*Universidad de las Ciencias Informáticas*
*Facultad 6*


# *Título:"Efecto de la variación de las Funciones de Transferencia en Perceptrones Multicapa"*

***Trabajo de Diploma para optar por el título de***

***Ingeniero Informático***

**Autores:** Yuniel Valenzuela Ramos
Osniel Tortosa Guerra

**Tutores:** Dr. Ramón Carrasco Velar
Ing. Yuleidys Mejías César

***Ciudad de la Habana, Cuba.***
***Junio ,2009***

***"Año del 50 Aniversario del triunfo de la Revolución."***

*"Son vanas y están plagadas de errores las ciencias que no han nacido del experimento, madre de toda certidumbre".*

*Leonardo da Vinci*

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yuniel Valenzuela Ramos

_____

Firma del Autor

Osniel Tortosa Guerra

_____

Firma del Autor

Dr. Ramón Carrasco Velar

_____

Firma del Tutor

Ing. Yuleidys Mejías César

_____

Firma del Tutor

**DATOS DE CONTACTO.**

**Tutores:**

Dr. Ramón Carrasco Velar
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
rcarrasco@uci.cu

Ing. Yuleidys Mejías César
Universidad de las Ciencias Informáticas, Ciudad de LaHabana, Cuba.
ymejias@uci.cu

**Agradecimientos.**

*A la UCI por ser nuestro  hogar, por educarnos y  darnos tantos amigos.*
*A nuestros tutores Yuly y Carrasco por la confianza y ayuda que nos brindaron siempre.*
*A los amigos, convertidos en hermanos muchas veces, por acompañarnos todo este tiempo.*
*A nuestras familias por ser siempre el mayor apoyo y por darnos un rayito de luz en los momentos más difíciles de la carrera.*

***Dedicatoria***

*A mis padres,incluyendo a mi padrasto, a mi hermana, a mi novia , a mi abuelos ,a mis primos y a mis tías, en fin ,a esa gran familia que siempre ha creído en mi.*
*A todas las personas que en un futuro utilicen esta tesis para su desarrollo profesional.*

*Osniel*

*A mi madre y a mi padre, porque siempre mi primer pensamiento va con ellos.*
*A mi hermano, para que se inspire y se esfuerce cada día más.*
*Y a mi primo Julio, por ser la primera persona que me habló sobre la informática.*

*Yuniel.*

**Resumen**

Durante mucho tiempo las Redes Neuronales Artificiales (RNA) han sido utilizadas en problemas de predicción de estructuras y actividad biológica. Es por ello que muchas personas se han dedicado a estudiar su funcionamiento con el fin de optimizar los algoritmos basados en ellas, de ahí que se conozca que una de las etapas fundamentales para el buen funcionamiento de una red es la del entrenamiento, donde las funciones de transferencias juegan un papel fundamental. Dentro de la amplia gama de RNA que existen se ha seleccionado el Perceptrón Multicapa como objeto de estudio ya que es uno de los que mejores resultados ha aportado en los estudios de relación estructura-actividad. Ha sido una incógnita conocer el comportamiento de una red Multicapa al variar las funciones de transferencia, por lo que esta investigación propone obtener esos resultados y dar una conclusión al respecto. Después de estudiar varias herramientas que trabajan con las RNA se decidió utilizar Matlab ya que es una herramienta muy potente y por sus características facilita nuestro estudio.. A partir de una serie de experimentos realizados se llegó a la conclusión de que funciones lineales como la *poslin, satlin y satlins* al combinarse con las funciones sigmoidales *tansig y logsig* en las capas ocultas brindan resultados satisfactorios, además se corroboró que la red no debe tener más de dos capas ocultas, así como que es recomendable colocar en las capas de entrada y salida la función *tansig*.

**Palabras claves**: Perceptrón Multicapa, Red Neuronal Artificial, Backpropagation, Matlab, función de transferencia.

## Índice

## Índice de Figuras

## Índice de Tablas

## Introducción

Imitar la inteligencia humana y lograr que un sistema informático tenga un comportamiento inteligente, ha sido uno de los principales retos de los científicos a lo largo de la historia, es por ello que por mucho tiempo se han dedicado al estudio minucioso del funcionamiento del cerebro como órgano rector de la inteligencia humana. Con el fin de lograr estos sistemas, surge la Inteligencia Artificial en el año 1956, la cual tiene como objetivo principal proveer soluciones a problemas que son bastante complejos para la tecnología actual y que en su mayoría no tienen una solución algorítmica.

Uno de los paradigmas de la Inteligencia Artificial sin dudas lo constituye las RNA las cuales imitan el funcionamiento del cerebro humano. El interés inicial en las redes neuronales artificiales proviene por la creencia de que permitirían mejorar el conocimiento del cerebro y la percepción humana con la esperanza de que se pudieran crear sistemas pensantes que tuvieran mejores resultados en tareas como clasificación, problemas de decisión, pronósticos y sistemas de control adaptables.

Las primeras ideas sobre RNA datan del año 1943 de manos de Warren MacCulloch y Walter Pitts las cuales fueron forjando un ambiente que dio lugar al descubrimiento del Perceptrón en el año 1959 por el psicólogo Frank Rosemblatt. Más adelante en el año 1984 debido a las limitantes del Perceptrón Simple se obtiene el Perceptrón Multicapa que está compuesto por al menos tres capas: una capa de entrada, una capa oculta y por último una de salida. Este tipo de red neuronal artificial ha dado lugar a varios estudios en distintas ramas de la ciencia.

Uno de los elementos principales del funcionamiento de las RNA son las funciones de transferencia o de activación. La función de activación de una neurona artificial simula la respuesta de una neurona biológica ante un cierto estímulo. Se conoce que el cerebro posee un comportamiento diferente ante cada estímulo, sin embargo, en la mayoría de los casos las RNA son utilizadas con una única función de transferencia. A raíz de esta situación surge la investigación actual, la cual está encaminada a conocer el efecto de la variación de las funciones de transferencia en el Perceptrón Multicapa.

El problema científico al que se le intentará dar solución durante la presente investigación es ver: ¿Cómo afectan las variaciones de las funciones de transferencia a los resultados de las Redes Neuronales Artificiales?, específicamente al Perceptrón Multicapa.

La hipótesis de la presente investigación es la siguiente: Es posible mejorar los resultados en un Perceptrón Multicapa variando las funciones de transferencia en sus diferentes capas. Después de plantear la hipótesis de la investigación es necesario tratar el objetivo general que no es más que: Evaluar el efecto de la variación de las funciones de transferencia en Perceptrones Multicapa.

A partir de un análisis del objetivo general se derivan los siguientes objetivos específicos:

- Evaluar un Perceptrón Multicapa (MLP) con una misma función de transferencia sigmoidal.
- Evaluar un Perceptrón Multicapa (MLP) variando las funciones de transferencia sigmoidales.
- Evaluar un Perceptrón Multicapa (MLP) combinando las funciones de transferencia sigmoidales con las lineales.
- Arribar a conclusiones a partir de los resultados obtenidos.

El problema científico de la investigación da lugar a la siguiente novedad científica: Investigar qué sucede cuando se utilizan varias combinaciones de funciones de transferencia durante el proceso de entrenamiento del Perceptrón Multicapa (MLP).

## Capítulo 1. Fundamentación Teórica

En el presente capítulo se hace un estudio de las Redes Neuronales Artificiales, partiendo de algunas definiciones de las mismas, además se hace un análisis más detallado del Perceptrón Multicapa porque es el tipo de red neuronal artificial que es objeto de estudio de la investigación. En el entrenamiento de una RNA, las funciones de transferencia juegan un papel fundamental, por lo que se se hace un estudio de estas y se presentan las más conocidas junto a una breve panorámica de las mismas.

Otros aspectos presentados en este capítulo y que también juegan un papel muy importante en el aprendizaje de una RNA son los tipos de entrenamiento existentes, así como el algoritmo de entrenamiento Backpropagation.

### 1.1 La Neurona Artificial

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.

Como se observa en la **Fig. 1**, dicha unidad de proceso consta de una serie de entradas Xi, que equivalen a las dendritas, en la neurona biológica, de donde reciben la estimulación, ponderadas por unos pesos Wi, que representan como los impulsos entrantes son evaluados y se combinan con la función de red que 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.

Como se puede ver en la siguiente ilustración, la neurona artificial se comporta como la neurona biológica pero de una forma muy simplificada.

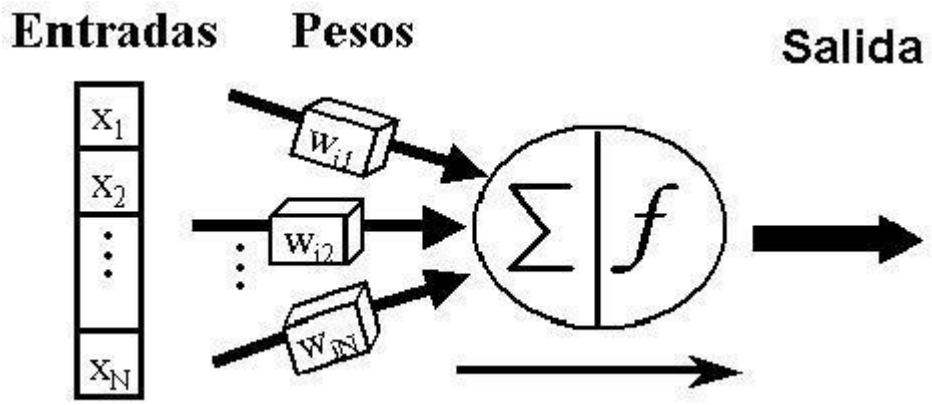


**Fig. 1** Neurona Artificial

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 través de las dendritas en la neurona biológica.

Los pesos que hay en las sinapsis Wi, 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 (Xi y Wi) 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 típicamente 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.

## 1.2 Red Neuronal Artificial

Hoy en día podemos encontrar diversas bibliografías que abordan el tema de las RNA, en todas ellas se brindan definiciones a partir del punto de vista del autor. Veamos a continuación algunas de las definiciones más actuales:

Según Hayking [1], una Red Neuronal Artificial es un procesador paralelo y distribuido que tiene la facilidad natural para almacenar conocimiento experimental y hacerlo útil para su uso, asemejando al cerebro en dos aspectos:

- El conocimiento es adquirido por la red a través de un proceso de Aprendizaje.
- La "fuerza" de las conexiones inter-neurona, conocida como pesos sinápticos son utilizados para almacenar el conocimiento.

Las Redes Neuronales Artificiales son modelos matemáticos multiparamétricos no-lineales, capaces de inducir una correspondencia entre conjuntos de patrones de información (la relación estímulo-respuesta) [2].

El procesamiento de la información se realiza a la manera de los sistemas neuronales biológicos. Una red neuronal puede verse como un grafo dirigido con las siguientes características:

- Los nodos del grafo se denominan elementos de procesamiento, unidades o neuronas.
- Las uniones entre los nodos se denominan conexiones. Cada conexión funciona en un momento determinado en una única dirección. A cada conexión se le asigna un factor de ponderación denominado peso o intensidad de conexión. Las conexiones pueden ser excitatorias o inhibitorias.
- Cada elemento puede recibir cualquier número de conexiones de entrada procedentes del resto de elementos de procesamiento que forman la red.
- Cada elemento de procesamiento tendrá una única salida que se podrá ramificar para ser aplicada a muchos otros elementos de procesamiento. Así, la salida de una unidad se tomará como entrada a cada uno de los elementos de procesamiento conectados con dicha unidad.
- Cada elemento de procesamiento puede tener una memoria local.

- Cada elemento de procesamiento posee una función de activación que usa la memoria local y las señales de entrada para producir una señal de salida. La salida producida por un elemento de procesamiento depende exclusivamente de los valores actuales de entrada que le llegan de otros elementos de procesamiento a través de las conexiones y los valores almacenados en su memoria local, es decir, el procesamiento de información en cada elemento de procesamiento es completamente local[2].

En general las Redes Neuronales Artificiales tienen un grupo de características como son:

- Procesamiento Paralelo (las neuronas operan en paralelo).
- Memoria Distribuida (información almacenada en las conexiones).
- Aprendizaje (modificación de pesos de conexiones en base a ejemplos de aprendizaje).

### 1.2.1 Tipos de redes neuronales artificiales.

Existen diferentes tipos de RNA, entre ellas podemos citar:

- El Perceptron Simple.
- La Red de Hopfield.
- El Perceptron Multicapa.
- Red neuronal Competitiva Simple.
- Redes Neuronales Online ART1.
- Redes Neuronales competitivas ART2.
- Redes neuronales autoorganizadas: Mapas de Kohonen.

Durante la investigación se hace un estudio profundo del Perceptrón Multicapa, como se muestra en el subepígrafe 1.2.4.

### 1.2.2 Clasificación de las Redes Neuronales Artificiales

Las Redes Neuronales Artificiales se clasifican según su Arquitectura y el Aprendizaje.

Según la arquitectura se clasifican en monocapas y multicapas, las monocapas son aquellas redes que solo cuentan con una capa, la más representativa es la red de Hopfield. Las redes monocapa han sido ampliamente utilizada en circuitos eléctricos ya que debido a su topología, son adecuadas para ser implementadas mediante hardware, usando matrices de diodos que representan las conexiones de las neuronas.

Las redes multicapas están formadas por varias capas de neuronas. Estas redes se pueden a su vez clasificar atendiendo a la manera en que se conexionan sus capas.

Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Ese tipo de conexiones se denominan conexiones feedforward o hacia delante.

Por el contrario existen algunas redes en que las capas aparte del orden normal algunas capas están también unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información. Las conexiones de este tipo se llaman conexiones hacia atrás, feedback o retroalimentadas.

Las redes con conexiones hacia adelante contienen solo se conectan con las capas hacia delante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella en la dinámica de la computación.

Ejemplos de estas redes son Perceptron, Adaline, Madaline, Backpropagation y los modelos LQV y TMP de Kohonen.

Las redes con conexión hacia atrás se diferencia en las anteriores en que si pueden existir conexiones de capas hacia atrás y por tanto la información puede regresar a capas anteriores en la dinámica de la red. Este Tipo de redes suelen ser bicapas ejemplos de estas redes son las redes ART, Bidirectional Associative Memory (BAM) y Cognitron.

Según el aprendizaje las RNA se clasifican en Redes con Aprendizaje Supervisado y No Supervisado. Estos tipos de aprendizaje se explican en los subepígrafes 1.3.2 y 1.3.3.

### 1.2.3 Funcionamiento de la Red Neuronal Artificial

En cuanto al modo interno de trabajo, las redes neuronales son modelos matemáticos recreados mediante mecanismos artificiales (como un circuito integrado, un ordenador o un conjunto de válvulas). El objetivo es conseguir que las máquinas den respuestas similares a las que es capaz de dar el cerebro que se caracterizan por su generalización y su robustez. Estos modelos matemáticos que emplean las redes son multivariantes que utilizan procedimientos iterativos, en general para minimizar funciones de error [3]. Las neuronas se agrupan en capas, constituyendo una red neuronal. Una determinada red neuronal es confeccionada y entrenada para llevar a cabo una labor específica. Los modelos neuronales utilizan varios algoritmos de estimación, aprendizaje o entrenamiento para encontrar los valores de los parámetros del modelo, que se denominan pesos sinápticos. Originalmente la red neuronal no dispone de ningún tipo de conocimiento útil almacenado. Para que la red neuronal ejecute una tarea es preciso entrenarla, en terminología estadística se diría que es necesario estimar los parámetros. En realidad todo el procedimiento es estadístico: primero se selecciona un conjunto de datos, o patrones de aprendizaje. Después se desarrolla la arquitectura neuronal, número de neuronas, tipo de red; es decir, se selecciona el modelo y el número de variables dependientes e independientes.

### 1.2.4 Perceptrón Multicapa

El Perceptrón Multicapa es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del Perceptrón Simple.

Las redes multicapas se pueden clasificar atendiendo a la manera en que se conexionan sus capas. Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Ese tipo de conexiones se denominan conexiones feedforward o hacia delante.

Por el contrario existen algunas redes en que las capas aparte del orden normal algunas capas están también unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información. Las conexiones de este tipo se llaman conexiones hacia atrás, feedback o retroalimentadas.

La arquitectura del Perceptrón Multicapa se define de la siguiente manera:

Esta red consta de una capa de entrada y una capa de salida y una o más capas ocultas. Dichas capas se unen de forma total hacia delante, esto es, la capa entrada se une con la primera capa oculta y esta con la siguiente y la última capa oculta se une con la capa de salida. Los valores que el Perceptrón Multicapa acepta son reales. [4]

Las capas pueden clasificarse en tres tipos:

- **Capa de entrada:** Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- **Capas ocultas:** Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capas posteriores.
- **Capa de salida:** Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

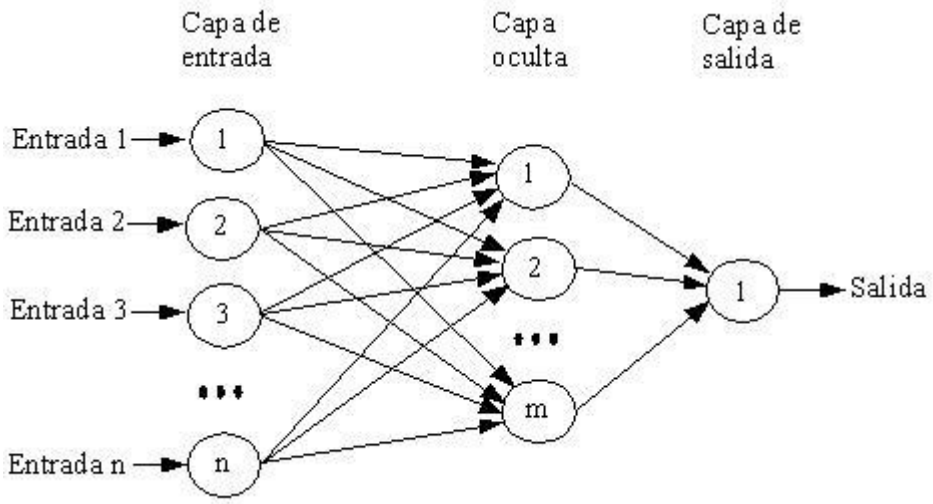


**Fig. 2** Perceptrón Multicapa

### 1.3 Entrenamiento de la Red Neuronal Artificial.

Las redes neuronales manejan dos tipos de información. La primera, es la información volátil que se refiere a los datos que se están usando y varían con la dinámica de la computación de la red, se encuentra almacenada en el estado dinámico de las neuronas.

El segundo tipo de información que manejan las redes neuronales, es la información no volátil que se mantiene para recordar los patrones aprendidos y se encuentra almacenada en los pesos sinápticos.

El aprendizaje de las redes neuronales, es el proceso de presentar los patrones a aprender, a la red y el cambio de los pesos de las conexiones sinápticas usando una regla de aprendizaje.

La regla de aprendizaje consiste en algoritmos basados en fórmulas matemáticas, que usando técnicas como minimización del error o la optimización de alguna "función de energía", modifican el valor de los pesos sinápticos en función de las entradas disponibles y con ello optimizan la respuesta de la red a las salidas que deseamos.

El aprendizaje se basa en el entrenamiento de la red con patrones, que usualmente son llamados patrones de muestra o entrenamiento. El proceso usual del algoritmo es que la red ejecuta los patrones iterativamente, cambiando los pesos de las sinapsis, hasta que convergen a un conjunto de pesos óptimos que representan a los patrones lo suficientemente bien, entonces mostrará una respuesta satisfactoria para esos patrones [5].

Sin embargo, hay que destacar que algunas redes no tienen un aprendizaje iterativo como el descrito en el párrafo anterior de presentar los patrones una y otra vez hasta que la red se establece para dar resultados correctos, si no que los pesos de las sinapsis son calculados previamente a partir de los patrones, como en la red de Hopfield.

Podemos distinguir tres tipos de aprendizaje, primeramente el Aprendizaje Supervisado, también se encuentra el no Supervisado, dentro de este últimos entran tres variantes: el Aprendizaje Reforzado, Aprendizaje por Componentes Principales y Aprendizaje Competitivo. Más adelante se hace énfasis en el Aprendizaje Supervisado.

### 1.3.1 Funciones de transferencia o activación y de entrenamiento.

La función de transferencia o de activación se encarga de calcular el nivel de activación de la neurona en función de la entrada total, también denota la salida de la neurona.

Básicamente, las funciones de transferencia realizan dos tareas importantes: la primera, es que sirven para limitar la salida de una neurona, y así los resultados no crezcan a valores demasiado grandes; y la segunda, es que proporciona características de no linealidad [6].

| Relación Entrada /Salida | Icono | Función |
|---|---|---|
| $a = 0\ n < 0$<br>$a = 1\ n \geq 0$ |  | *hardlim* |
| $a = -1\ n < 0$<br>$a = +1\ n \geq 0$ |  | *hardlims* |
| $a = 0\ n < 0$<br>$a = n\ 0 \leq n$ |  | *poslin* |
| $a = n$ |  | *purelin* |
| $a = 0\ n < 0$<br>$a = n\ 0 \leq n \leq 1$<br>$a = 1\ n > 1$ |  | *satlin* |
| $a = -1\ n < -1$<br>$a = n\ -1 \leq n \leq 1$<br>$a = +1\ n > 1$ |  | *satlins* |

| | | |
|---|---|---|
| $a = \frac{1}{1+e^{-n}}$ |  | *logsig* |
| $a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$ |  | *tansig* |
| $a = 1$ *Neurona con n max*<br>$a = 0$ *El resto de neuronas* |  | *compet* |
| a= exp (-n2) |  | *radbas* |
| a = 1-abs(n), if -1 <= n <= 1; = 0 |  | *tribas* |

**Tabla 1** Relación de las principales funciones de transferencia empleadas en el entrenamiento de redes neuronales.

De las funciones de transferencia que se muestran en la tabla hay algunas que poseen un uso específico, por ejemplo: la función *radbas* y la *tribas* se utilizan en Redes Radiales Básicas [6].

La función de aprendizaje utilizada durante el entrenamiento de una red Feed- forward Backpropagation en esta investigación es la *traingd*, que se define como una función de aprendizaje de la red que actualiza los pesos acorde al gradiente descendiente.

### 1.3.2 Aprendizaje Supervisado

El aprendizaje es una de las partes más importantes de las RNA, dado que el esquema de aprendizaje utilizado proporcionará a la red la habilidad para resolver uno u otro tipo de problemas.

El aprendizaje supervisado es aquel en el que el entrenador de redes neuronales, es decir, el diseñador, ha de indicar a la red tanto las entradas como las salidas que desea obtener, o sea, el diseñador ha de mostrar a la red las entradas, y corregir sus salidas para que coincidan con unas salidas deseadas. Para ello se regulan los pesos de la red, con el fin de obtener dicho resultado, en un proceso que se denomina entrenamiento de la red [7].

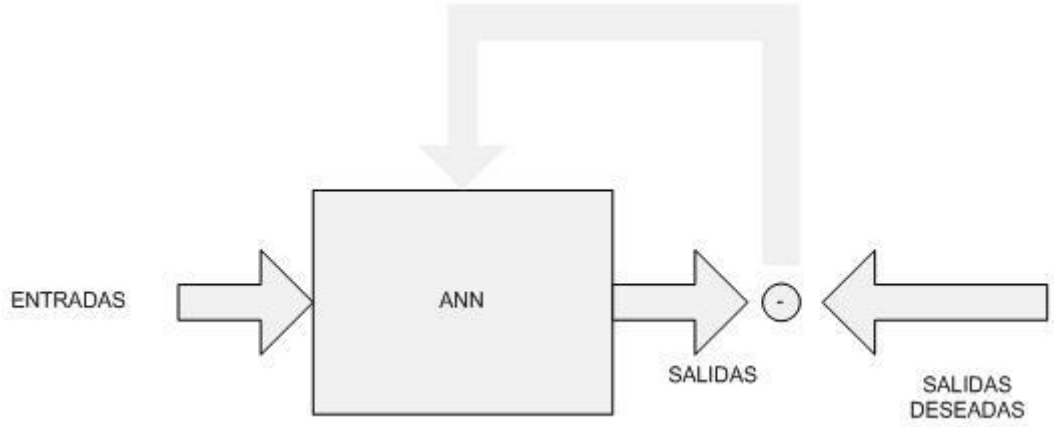


**Fig. 3** Aprendizaje Supervisado

### 1.3.3 Aprendizaje no supervisado en Redes Neuronales

En el aprendizaje no supervisado no existe información referente a las salidas de los patrones que vamos introduciendo, sólo sus entradas. Ha de ser la red, por lo tanto, la que vaya adecuando sus pesos en función de la información interna que vaya recogiendo de las entradas.

Este tipo de entrenamiento se utiliza principalmente para clasificar y diferenciar rasgos significativos de un conjunto de datos no clasificado a priori, dado que la red internamente intenta encontrar redundancias y rasgos significativos para agrupar los datos [7].

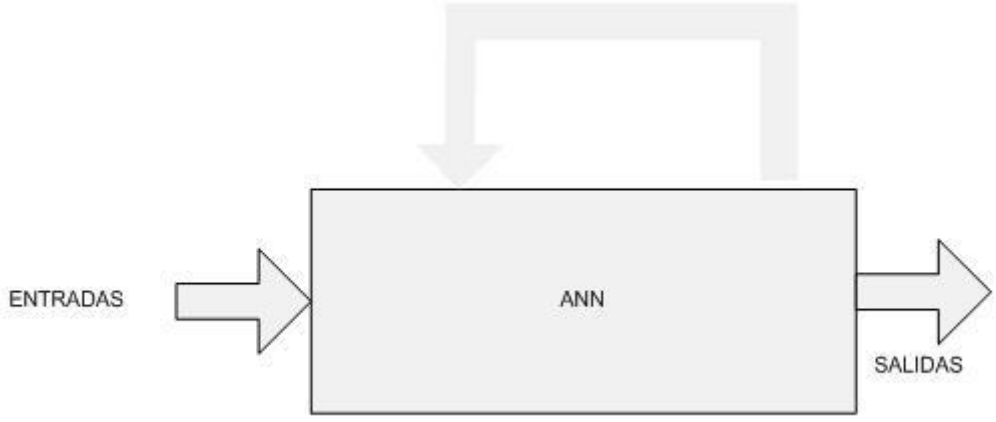


**Fig. 4** Aprendizaje no Supervisado

### 1.3.4 Backpropagation

El algoritmo *back-propagation* (o *entrenamiento hacia atrás*) es un sistema automático de entrenamiento de redes neuronales con *capas ocultas*, perfeccionado en la década de los 80.

En este tipo de redes, el problema a la hora de entrenarlas estriba en que solo conocemos la salida de la red y la entrada, de forma que no se pueden ajustar los pesos sinápticos asociados a las neuronas de las capas ocultas, ya que no se puede inferir a partir del estado de la capa de salida como tiene que ser el estado de las capas ocultas [8].

Las principales características del algoritmo son [9]:

- El algoritmo busca el mínimo de la función error a partir de un conjunto de patrones de entrenamiento
- Entrenar a consiste en modificar los pesos de la red.
- Los pesos se modifican hacia la dirección descendente de la función error.

El sistema de entrenamiento mediante back-propagation consiste en:

- Empezar con unos pesos sinápticos cualquiera (generalmente elegidos al azar).
- Introducir unos datos de entrada (en la capa de entradas) elegidos al azar entre los datos de entrada que se van a usar para el entrenamiento.
- Dejar que la red genere un vector de datos de salida (propagación hacia delante).
- Comparar la salida generada por la red con la salida deseada.
- La diferencia obtenida entre la salida generada y la deseada (denominada error) se usa para ajustar los pesos sinápticos de las neuronas de la capa de salidas.
- El error se propaga hacia atrás (back-propagation), hacia la capa de neuronas anterior, y se usa para ajustar los pesos sinápticos en esta capa.
- Se continua propagando el error hacia atrás y ajustando los pesos hasta que se alcance la capa de entradas [8].

Para conocer cómo el algoritmo Backpropagation determina el error cuadrático medio se presenta a continuación en la **Fig. 1.5** [9]:

- *Sean* $E = \text{Error total cometido}$
  $t_k = \text{Salida deseada}$
  $o_k = \text{Salida obtenida}$
- *El error total cometido será*

$$E = \frac{1}{2}\sum_{k}(e_k)^2 \quad \text{donde} \quad e_k = t_k - o_k$$

**Fig. 5** Definición del error con Backpropagation

### 1.3.5 Arquitectura de la red para un eficiente entrenamiento

Durante el entrenamiento del Perceptrón Multicapa es necesario tener en cuenta que para buscar la eficiencia en este proceso es necesario conocer cuál es el número de capas ocultas que deben utilizarse. Sobre esto hay quienes consideran que problemas que requieren dos capas ocultas son raramente encontrados, sin embargo, las redes neuronales con dos capas ocultas pueden representar funciones con cualquier tipo de forma. Actualmente no existe ninguna razón teórica para utilizar redes neuronales con más de dos capas ocultas. De hecho, para muchos problemas prácticos, no hay razón para usar nada más que una capa oculta.

Decidir el número de neuronas en las capas ocultas es una parte muy importante para la decisión general de la arquitectura de la red neuronal, a pesar de que estas capas no interactúan directamente con el ambiente externo, tienen una enorme influencia en el resultado final. Tanto el número de capas ocultas y el número de neuronas en cada una de estas capas ocultas debe considerarse cuidadosamente.

En general los métodos para determinar el número correcto de las neuronas para su utilización en las capas ocultas, son las siguientes [10]:

- El número de neuronas ocultas debe estar entre el tamaño de la capa de entrada y el tamaño de la capa de salida.
- El número de neuronas ocultas deben ser de 2 / 3 del tamaño de la capa de entrada, más el tamaño de la capa de salida.

- El número de neuronas ocultas debe ser inferior a dos veces el tamaño de la capa de entrada.

Estas tres normas proporcionan un punto de partida para que usted considere.

## Conclusiones

En este capítulo se ha presentado una breve introducción a las Redes Neuronales Artificiales, brindando inicialmente algunas definiciones de las mismas, dadas por autores reconocidos en el tema. También se hizo una presentación de los tipos de RNA más comunes en la actualidad, haciendo énfasis en el Perceptrón Multicapa por ser la red que se utilizará en el transcurso de la investigación. Durante el desarrollo del capítulo se realizó un profunda revisión sobre el entrenamiento de una RNA, específicamente en los puntos necesarios para resolver el problema científico de la investigación, como son: las funciones de transferencia, el tipo de aprendizaje que se utilizará durante la investigación (aprendizaje supervisado), además se brindan aspectos de interés sobre Backpropagation ya que es el algoritmo que se utilizará en el entrenamiento de la red. En el presente capítulo también se hizo un análisis sobre algunos puntos esenciales en la arquitectura de la red para lograr que el proceso de entrenamiento del Perceptrón Multicapa sea lo más eficiente posible.

## Capítulo 2. Programas y Metodologías

En este capítulo dará una breve panorámica de la herramienta que se utilizará en el desarrollo de toda la investigación, el Matlab, explicando sus principales características, además se explicará el procedimiento que se utilizó para el desarrollo de la investigación utilizando dicha herramienta.

Los primeros pasos de la investigación se dieron con el Weka, que es un conocido software de aprendizaje automático y minería de datos desarrollado en Java, esta variante fue apartada pues no brinda la posibilidad de variar las funciones de transferencia en el perceptrón multicapa durante el entrenamiento. Se decidió estudiar el Matlab ya que además de permitir la creación, entrenamiento y simulación de una RNA sí brinda la posibilidad de variar las funciones de transferencia por capas.

### 2.1 Matlab

MATLAB en abreviatura MATrix LABoratory (laboratorio de matrices), es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Este software es multiplataforma y propietario, entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

MATLAB es un programa de cálculo numérico orientado a matrices. Por tanto, será más eficiente si se diseñan los algoritmos en términos de matrices y vectores.[11]

Durante la investigación se utiliza específicamente la versión Matlab 7.04. Queda claro que esta herramienta brinda una serie de funciones ya implementadas para el proceso de creación, entrenamiento y simulación de una red neuronal, específicamente el Perceptrón Multicapa, además un programa escrito en MATLAB admite la mayoría de las estructuras de programación tales como las estructuras (if, for, while).

Para el caso de la creación de una red Perceptrón Multicapa con conexión hacia adelante y con el algoritmo de entrenamiento Backpropagation (retro-propagación del error), se utiliza la función *newff()*,

para entrenar esta red se usa la función *train*() y finalmente para simular la red la función utilizada es *sim().*

Matlab posee un grupo muy amplio de funciones de transferencia y de funciones de entrenamiento, brindando la posibilidad de seleccionar las adecuadas para cada caso de entrenamiento de una red neuronal.

### 2.2 Procedimiento para crear y entrenar El Perceptrón Multicapa con Matlab.

Resulta de mucha importancia antes de comenzar a explicar detalladamente el proceso de creación y entrenamiento de la red, es muy importante destacar que a pesar de las muchas ventajas que brinda el toolbox de redes neuronales de Matlab, este no permite durante la creación de la red pasarle a dicha red más de una función de transferencia, lo cual es necesario para el desarrollo de la investigación, debido a estas desventajas se decidió no utilizar el mismo, además como se muestra a continuación en la **Fig. 6**, el combobox de las funciones de transferencia solo brinda la opción de escoger entre tres funciones de transferencia y el objetivo es tener acceso a todas las funciones que Matlab posee.

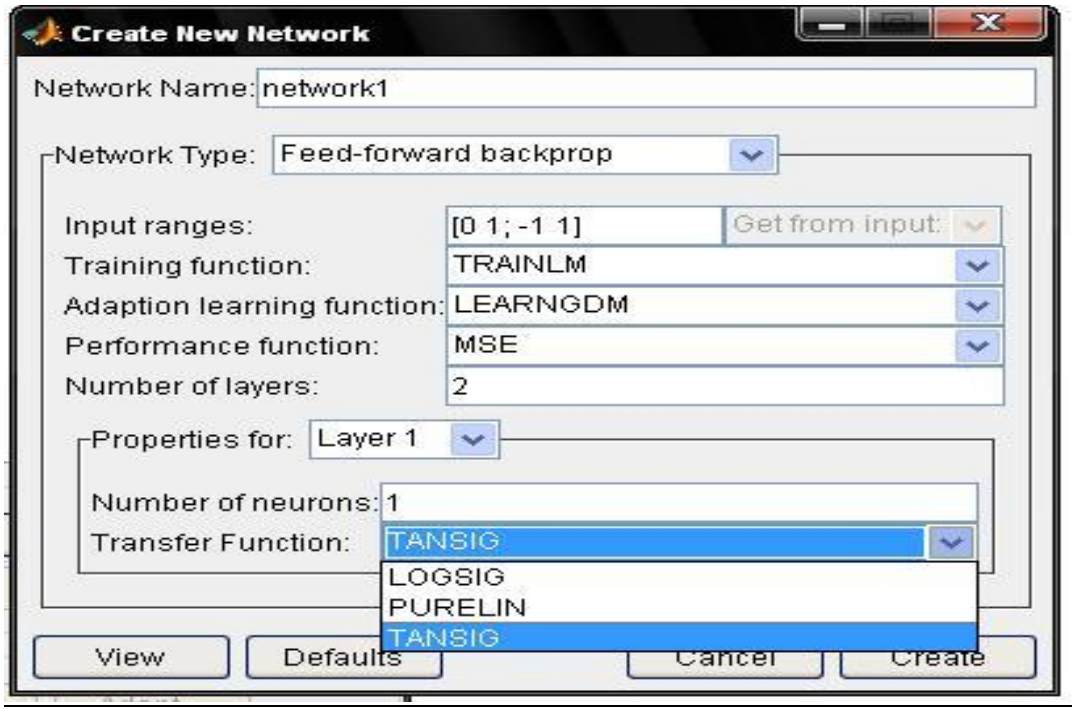


**Fig. 6** Toolbox de redes neuronales de Matlab

Ante las limitantes del Toolbox de Redes Neuronales de Matlab, es posible crear y entrenar una red con los requisitos necesarios de una manera sencilla.

El primer paso para crear un Perceptrón con las cualidades requeridas es crear un nuevo M-file Editor (ventana de implementación) y crear dentro del mismo una nueva función con el nombre deseado donde se implementará el Perceptrón Multicapa utilizando la función *newff*() predeterminada por Matlab, y luego se procede de la siguiente manera*:*

net = newff([SD1, SD2, …SDn], [NCE, NCO1, NCO2,1], {'FTE' 'FTC1' 'FTC2' 'FTS' }, 'FE');

SD: intervalo de las salidas deseadas

NCE: número de neuronas en la capa de entrada

NCO1: número de neuronas en la primera capa oculta

NCO2: número de neuronas en la segunda capa oculta

FTE : función de transferencia de la capa de entrada

FTC1: función de transferencia de la primera capa de oculta

FTC2: función de transferencia de la segunda capa de oculta

FTS : función de transferencia de la capa de salida

FE: función de aprendizaje

Para el entrenamiento se determinan algunos parámetros a decisión del investigador. A continuación se ejemplifica cómo definir los parámetros mencionados.

net.trainParam.show = 1000; // cantidad de muestras, cada mil iteraciones muestra un resultado.

net.trainParam.lr = 0.1; //coeficiente de aprendizaje.

net.trainParam.epochs = 10000; //número de iteraciones.

net.trainParam.goal = 1e-5; // se establece el mayor error que puede alcanzar el entrenamiento.

net = train(net, p, t); //se entrena la red *net*, con los valores de la matriz **p(entrada)** y **t(salida).**

Y = *sim*(net, pp); //se simula el aprendizaje de la red con la matriz *pp* que va a tener 8 vectores de prueba para obtener 8 resultados.

En el momento en que se crea el Perceptrón Multicapa es necesario pasarle tanto las funciones de transferencia como la función de entrenamiento deseada.

La función de transferencia de una capa se establece al crear la red o bien alterando el valor del parámetro NET.layers {i}.transferFcn en una red existente.

Después de creado el Perceptrón Multicapa se procede al entrenamiento del mismo usando la sentencia *train(red, entradas, salidas)* y a continuación se simula el entrenamiento usando la sentencia *sim(red,p)*, donde *p* van a ser los vectores con los que se desea probar la red. Seguidamente se

muestra un ejemplo de gráfica devuelta por Matlab en la cual se muestran todos los aspectos que se tienen en cuenta para cada entrenamiento entre los que se encuentran el *performance* (comportamiento) o error cuadrático medio de la red según los parámetros entrados, las cantidad de iteraciones o Epochs y el Goal o error, el cual en el entrenamiento se usa como punto de referencia o error a alcanzar.

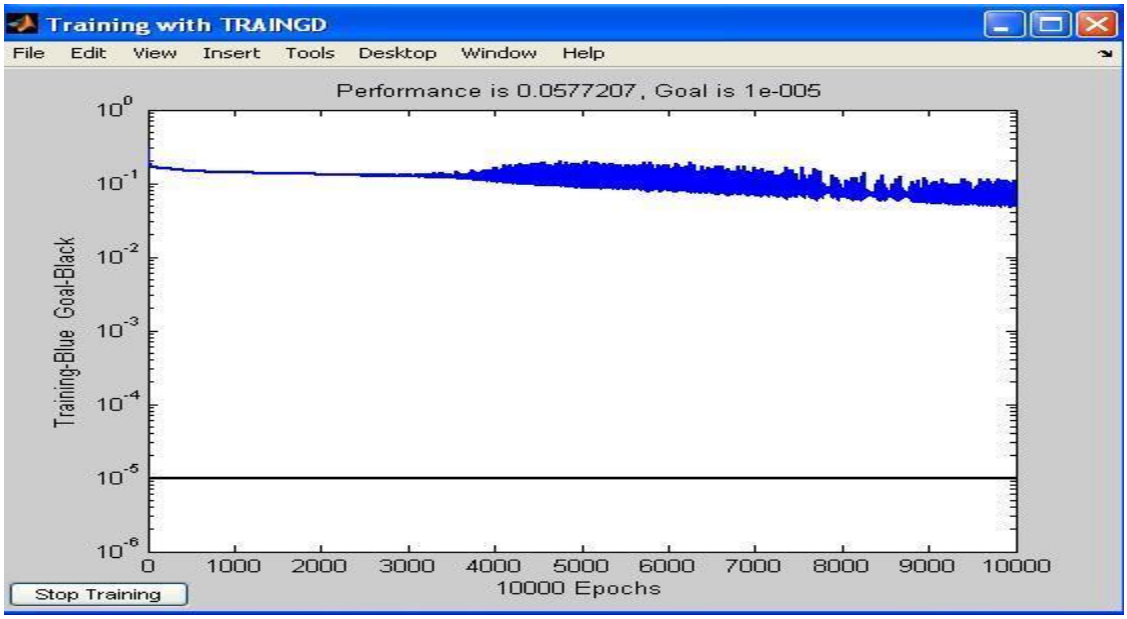


**Fig.1.7** Comportamiento del entrenamiento de la red.

Otro aspecto que es importante tener en cuenta para el entrenamiento de la red en Matlab es que existen dos tipos de entrenamiento, el estático para el cual se utiliza la sentencia *train* donde se dispone de todos los datos desde el principio; existen dos fases para el mismo:

- Entrenamiento de la red a partir de un conjunto de pares <entrada, salida>
- Uso de la red, aplicando unos datos de entrada para obtener una salida

También existe el adaptativo en el que se utiliza *adapt(),* donde los pesos se modifican a la vez que van llegando datos. Por tanto, el entrenamiento de la red se realiza a la vez que se usa. En el caso de esta investigación el entrenamiento es estático por lo que se utiliza el *train()* [12].

### 2.2.1 Diseño para el entrenamiento.

Para comenzar a realizar los experimentos fue necesario definir una forma organizada. Para todos los experimentos se utilizarán seis funciones de transferencia, tomadas de Matlab y agrupadas en dos grupos, las sigmoidales (*tansig, logsig)* y las lineales (*purelin, poslin, satlin, satlins*).

Los experimentos se dividirán en dos grupos específicos: experimentos cualitativos y cuantitativos, en ambos casos se harán pruebas con una y dos capas ocultas.

Inicialmente se comenzará colocando la misma función de transferencia en las capas de entrada y salida, comenzando siempre con la función *tansig.* Para el caso en que los experimentos se realizan con un Perceptrón Multicapa de dos capas ocultas, se comenzará variando la segunda capa oculta, manteniendo la primera fija y al concluir se invertirá este mismo proceso, o sea, se variarán las de la primera y se mantendrá fija la función transferencia de la segunda.

A continuación se tomará la función de transferencia *logsig* y se realizarán los mismos pasos que con la *tansig.*

Para observar si al variar las capas de entrada y salida ocurre algún cambio favorable para los resultados, se mezclarán las dos funciones sigmoidales de la siguiente manera: primero la *tansig* en la capa de entrada y la *logsig* en la de salida y después a la inversa, variando siempre la capa o las capas ocultas.

## Capítulo 3. Resultados y discusión

En el presente capítulo se mostrarán dos tipos de resultados para el entrenamiento del Perceptrón Multicapa con una y dos capas ocultas, éstos resultados son *cualitativos y cuantitativos*.

Para llevar a cabo esta investigación se tomaron sesenta muestras de una sustancia para E. Cloacae y otras sesenta para S. Aureus con el objetivo de predecir la actividad biológica de la sustancia según los descriptores de la misma. Aunque el entrenamiento del Perceptrón Multicapa se realiza con una matriz de 9X60 descriptores, solo se toman los primeros 8 vectores para la simulación de la red, o sea

se tendrán 8 salidas deseadas y saldrán 8 valores después de realizada la simulación del entrenamiento de la red .

Entre las salidas deseadas y las que realmente devuelve la red se realiza una comparación que genera un error.

La mayoría de los experimentos se realizan con el nivel de actividad biológica de la bacteria S. Aureus, aunque antes de finalizar con todos los experimentos se hacen un grupo pequeño de pruebas con el de la E. Cloacae para confirmar los resultados obtenidos con la otra bacteria.

Para mostrar la arquitectura, el error cuadrático medio (MSE) y las salidas de cada red para cada uno de los experimentos se utiliza una tabla. Los experimentos están divididos en grupos según las combinaciones que se definen por el entrenador de la red. Las tablas van a contar con un máximo de seis experimentos y con un mínimo de cuatro.

Con el objetivo de poder observar el comportamiento visual de una red durante su entrenamiento, se muestra debajo de cada tabla una figura con el comportamiento de la red con la que se obtuvo el menor error en cada grupo de experimentos.

### 3.1 Resultados cualitativos para S.Aureus

El entrenamiento del Perceptrón Multicapa siempre devuelve valores en correspondencia con la función de transferencia de la capa de salida, aunque generalmente los valores deben ser entre 0 y 1 puede darse el caso de que se obtenga un valor negativo cuando se usa la función *tansig* en alguna de las capas de la red, ya que esta puede devolver valores entre -1 y 1. En dependencia de los resultados, estas sustancias se clasifican en activas e inactivas, si los valores son menores que 0.5 se consideran inactivas y si están entre 0.5 y 1 se consideran activas.

Para determinar cuál o cuáles funciones resultan más eficientes durante el entrenamiento en cuestión es importante hacer distintas combinaciones de funciones transferencia.

Tanto en este epígrafe como en el siguiente se sigue el mismo orden para las combinaciones de las funciones de transferencia.

En cada una de las tablas se mostrará la combinación de funciones utilizada y el MSE (error cuadrático medio).

### 3.1.1 Experimentos en un Perceptrón Multicapa con una sola capa oculta.

#### 3.1.1.1 Experimentos con la función de transferencia tansig en las capas de entrada y salida, y variando la función de la capa oculta.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | tansig | tansig | tansig | 0.0744 |
| 2 | tansig | logsig | tansig | 0.1038 |
| 3 | tansig | poslin | tansig | 0.0967 |
| 4 | tansig | purelin | tansig | 0. 1064 |
| 5 | tansig | satlin | tansig | 0.0856 |
| 6 | tansig | satlins | tansig | 0.0758 |

**Tabla 2** Resultados Experimentales 1.

Como se observa en la tabla 2, para los primeros seis experimentos realizados el menor error se obtuvo en el primero, donde la red está compuesta solamente por la función de transferencia *tansig.*

A continuación se muestra el comportamiento de la red compuesta por la función tansig en las tres capas del Perceptrón Multicapa:

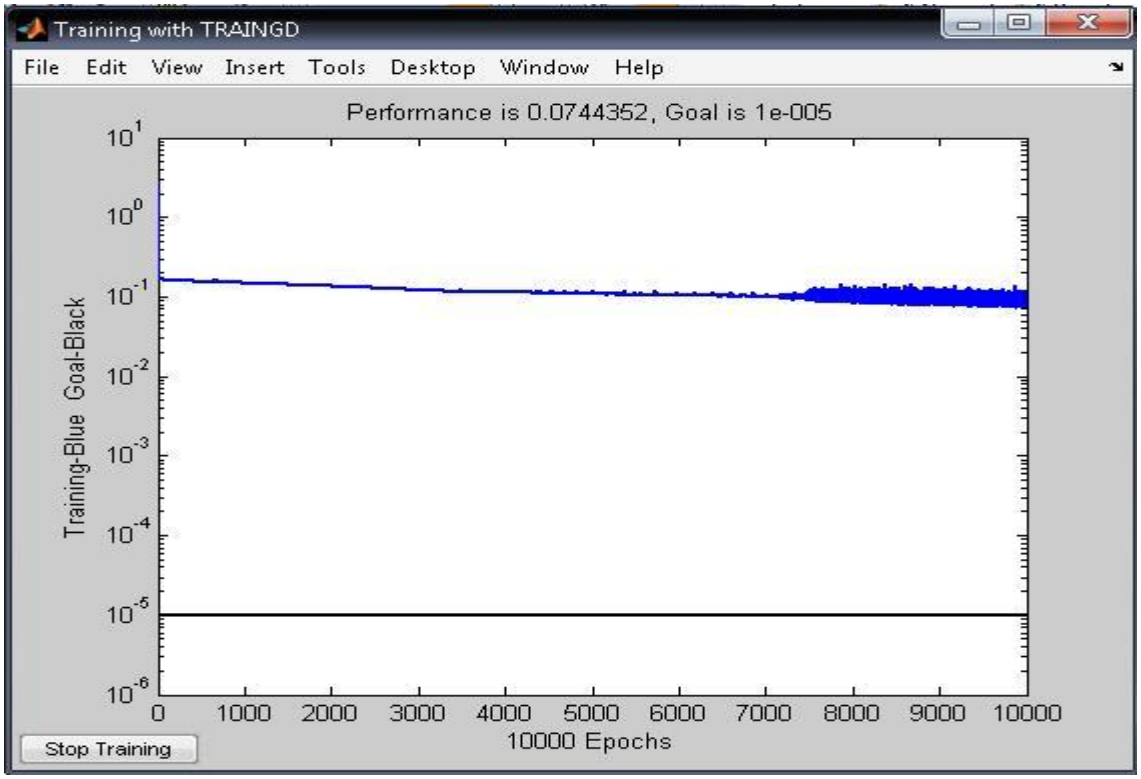


**Fig. 7** Comportamiento del experimento 1 - Resultados experimentales 1.

### 3.1.1.2 Experimentos con la función de transferencia *logsig* en las capas de entrada y salida, y variando la función de la capa oculta.

A diferencia de la tabla anterior, en la presente tabla la función de transferencia de las capas de entrada y salida es la *logsig*, con el objetivo de provocar una disminución del error, usando las mismas funciones de transferencia en la capa oculta de la red.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | logsig | logsig | logsig | 0.1391 |
| 2 | logsig | tansig | logsig | 0.1390 |
| 3 | logsig | poslin | logsig | 0.1394 |
| 4 | logsig | purelin | logsig | 0.1397 |
| 5 | logsig | satlin | logsig | 0.1379 |
| 6 | logsig | satlins | logsig | 0. 1396 |

**Tabla 3** Resultados Experimentales 2.

A diferencia de los experimentos en los que se utilizó la función *tansig* en las capas de entrada y salida, en éstos, el cambio de dicha función provocó un aumento considerable del porciento de error, por ejemplo, anteriormente con la función *tansig* en la capa oculta se obtuvo el mejor resultado, y ahora la función *satlin* en la capa oculta es la que menor error de aprendizaje tiene.

Aunque ninguno de los errores obtenidos es pequeño, es importante ver cómo influye la función *satlin* en el comportamiento de la red.

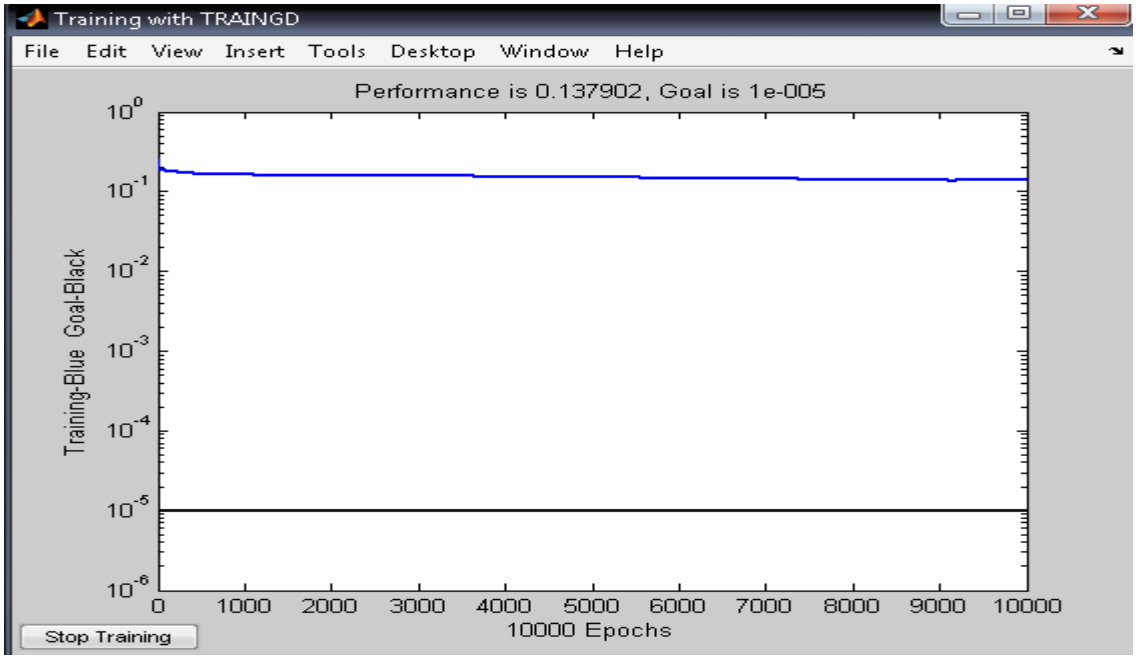


**Fig. 8** Comportamiento del experimento 5 - Resultados experimentales 2.

### 3.1.1.3 Experimentos con la función de transferencia *logsig* en las capa de entrada, la *tansig* en la de salida, y variando la función de la capa oculta.

En los experimentos realizados anteriormente la función de la capa de entrada era la misma que se usaba en la capa de salida, pero como se puede ver en la tabla 4 que se muestra a continuación, ese esquema se rompe y se comienza a variar además de la función de la capa oculta, las funciones de la capa de entrada y la de salida.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | logsig | tansig | tansig | 0.1315 |
| 2 | logsig | logsig | tansig | 0. 1208 |
| 3 | logsig | purelin | tansig | 0. 1268 |
| 4 | logsig | satlin | tansig | 0.1060 |
| 5 | logsig | poslin | tansig | 0.1364 |
| 6 | logsig | satlins | tansig | 0.1364 |

**Tabla 4** Resultados Experimentales 3.

En la presente tabla se observa con facilidad que todas las combinaciones de funciones devuelven errores muchos mayores que los obtenidos en los seis experimentos iniciales, pero a diferencia de los experimentos en los que se utilizó la función *logsig* en las capas de entrada y salida, en éstos el menor error se obtuvo con la función *satlin.*

El comportamiento visual de la red que menor error arrojó es el siguiente:

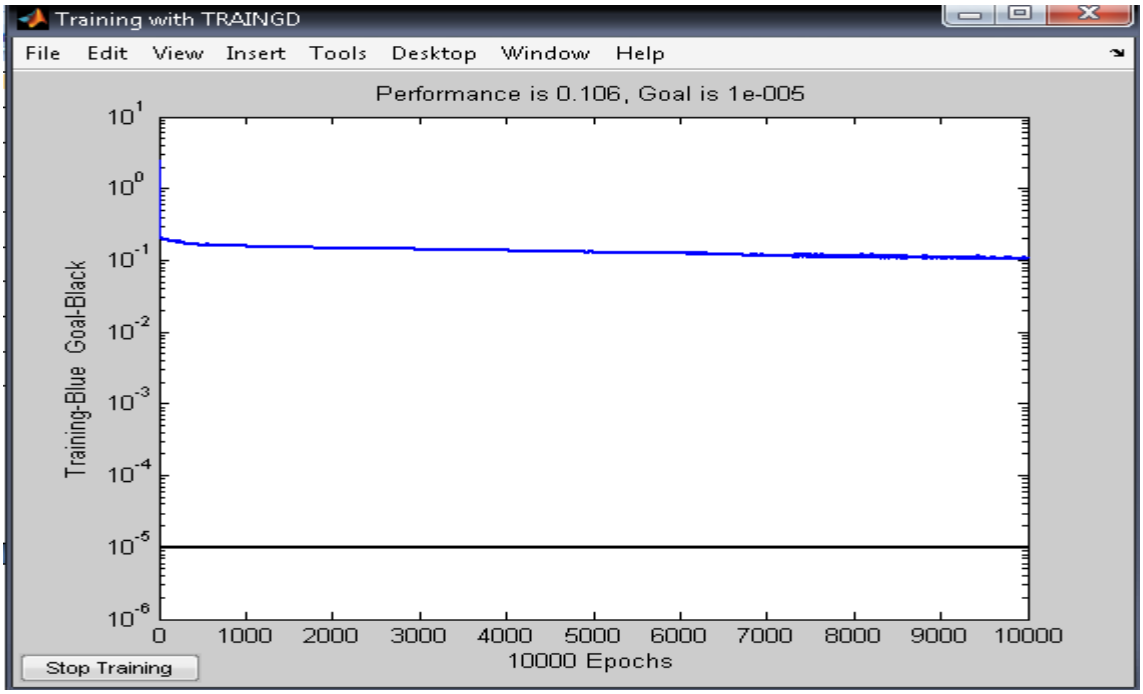


**Fig. 9** Comportamiento del experimento 4 - Resultados experimentales 3.

### 3.1.1.4 Experimentos con la función de transferencia *tansig* en las capa de entrada, la *logsig* en la de salida, y variando la función de la capa oculta.

Hasta el momento se ha observado que la función de transferencia *tansig* es mucho más eficiente que la *logsig* en la capa de entrada, por lo que nuevamente pasa a estar en dicha capa y la *logsig* pasa a la de salida, o sea, se invierte el orden de las funciones de los seis experimentos que anteceden a los que se muestran en la siguiente tabla:

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | tansig | tansig | logsig | 0.0931 |
| 2 | tansig | logsig | logsig | 0.1073 |
| 3 | tansig | purelin | logsig | 0.0932 |
| 4 | tansig | poslin | logsig | 0.1132 |
| 5 | tansig | satlin | logsig | 0.1310 |
| 6 | tansig | satlins | logsig | 0.0941 |

**Tabla 5** Resultados Experimentales 4.

En los valores mostrados en esta tabla, se observa que al colocar nuevamente la función *tansig* en la capa de entrada las salidas obtenidas son más cercanas a las deseadas, pero a pesar de la mejoría en los resultados, el menor error obtenido en estos experimentos no es inferior al obtenido en los primeros seis experimentos, o sea, el hecho de que aún se obtengan resultados poco convenientes viene dado por el uso de la función de transferencia *logsig* en la capa de salida.

En la siguiente figura se observa el comportamiento de la mejor combinación de funciones:

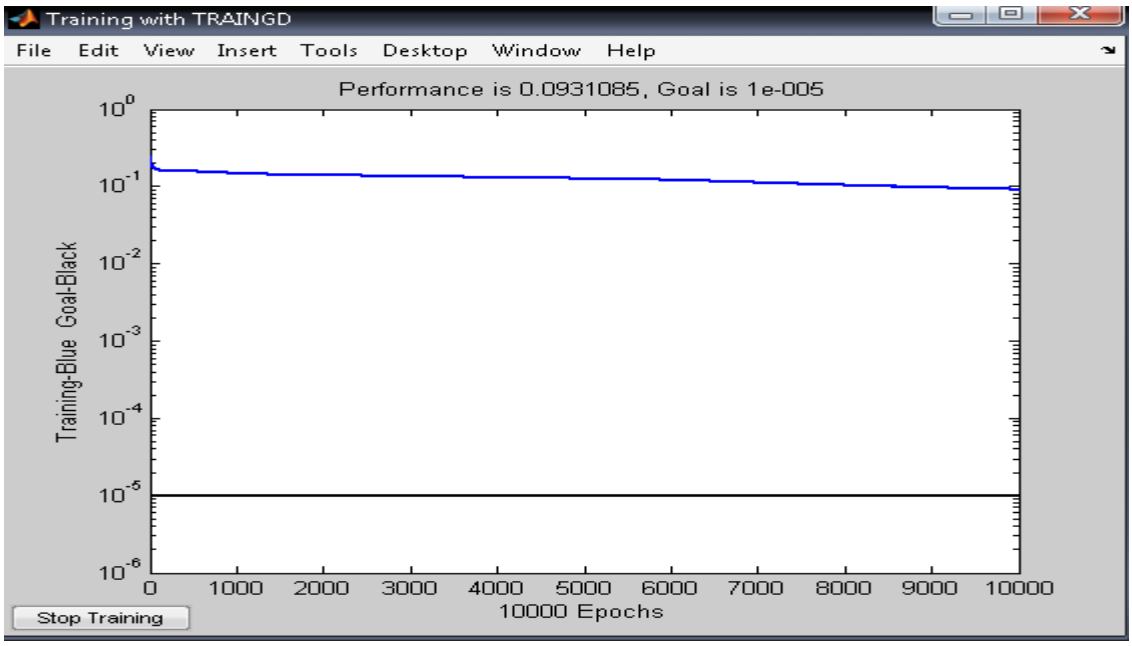


**Fig. 10** Comportamiento del experimento 3 - Resultados experimentales 4.

### 3.1.1.5 Experimentos con las funciones lineales en las capas de entrada y salida, y la *tansig* en la capa oculta.

Teniendo en cuenta que en algunos de los experimentos realizados anteriormente se ha observado que las funciones lineales provoca una disminución del error cuando son utilizadas en la capa oculta se decidió probar su efecto cuando se colocaban en las capas de entrada y salida. Para los presentes experimentos no hay ninguna variación en cuanto a la función de transferencia usada en la capa de entrada y la de salida en una misma red, sino que siempre se mantiene fija en ambas capas, ya que el objetivo principal de este grupo de experimentos es analizar qué sucede en estos casos, pero en cada uno de los experimentos la función de transferencia de las capas de entrada y salida es diferente y la función de la capa oculta no varía, es por ello que solo se muestran en la tabla resultados de cuatro experimentos .

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | poslin | tansig | poslin | 0.1012 |
| 2 | purelin | tansig | purelin | 0.1247 |
| 3 | satlin | tansig | satlin | 0.1209 |
| 4 | satlins | tansig | satlins | 0.0991 |

**Tabla 6** Resultados Experimentales 5.

Ninguno de los errores obtenidos es significativo a pesar de que la función *satlins* devolvió un error más pequeño que con el resto de las funciones, no obstante se muestra el comportamiento con dicha función de transferencia.

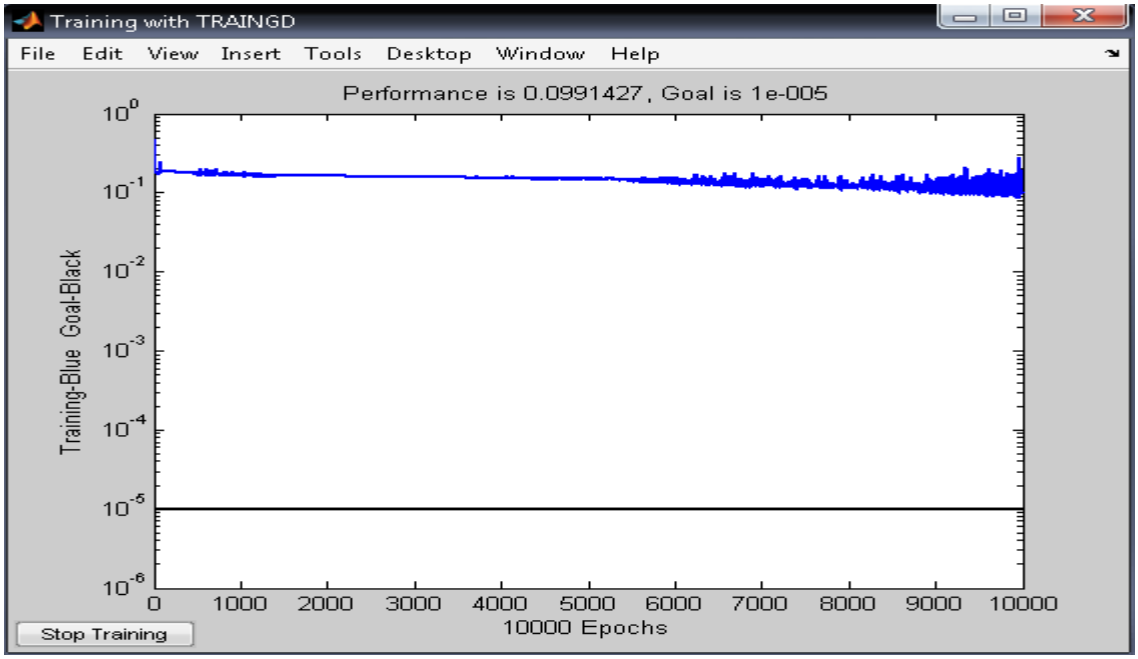


**Fig. 11** Comportamiento del experimento 4 - Resultados experimentales 5.

### 3.1.2 Experimentos en un Perceptrón Multicapa con dos capas ocultas.

#### 3.1.2.1 Experimentos con la función de transferencia *tansig* en las capas de entrada y salida y en la primer a capa oculta, y variando la segunda capa oculta.

En esta sección se analizarán los resultados de los experimentos realizados con 2 capas ocultas. Para lograr una uniformidad se decidió hacer lo mismo que en el 1er experimento cualitativa con 1 sola capa oculta, o sea, tomar como punto de partida el uso la función de transferencia tansig en todas las capas

de la red, y en el resto de los experimentos mostrados se hace una variación en la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | tansig | tansig | 0.0577 |
| 2 | tansig | tansig | logsig | tansig | 0.0698 |
| 3 | tansig | tansig | satlin | tansig | 0.1351 |
| 4 | tansig | tansig | purelin | tansig | 0.0567 |
| 5 | tansig | tansig | poslin | tansig | 0.0537 |
| 6 | tansig | tansig | satlins | tansig | 0.1047 |

**Tabla 7** Resultados Experimentales 6.

Los resultados mostrados por esta tabla permiten darse cuenta rápidamente que el error cuadrático medio tiene una disminución significativa al incluir otra capa oculta en la red, o sea, que a diferencia de los experimentos cualitativos con una capa oculta, ahora se han obtenido varios errores con un porciento de error más pequeño, como ejemplos tenemos los experimentos 1, 4 y 5 los cuales utilizan en la segunda capa oculta las funciones *tansig*, *purelin* y *poslin* respectivamente.

A continuación se muestra visualmente el comportamiento de la red con la función de transferencia *poslin* .

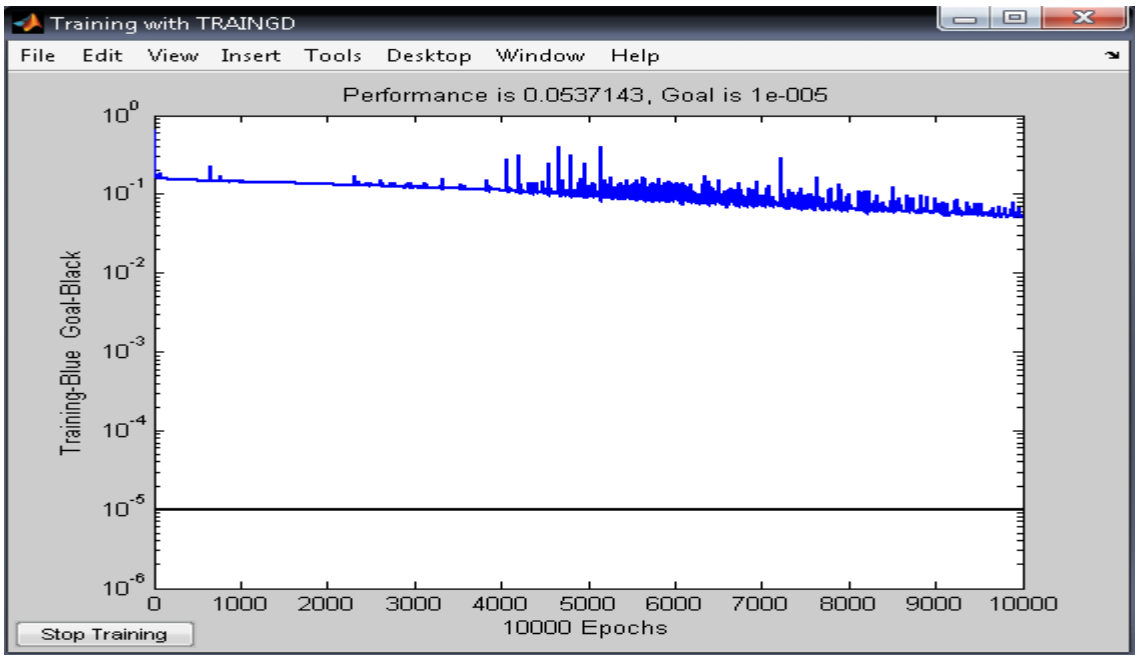


**Fig. 12** Comportamiento del experimento 5 - Resultados experimentales 6.

### 3.1.2.2 Experimentos con la función de transferencia tansig en las capas de entrada y salida y la segunda capa oculta, y variando la primera capa oculta.

Anteriormente se observó una mejoría de los resultados obtenidos utilizando dos capas ocultas, por lo que el momento sugiere ver si variando la primera capa se pueden obtener mejores resultados.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | poslin | tansig | tansig | 0.0929 |
| 2 | tansig | purelin | tansig | tansig | 0.0692 |
| 3 | tansig | satlins | tansig | tansig | 0.1253 |
| 4 | tansig | logsig | tansig | tansig | 0.1063 |
| 5 | tansig | satlin | tansig | tansig | 0.1165 |

**Tabla 8** Resultados Experimentales 7.

A diferencia de los experimentos en los que se varió la función de transferencia en la segunda capa oculta, ahora los resultados variando la primera, no son semejantes. Si se observan y se comparan los errores de cada uno de los experimentos con los de la tabla anterior a ésta, es fácil darse cuenta que son mayores. A continuación se muestra la gráfica del comportamiento de la red en el experimento 2.

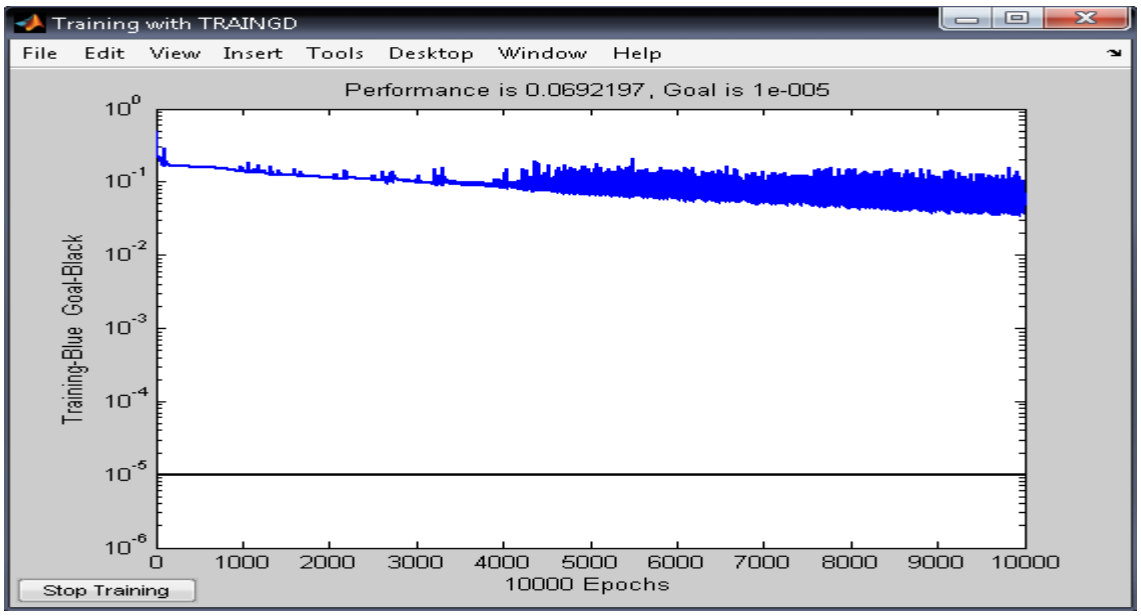


**Fig. 13** Comportamiento del experimento 2 - Resultados experimentales 7.

### 3.1.2.3 Experimentos con la función de transferencia *tansig* en las capas de entrada y salida y la *logsig* en la primera capa oculta, y variando la segunda capa oculta.

Para los presentes experimentos, como se observa en la tabla 9 que está a continuación, el primer experimento tiene en ambas capas ocultas la función de transferencia *logsig*, o sea, que la red del primer experimento no está compuesta por ninguna función lineal, por lo que servirá como punto de análisis al comparar sus resultados con las combinaciones de los demás experimentos.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | logsig | logsig | tansig | 0.0921 |
| 2 | tansig | logsig | poslin | tansig | 0.0762 |
| 3 | tansig | logsig | purelin | tansig | 0.0783 |
| 4 | tansig | logsig | satlin | tansig | 0.0865 |
| 5 | tansig | logsig | satlins | tansig | 0.0921 |

**Tabla 9** Resultados Experimentales 8.

De los errores mostrados en esta tabla el único que llama la atención es el del segundo experimento, en el cual está colocada la función *poslin* en la segunda capa oculta.

Nuevamente la función *poslin* provocó una disminución del MSE. En la figura que se muestra a continuación se observa el comportamiento del segundo experimento.

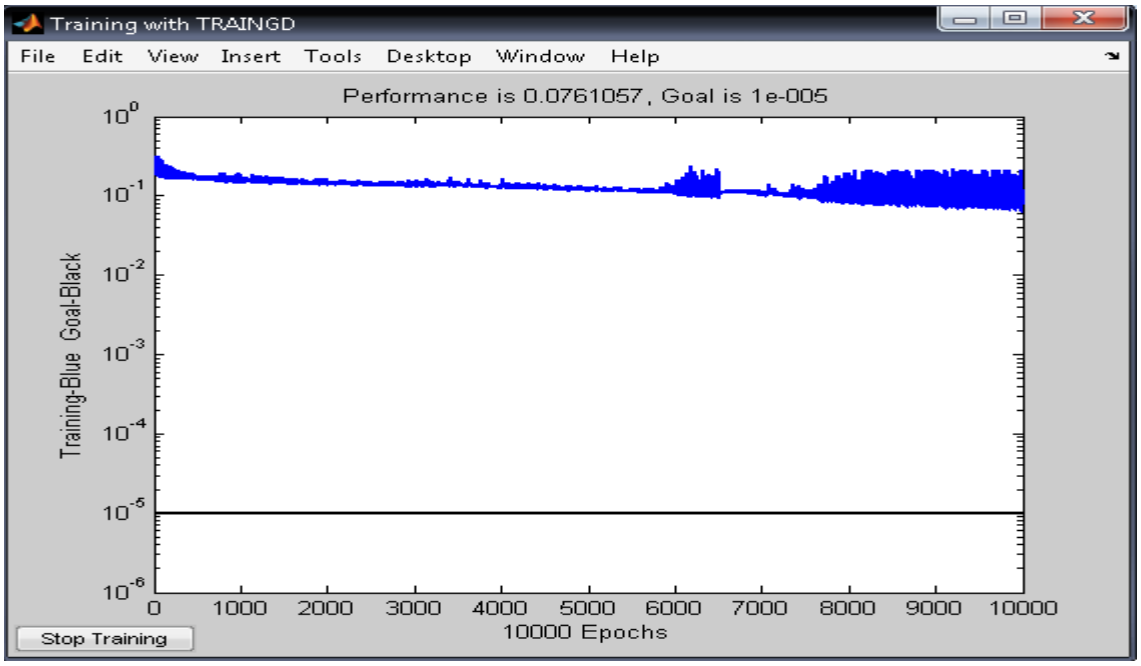


**Fig. 14** Comportamiento del experimento 2 - Resultados experimentales 8.

### 3.1.2.4 Experimentos con la función de transferencia tansig en las capas de entrada y salida y la logsig en la primera capa oculta, y variando la segunda capa oculta.

Debido a que en la tabla anterior se muestra el comportamiento de la red con la función *logsig* en ambas capas ocultas, en la siguiente tabla solo habrá cuatro experimentos, o sea, cada función lineal en la primera capa oculta combinándose con la *logsig* de la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | poslin | logsig | tansig | 0.1252 |

| | | | | | |
|---|---|---|---|---|---|
| 2 | tansig | purelin | logsig | tansig | 0.0739 |
| 3 | tansig | satlin | logsig | tansig | 0.0729 |
| 4 | tansig | satlins | logsig | tansig | 0.0738 |

**Tabla 10** Resultados Experimentales 9.

Como se observa en los resultados mostrados por la tabla 10, aunque los errores no son muy pequeños, todos son muy parecidos a excepción del primer experimento, el cual el error fue un poco desproporcionado.

A continuación se presenta la figura con el comportamiento de la mejor combinación de funciones de transferencia de los experimentos realizados.

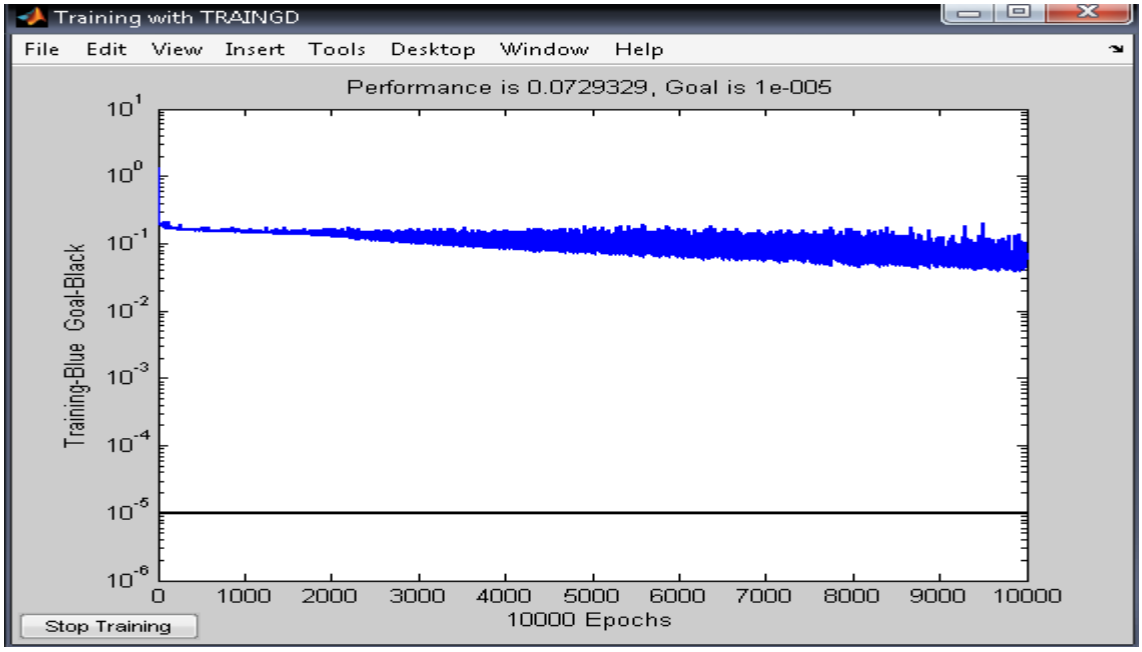


**Fig. 15** Comportamiento del experimento 3- Resultados experimentales 9.

### 3.1.2.5 Experimentos con la función *tansig* en la capa de entrada y la primera capa oculta, la función *logsig* en la capa de salida, y variando la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | tansig | logsig | 0.0704 |

| | | | | | |
|---|---|---|---|---|---|
| 2 | tansig | tansig | logsig | logsig | 0.0807 |
| 3 | tansig | tansig | poslin | logsig | 0.0495 |
| 4 | tansig | tansig | purelin | logsig | 0.0870 |
| 5 | tansig | tansig | satlin | logsig | 0.0497 |
| 6 | tansig | tansig | satlins | logsig | 0.0753 |

**Tabla 11** Resultados Experimentales 10.

De los resultados mostrados en la presente tabla lo más significativo son los errores devueltos por las redes compuestas por la función *poslin* y *satlin*, debido a que marcan una diferencia en el porciento de error respecto a las demás redes .

A continuación se presenta el comportamiento de la red compuesta por la función de transferencia *poslin*.

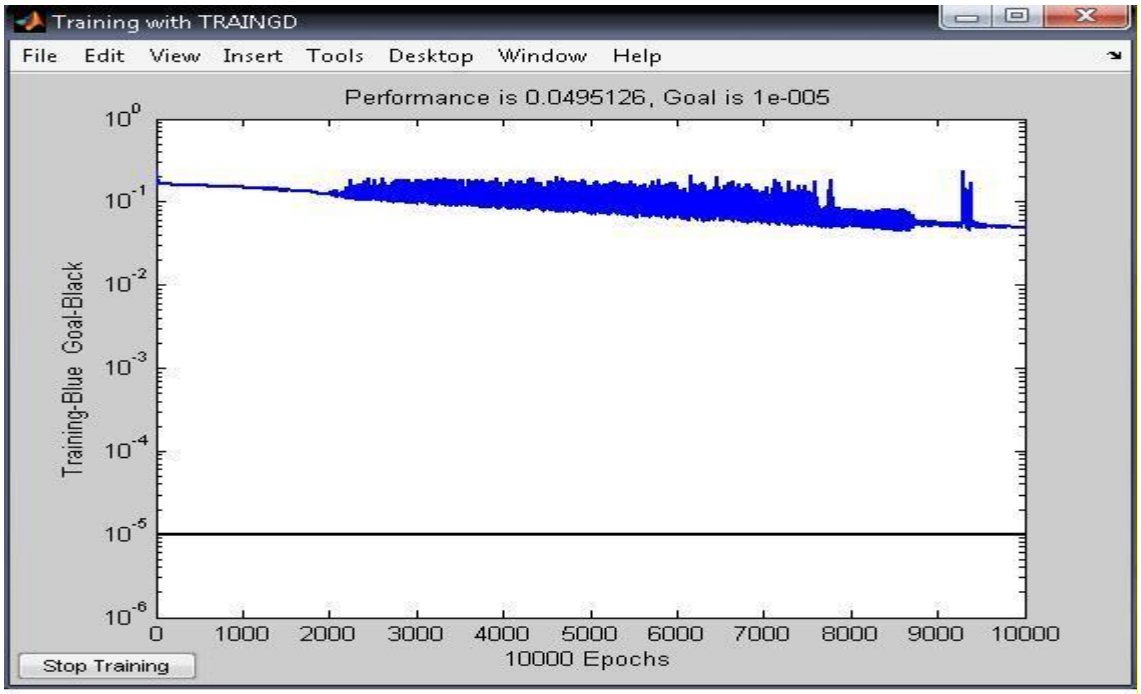


**Fig. 16** Comportamiento del experimento 5- Resultados experimentales 10.

### 3.1.2.6 Experimentos con la función *tansig* en la capa de entrada y la segunda capa oculta, la función *logsig* en la capa de salida, y variando la función de la primera capa oculta capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | logsig | tansig | logsig | 0.0808 |
| 2 | tansig | poslin | tansig | logsig | 0.0991 |
| 3 | tansig | purelin | tansig | logsig | 0.1081 |
| 4 | tansig | satlin | tansig | logsig | 0.0919 |
| 5 | tansig | satlins | tansig | logsig | 0.1011 |

**Tabla 12** Resultados Experimentales 11.

En los resultados mostrados se observa que ninguna de las combinaciones de funciones de transferencia realizadas devuelve un error pequeño.

A continuación se muestra la figura que corresponde al comportamiento de la red que menor error devolvió, o sea, el del primer experimento mostrado en la tabla.

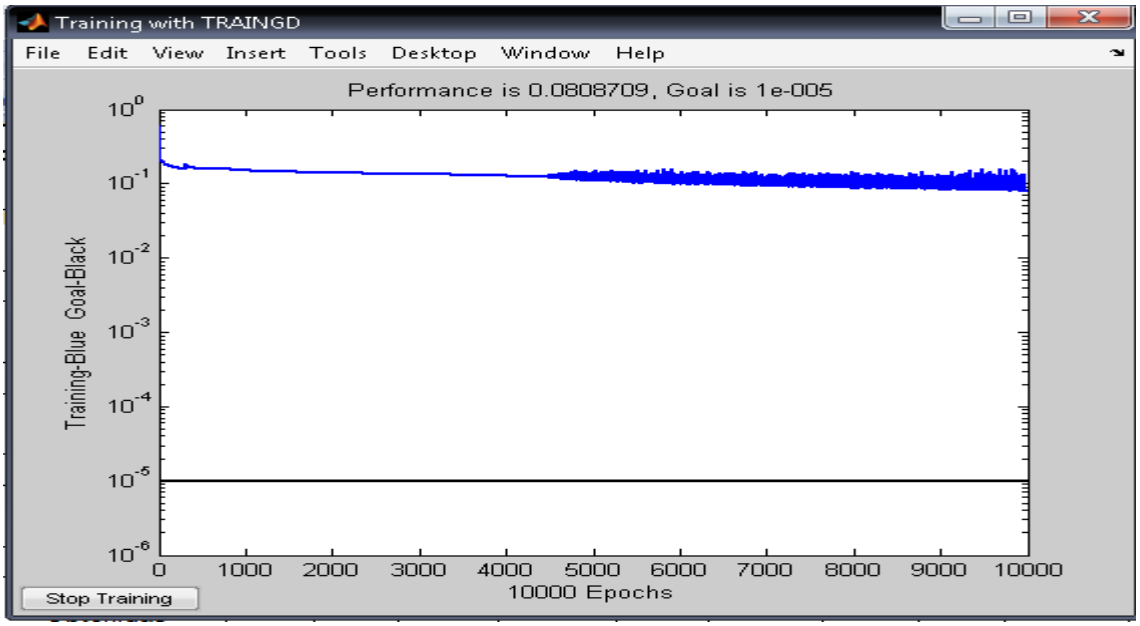


**Fig. 17** Comportamiento del experimento 1 - Resultados experimentales 11.

### 3.1.2.7 Experimentos con la función *logsig* en la capa de entrada, la función *tansig* en la capa de salida y en la primera capa oculta, y variando la función de la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | logsig | tansig | tansig | tansig | 0.0901 |
| 2 | logsig | tansig | logsig | tansig | 0.1231 |
| 3 | logsig | tansig | poslin | tansig | 0.0661 |
| 4 | logsig | tansig | purelin | tansig | 0.1331 |
| 5 | logsig | tansig | satlin | tansig | 0.1240 |
| 6 | logsig | tansig | satlins | tansig | 0.0868 |

**Tabla 13** Resultados Experimentales 12.

En la tabla que se muestra que los errores devueltos por la red no son los mejores, no obstante el experimento 3 devolvió un error bastante aceptable. A continuación se observa el comportamiento de este experimento.

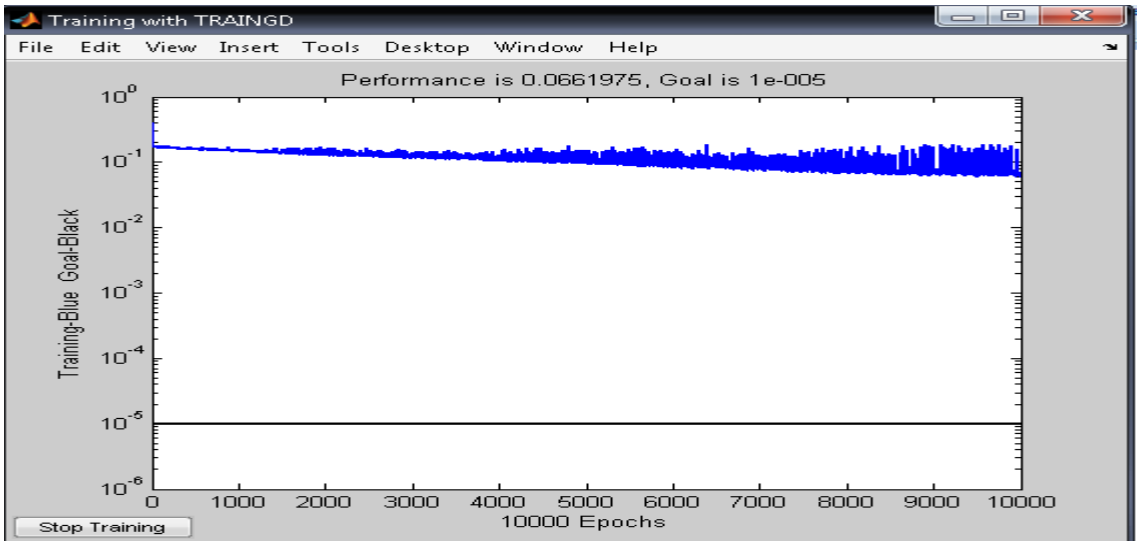


**Fig1.18** Comportamiento del experimento 3 - Resultados experimentales 12.

### 3.1.2.8 Experimentos con la función *logsig* en la capa de entrada, la función *tansig* en la capa de salida y en la segunda capa oculta, y variando la función de la primera capa oculta.

A pesar de que en los experimentos con una sola capa oculta poner la función *logsig* en la capa de entrada no arrojó buenos resultados, para los siguientes experimentos se vuelve a colocar dicha función en la primera capa con el objetivo de realizar todos los experimentos planificados.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | logsig | logsig | tansig | tansig | 0.1031 |
| 2 | logsig | poslin | tansig | tansig | 0.0992 |
| 3 | logsig | purelin | tansig | tansig | 0.1140 |
| 4 | logsig | satlin | tansig | tansig | 0.1133 |
| 5 | logsig | satlins | tansig | tansig | 0.1307 |

**Tabla 14** Resultados Experimentales 13.

Los resultados mostrados en la tabla demuestran que al igual que en un Perceptrón Multicapa de una sola capa oculta, los resultados en cada experimento no cumplen a plenitud con lo deseado.

El comportamiento visual del segundo experimento es el siguiente:

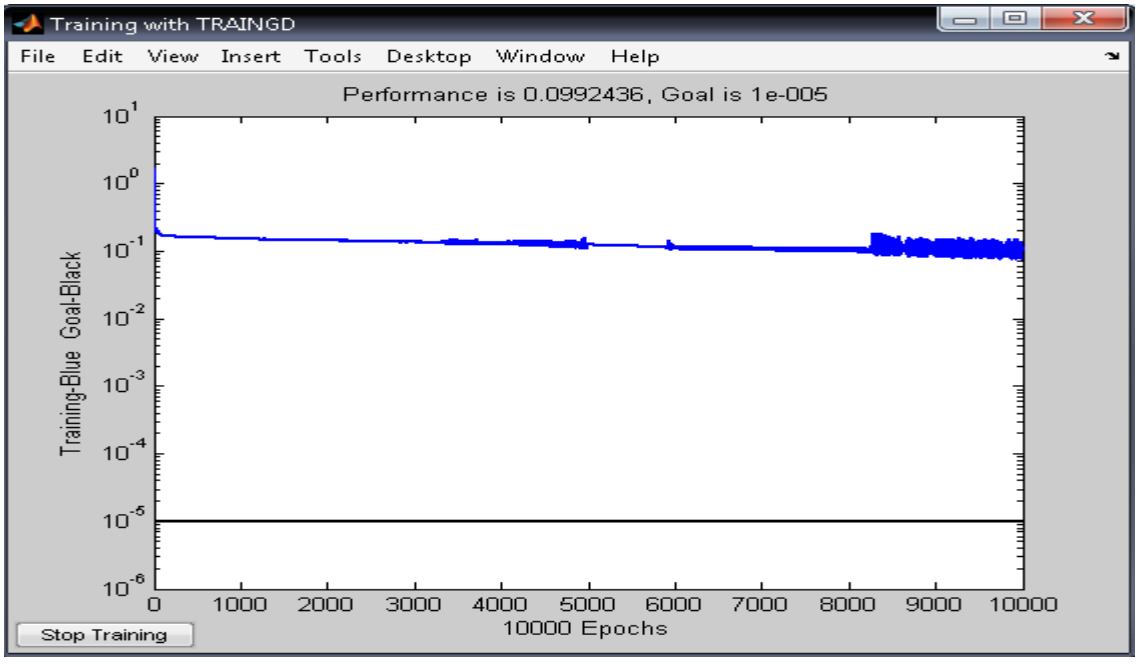

**Fig. 18** Comportamiento del experimento 2 - Resultados experimentales 13.

### 3.2 Resultados cuantitativos para S.Aureus

Después de haber realizado un estudio cualitativo sobre las muestras, se continuará con el análisis cuantitativo. Para obtener este tipo de resultados es necesario normalizar, tanto la matriz de entrada a la red como la de las salidas deseadas, utilizando una función normaliza(), la cual se tuvo que implementar para este tipo de experimentos, debido a que el Perceptrón Multicapa trabaja generalmente con valores entre -1 y 1; además para poder comparar las salidas deseadas con las obtenidas se desnormaliza las matriz devuelta por la red, porque estos valores van a estar en el rango de -1 a 1, utilizando una función desnormaliza(), también previamente implementada para estos experimentos. En estos experimentos no se compararán los valores deseados con los obtenidos ya que lo más importante es la comparación de los errores entre los distintos experimentos.

#### 3.2.1 Experimentos Cuantitativos con una capa oculta.

##### 3.2.1.1 Experimentos con la función *tansig* en la capa de entrada y salida y en la segunda capa oculta de la red.

Al igual que en los resultados cualitativos con una sola capa oculta en los que se encuentra la función *tansig* en las capas de entrada y salida, la primera función que se comprueba en la capa oculta es la misma función *tansig*, la cual sirve de punto de partida para llevar a cabo un análisis en los otros cinco experimentos que le siguen.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | tansig | tansig | tansig | 0.0606 |
| 2 | tansig | logsig | tansig | 0.0640 |
| 3 | tansig | poslin | tansig | 0.0577 |
| 4 | tansig | purelin | tansig | 0.0648 |
| 5 | tansig | satlin | tansig | 0.0560 |
| 6 | tansig | satlins | tansig | 0.0589 |

**Tabla 15** Resultados Experimentales 14.

En la presente tabla se muestran los resultados de los primeros seis experimentos cuantitativos con una sola capa oculta. Como se puede observar hay varias funciones de transferencia lineales que colocadas en la capa oculta, mejoran los resultados de la red. En cuanto a lo mencionado, la función de transferencia más significativa es la *satlin.*

A continuación se muestra visualmente el comportamiento de la red del segundo experimento presentado en la tabla 15.

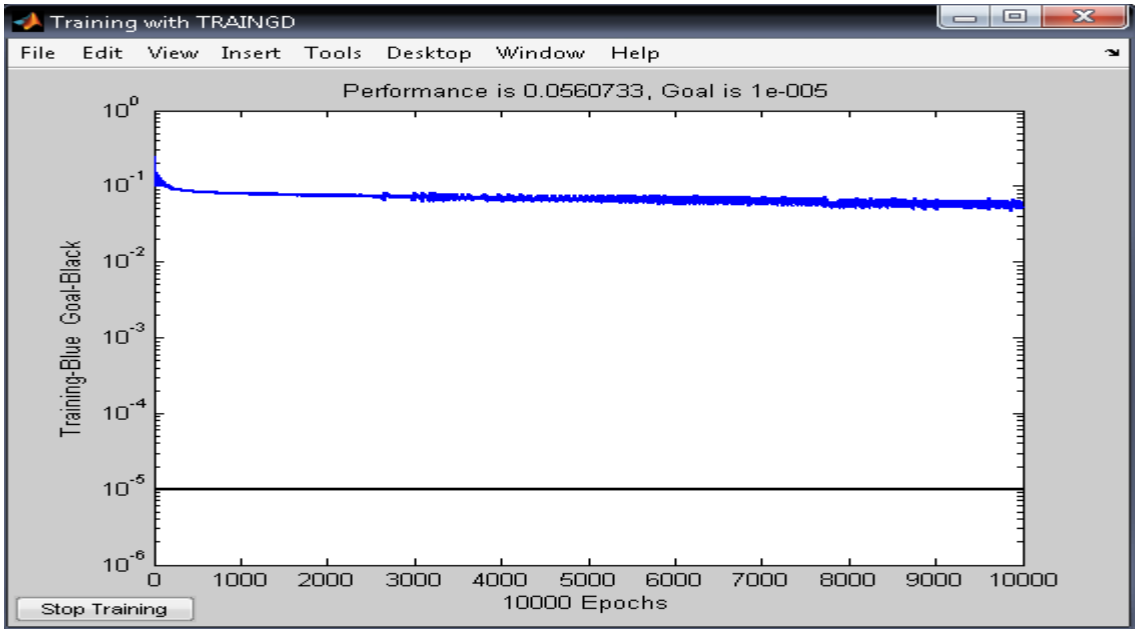


**Fig. 19** Comportamiento del experimento 5 - Resultados experimentales 14.

### 3.2.1.2 Experimentos en un Perceptrón Multicapa con la función *logsig* en la capa de entrada y salida.

A pesar de que en los experimentos cualitativos realizados con una sola capa oculta, colocando la función de transferencia *logsig* en las capas de entrada y salida no provocaron una disminución del error, ni una mejoría en los resultados obtenidos, vale la pena comprobar si cuantitativamente los resultados continúan siendo poco significativos.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | logsig | tansig | logsig | 0.0770 |
| 2 | logsig | logsig | logsig | 0.0774 |
| 3 | logsig | poslin | logsig | 0.0733 |
| 4 | logsig | purelin | logsig | 0.0719 |
| 5 | logsig | satlin | logsig | 0.0755 |
| 6 | logsig | satlins | logsig | 0.0751 |

**Tabla 16** Resultados Experimentales 15.

Como se observa en la tabla y al igual que en las pruebas cualitativas, el cambio de la función de la capa de entrada y salida provocó un aumento considerable del error, no obstante es importante señalar que con el uso de la función de transferencia *purelin* es con la que se obtienen mejores resultados. Para observar cómo se comportó la red durante el entrenamiento, se muestra la siguiente figura

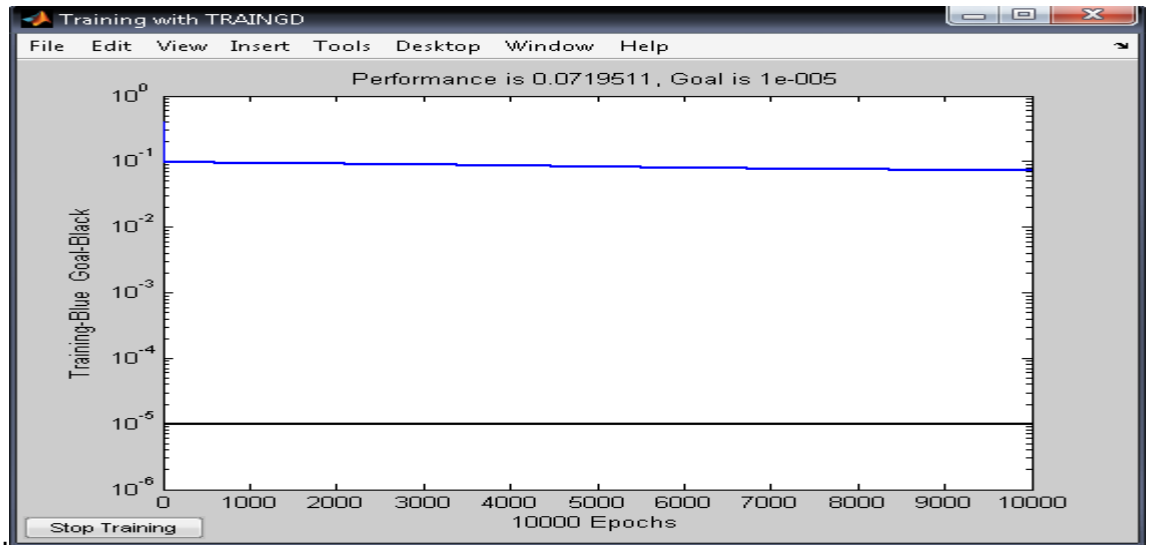


**Fig. 20** Comportamiento del experimento 4 - Resultados experimentales 15.

### 3.2.1.3 Experimentos con la función *logsig* en la capa de entrada y *tansig* en la de salida de la red.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | logsig | tansig | tansig | 0.0661 |
| 2 | logsig | logsig | tansig | 0.0766 |
| 3 | logsig | poslin | tansig | 0.0681 |
| 4 | logsig | purelin | tansig | 0. 0727 |
| 5 | logsig | satlin | tansig | 0.0698 |
| 6 | logsig | satlins | tansig | 0.0823 |

**Tabla 17** Resultados Experimentales 16.

En los resultados mostrados en esta tabla se observa que los errores no son mejores que los que se obtuvieron al usar la función *tansig* en las capas de entrada y salida, por lo que hasta el momento utilizar la función *logsig* en las capas de entrada y salida no ha sido eficiente.

Debido a que el primer experimento fue el de menor error obtenido, a continuación se muestra cómo se comportó la red durante todas las etapas de su aprendizaje.

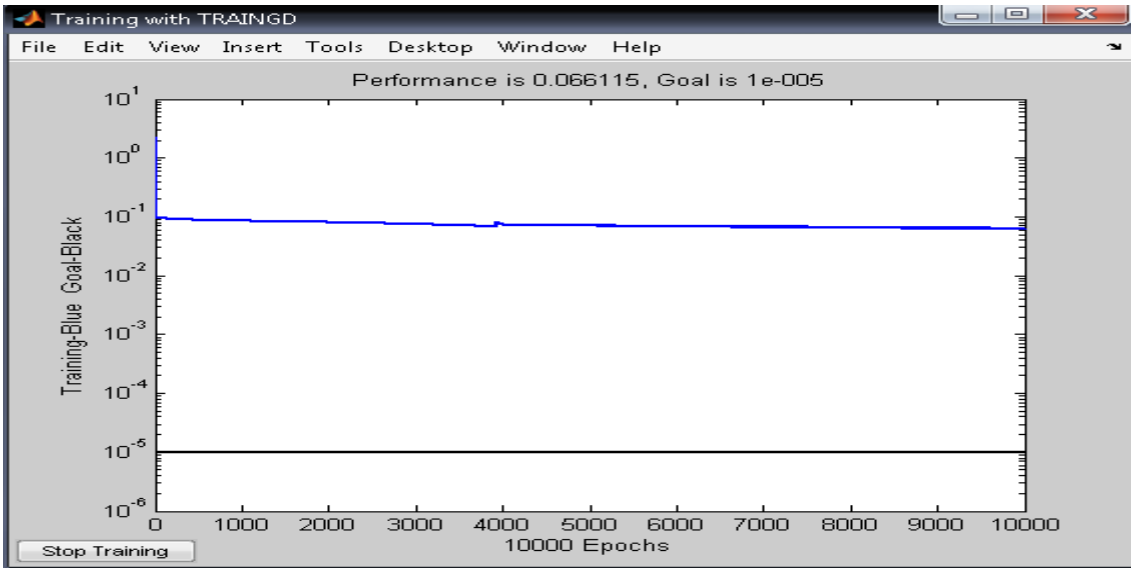


**Fig. 21** Comportamiento del experimento 1- Resultados experimentales 16.

### 3.2.1.4 Experimentos con la función *tansig* en la capa de entrada y *logsig* en la de salida de la red.

| Experimento | Capa Entrada | Capa Oculta | Capa Salida | MSE |
|---|---|---|---|---|
| 1 | tansig | tansig | logsig | 0.0766 |
| 2 | tansig | logsig | logsig | 0.0854 |
| 3 | tansig | poslin | logsig | 0.0629 |
| 4 | tansig | purelin | logsig | 0.0664 |
| 5 | tansig | satlin | logsig | 0.0628 |
| 6 | tansig | satlins | logsig | 0.0635 |

**Tabla 18** Resultados Experimentales 17.

En esta tabla se observa que no hay un error inferior al menor error obtenido en uno de los seis primeros experimentos cuantitativos con una capa oculta, o sea se demuestra que ninguna de las seis combinaciones de funciones mostradas en la tabla es una buena opción si se desea obtener una mejoría de los resultados en el entrenamiento del Perceptrón Multicapa, pero sí se puede notar que algunas funciones lineales tales como la *satlin, satlins* y la *poslin* provocan que el error devuelto por la red en cada caso disminuya, respecto a los que devuelve con otra función .

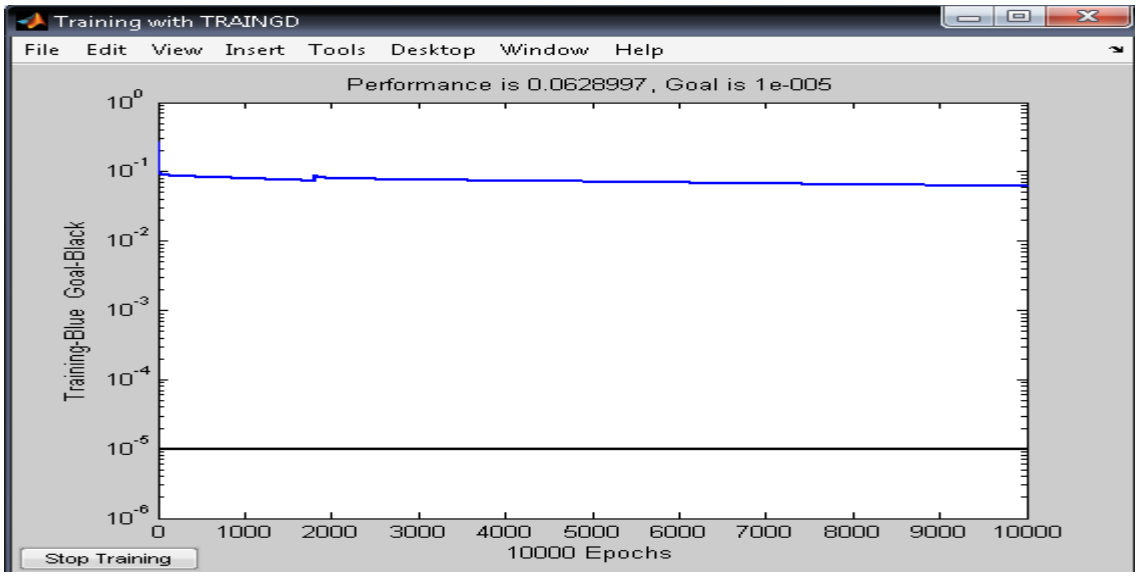


**Fig. 22** Comportamiento del experimento 5 - Resultados experimentales 17.

### 3.2.2 Resultados Cuantitativos con dos capas ocultas

Para este tipo de experimentos se sigue la misma secuencia de combinaciones utilizadas en los experimentos cualitativos con dos capas ocultas.

#### 3.2.2.1 Experimentos con la función *tansig* en las capa de entrada y salida, y en la segunda capa oculta, variando la primera capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | tansig | tansig | 0.0501 |
| 2 | tansig | logsig | tansig | tansig | 0.6119 |
| 3 | tansig | poslin | tansig | tansig | 0.0543 |
| 4 | tansig | purelin | tansig | tansig | 0.0858 |
| 5 | tansig | satlin | tansig | tansig | 0.0494 |
| 6 | tansig | satlins | tansig | tansig | 0.0487 |

**Tabla 19** Resultados Experimentales 18.

En los resultados mostrados por esta tabla se observa que las funciones de transferencia *tansig* inicialmente mejora los resultados obtenidos hasta el momento, pero en el último experimento de la tabla en la cual interviene la función *satlins*, el error es aún más pequeño, además hay otras funciones que aunque devolvieron un error un poco mayor, tienen un buen comportamiento.

Para mostrar visualmente lo antes mencionado se presenta la siguiente figura:

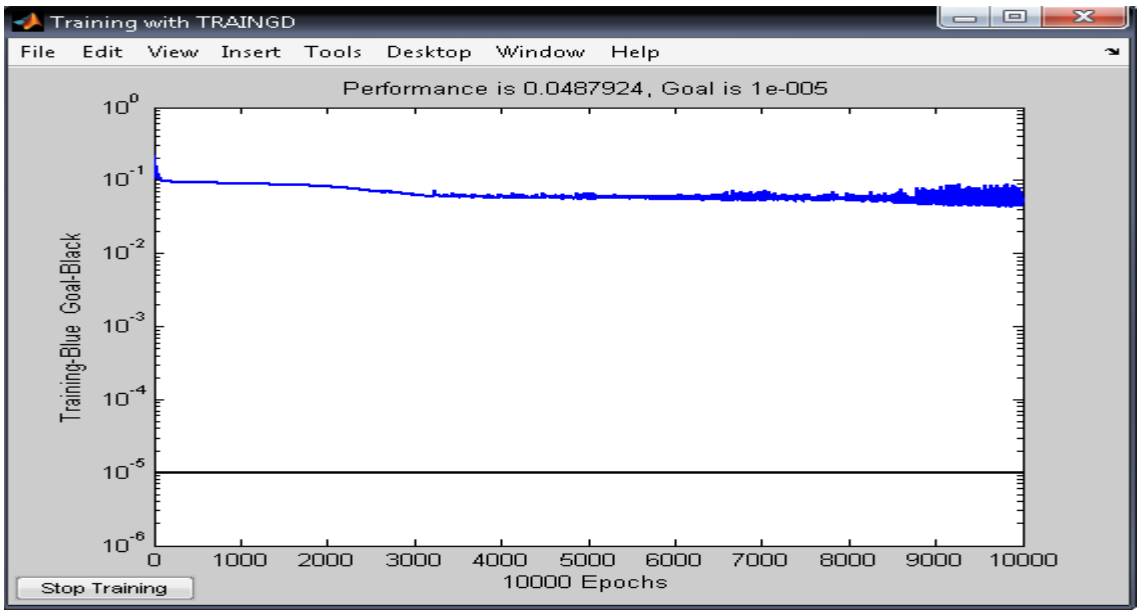


**Fig. 23** Comportamiento del experimento 6 - Resultados experimentales 18.

### 3.2.2.2 Experimentos con la función *tansig* en la capa de entrada y salida, variando la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | logsig | tansig | 0.0517 |
| 2 | tansig | tansig | poslin | tansig | 0.0317 |
| 3 | tansig | tansig | purelin | tansig | 0.0590 |
| 4 | tansig | tansig | satlin | tansig | 0.0346 |
| 5 | tansig | tansig | satlins | tansig | 0.0511 |

**Tabla 20** Resultados Experimentales 19.

Los resultados mostrados en la tabla reflejan muy buenos resultados, basta ver el error devuelto en el segundo experimento usando la función *poslin* y el cuarto con la *satlin*, ambas en la segunda capa oculta.

Es importante tener en cuenta que hasta el momento los resultados obtenidos son los mejores, además cabe señalar que todas las funciones de transferencia tuvieron un buen comportamiento. Si se observan detenidamente los resultados devueltos por la red y se hace una comparación con los deseados, es muy fácil darse cuenta que la combinación de funciones por capa presentada, es muy buena.

Seguidamente se muestra el comportamiento de la red durante todas las etapas de su aprendizaje.

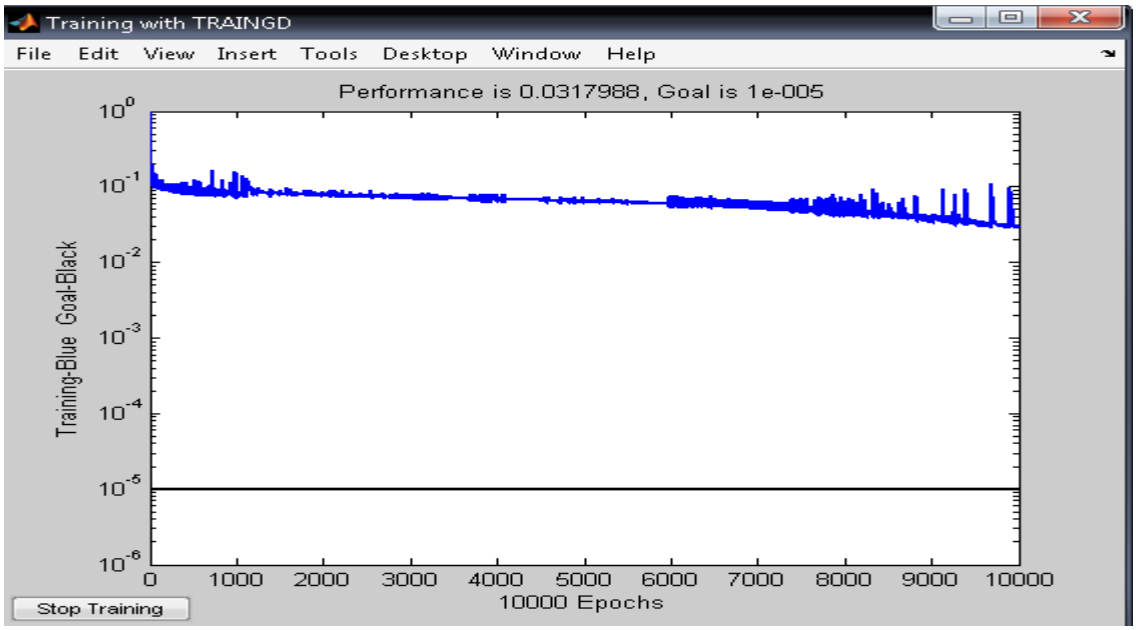


**Fig. 24** Comportamiento del experimento 2 - Resultados experimentales 19.

### 3.2.2.3 Experimentos con la función *tansig* en la capa de entrada y salida, la *logsig* en la primera capa oculta y variando la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | logsig | poslin | tansig | 0.0529 |
| 2 | tansig | logsig | purelin | tansig | 0.0642 |
| 3 | tansig | logsig | satlins | tansig | 0.0628 |

| | | | | | |
|---|---|---|---|---|---|
| 4 | tansig | logsig | satlin | tansig | 0.0564 |

**Tabla 21** Resultados Experimentales 20.

Los resultados mostrados en esta tabla demuestran lo mismo que se observó en los experimentos cualitativos en los que se intercambió la función *tansig* por la *logsig* para combinarla con otras funciones en las capas ocultas.

Lo más significativo de los experimentos mostrados es que nuevamente se observa que las funciones de transferencia *poslin* y *satlin* al combinarlas con las sigmoidales provocan una mejoría considerable del error.

A continuación se muestra el comportamiento de la red compuesta por la función *poslin* del primer experimento de la tabla 21:

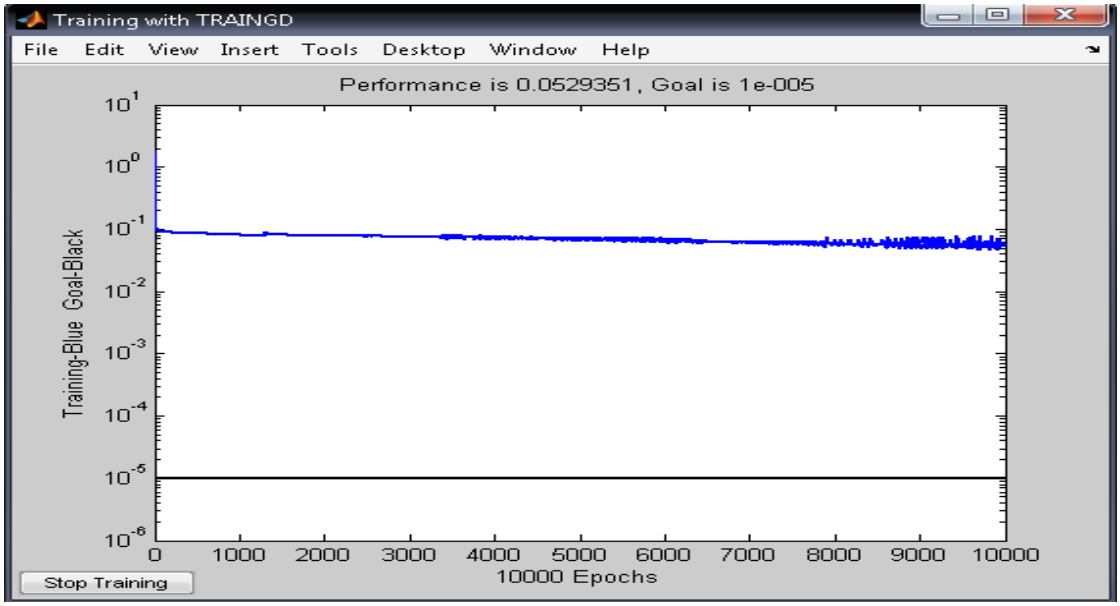


**Fig. 25** Comportamiento del experimento 1 - Resultados experimentales 20.

### 3.2.2.4 Experimentos con la función *tansig* en la capa de entrada y salida, la logsig en la segunda capa oculta y variando la primera capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | logsig | logsig | tansig | 0. 0639 |
| 2 | tansig | poslin | logsig | tansig | 0.0545 |
| 3 | tansig | purelin | logsig | tansig | 0.0488 |
| 4 | tansig | satlin | logsig | tansig | 0.0470 |
| 5 | tansig | satlins | logsig | tansig | 0.0481 |

**Tabla 22** Resultados Experimentales 21.

Como se puede observar en esta tabla, se han obtenido varios errores pequeños, aunque ninguno por debajo del menor obtenido hasta el momento. Resulta significativo señalar que nuevamente la función *satlin* provocó una disminución del error.

A continuación se muestra el comportamiento durante todas las etapas de entrenamiento de la red mostrada en el cuarto experimento.

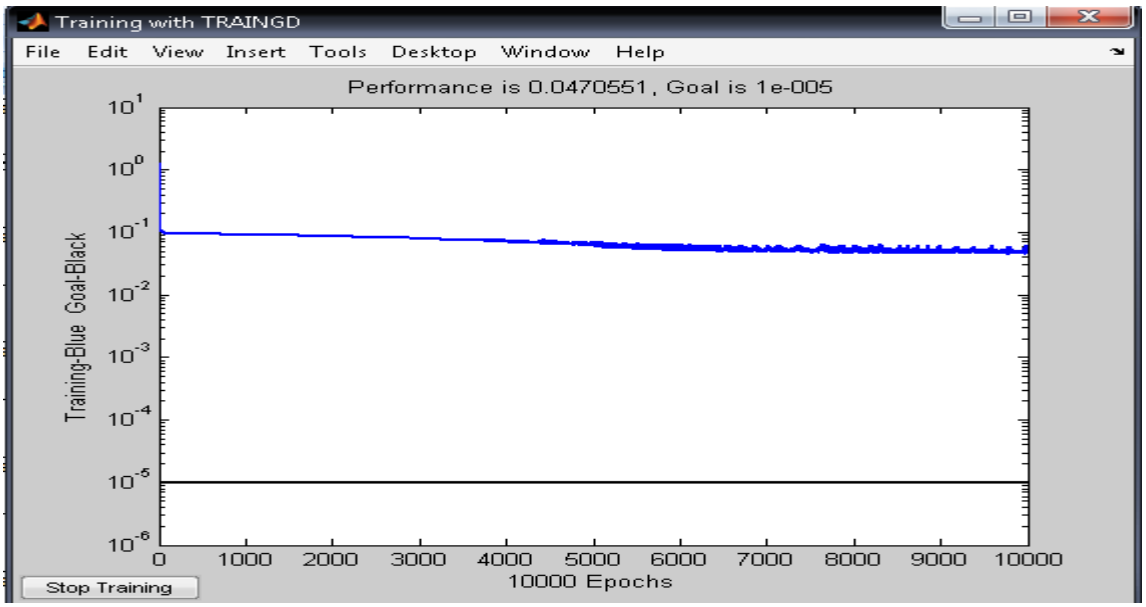


**Fig. 26** Comportamiento del experimento 4 - Resultados experimentales 21.

### 3.2.2.5 Experimentos con la función *tansig* en la capa de entrada y salida y una función lineal en ambas capas ocultas de la red.

Debido al buen resultado de funciones lineales como la *poslin* y la *satlin* se decidió realizar experimentos con las cuatro funciones lineales que forman parte de la investigación para ver si el hecho de que no sean combinadas con las funciones de transferencia sigmoidales en las capas ocultas provoca un resultado a favor de la mejoría de los resultados.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | Error |
|---|---|---|---|---|---|
| 1 | tansig | poslin | poslin | tansig | 0.0491 |
| 2 | tansig | satlin | satlin | tansig | 0.0632 |
| 3 | tansig | satlins | satlins | tansig | 0.0623 |
| 4 | tansig | purelin | purelin | tansig | 0.6883 |

**Tabla 23** Resultados Experimentales 22.

Los resultados obtenidos demuestran que aunque las funciones lineales provocan una mejoría de los resultados de la red, así como una disminución del error cuadrático medio cuando se utilizan en las capas ocultas del Perceptrón Multicapa, no se debe poner en ambas capas ocultas una función lineal.

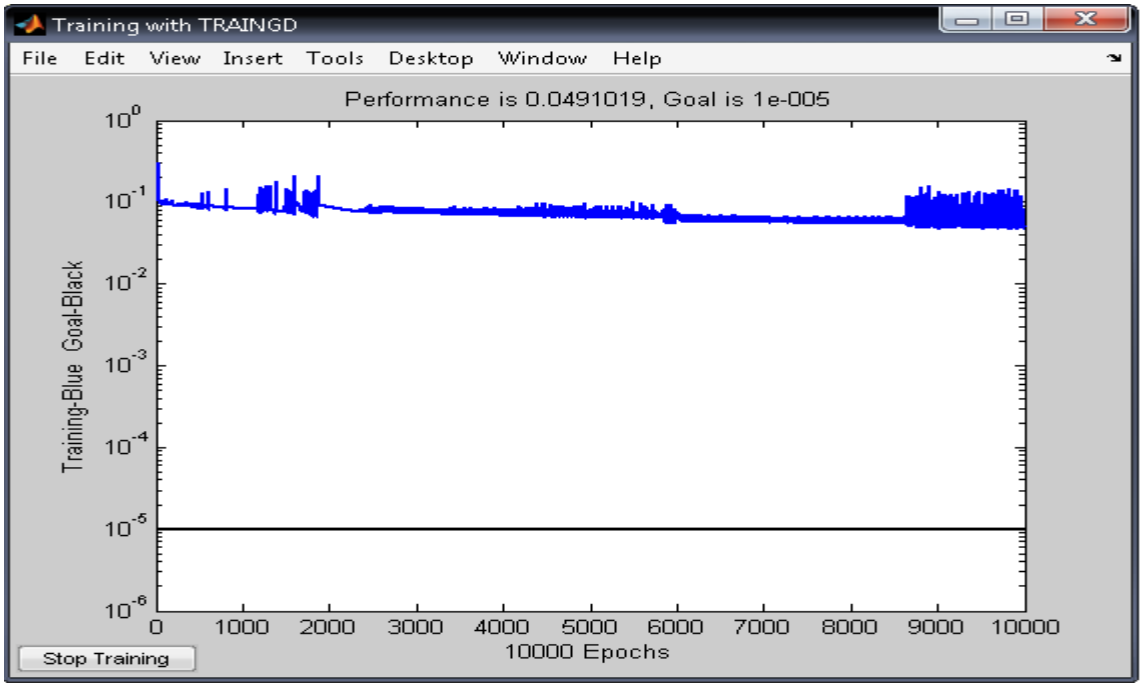


**Fig. 27** Comportamiento del experimento 1 - Resultados experimentales 22.

### 3.2.2.6 Experimentos con la función *tansig* en la capa de entrada y la primera capa oculta, la *logsig* en la de salida, y variando las segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | tansig | logsig | 0.0572 |
| 2 | tansig | tansig | logsig | logsig | 0.0627 |
| 3 | tansig | tansig | poslin | logsig | 0.0501 |
| 4 | tansig | tansig | purelin | logsig | 0.0674 |
| 5 | tansig | tansig | satlin | logsig | 0.0506 |

| | | | | | |
|---|---|---|---|---|---|
| 6 | tansig | tansig | satlins | logsig | 0.0626 |

**Tabla 24** Resultados Experimentales 23.

Como se observa en la tabla, ninguno de los experimentos arroja un error pequeño si se comparan con algunos que se han obtenido usando la función *tansig* en la capa de entrada y en la de salida. Lo más significativo de estos experimentos es que con las funciones que menor error se obtuvo fue con la *poslin* ya la *satlin* respectivamente. A continuación mostramos el comportamiento usando la función *poslin*.

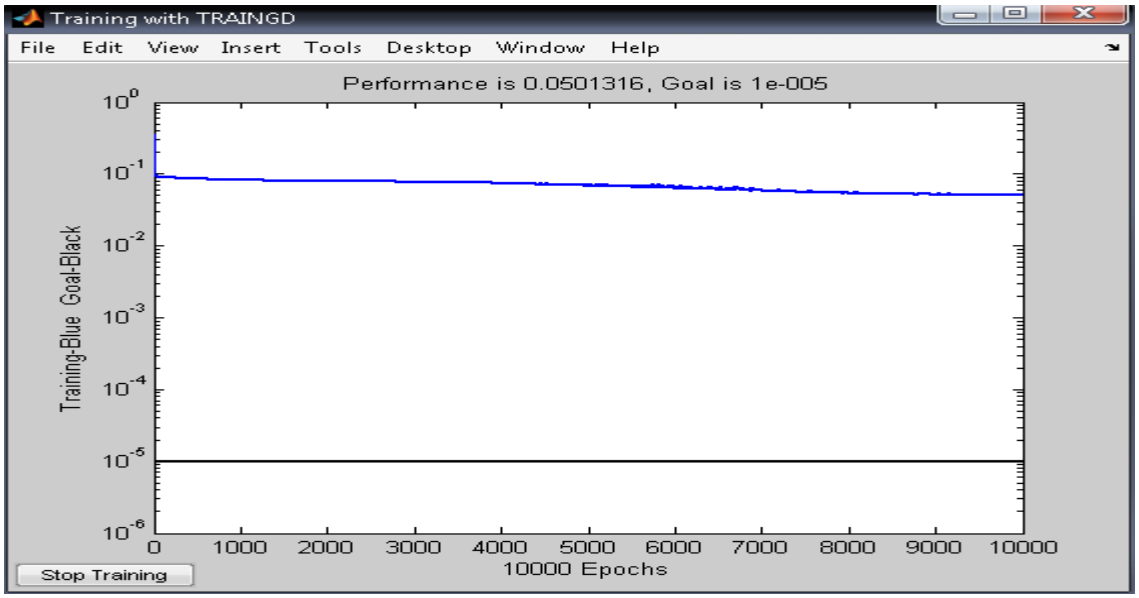


**Fig. 28** Comportamiento del experimento 3 - Resultados experimentales 23.

### 3.2.2.7 Experimentos con la función *tansig* en la capa de entrada y la segunda capa oculta, la *logsig* en la de salida, y variando las primera capa oculta.

A diferencia de los seis experimentos anteriores, ahora se observará la influencia que puede tener variar la función de transferencia de la primera capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1 | tansig | logsig | tansig | logsig | 0.0631 |
| 2 | tansig | poslin | tansig | logsig | 0.0554 |
| 3 | tansig | purelin | tansig | logsig | 0.0658 |
| 4 | tansig | satlin | tansig | logsig | 0.0548 |
| 5 | tansig | satlins | tansig | logsig | 0.0625 |

**Tabla 25** Resultados Experimentales 24.

En la presente tabla se observa se observa que el hecho de variar la primera capa oculta provoca que ningún experimento retornó un error menor que al variar la segunda capa oculta, no obstante, nuevamente las funciones de transferencia que mejores resultados arrojaron fueron la *satlin* y la *poslin* respectivamente.

A continuación se muestra el comportamiento con la función *satlin*.

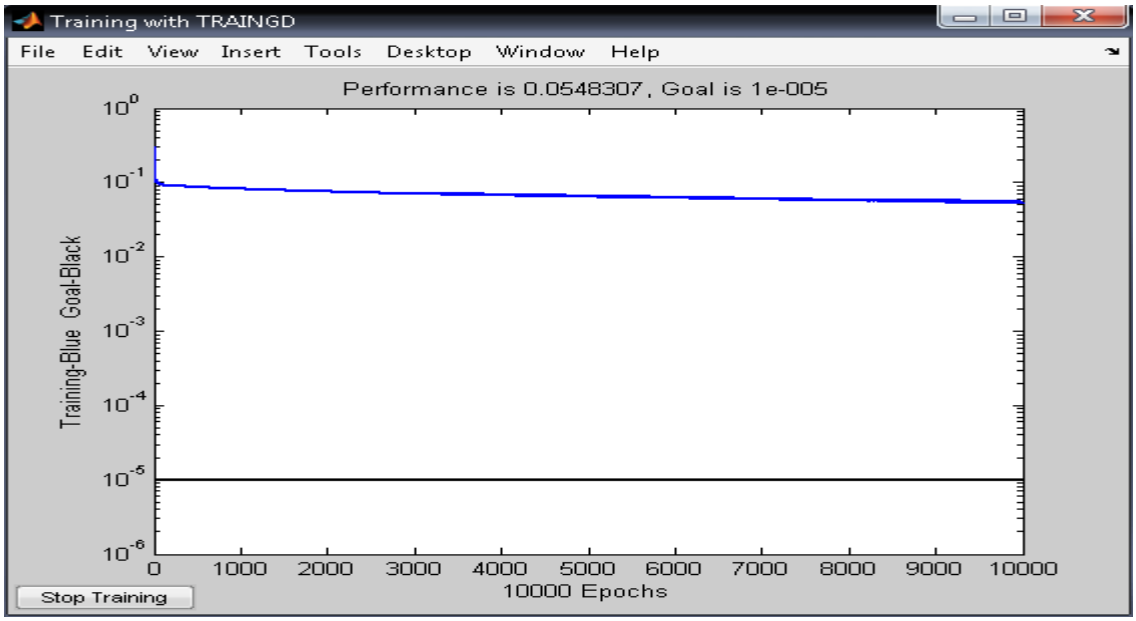


**Fig. 29** Comportamiento del experimento 4 - Resultados experimentales 24.

### 3.2.2.8 Experimentos con la función *logsig* en la capa de entrada, la *tansig* en la primera capa oculta y en la capa de salida, y variando la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | logsig | tansig | tansig | tansig | 0.0503 |
| 2 | logsig | tansig | logsig | tansig | 0.0855 |
| 3 | logsig | tansig | poslin | tansig | 0.0493 |
| 4 | logsig | tansig | purelin | tansig | 0.0631 |
| 5 | logsig | tansig | satlin | tansig | 0.0599 |
| 6 | logsig | tansig | satlins | tansig | 0.0634 |

**Tabla 26** Resultados Experimentales 25.

Como se observa en la tabla, a pesar de que en ningún experimento devuelve buenos resultados, ni un error pequeño, se puede observar que el experimento número 3 en el cual se utiliza la función *poslin* es el que menor MSE presenta.

A continuación se muestra el comportamiento visual del tercer experimento.

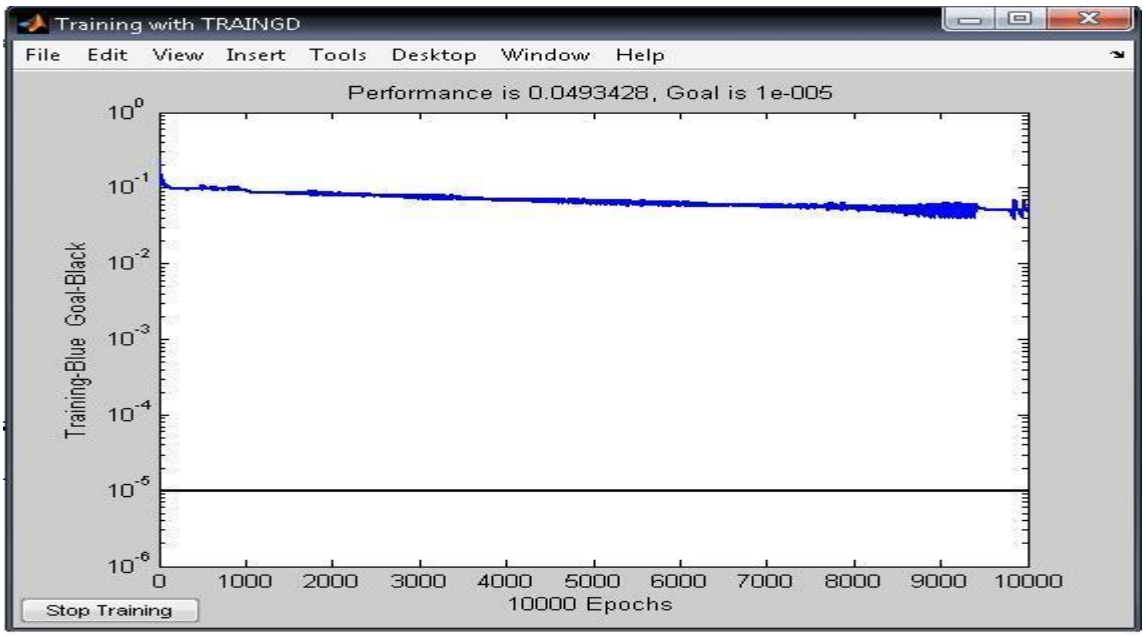


**Fig. 30** Comportamiento del experimento 3 - Resultados experimentales 25.

### 3.2.2.9 Experimentos con la función *logsig* en la capa de entrada, la *tansig* en la segunda capa oculta y en la capa de salida, y variando la primera capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | logsig | logsig | tansig | tansig | 0.0813 |
| 2 | logsig | poslin | tansig | tansig | 0.0810 |
| 3 | logsig | purelin | tansig | tansig | 0.0647 |
| 4 | logsig | satlin | tansig | tansig | 0.0788 |
| 5 | logsig | satlins | tansig | tansig | 0.0656 |

**Tabla 27** Resultados Experimentales 26.

En esta tabla se observa que en ninguno de los experimentos se obtuvo un error inferior a los que proporcionan otro tipo de combinaciones vistas en el transcurso de la investigación.

Para mostrar el comportamiento visual durante todas las etapas del entrenamiento de la red que se observa en el experimento 3 se presenta la siguiente figura.

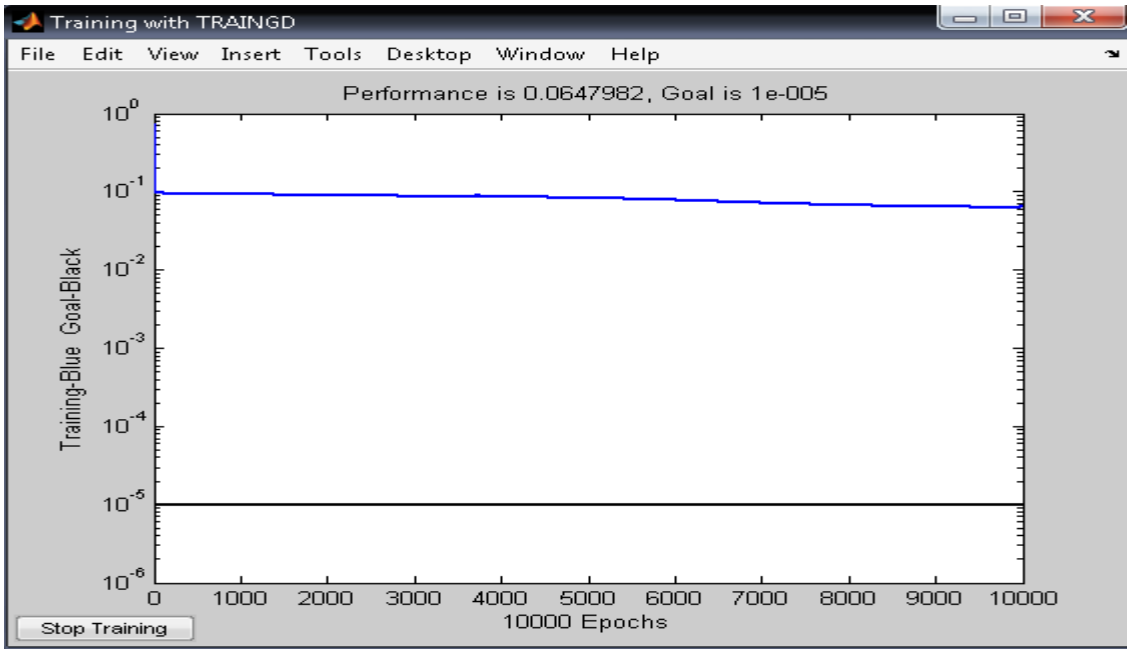


**Fig. 31** Comportamiento del experimento 3 - Resultados experimentales 26.

### 3.2.3 Experimentos cuantitativos con dos capas ocultas para E. Cloacae.

Para reafirmar los resultados obtenidos se hace necesario realizar un pequeño grupo de experimentos con otra bacteria, para esto se hace uso de las mejores combinaciones de funciones de transferencia que se han obtenido hasta el momento.

#### 3.2.3.1 Experimentos con la función *tansig* en la capa de entrada, primera capa oculta y en la de salida, variando la segunda capa oculta.

| Experimento | Capa Entrada | Capa Oculta 1 | Capa Oculta 2 | Capa Salida | MSE |
|---|---|---|---|---|---|
| 1 | tansig | tansig | tansig | tansig | 0.0332 |
| 2 | tansig | tansig | logsig | tansig | 0.0378 |
| 3 | tansig | tansig | poslin | tansig | 0.0298 |
| 4 | tansig | tansig | satlin | tansig | 0.0307 |
| 5 | tansig | tansig | satlins | tansig | 0.0466 |

**Tabla 28** Resultados Experimentales 27.

Como se puede observar en la tabla, utilizando otras muestras, también las funciones *poslin* y la *satlin* provocaron una mejoría del error cuadrático medio.

A continuación se muestra cómo se comportó la red con la función *poslin* en la segunda capa oculta.

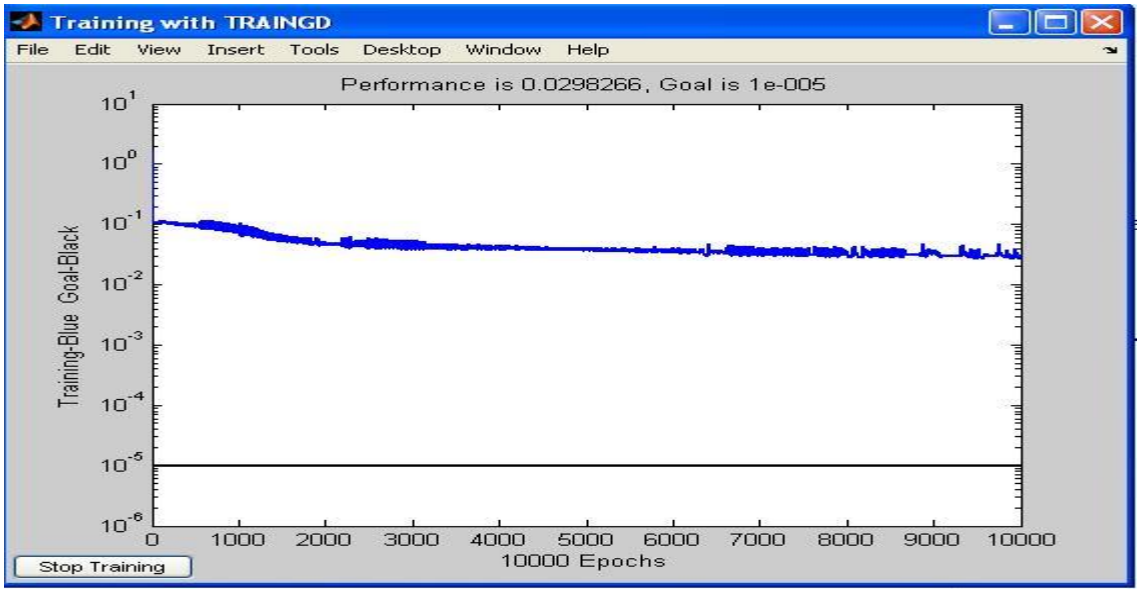


**Fig. 32** Comportamiento del experimento 3 - Resultados experimentales 27.

### 3.3 Conclusiones

- Se demostró, para la muestra analizada, que al variar las funciones de transferencia en el Perceptrón Multicapa pueden mejorarse los resultados, si se realiza el entrenamiento usando solamente dos capas ocultas y colocando la función de transferencia *tansig* en las capas de entrada y salida, la función *tansig* en la primera capa oculta, y la *poslin y satlin* en la segunda capa oculta.
- Se corroboró que no es recomendable que las funciones de entrada y salida sean distintas ya que el error aumenta considerablemente.

### 3.4 Recomendaciones

- Estudiar otras herramientas que permitan trabajar con el Perceptrón Multicapa con el fin de tenerlas bien definidas para posteriores experimentos.
- Continuar con grupos investigativos que profundicen más sobre el tema tratado.
- Extender los estudios realizados sobre el efecto de la variación de las funciones de transferencia en el Perceptrón Multicapa a otros tipos de investigaciones relacionadas con la IA.
- Realización de un diseño de experimento.

### 3.5 Bibliografía


1. **HAYKIN.** *Neural networks: a comprehensive foundation New York: Maxwell Macmillan.* 1994.

2. **VALDIVIA.** *Algoritmo LVQ aplicado a tareas de procesamiento de lenguaje natural.* Malaga : s.n., 2004.

3. **ESPINOZA.** *Las Redes Neuronales Artificiales y su importancia como herramienta en la toma de decisiones.* Universidad Nacional Mayor de San Marcos : s.n., 2002.

4. **Ballesteros, Alfonso.** Neural Network Framework. [En línea] [Citado el: 2009 de enero de 20.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/tipos-de-redes-neuronales.htm.

5. —. Neural Network Framework. [En línea] [Citado el: 26 de enero de 2009.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/tipos-de-redes-neuronales.htm.

6. Ayuda del Matlab. [En línea] [Citado el: 15 de enero de 2009.]

7. **Ceballo, D. and Y. Del Carmen.** *Categorización de texto usando Redes Neuronales Artificiales.* Ciudad de la Habana : s.n., 2008.

8. **Herrera, A. and Y. Prevot.** *Aplicaciones de las Redes Neuronales en Entornos Virtuales.* Ciudad de la Haban : s.n., 2008.

9. **Galindo, Pedro L.** Redes multicapa Algoritmo BackPropagation. [En línea] [Citado el: 2 de febrero de 2009.] http://www2.uca.es/dept/leng_sist_informaticos/preal/23041/transpas/E-Backpropagation/.

10. **jeffheaton.** Heatonresearch. [En línea] [Citado el: 16 de febrero de 2009.] http://www.heatonresearch.com/online/introduction-neural-networks-java-edition-2/chapter-5.

11. **Rey, Juan-Antonio Infante y José María.** Introducción a MATLAB. [En línea] [Citado el: 2 de febrero de 2009.] http://www.mat.ucm.es/~jair/matlab/notas.htm#programacion.

12. Electrical Engineering. [En línea] [Citado el: 25 de febrero de 2009.] http://www.eecs.umich.edu/.

## 3.6 Anexos

### 3.6.1 Anexos 1

**Funciones de entrenamiento que ofrece Matlab**

| **Función de entrenamiento** | **Nombre de la función** |
|---|---|
| trainb | Batch training with weight and bias learning rules. |
| trainbfg | BFGS quasi-Newton backpropagation |
| trainbr | Bayesian regularization. |
| trainc | Cyclical order incremental update. |
| traincgb | Powell-Beale conjugate gradien backpropagation |
| traincgf | Fletcher-Powell conjugate gradien backpropagation. |
| traincgp | Polak-Ribiere conjugate gradien backpropagation |
| traingd | Gradient descent backpropagation |
| traingda | Gradient descent with adaptive l backpropagation. |
| traingdm | Gradient descent with momentum backpropagation. |
| traingdx | Gradient descent with momentum & adaptive l backprop. |
| trainlm | Levenberg-Marquardt backpropagation |
| trainoss | One step secant backpropagation. |
| trainr | Random order incremental update. |
| trainrp | Resilient backpropagation (Rprop). |
| trains | Sequential order incremental update. |
| trainscg | Scaled conjugate gradient backpropagation |

### 3.6.2 Anexo 2

**Código en matlab para normalizar una matriz.**

```
function [normalizado,k,p,neg] = normaliza(matriz)

%##########################################################################
%##          Normaliza una matriz de orden [nt,ne]
%##-------------------------------------------------------------------
%##        Variables de entrada:
%##          matriz: Matriz de orden matriz[nt,ne]
%##          nt: Número de filas
%##          ne: Número de columnas
%##-------------------------------------------------------------------
%##        Variables de salida:
%##          normalizado: matriz con datos normalizados entre [-1,1]
%##          k y p: Parámetros utilizados en la normalización
%##          neg: Matriz columna de orden [nn,1]
%##          nn: Número de columnas negadas, cada valor identifica a la
%##              columna que se ha negado para normalizar
%##-------------------------------------------------------------------
%##########################################################################
[nt,ne]=size(matriz);
nn=0;
neg(1,1)=0;
 for i=1:ne
            matriz_max(i,1)=max(matriz(:,i));
            matriz_min(i,1)=min(matriz(:,i));
            if ((matriz_max(i,1)<=0) && (matriz_min(i,1)<=0))
```

```
  matriz(:,i)=-matriz(:, i);
  matriz_max(i,1)=-matriz_max(i,1);
  matriz_min(i,1)=-matriz_min(i,1);
  nn=nn+1;
  neg(nn,1)=i;
 end

 % Constantes para normalizar:
 k(i, 1)=2/(matriz_max(i,1)-matriz_min(i,1));
 p(i, 1)=-((matriz_max(i,1)+matriz_min(i,1))/(matriz_max(i,1)-matriz_min(i,1)));

 % Normaliza
 normalizado(:, i)=matriz(:,i).  *k(i,1)+p(i,1);


end
```

### 3.6.3 Anexo 3

**Código en Matlab para desnormalizar una matriz.**

```
function [matriz] = desnormaliza(normalizado,k,p, neg)

%#########################################################################
%##          Normaliza una matriz de orden [nt,ne]
%##----------------------------------------------------------------------
%##        Variables de entrada
%##          normalizado matriz con datos normalizados entre [-1,1]
%##          k y p Parámetros utilizados en la normalización
%##          neg Matriz columna de orden [nn,1]
%##          nn Número de columnas negadas, cada valor identifica a la
%##              columna que se ha negado para normalizar
%##
%##----------------------------------------------------------------------
%##        Variables de salida
%##          matriz Matriz de orden matriz[nt,ne]
%##          nt  Número de filas
%##          ne  Número de columnas
%##----------------------------------------------------------------------
%#########################################################################
[nt,ne]=size(normalizado);
[nn,nf]=size(neg);

for i=1:ne
    matriz(:,i)=(normalizado(:,i)-p(i,1)).  /k(i,1);
    for j=1:nn
       if (i==neg(j,1))
          matriz(:,i)=-matriz(:,i);
       end
    end
end
```

**Glosario de Términos:**

**pesos sinápticos**: peso determinado entre las *conexiones* de las neuronas .

**conexiones sinápticas**: conexiones entre neuronas que permiten la comunicación entre estas.

**Aprendizaje de Hebb**: regla de aprendizaje normalmente usada en redes de aprendizaje Competitivo.

**conexiones excitatorias**: aumentan la posibilidad de que se dispare la célula objetivo.

**conexiones inhibitorias**: disminuyen la posibilidad de que se dispare la célula.

**factor de ponderación**: peso o intensidad de conexión entre las neuronas.

**newff**: sintaxis que define Matlab para crear una nueva red neuronal artificial con conexión hacia delante.

**sim**: sintaxis que define Matlab para simular una red neuronal artificial.

**train**: sintaxis que define Matlab para el entrenamiento de la red neuronal artificial.

**Procesamiento distribuido**: es un conjunto de computadoras autónomas interconectadas que cooperan compartiendo recursos (físicos y datos).

**Procesamiento paralelo**: es una tarea utilizando múltiples procesadores en el mismo tiempo con el fin de reducir el tiempo de ejecución.

**MLP**: Perceptrón Multicapa.

**RNA:** Red Neuronal Artificial**.**

**MSE:** Error Cuadrático Medio.