bias de los nodos ocultos igual a 2.5. La convergencia del error se muestra en la figura 20. Hay que tener en cuenta que la red neuronal con 80 nodos ocultos, 120 nodos de entrada y un nodo de salida, tiene 9680 conexiones, siendo una arquitectura mucho más grande que la que se obtuvo con el método de pruning del capitulo 5.

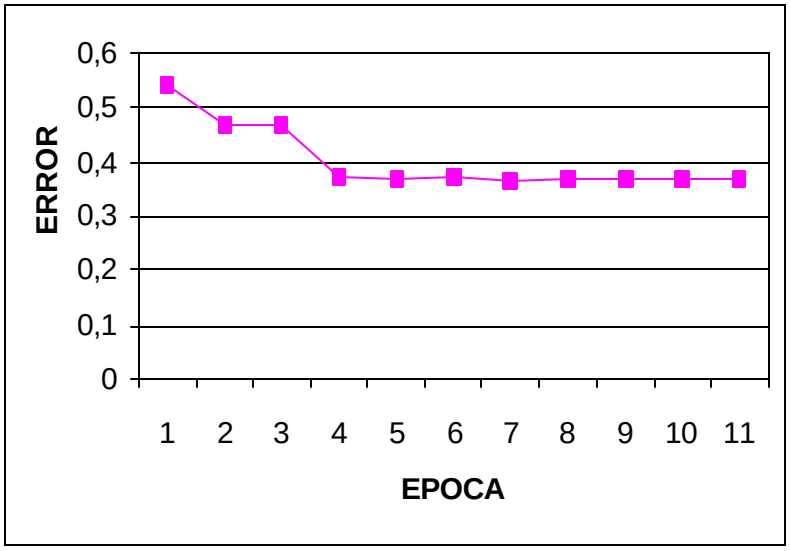

Figura 20

Al comparar este tipo de red con la encontrada en el capitulo 5 (Figura 15), se observa que a pesar de tener 80 nodos en la capa oculta, mucho más que los 16 nodos de la red del capitulo 5, esta nueva red no ofrece mejores resultados; así que es posible mejorar la calidad de estos usando redes neuronales RBF con conexiones adicionales entre la capa de entrada y de salida (Ref 5).

CAPITULO 8 ALGORITMO STA –LTA vs. REDES NEURONALES

8.1 DESCRIPCIÓN DEL ALGORITMO STA – LTA

Actualmente este algoritmo es el más usado en sismología para la detección de ondas regionales (como por ejemplo ondas Pn y Sn). Este continuamente calcula el promedio de las amplitudes absolutas de señal sísmica en dos consecutivas ventanas de tiempo. El término STA se refiere al promedio en la venta de tiempo pequeña (short time average), y es sensitivo a eventos sísmicos, mientras que el termino LTA se refiere al promedio en la ventana grande de tiempo, el cual tiene información acerca de la amplitud temporal del ruido sísmico. Cuando la razón de ambos excede un valor determinado (threshold), un evento es declarado.

Para mejorar el algoritmo, no solo se tiene en cuenta la amplitud absoluta de la señal, también se calcula la primer derivada temporal de la traza con respecto al tiempo, esto hace que el algoritmo sea sensitivo también al cambio de la frecuencia; por lo tanto el promedio se realiza con la llamada función característica E(t)

$$E(t) = f(t)^2 + C_2 * f'(t)^2$$

donde f(t) es la amplitud de la señal en el tiempo t, f ’(t) es la derivada temporal, y C_2 es un coeficiente que varía la relativa importancia entre el promedio de la señal en el tiempo y su derivada.

Para el algoritmo STA-LTA, se usaron los parámetros establecidos en el sistema de adquisición de datos, de la Red Sismológica Nacional, correspondientes a la estación de Barichara:

- STA 0.25 segundos

- LTA 1 segundo
- Treshold 60
- C_2 0.5

8.2 COMPARACIÓN ENTRE EL ALGORITMO STA - LTA Y LAS REDES NEURONALES

Se escogieron al azar 10 patrones de validación, a los cuales se les calculó el algoritmo STA-LTA; y se guardó la muestra con el valor respectivo de STA/LTA para todos los 10 patrones.

Por otro lado, el software SNNS (anexo C) permite salvar los resultados en archivos con extensión ".res", en los cuales aparece para cada patrón (de validación) las salidas que la red neuronal calculó. Así que por cada muestra Ai en el patrón (total 120 muestras), se creó una ventana de tiempo de 2 segundos (120 muestras) centrada en la muestra Ai, por lo tanto, se obtuvieron 10*120 =1200 nuevos patrones. Estos 1200 patrones se procesaron con la red neuronal que corresponde a la figura 14, y se guardó el archivo de resultados.

Ahora se tiene para cada muestra de los 10 patrones escogidos al azar, un valor STA/LTA, una salida de la red neuronal, y también sabemos que para todos los 10 patrones, la muestra 60 fue la marcada por el humano como arribo de la onda Pn (Ver capítulo de preprocesamiento.).

Conociendo las tres muestras: para STA-LTA es la primer muestra donde STA/LTA > umbral, para el humano es la muestra 60, y para la red neuronal es la muestra con mayor valor de salida (la cual esta entre cero para ruido y 1 para arribo de onda Pn); se puede calcular la difererencia absoluta entre las muestras del STA-LTA y humano; y muestras de la red neuronal y humano; y así saber cual método (STA-LTA o Redes neuronales) da resultados más cercanos a la realidad

(de ahí la necesidad de revisar con mucho cuidado los patrones iniciales para evitar que haya errores humanos en la detección del arribo de la onda Pn).

Los resultados fueron los siguientes:

Promedio de la diferencia absoluta entre la muestra humano y muestra de STA/LTA	**Promedio de la diferencia absoluta entre la muestra humano y muestra de red neuronal**
4.5 muestras	6.8 muestras

Tabla 2

Según los anteriores resultados el algoritmo STA-LTA funciona un poco mejor que la red neuronal obtenida con el método de *prunnig,* sin embargo, existen posibles métodos mencionados en el ultimo capitulo, que permiten a la red neuronal tener mejores resultados.

CAPITULO 9 CONCLUSIONES Y TRABAJOS FUTUROS

9.1 CONCLUSIONES

- Con eL algoritmo STA-LTA se obtienen mejores resultados que con las redes neuronales multicapa entrenadas con el algoritmo *backpropagantion.* Lo cual se puede deber ha varios factores, tales como: La complejidad de los patrones y en especial la del ruido hace que la arquitectura de las redes neuronales de tipo Kohonen sean muy simples para clasificar este tipo de patrones. Las redes neuronales RBF fueron inicializadas con un algoritmo que escoge aleatoriamente, del conjunto de patrones, los centros de las funciones. Este método talvez no el mas optimo ya que no tiene en cuenta todo el conjunto de patrones para establecer las clases.

- El término del *momentum* no mejoró significativamente la velocidad de convergencia del proceso de aprendizaje en la red entrenada con el algoritmo Backpropagation.

- El método de pruning usado fue muy sencillo pero eficiente, ya que redujo enormemente el número de conexiones sin que se perdiera generalidad, aunque sí aumento el número de épocas que se necesitan para llegar a obtener un buen entrenamiento de la red neuronal.

- Las redes Kohonen por si solas no fueron suficientes para clasificar los dos tipos de patrones (sismo o ruido.). Se requirió realizar decimación para reducir la complejidad en los patrones, y usar una red neuronal un poco más sofisticada, como lo es la red tipo counterpropagation, la cual consiste en una capa oculta de tipo Kohonen y una capa de salida de tipo *Grossberg*.

- Con las redes Counterpropagation no se obtiene mejores resultados que con

las redes multi-capa de perceptrones.

- El ancho de la ventana de tiempo que se usa para escoger las muestras que serán introducidas en la red neuronal, es un parámetro muy importante en el modelo, ya que este determina en gran medida varios aspectos, tales como: complejidad de la red neuronal, tiempo de convergencia, si es muy pequeña puede generar falsas alarmas, si es muy grande puede provocar la perdida de señal sísmica.

- Las redes neuronales RBF con un número de nodos ocultos entre 10 y 40 convergía más rápido que la red neuronal entrenada con el algoritmo de Backpropagation. Sin embargo cuando se tenía 80 nodos en la capa oculta, la convergencia y la inicialización de la red, presentaban más costo temporal.

- Los mejores resultados usando redes neuronales RBF, fueron usando 80 nodos en la capa oculta, sin embargo estos no son mejores que los obtenidos con las redes neuronales multicapa, entrenadas con el algoritmo de Backpropagation.

- Tal vez sea posible mejorar la calidad de los resultados con las redes neuronales RBF, incluyendo conexiones adicionales entre los nodos de entrada y los nodos de salida (Ref 5).

9.2 TRABAJOS FUTUROS

- Este trabajo se desarrolló solo con una estación sismológica, será necesario realizar trabajos más exhaustivos, para saber si una sola arquitectura de red neuronal, es suficiente para que determine el arribo de las ondas Pn. Si no es el caso, es posible que sea necesario hacer un proceso diferente para cada estación sismológica.

- Si el tiempo de arribo determinado para la fase Pn por una red neuronal, resulta ser efectivamente mejor que el algoritmo STA-LTA para todas las estaciones sismológicas de interés, es conveniente implementar un sistema de localización de sismos automático en tiempo real; basado en Redes Neuronales.

- Es posible extender esta metodología en posteriores trabajos, al reconocimiento de ondas sísmicas diferentes de Pn, como por ejemplo ondas Sn.

- Tal vez sea posible mejorar la eficiencia de la red neuronal con la combinación de mejores métodos de preprocesamiento, por lo cual es interesante realizar trabajos futuros que usen procesos más sofisticados como los del tratamiento del sonido (Ref 7).

- Con respecto al uso de redes RBF, es conveniente modificar la arquitectura usada en este trabajo, agregando conexiones entre los nodos de la capa de entrada y la capa de salida, lo cual puede mejorar la convergencia de la red (Ref 5).

- Teniendo en cuenta que la etapa de inicialización en las redes neuronales RBF es bastante importante, es posible que se mejore los resultados si se escoge un método diferente de inicialización, como por ejemplo el RBF_Weights_Kohonen, el cual consiste en usar métodos asociativos como lo es el método de Kohonen, para seleccionar los centros de una manera más apropiada.

BIBLIOGRAFÍA

1. INTELIGENCIA ARTIFICIAL Y MINIROBOTS, Alberto Delgado, Ecoe ediciones.

2. RADAR TARGET RECOGNITION USIGN A RADIAL BASIS FUNCTION NEURAL NETWORK, Qun Zhao, Neural Networks,Vol 9 no.4 pp 709-720,1996

3. SEISMIC DISCRIMINATION WITH ARTIFICIAL NEURAL NETWORK. PRELIMINARY RESULTS WITH REGIONAL SPECTRAL DATA. Farid U. Dowla, Bulletin of the Seismological Society of America, vol 80, no 5 pp 1346-1373,1990.

4. JNN, A RANDOMIZED ALGORITHM FOR TRAINIG MULTILAYER NETWORKS IN POLYNOMIAL TIME. Andre Elisseeff, Neurocomputing, Vol 29, pp 3-24,1999.

5. STUTTGART NEURAL NETWORK SIMULATOR. Universidad de Stuttgart.

6. AN ARTIFICIAL NEURAL NETWORK APPROACH FOR BROADBAND SEISMIC PHASE PICKING.Yue Zhao, Bulletin of the Seismological Society of America, vol 89, no 3, pp 670-680,1999.

7. A SPEECH RECOGNITION METHOD BASED ON THE SEQUENTIAL MULTI-LAYER PERCEPTRONS. Wen Yuan Chen, Neural Networks,Vol 9 no.4 pp 655-669,1996

8. AUTOMATIC EARTHQUAKE RECOGNITION AND TIMING FROM SINGLE TRACES. Rex V. Allen, Bulletin of the Seismological Society of America, vol 68, no 5, pp 1521-1532,1978.

9. MODEL SELECTION IN NEURAL NETWORKS. U. Anders, Neural Networks, vol 12, pp 309-323,1999

10. GLOBAL MODERN SISMOLOGY. Richard E. Woods, Addison-Wesley,1987.

11. INTELIGENCIA ARTIFICIAL. 3RA EDICIÓN. Patrick Henry Winston. Addison Wesley Iberoamericana.

12. NEW MANUAL OF SEISMOLOGICAL OBSERVATORY PRACTICE. http://www.seismo.com/msop/nmsop/nmsop.html. Global Seismological Services.

13. DIFFERENT CRITERIA FOR ACTIVE LEARNIG IN NEURAL NETWORKS: A COMPARATIVE STUDY. Jan Poland, Andreas Zell. Neural Networks.

14. APPLICATION OF FEED-FORWARD ARTIFICIAL NEURAL NETWORKS TO THE IDENTIFICATION OF DEFECTIVE ANALOG INTEGRATED CIRCUITS. D. Micusik, V.Stopjakova, L. Benuskova. Neural computing and applications. Vol 11, pp 71-79, 2002.

15. INTRODUCTION TO TOPOLOGY AND MODERN ANÁLYSIS. G. F. Simmons. McGraw-Hill international editions.

16. Training radial basis neural networks with the extended Kalman filter. Dan Simon. Neurocomputing. Vol 1, pp 1- 25, 2002

ANEXO A. CARACTERÍSTICAS TÉCNICAS DE LA ESTACIÓN SISMOLÓGICA DE BARICHARA

La estación sismológica de Barichara hace parte de la Red Sismológica Nacional (Ingeominas) y su ubicación geográfica es de 73.5 grados oeste y 6.5 grados norte (figura 21.).

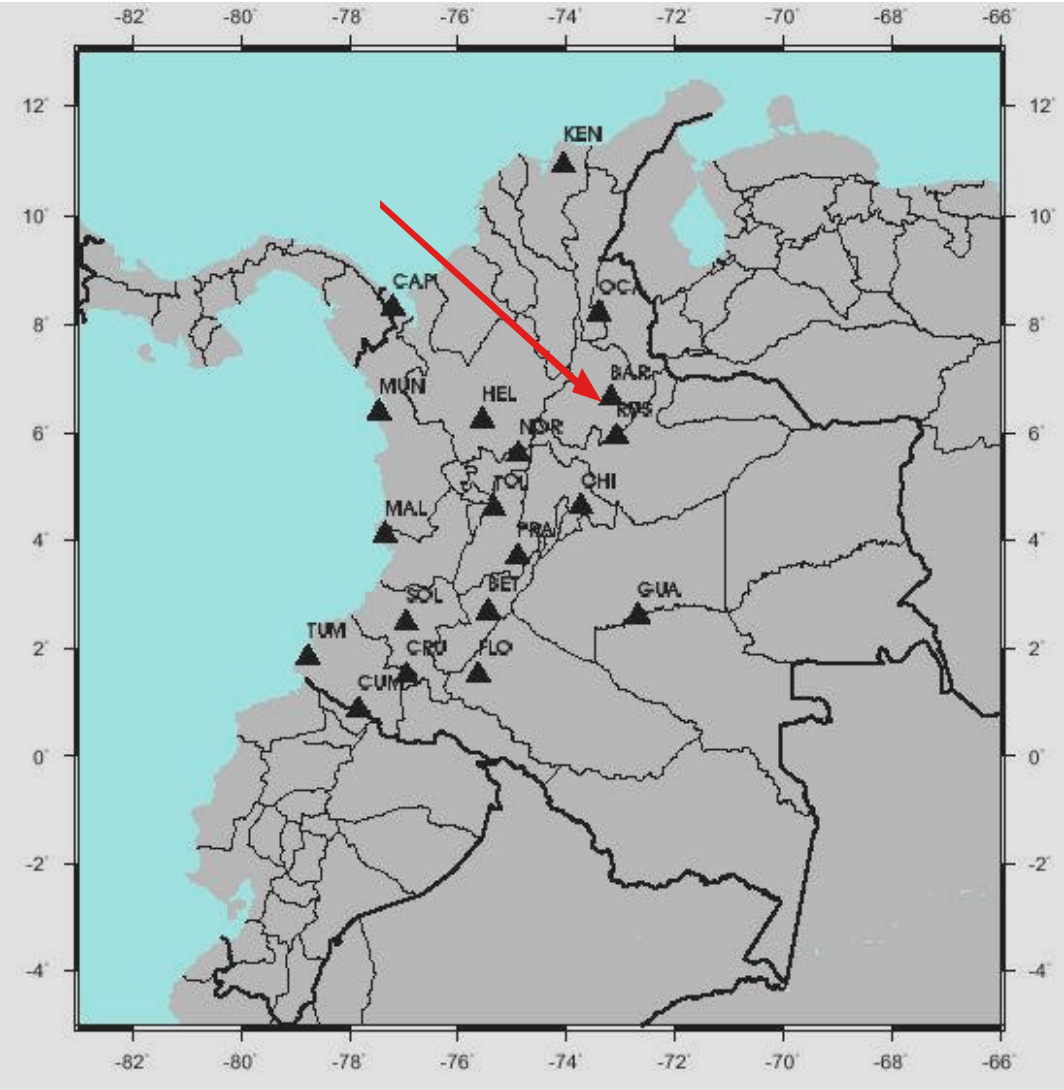

Figura 21

Esta estación sismológica cuenta con un sensor llamado sismógrafo el cual se

encarga de medir la velocidad relativa del suelo con respecto a una masa que tiende a estar en reposo debido a su inercia (figura 22), el sismógrafo tiene una frecuencia natural de 1 seg, y responde aceptablemente para frecuencias entre 0.1 y 10 Hz; además solo es de componente Z, es decir registra únicamente el movimiento en el eje vertical (para mas información visitar www.ingeominas.gov.co.).

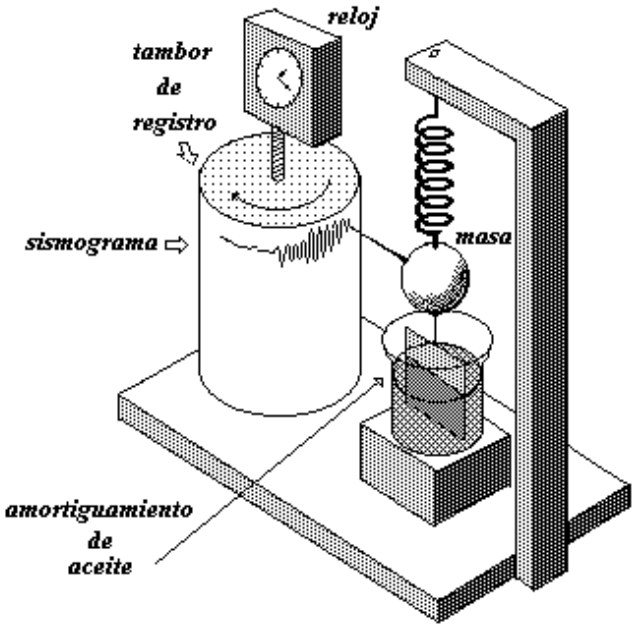

Figura 22

Además, existe en el lugar un digitalizador que se encarga de hacer un muestreo de la señal análoga del sensor a 60 muestras / segundo, o sea que por el teorema de Nyquist solo es capaz de digitalizar sin perder información, señales con frecuencias menores o iguales a 30 Hz.

ANEXO B. RUTINAS ELABORADAS EN LENGUAJE C PARA EL PREPROCESAMIENTO DE LOS DATOS Y ANÁLISIS DE LOS RESULTADOS

El cdrom que acompaña esta tesis contiene la información necesaria para reproducir y/o perfeccionar los resultados. La estructura de directorios es la siguiente:

- En el directorio sismos: Se encuentran los datos simológicos registrados por la RSNC que se seleccionaron para crear los patrones de entrenamiento, en formato ASCII.

- En el directorio patrones: Están los patrones de entrenamiento que se usaron para las redes neuronales.

- En el directorio topologías: Están las estructuras de las redes neuronales.

- En el directorio rutinas: Se encuentra las rutinas escritas en lenguaje C para linux, que permiten realizar el preprocesamiento de las señales sísmicas; las rutinas son:

La rutina filtra.c revisa que las señales sísmicas seleccionadas estén correctas (no tengas Gaps y tengan localización asociada.).

La rutina procesa.c realiza el preprocesamiento de las señales sísmicas antes de convertirse a formato SNNS.

La rutina doPatron.c crea el archivo de patrones en formato SNNS para cada señal sísmica seleccionada.

La rutina shortPatterns.c distribuye de manera aleatoria los patrones individuales en un solo archivo de patrones.

La rutina stalta.c calcula la razón STA/LTA para una señal sísmica.

ANEXO C. SNNS STTUGART NEURAL NETWORK SIMULATOR

En el desarrollo de esta tesis se uso el software SNNS versión 4.2 para Linux. Este software es un simulador para el desarrollo de redes neuronales desarrollado en la universidad de Sttugart. El SNNS consiste en cuatro principales componentes, los cuales son: kernel, interfase grafica XGUI, lenguaje de comandos, compilador de redes snns2c. Este permite crear complejas redes neuronales y controlar la simulación de esta. El software es implementado totalmente en lenguaje ANSI-C y puede ser adquirido libremente en la dirección: ftp.informatik.uni-sttugart.de.

A pesar de que esta versión a sido probado en varios sistemas operativos, al usarlo bajo Linux Red Hat 7.3 la instalación presento un problema de compilación, este fue debido a que en la librería string2.h de Linux existe una constante con el mismo nombre que una variable local usada por SNNS, por lo que es necesario cambiar el nombre de dicha variable local en el código fuente.