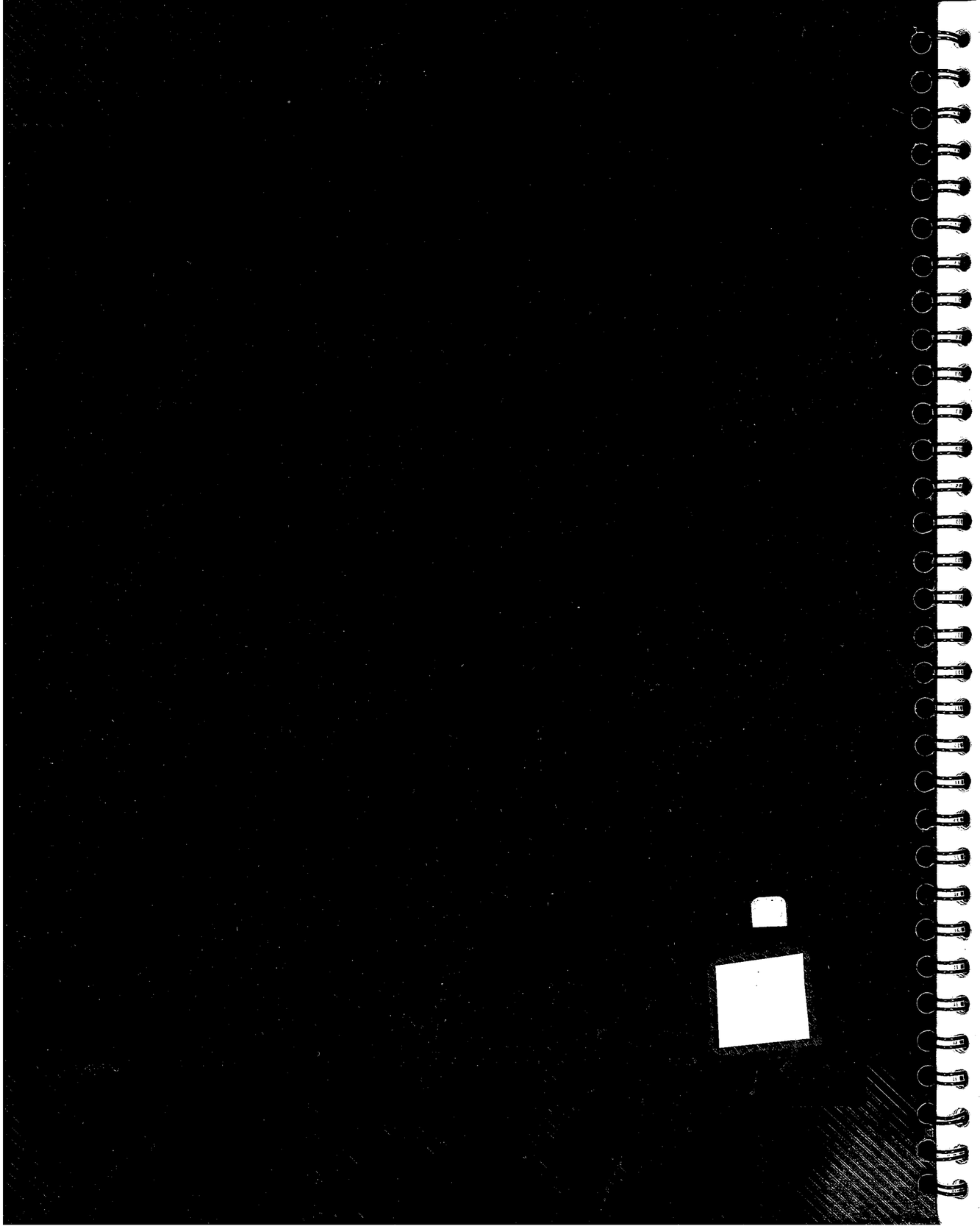


CIMAT

Centro de Investigación en Matemáticas, A.C.

Segmentación de Imágenes Utilizando Redes Neuronales Recurrentes

Tesis

que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

presenta

Edmar Mota García

Director de Tesis

Dr. José Luis Marroquín Zaleta

Guanajuato, Gto., Agosto del 2001

Abstract

This work presents an innovative structure and algorithms for image processing using recurrent neural networks with backpropagation.

The use of Neural Networks for image processing, principally for the task of segmentation, is not new. Simple Backpropagation Feed-Forward Networks, Bayesian Learners, RBF Nets, Hopfield Networks, Kohonen Maps and many other architectures have been developed and tested with great results in this field, but as far as we know, a recurrent architecture has never been used in the way its introduced in this thesis.

The aim of this work is first the introduction of a recurrent structure capable of segmenting a gray-scale image and second, to prove that the use of recursive feedback in the training of common, small sized feed-forward networks improves their performance in the presence of conflictive patterns.

This work proposes the use of a simple feed-forward network as a window operator to produce a classification. This intermediate classification is used by the network to produce a new classification that improves the last one. Two algorithms for the learning process, a simpler recurrent algorithm and a modified recurrent algorithm are introduced and compared with a similar network without the use of feedback.

Two kinds of experiments where carried out in this work: initially a simpler 2 class problem with 96 binary images of many representations of a handwritten number two, and finally a more complex 3 class problem using 10 synthetic gray-scale images of random-shaped polygons.

The results tell us that the use of feedback is worthy and in conjunction with the modified algorithm proves to be an alternative that is both innovative and simple to implement.

Resumen

Este trabajo introduce una innovadora estructura y algoritmos para el procesamiento de imágenes utilizando redes neuronales recurrentes con retropropagación.

El uso de las redes neuronales para el procesamiento de imágenes, principalmente en el campo de la segmentación, no es novedoso. Redes simples del tipo feed-forward entrenadas con retropropagación, redes Bayesianas, redes Hopfield, redes RBF, mapas de auto-organización de Kohonen y muchas otras arquitecturas han sido desarrolladas y probadas con éxito en este campo, pero hasta donde sabemos no se han utilizado redes recurrentes tal y como se desarrolla en esta tesis.

El objetivo de este trabajo es, primero, el introducir una estructura recurrente capaz de segmentar imágenes en escala de gris, y segundo, el probar que la utilización de la retroalimentación mejora el desempeño de una red normal y pequeña del tipo feed-forward ante la presencia de patrones conflictivos.

Este trabajo propone la utilización de una red básica del tipo feed-forward como operador de ventana para producir una clasificación. Esta clasificación es utilizada por la red para obtener sucesivas clasificaciones que mejoren la anterior. Dos algoritmos de aprendizaje, un algoritmo simple recurrente y un algoritmo recurrente modificado, se presentan y se comparan con una red similar sin la utilización de retroalimentación.

Dos clases de experimentos se llevaron a cabo: inicialmente uno más sencillo de 2 clases utilizando 96 imágenes binarias de distintas representaciones manuscritas del número dos y al final uno más complejo de 3 clases utilizando imágenes sintéticas en escala de gris de polígonos con formas aleatorias.

Los resultados obtenidos expresan que vale la pena el uso de la retroalimentación y que en conjunto con la aplicación del algoritmo modificado prueban ser una alternativa tanto innovadora como simple de programar.

Agradecimientos

A todas las personas que contribuyeron a que se pudiera terminar con satisfacción el presente trabajo.

Al Dr. José Luis Marroquín, quién me ayudó en gran medida a la elaboración del trabajo de tesis y que con sus explicaciones y comentarios me hizo entender la importancia de la calidad dentro de la investigación y de la experimentación, todo esto necesario para poder continuar y sobresalir con éxito en este difícil camino.

Al Dr. Mariano J. Rivera, al Dr. Johan Van Horebeek, al Dr. Salvador Botello Rionda, al Dr. Carlos Montes de Oca y al Dr. Francisco Sánchez, quienes durante el tiempo que estuve en Cimat me transmitieron sus conocimientos y experiencias en distintos ámbitos del cómputo científico, la gran mayoría de los cuales yo desconocía completamente y que me abrieron los ojos hacia una visión más amplia de la ciencia y la labor de investigación.

Al Dr. Raúl Rojas, quién a través de sus seminarios a la comunidad de Cimat y en las clases que nos impartió, logró que me interesara en el apasionante estudio de las redes neuronales artificiales.

A mis compañeros durante la maestría, Jesús E. Navarro, José Antonio Muñoz, Javier Vigueras y Pedro Elizarraraz, con los que pase ratos muy agradables de estudio y convivencia que ayudaron para que lograra finalizar con éxito mis estudios, y que además encontrara muchos amigos de verdad.

A mi esposa Adriana por el apoyo y cariño durante la realización de este trabajo y de mis estudios, no importando las distancias físicas que durante un tiempo nos separaron.

A mis padres y hermano, pues con su ejemplo me han impulsado a ser trabajador, honesto y sincero, ya que estas son acciones con las cuales uno puede alcanzar cualquier cosa.

Contenido

Capítulo 1

Introducción.

Utilizar redes neuronales para segmentar imágenes no es novedoso. Durante los últimos años se han utilizado distintos modelos de redes neuronales para obtener segmentaciones más o menos exitosas. Se han utilizado modelos de redes de retropropagación [19], asociativas [1],[18], de Funciones de Base Radial (RBF) y Hopfield, así como redes neuronales Bayesianas [24] y redes con neuronas de codificación de pulso (Pulse-Coding)[27]. Se han utilizado redes en combinación con otras técnicas conocidas como filtros de Gabor o MAP, para obtener los vectores de entrada [22].

La utilización de redes entrenadas con retropropagación para la segmentación de imágenes ha sido limitada a utilizar las redes con múltiples capas para realizar el mapeo de una serie de valores representativos de la imagen o características a su correspondiente clasificación previamente validada. Esto es porque la limitada capacidad de las redes con retropropagación a ser invariantes a desplazamientos, homotecias, etc, hace necesario el cálculo de características que suplan estas deficiencias (que ya tengan estas propiedades de invariabilidad) para poder realizar segmentaciones exitosas.

Los intentos por utilizar las redes neuronales entrenadas con retropropagación directamente a los datos, en este caso los píxeles de la imagen, dentro de una pequeña vecindad local, se han topado con el problema en la convergencia del aprendizaje de la red al presentársele patrones conflictivos; llamados estos los que presentan un mapeo de uno a muchos, algo que es común si visualizamos la

correspondencia entre los píxeles originales y su clasificación; efectivamente desaprende los patrones previamente mostrados [23], haciendo sumamente difícil que ésta pueda con cierto grado de certidumbre hacer el correspondiente mapeo del espacio de la imagen original al espacio de la imagen de clasificación.

Una solución ya probada es realizar una selección de zonas representativas de la imagen para evitar tales patrones [1],[15], lo cual puede ser difícil en imágenes muy ruidosas o degradadas además de agregar algún trabajo extra al proceso. Se han utilizado también otras estrategias del tipo "divide y conquista" con éxito en el ámbito del reconocimiento de caracteres [2].

1.1 Motivación y Objetivos.

La utilización de redes neuronales recurrentes, conocidas estas como toda red neuronal que recibe retroalimentación de una o más de sus unidades internas, se ha probado como una manera eficaz de incluir algún tipo de información extra en el conjunto de aprendizaje que ayude a mejorar la correspondencia entre ambos espacios [23],[6], y varias variantes de ella han sido investigadas por Jordan[8], Elman[4]y otros pero hasta donde sabemos no se han utilizado en el procesamiento de imágenes de la manera que presentamos.

La propuesta presentada en este trabajo de tesis es usar una red de múltiples capas entrenada con retropropagación como un operador de ventana sobre una imagen, utilizando las salidas obtenidas por la red como retroalimentación hacia la misma, un número definido de veces, para lograr una segmentación de una imagen; esta idea aunada a un algoritmo de entrenamiento novedoso, son nuestras principales contribuciones.

Uno de los objetivos principales de este trabajo, además, es conocer el alcance de las redes neuronales recurrentes de aplicación local para realizar segmentaciones de gran calidad dentro del ámbito del aprendizaje supervisado en redes neuronales.

1.2 Desarrollo de la tesis.

El presente trabajo se encuentra divido en 6 capítulos y un apéndice.

En el capítulo 2 se presentan las bases del campo de las redes neuronales. En este capítulo se describe la motivación detrás de este paradigma así como una breve historia de su desarrollo. Se desarrolla la teoría básica detrás de las redes de múltiples capas, el algoritmo de aprendizaje más utilizado de esta arquitectura: el algoritmo de *Retropropagación*; así como algunas técnicas de aceleración de convergencia más utilizadas. También se describen las características y arquitecturas más comunes de las redes recurrentes.

El capítulo 3 da un breve vistazo al estado del arte en cuanto a redes neuronales y procesamiento de imágenes se refiere, haciendo una breve descripción de algunas interesantes propuestas que se han desarrollado para resolver algunos de los problemas dentro del campo del procesamiento de imágenes y de visión como segmentación, clasificación y análisis. El mayor énfasis se da en los métodos de segmentación, lugar que ocupa esta tesis en el campo de procesamiento de imágenes.

El capítulo 4 describe a detalle la estructura que este trabajo propone, así como los algoritmos que se desarrollaron para el proceso de segmentación y los conceptos de visión utilizados.

En el capítulo 5 se muestran los experimentos realizados. Aquí se presentan todas las características de tales pruebas así como los resultados obtenidos. También se da una explicación sobre el equipo utilizado para su realización.

Nuestras conclusiones a los experimentos realizados se expresan en el capitulo 6. Las referencias utilizadas en la elaboración de este trabajo de tesis se encuentran en el capítulo 7.

Capítulo 2

Redes Neuronales Artificiales.

En este capítulo se pretende explicar, inicialmente, las bases generales de todas las redes neuronales artificiales, el origen de su desarrollo, así como su historia y sus más importantes conceptos teóricos como arquitecturas, funciones de activación, de propagación, pesos, etc. Más adelante, se explicarán a detalle las redes neuronales con retropropagación, uno de los aspectos centrales de esta tesis, así como el algoritmo de entrenamiento que le da el nombre y que permite que esta red pueda realizar la aproximación del mapeo de espacios que requerimos para nuestra estructura.

Una de las partes importantes de la novel aproximación que presentamos en este trabajo es utilizar la retroalimentación, por lo que se explicará los aspectos más relevantes y algunas propiedades de las redes neuronales que utilizan este tipo de conexiones: las redes neuronales recurrentes.

Se describirán también algunas de las más utilizadas arquitecturas de redes recurrentes que se han propuesto hasta ahora para tener un marco de referencia de el lugar que ocupa nuestra propuesta en el siempre cambiante ámbito de las redes neuronales artificiales recurrentes.

2.1 Introducción.

2.1.1 Motivación

Las Redes Neuronales Artificiales (RNA) son un intento de modelar las capacidades de procesamiento de información de los sistemas nerviosos de los seres vivos[20].

Existe un número muy grande de problemas que son extremadamente difíciles de tratar por una computadora o un procesador secuencial. Algunos de ellos son del tipo *no-polinomial*, pues el tiempo necesario para resolverlos en una computadora crece a tazas que no pueden expresarse como una función polinomial de su tamaño. Otros son llamados *intratables* pues por su naturaleza no pueden ser expresados en términos de la arquitectura computacional convencional introducida por Von-Neumann. Algunos ejemplos de estos problemas incluyen la predicción del tiempo; interpretación de imágenes y de visión; comprensión de frases y reconocimiento de caracteres manuscritos.

Los sistemas nerviosos de los seres vivos, desde el más simple, parecen poder procesar instantáneamente muchos de estos problemas sin ninguna dificultad, utilizando únicamente (hasta donde se conoce) unidades de activación química llamadas *neuronas* que reaccionan en tiempos mucho menores que nuestras más poderosas computadoras.

La diferencia tan grande entre las capacidades de un sistema nervioso animal y una computadora secuencial parece radicar en las diferencias en arquitectura entre ambos enfoques. Mientras nuestras computadoras procesan toda la información secuencialmente, los sistemas nerviosos lo hacen en paralelo.

El estudio profundo de los procesos que se realizan dentro de los sistemas nerviosos; cómo se aprende, reconoce, memoriza, etc; han permitido desarrollar modelos que, matemáticamente y físicamente, intentan simular tales procesos, así como implementarlos tanto en nuestras computadoras como en procesadores paralelos.

2.1.2 Historia.

Las raíces del trabajo que se ha venido realizando dentro del campo de las RNA se pueden encontrar en investigaciones sobre los procesos cerebrales realizados a finales del siglo XIX [13].

Se les acredita a McCulloch y Pitts (1943) el desarrollo del primer modelo matemático de una neurona artificial. Dicho modelo utiliza simples funciones de binarias de paso (threshold) dentro de sus unidades de procesamiento. Inspirados por la neurobiología estudiaron el problema de abstraer conceptos universales de percepciones específicas. Aún cuando sus conceptos eran muy sencillos, estas unidades probaron ser capaces de resolver un gran número de problemas, desde simple lógica binaria hasta autómatas finitos, al enlazarse en complejas estructuras o redes [20].

En 1958 Frank Rosenblatt propone el *perceptrón*, un modelo que mejora los conceptos introducidos por McCulloch y Pitts, agregando además un método de aprendizaje. El *Perceptrón* introduce el concepto de pesos numéricos y un patrón de interconexión especial. En el modelo original de Rosenblatt las unidades de cálculo son elementos de paso y la conectividad es determinada de manera estocástica. El aprendizaje toma lugar al adaptar los pesos de la red con un algoritmo numérico.

El trabajo realizado por Minsky y Papert (1969) demostró los límites de los perceptrones como unidades de aprendizaje, lo que abrió el camino al desarrollo de mejoras novedosas en distintos campos que trataran de minimizar las limitantes de las redes de perceptrones, así como reducir por un tiempo la cantidad de trabajo de investigación dentro del campo.

Muchos investigadores se han dedicado a distintas líneas de trabajo en los últimos 30 años, tales como: reglas de aprendizaje aplicables a redes neuronales artificiales de alta complejidad desarrolladas por Dreyfus (1962), Bryson y Ho (1969) y popularizadas por McClelland y Rumelhart (1986), basadas en descenso de gradiente; aprendizaje basado en métodos probabilísticos, aleatorios o estocásticos

como mejora a ciertos problemas de descenso de gradiente desarrollados por Ackley, Hinton y Sejnowski (1985) y otros; resultados teóricos para entender las capacidades de redes no triviales por Cybenko(1988), Valiant (1985) y Baum y Haussler(1988); sistemas híbridos combinando redes neuronales artificiales y componentes simbólicos como el trabajo de Gallant (1985); nuevas estructuras y patrones de conexión desarrollados por Hopfield (1985), Kohonen (1988) y otros en el campo de mapas de auto-organización (self-organizing maps) y memoria asociativa [13].

En las últimas décadas, el rápido avance de las redes neuronales fue determinado por los desarrollos dentro del campo de la estadística computacional, principalmente la de los grupos de reconocimiento de patrones y de clasificación; dentro del estudio de la estadística de redes completamente conectadas con conexiones simétricas; así como también por los avances dentro de la teoría de control, proceso de señales y en menor grado por la facilidad y bajo costo actual de implementar las redes neuronales en hardware.

2.1.3 Fundamentos.

En la literatura existen varias definiciones generales de los que es una red neuronal. Al iniciar este capítulo dimos una de ellas: "*Las redes neuronales artificiales son un intento por modelar las capacidades de los sistemas nerviosos.*". Esta definición inició la primera explosión dentro de la investigación de las redes neuronales artificiales en las décadas del 50 y del 60.

Existen otras definiciones de lo que representa una red neuronal artificial. Desde el punto de vista matemático y estadístico una red neuronal artificial es un grafo dirigido que realiza trasformaciones de ciertos patrones representados como valores numéricos incrustados en sus nodos, de un espacio a otro a través de simples algoritmos de transmisión de mensajes [9].

Desde el punto de vista computacional podemos definirla como una serie de arquitecturas que permiten resolver problemas complejos a través de unidades de

cómputo sencillas y conexiones masivas en paralelo adecuadamente pesadas entre ellas, es decir la implementación de un modelo paralelo distribuido.

Esta ultima definición nos permitirá describir a continuación algunos conceptos importantes, normales a todas (o casi todas) las redes neuronales artificiales propuestas hasta el momento no importando su arquitectura.

En toda red neuronal artificial se distinguen los siguientes aspectos [10]:

- Unidades de proceso, denominadas neuronas, células, nodos o simplemente unidades.
- Cada unidad tiene un único valor de salida o_i, también denominado estado de activación, independientemente de su posición.
- Conexiones entre las unidades, definidas por pesos w_{ij} que determinan el efecto de cada señal proveniente de una unidad i en la unidad j.
- Reglas de propagación s_i y funciones de activación F_i que determinan la salida de la unidad de proceso de acuerdo a las entradas recibidas.
- Métodos de recopilación de la información o reglas de aprendizaje. En este punto se pueden dividir en métodos supervisados y no supervisados.

2.1.3.1 Unidades de Proceso.

Cada unidad de proceso dentro de una red neuronal artificial realiza el trabajo de recibir las salidas de las unidades vecinas de acuerdo a su topología y utilizarlas para calcular una señal de salida que será propagada a otras unidades, inclusive hacia ella misma.

Es preciso diferenciar tres tipos de unidades:

- Unidades de entrada, que reciben datos del exterior, estas unidades normalmente se consideran como unidades de paso ya que únicamente transfieren dichas entradas hacia el interior de la red.
- Unidades de salida, que envían datos fuera de la red.
- Unidades ocultas, que reciben entradas y envían salidas a unidades dentro de la red.

- Unidades de sesgo (*offset* ó *bias*), estas unidades siempre están activas, es decir su valor es siempre 1 y permiten incluir en la arquitectura los valores tope de activación o de umbral de una unidad fuera de la misma como un peso más.

Cada unidad puede activarse de manera síncrona o asíncrona, es decir simultáneamente dentro de una unidad de tiempo t o de acuerdo a cierta probabilidad usualmente fija.

Una característica común que hay que tomar en cuenta es que todas las entradas a la unidad llegan a ella al mismo tiempo o quedan activas lo suficiente para que el cómputo de la salida pueda ocurrir.

2.1.3.2 Reglas de propagación.

En la mayoría de las arquitecturas de redes neuronales artificiales las funciones de propagación son simplemente la suma pesada de las entradas recibidas lo que se puede expresar con la siguiente ecuación:

$$s_i(t) = \sum_{i=1}^{n+1} w_{ij}(t) o_j(t) \tag{2.1}$$

En la expresión anterior $o_j(t)$ representa la salida de alguna otra unidad $j-ésima$ de la red con conexión a la unidad actual i. La unidades que utilizan esta ecuación se les denomina *unidades sigma*.

Contribuciones de pesos positivos se consideran contribuciones *excitatorias* y de pesos negativos contribuciones *inhibitorias*.

Existe otro tipo de reglas de propagación introducidas por Feldman y Ballard [21]. Estas reglas no toman la suma pesada de los entradas, por el contrario las multiplican de acuerdo a la siguiente expresión:

$$s_i(t) = \sum_j w_{ij}(t) \prod_m y_{jm}(t) \tag{2.2}$$

En la ecuación anterior y_{jm} puede ser pesado antes de realizar la multiplicación. Las unidades que implementan esta función se conocen como *unidades sigma-pi*.

2.1.3.3 Funciones de activación.

La función de activación nos dirá el grado de excitación de la unidad de proceso al estímulo enviado por otras unidades.

De forma general una unidad determina su grado de activación directamente de las entradas de acuerdo a la siguiente expresión produciendo un nuevo valor en un tiempo t determinado:

$$o_i(t) = F_i(o_i(t-1), s_i(t)) \tag{2.3}$$

La expresión anterior permite que la activación de una unidad sea también función de su propia salida en un paso anterior, pero en la mayoría de los casos la activación es únicamente función de las entradas por lo que la ecuación anterior se modifica resultando la siguiente expresión:

$$o_i(t) = F_i(s_i(t)) \tag{2.4}$$

La selección de una determinada función de activación F_i dependerá de algunos factores, a saber: a) el tipo de red a utilizar; b) la función realizada por la unidad y c) la interpretación que se requiera de la salida de la red.

Existen numerosas funciones de activación; en realidad cualquier función puede ser utilizada para activar una unidad; a continuación las más populares:

a) Función lineal. Básicamente representa una función de paso: todo lo que entra a la unidad sale. Esta función es una de las más utilizadas puesto que es la que interviene en las unidades de entrada y esta expresada por la ecuación:

$$F_i(s_i(t)) = s_i(t) \tag{2.5}$$

b) Función de umbral o paso. Esta función permite dos estados estables y queda expresada por la siguiente ecuación:

$$F_i(s_i(t)) = \begin{cases} a & \text{si } s_i(t) < \theta \\ b & \text{si } s_i(t) > \theta \end{cases} \tag{2.6}$$

c) Función sigmoidea. Esta función también cuenta con dos estados estables pero además presenta una zona de transición suave y es continua, por lo

que es diferenciable en cualquier punto. Las dos expresiones mayormente utilizadas son las siguientes:

$$F_i(s_i(t)) = \frac{1}{1+e^{-s_i(t)}} \tag{2.7}$$

y

$$F_i(s_i(t)) = \frac{1-e^{-s_i(t)}}{1+e^{-s_i(t)}} \tag{2.8}$$

Conocidas como sigmoidea y sigmoidea simétrica. La ecuación (2.8) se recomienda pues presenta algunas ventajas en el aprendizaje [20]

d) Función Gaussiana. Esta función es comúnmente utilizada dentro de unidades de clasificación y actúa como un filtro, dejando pasar únicamente aquellas entradas deseadas, a la vez que permite interpretar el grado de pertenencia de cada entrada en la unidad. La ecuación general es la siguiente:

$$F_i(s_i(t)) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{s_i(t)-\mu}{\sigma}\right)^2\right] \tag{2.9}$$

Esta función es una de las más utilizadas en las redes de funciones de base radial (RBF).

En la Figura 2.1 se muestran las gráficas de estas funciones.

2.1.4 Arquitecturas de Redes Neuronales Artificiales.

Una única unidad de proceso es insuficiente para resolver cualquier problema práctico, por lo que es necesario agrupar las unidades en arquitecturas o topologías. La manera en la que los nodos se conectan entre si e interactúan es una parte importante de la implementación inicial de los modelos neuronales.

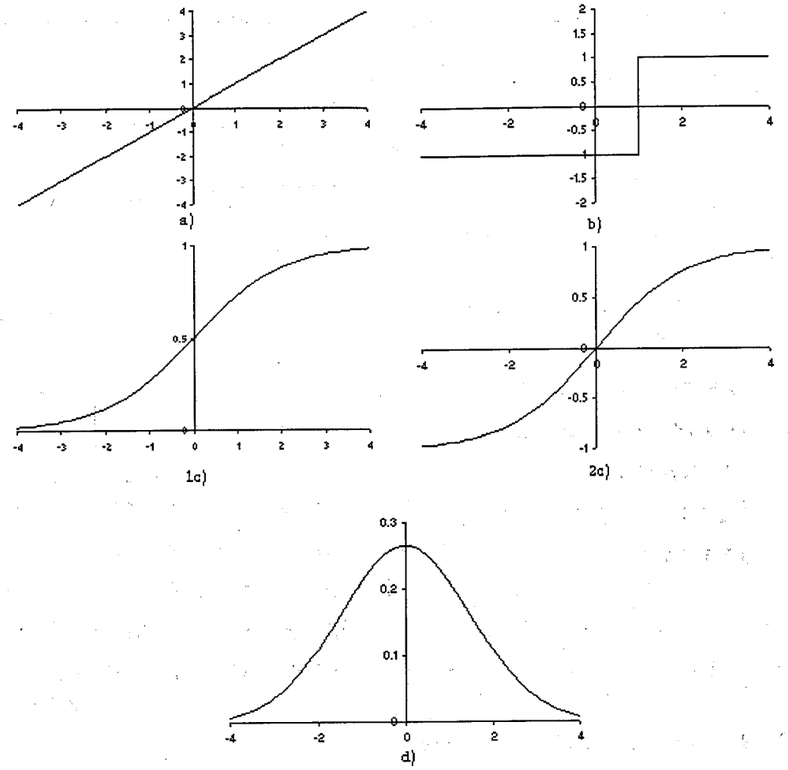


Figura 2.1. Gráficas de las distintas funciones de activación. a) Función lineal. b) Función de umbral. 1c) Función sigmoidea. 2c) Función sigmoidea simétrica. d) Función Gaussiana.

Existen numerosas arquitecturas de redes neuronales artificiales, algunas se derivan de los estudios de los sistemas nerviosos y otras de las características de los problemas que intentan resolver.

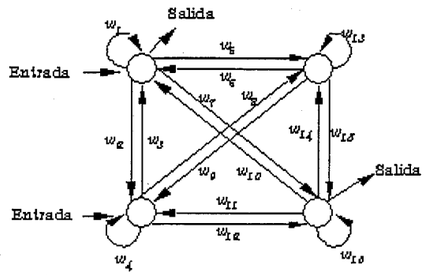


Figura 2.2. Una red completamente conectada.

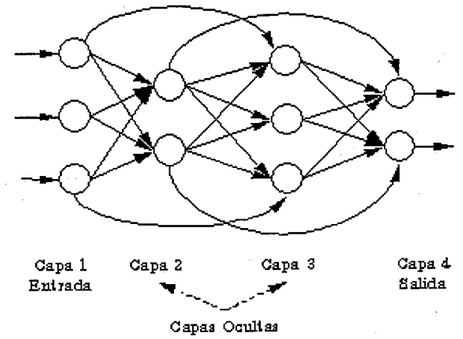
2.1.4.1 *Redes completamente conectadas.*

La arquitectura más general posible de redes neuronales artificiales es aquella en la que cada unidad esta conectada a cada unidad, teniendo en este caso tanto conexiones excitatorias como inhibitorias e incluso conexiones iguales a cero. Esta arquitectura se muestra en la Figura 2.2 y todas las demás arquitecturas pueden lograrse a partir de ella al asignar valores nulos a algunas de las conexiones.

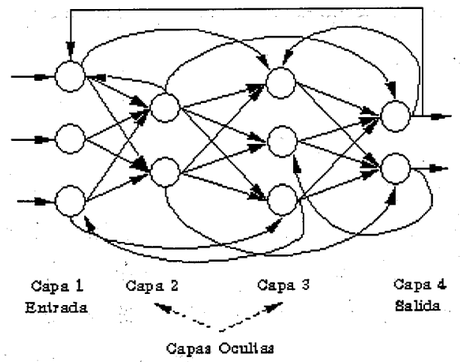
2.1.4.2 *Redes de múltiples capas*

Estas son redes en las que las unidades de las que se compone están divididas en partes separables denominadas capas.

Cada capa realiza una función específica dependiendo de los resultados que se desean y las unidades empleadas por lo que en una red de múltiples capas las funciones de activación y propagación pueden ser iguales o estar mezcladas de tal manera que el comportamiento de cada capa sea homogéneo.



a) Red Feed-Forward



b) Red Recurrente

Figura 2.3. Ejemplos de redes de múltiples capas. En a) una red del tipo *feed-forward* con 2 capas ocultas. Como se observa no existen conexiones dentro de una misma capa. En b) se muestra un tipo de red de múltiples capas recurrente, aún cuando no aparecen en la figura las conexiones dentro de una misma capa son válidas

Ejemplos de esta arquitectura se presentan en la Figura 2.3.

Las redes de múltiples capas se pueden dividir de acuerdo a los tipos de conexiones que presentan en:

- Redes *feed-forward,* donde el flujo de datos se da en una única dirección, esto es, no existen conexiones entre unidades de la capa i a la capa j si se cumple que $i > j$. Esta se muestra en la Figura 2.3a. Una consideración

importante es que no se permiten conexiones dentro de la misma capa. La variante más utilizada de este tipo de redes se explica más adelante.

- Redes recurrentes, que contienen conexiones de retroalimentación. Estas redes se explican más a detalle en la sección 2.3. Su topología se muestra en la Figura 2.3b.

2.1.5 Aprendizaje en las Redes Neuronales.

En la literatura es común hacer una diferenciación entre las arquitecturas de acuerdo al tipo de aprendizaje que utilizan, principalmente en dos clases bien definidas de aprendizaje:

a) Aprendizaje Supervisado: En este caso entran aquellos métodos que requieren la presencia de parejas de ejemplos representativos de nuestro espacio de datos de entrada y de salida con los que se entrenará la red para encontrar la mejor correspondencia posible. Ejemplos de este tipo son la red neuronal con retropropagación, algunos tipo de redes recurrentes que utilicen retropropagación como BPTT y las redes de Elman y Jordan, redes de funciones de base radial, redes adaptables multicapa, redes hetero-asociativas y muchos otras arquitecturas.

b) Aprendizaje No Supervisado: En el aprendizaje no supervisado se desconoce por completo la solución exacta que debe obtener la red, por lo que construir parejas de datos es imposible. En este tipo de aprendizaje la red modifica sus pesos y conexiones de acuerdo a la información que obtiene de los datos de entrada únicamente. En este caso entran las memorias auto-asociativas, las redes de Hopfield, cuantificadores de aprendizaje vectorial (LVQ), redes de contrapropagación (counterpropagation), redes de modelos ART (Adaptive Resonance Theory), Self-Organizing Maps o redes de Kohonen, redes de análisis de componentes principales y muchos otras arquitecturas que se utilizan en aplicaciones de agrupamiento principalmente.

Muchos de los modelos que se presentan más adelante en el capítulo utilizan alguno de estos dos modelos de aprendizaje para realizar su trabajo satisfactoriamente.

2.2 Redes Neuronales con Retropropagación.

Una red neuronal con retropropagación (RNR, BPN en inglés) es una red de múltiples capas del tipo *feed-forward* entrenada con el algoritmo de retropropagación. Este método numérico fue usado por distintas comunidades de desarrollo en diferentes campos y fue descubierto y redescubierto hasta que en 1985 encontró su camino dentro del grupo de inteligencia artificial a través de los trabajos del grupo de procesamiento paralelo distribuido [20].

Una RNR cae dentro del conjunto de redes neuronales con aprendizaje supervisado por lo que requieren parejas de ejemplos representativos para el entrenamiento como se explicó en otro punto de este capítulo.

La topología general se muestra en la Figura 2.3a.

2.2.1 Teoría.

Si recordamos una de las definiciones expresadas previamente, podemos decir que una RNR tiene la función de obtener la correspondencia entre dos espacios, el espacio de los datos de entrada y el de los datos de salida. Demostraciones teóricas realizadas por Cybenko (1989), Funahashi (1989), Hortnik et al. (1989), Carroll y Dickinson (1989), Stinchcombe y White (1989) y muchos otros autores posteriores nos muestran que cualquier función continua uniforme en un compacto puede ser aproximada por una red neuronal con una única capa oculta utilizando funciones de activación no-lineales (sigmoideas por ejemplo), denominándose a este teorema como el *teorema de la aproximación universal.* Una demostración muy sencilla de dicho teorema se puede encontrar en [19].

Utilizando este teorema muy importante podemos concluir que el utilizar una única capa oculta es suficiente para cualquier problema de correspondencia que se quiera resolver.

Al realizar la propagación de los datos dentro de una RNR con una única capa oculta se obtendrán unas salidas $o^{(2)}$ de acuerdo a la siguiente expresión:

$$o_k^{(2)} = F_k^{(2)}\left(\sum_j w_{jk}^{(2)} F_j^{(1)}\left(\sum_i w_{ij}^{(1)} x_i + \alpha_i^{(1)}\right) + \alpha_j^{(2)}\right) \tag{2.10}$$

En la anterior ecuación $F^{(1)}$ y $F^{(2)}$ corresponden a las funciones de activación de las capas oculta y de salida respectivamente, $w^{(1)}$ y $w^{(2)}$ corresponden a los pesos de la capa de entrada a la oculta y de la capa oculta a la de salida respectivamente, x_i representa la entrada a la red y por último α_i y α_j representan los sesgos de las unidades de las capas oculta y de salida.

Dentro del análisis teórico de esta expresión se ha encontrado que puede ser interpretado como un caso especial de regresión proyectiva persecutoria por lo que mucha de la teoría de este campo puede aplicarse a las redes neuronales con retropropagación [19].

Saber que podemos aproximar cualquier función con una red con una sola capa oculta, y por tanto realizar la correspondencia requerida, no es de gran ayuda sin encontrar alguna manera de poder conocer la correcta relación de pesos que haga el trabajo. Una forma sencilla de resolver esto es tratar este problema como un problema de optimización y utilizar para esto mínimos cuadrados. Esta es la aproximación tomada por la escuela de Rumelhart y McClelland [19].

En un problema de optimización por mínimos cuadrados tendremos la siguiente ecuación a minimizar para una serie de ejemplos p cualquiera:

$$E(w) = \frac{1}{2}\sum_{p=1}^{n} \left\| f(x^p, w) - t^P \right\|^2 \tag{2.11}$$

Dados (x,t) los ejemplos que me describen una función a aproximar por nuestra red neuronal y $f(x^p,w)$ representando la ecuación (2.10) de salida de una red neuronal en función de las entradas y los pesos.

Entonces la actualización de los pesos w para la iteración $t+1$ puede darse utilizando una forma de descenso de gradiente por lo que se tendrá la siguiente expresión:

$$w^{t+1} = w^t - \eta \frac{\partial E}{\partial w} \tag{2.12}$$

con η el tamaño de paso y w^t los pesos en la actual iteración.

En este caso requerimos las derivadas parciales del error con respecto a todos los pesos de la red.

A continuación presentamos el algoritmo de retropropagación como una manera de obtener las derivadas parciales de los pesos y de esta manera obtener la combinación de pesos requerida en nuestro problema de correspondencia.

2.2.2 El algoritmo de Retropropagación (Backpropagation).

Uno de los métodos de aprendizaje más utilizado dentro del grupo de redes neuronales artificiales es el método denominado de *Retropropagación* o en inglés *Backpropagation* (BP). El método BP es utilizado para obtener una combinación correcta de pesos en una red neuronal artificial del tipo feed-forward tal que minimice una función de error. La base del método es la retropropagación del error a través de la red para encontrar las derivadas parciales necesarias para realizar dicha minimización.

Al observar la ecuación (2.10) podemos ver que en la etapa de presentación estamos realizando una composición de funciones cuyo parámetro desconocido son los pesos w, la parte de retropropagación es entonces aplicar la regla de la cadena del cálculo para cada rama de la red y de esta manera obtener las derivadas parciales de la ecuación con respecto a dichos pesos.

Consideremos $w_{ij}^{(1)}$ y $w_{jk}^{(2)}$, los pesos entre la capa de entrada y la oculta y entre la capa oculta y la de salida respectivamente, por separado para facilitar la explicación.

Para el caso de los pesos $w_{jk}^{(2)}$, la derivada parcial $\partial E/\partial w_{jk}^{(2)}$ será dependiente únicamente de $w_{jk}^{(2)}$ a través de la salida $o_k^{(2)}$ por lo que es necesario inicialmente obtener la derivada del error con respecto a cada salida. Si el error es calculado con la expresión (2.11) entonces su derivada será:

$$\frac{\partial E}{\partial o_k^{(2)}} = \left(o_k^{(2)} - t_k\right) \tag{2.13}$$

Ahora habrá que diferenciar cada salida con respecto a los pesos. Esta derivada se obtiene a través de las funciones de propagación por lo que utilizando la regla de la cadena y sabiendo que $\partial o_k^{(2)}/\partial s_k^{(2)} = F_k'(s_k^{(2)})$ se tendrá:

$$\frac{\partial o_k^{(2)}}{\partial w_{jk}^{(2)}} = \frac{\partial o_k^{(2)}}{\partial s_k^{(2)}} \frac{\partial s_k^{(2)}}{\partial w_{jk}^{(2)}} = F_k'(s_k^{(2)}) o_j^{(1)} \tag{2.14}$$

Ahora aplicando la regla de la cadena nuevamente para obtener las derivadas buscadas $\partial E/\partial w_{jk}^{(2)}$ tendremos:

$$\frac{\partial E}{\partial w_{jk}^{(2)}} = \frac{\partial E}{\partial o_k^{(2)}} \frac{\partial o_k^{(2)}}{\partial s_k^{(2)}} \frac{\partial s_k^{(2)}}{\partial w_{jk}^{(2)}} = (o_k - t_k) F_k'(s_k^{(2)}) o_j^{(1)} \tag{2.15}$$

En ambas ecuaciones $o_j^{(1)}$ son las salidas de las unidades de la capa oculta, además $s_k^{(2)}$ son las funciones de propagación de las unidades en la capa de salida y tiene la expresión mostrada en la ecuación (2.1).

Ahora para el caso de los pesos $w_{ij}^{(1)}$ la derivada parcial $\partial E/\partial w_{ij}^{(1)}$ será dependiente de $w_{ij}^{(1)}$, de manera similar, a través de las salidas $o_j^{(1)}$ de las unidades de la capa oculta, que a su vez aparecen en cada una de las salidas de la red $o_k^{(2)}$ por lo que utilizando la regla de la cadena tendremos:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^{(1)}} &= \sum_k \frac{\partial E}{\partial o_k^{(2)}} \frac{\partial o_k^{(2)}}{\partial s_k^{(2)}} \frac{\partial s_k^{(2)}}{\partial o_j^{(1)}} \frac{\partial o_j^{(1)}}{\partial s_j^{(1)}} \frac{\partial s_j^{(1)}}{\partial w_{ij}^{(1)}} \\ &= \sum_k (o_k - t_k) F_k'(s_k^{(2)}) w_{jk}^{(2)} F_j'(s_j^{(1)}) x_i \end{aligned} \tag{2.16}$$

Finalmente la actualización de los pesos, utilizando descenso de gradiente, se realiza con las siguientes expresiones finales:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} x_i \tag{2.17}$$

$$\frac{\partial E}{\partial w_{jk}^{(2)}} = \delta_k^{(2)} o_j^{(1)} \tag{2.18}$$

Con

$$\delta_k^{(2)} = \left(o_k^{(2)} - t_k\right) F_k'(s_k^{(2)}) \tag{2.19}$$

el error retropropagado hasta la capa de salida y

$$\delta_j^{(1)} = F_j'(s_j^{(1)}) \sum_k w_{jk}^{(2)} \delta_k^{(2)} \tag{2.20}$$

el error retropropagado hasta la capa oculta.

En un resumen final el algoritmo BP consta de 3 etapas [20] :

1. Etapa *feed-forward* o etapa de presentación. Se alimenta la red con una pareja de ejemplos y se calcula el error de mínimos cuadrados dada la salida obtenida.
2. Etapa *backpropagation* (retropropagación) . Se calculan las derivadas parciales de la capa de salida a la oculta y de la oculta a la de entrada.
3. Etapa de actualización: Se actualizan los pesos de acuerdo a la ecuación (2.12).

2.2.2.1 Retropropagación desde el punto de vista matricial.

Una manera muy sencilla de presentar el algoritmo BP completo es a través de operaciones matriciales [20].

Iniciemos nombrando a $\overline{W}_1$ como la matriz de $(n+1)\times k$ pesos que conectan cada una de las n unidades de entrada con cada una de las k unidades en la capa oculta y a $\overline{W}_2$ como la matriz de $(k+1)\times m$ pesos que conectan cada unidad en la capa oculta con cada una de las m unidades de salida. El renglón adicional en estas matrices corresponde a los valores de los sesgos de cada unidad oculta y de salida.

Ahora teniendo el vector renglón $\partial = \{x,1\}$ el vector extendido de entrada de la red, con x el vector renglón de entrada original, y el vector renglón $\partial^{(1)} = \{o^{(1)},1\}$ el vector extendido de salida de las unidades de la capa oculta, con $o^{(1)}$ el vector renglón de salidas original, la excitación de una unidad oculta, así como la de una unidad de salida utilizando unidades de propagación *sigma* se pueden calcular por:

$$o^{(1)} = F^{(1)}(\partial \overline{W}_1) \tag{2.21}$$

y por

$$o^{(2)} = F^{(2)}(\partial^{(1)} \overline{W}_2) \tag{2.22}$$

donde $F^{(1)}$ y $F^{(2)}$ son funciones vectoriales de activación: $F(\bar{x}) = (F_1(x_1), F_2(x_2), \ldots)$ donde F_i está dado por las ecuaciones (2.7) ó (2.8).

Las derivadas almacenadas en el paso *feed-forward* de las k unidades ocultas y las m unidades de salida se pueden escribir como dos matrices diagonales $D_1^{(k \times k)}$ y $D_2^{(m \times m)}$.

Ahora en el paso de retropropagación no ocupamos los sesgos por lo que las matrices aumentadas $\overline{W}_1$ y $\overline{W}_2$ se convierten en las matrices W_1 y W_2 con únicamente los pesos.

Definimos al vector e de las derivadas de las desviaciones cuadradas como:

$$\mathbf{e} = \begin{pmatrix} (o_1^{(2)} - t_1) \\ (o_2^{(2)} - t_2) \\ \vdots \\ (o_m^{(2)} - t_m) \end{pmatrix} \tag{2.23}$$

Con esto el vector del error propagado hasta la capa de salida esta dado por:

$$\boldsymbol{\delta}^{(2)} = \mathbf{D}_2 \mathbf{e} \tag{2.24}$$

y el error hasta la capa oculta por:

$$\boldsymbol{\delta}^{(1)} = \mathbf{D}_1 \mathbf{W}_2 \boldsymbol{\delta}^{(2)} \tag{2.25}$$

Entonces las correcciones para las matrices $\overline{W}_1$ y $\overline{W}_2$ de pesos están dadas por:

$$\Delta \overline{\mathbf{W}}_2^T = -\gamma \boldsymbol{\delta}^{(2)} \partial^{(1)} \tag{2.26}$$

y

$$\Delta \overline{\mathbf{W}}_1^T = -\gamma \boldsymbol{\delta}^{(1)} \partial \tag{2.27}$$

donde γ es el tamaño de paso del paso de descenso de gradiente.

Como se puede observar el método de retropropagación visto de esta manera ocupa únicamente operaciones como multiplicaciones vector-matriz, vector-vector y matriz-vector por lo que se consideró como base para la implementación de los algoritmos de la tesis.

2.2.3 Algoritmos de aceleración.

Uno de los problemas asociados al algoritmo de Retropropagación es su lenta convergencia en la mayoría de los problemas en los que se implementa.

Existen numerosas técnicas de aceleración para las redes que utilizan descenso de gradiente para minimizar su función de error, la gran mayoría son técnicas desarrolladas dentro del ámbito de la optimización y que han sido ampliamente probadas dentro del campo de las redes neuronales satisfactoriamente.

2.2.3.1 Rprop.

Una variante al algoritmo de Silva y Almeida es el algoritmo Rprop.

La idea central en el algoritmo es la de actualizar los pesos de la red usando únicamente la taza de aprendizaje y el signo de la derivada parcial de la función de error con respecto a cada peso. Este procedimiento acelera el aprendizaje principalmente en regiones planas de la función de error así como cuando la iteración se acerca a un mínimo local.

Para evitar acelerar o desacelerar mucho se utilizan valores mínimo γ_{min} y máximo γ_{max} de la taza de aprendizaje.

Las tazas de aprendizaje son actualizadas en cada iteración k de la siguiente forma:

$$\gamma_i^{(k+1)} = \begin{cases} \min(\gamma_i^{(k)} u, \gamma_{\max}) & si \quad \nabla_i E^{(k)} \cdot \nabla_i E^{(k-1)} > 0 \\ \max(\gamma_i^{(k)} d, \gamma_{\min}) & si \quad \nabla_i E^{(k)} \cdot \nabla_i E^{(k-1)} < 0 \\ \gamma_i^k & si\ no \end{cases} \tag{2.28}$$

Aquí las constantes u y d satisfacen la condición de que $u > 1$ y $d < 1$. Cuando $\nabla_i E^{(k)} \cdot \nabla_i E^{(k-1)} \geq 0$ los pesos se actualizan con la siguiente expresión:

$$\Delta^{(k)} w_i = -\gamma_i^{(k)} \operatorname{sgn}(\nabla_i E^{(k)}) \tag{2.29}$$

de otra manera $\Delta^{(k)} w_i$ y $\nabla_i E^{(k)}$ se colocan a cero.

2.3 Redes Recurrentes.

Una red neuronal artificial recurrente es aquella en la que conexiones de retroalimentación se implementan entre capas. En su forma más simple, las redes recurrentes tienen retroalimentación solamente de la capa de salida hacia la capa de entrada. El denominador común en todas las redes recurrentes con múltiples capas es la función de las conexiones de retroalimentación: se usan como indicadores del estado agregándole a la red la habilidad de anticipar el siguiente estado a partir del actual [23].

La motivación detrás de la exploración reciente de arquitecturas con recursividad dentro del campo de las redes neuronales es el potencial que presentan para tratar problemas complejos. Las redes recurrentes son capaces de llegar a una solución que satisface muchas restricciones, como en un sistema de visión que se estabiliza en una interpretación que satisface mayoritariamente complejos conjuntos de restricciones conflictivas [17].

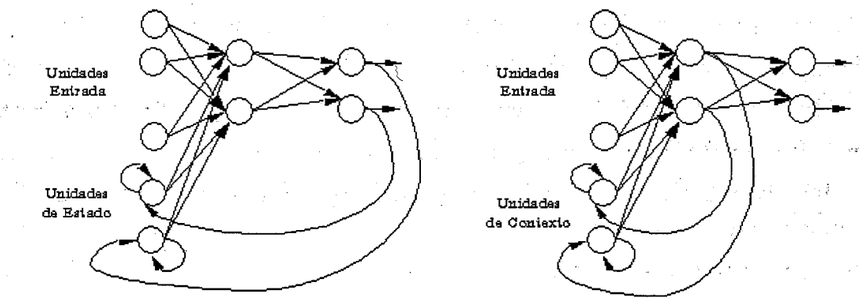
Una red neuronal artificial del tipo *feed-forward*, dado su comportamiento; el convertir una entrada dada en una salida específica; tiene el problema de desaprender las entradas previamente mostradas cuando recibe alguna entrada conflictiva, siendo en este caso sumamente difícil llegar a una aproximación relativamente buena de las salidas. Un vector, patrón o entrada conflictiva es aquella que presenta una semejanza muy significativa con otra entrada pero cuya

salida es completamente distinta. La utilización de recurrencia permite eliminar parte de este problema al incluir un componente dinámico al problema de la correspondencia.

Además otros problemas como series de tiempo, aprendizaje no supervisado, control en el tiempo, asociación de patrones y problemas de búsqueda pueden ser mucho mejor representados y resueltos con modelos recurrentes que con los modelos *feed-forward*.

Existen diferentes tipos de redes recurrentes; aquellas que están basadas en la búsqueda de algún punto de atracción, esto es un punto de estabilidad, por lo que se actualizan repetidamente hasta alcanzar dicho punto y una vez que se alcanza éste se adaptan sus pesos; y aquellas que usan los algoritmos de aprendizaje después de cada propagación, por lo que las conexiones recursivas en ellas pueden considerarse como simples entradas extra a la red.

A continuación daremos un vistazo a algunos tipos de redes recurrentes.



a) Red Jordan **b) Red Elman**

Figura 2.4. Ejemplos de redes a) Jordan y b) Elman.

2.3.1 Redes con Retropropagación Recurrentes.

Como se ha demostrado en la literatura, principalmente en el trabajo de Rumelhart [21], toda red completamente conectada como la mostrada en la Figura 2.2, y por tanto recurrente, puede ser representada como una red del tipo *feed-forward*, al extenderla en cada unidad de tiempo t. Este desdoble de una red recurrente

permite la aplicación del algoritmo de retropropagación, denominándosele como *Retropropagación a través del tiempo* (BPTT en inglés).

Un tipo especial de redes recurrentes que consideran la recursividad únicamente durante valores discretos de t y además utilizan retropropagación son las redes propuestas por Jordan [8] y Elman [4].

La red Jordan presenta conexiones directas desde la salidas de la red a unas unidades dentro de la capa de entrada denominadas de estado, teniendo los pesos de dichas conexiones fijos en la unidad. El aprendizaje se realiza para toda unidad excepto para las unidades de estado. Esta red se considera como un caso especial de una metodología conocida como *Teacher Forcing*.

La red Elman presenta retroalimentación directamente de las unidades ocultas hacia la capa de entrada a través de una serie de unidades denominadas de contexto, por lo que se puede considerar como una versión discreta de BPTT en la que la historia temporal no cuenta más allá de unos poco pasos a juicio del diseñador de la red. Esto hace que se considere a las redes de Elman como versiones *on-line* del algoritmo BPTT.

Una discusión profunda de estas arquitecturas y del aprendizaje dentro de las redes recurrentes en general se puede encontrar en el trabajo de Pearlmutter [17]

Una representación gráfica de dichas redes se puede observar en la Figura 2.4

2.3.2 Redes Asociativas.

En la literatura este tipo de redes también se les denomina como *Memorias Asociativas*. Aún cuando la arquitectura de estas redes es claramente recurrente, en muchos textos se les considera como modelos que por su importancia merecen ser considerados a parte de las redes recurrentes.

El objetivo de estos modelos es *asociar* un conjunto conocido de vectores de entrada con algún conjunto de vectores de salida, y además que realicen el mapeo de una vecindad de algún vector de entrada exactamente a un vector salida específico.

Matemáticamente esto es que si tenemos una función de mapeo Φ, resultante de entrenar la memoria asociativa, de x a y tal que $\Phi(x_i) = y_i$ dado un x_i del conjunto de datos y se presentan un nuevo valor x^* tal que $\|x^* - x_i\|^2 < \varepsilon$ entonces la salida de una memoria asociativa será $\Phi(x^*) = y_i$.

Esta propiedad le confiere a las redes asociativas la capacidad de asociar o memorizar dentro sus conexiones los patrones exactos que permiten tal correspondencia de tal manera que datos de entrada ruidosos serán asociados con un dato de salida específico y no necesariamente con una aproximación continua cercana a algunos de los datos de salida utilizados en el entrenamiento como sucede en las redes convencionales.

Las memorias asociativas pueden implementarse utilizando redes que utilicen retroalimentación o no, pero la primera aproximación da mejor resultado [20].

La mayoría de las arquitecturas que implementan las funciones de las memorias asociativas utilizan un tipo de regla de aprendizaje conocido como *aprendizaje de Hebb. (Hebbian Learning HL)*.

La idea dentro del HL es que dos neuronas que se activan simultáneamente bajo la acción de cierto vector de entrada deberían desarrollar cierto grado de interacción mucho más grande que aquellas neuronas cuyas actividades no están correlacionadas, además tendiendo esta última interacción a ser muy pequeña o cero [20].

Utilizando esto los pesos de una memoria asociativa se actualizan utilizando la siguiente regla:

$$\Delta w_{ij} = \gamma x_i y_j \tag{2.30}$$

Con (x_i, y_j) las parejas de datos que se requieren aprender, y γ una constante de aprendizaje. El aprendizaje mediante la regla de *Hebb* funciona muy bien si los vectores a entrenar son ortogonales.

Las memorias asociativas se pueden dividir de acuerdo a la función que realizan en:

a) Memorias Hetero-asociativas: Aquellas que realizan la correspondencia tal y como se explicó anteriormente entre parejas de datos. En este tipo caen aquellas que realizan reconocimiento de patrones.

b) Memorias Auto-asociativas: Aquellas en las que la salida y_j es la misma entrada x_i. Estas memorias tienen la función de corregir entradas ruidosas.

A continuación describimos brevemente algunas de las arquitecturas de redes asociativas más utilizadas. Discusiones más profundas de estas arquitecturas, así como muchas de las demostraciones de sus propiedades se pueden encontrar en [20][13]y [23]así como en los trabajos específicos de Hopfield(1982) y Hinton y Sejnowski(1985).

2.3.2.1 Redes BAM.

Las redes BAM se refieren a las memorias asociativas bidireccionales. Son una de las arquitecturas más sencillas de memoria hetero-asociativa. Un ejemplo de una red BAM se presenta en la Figura 2.5a. La función de activación utilizada en las redes BAM es la función signo y la información es codificada utilizando valores bipolares (-1,1).

El funcionamiento de una red BAM consiste en dejar iterar un cierto numero de iteraciones t, iniciando con un vector de entrada x_i arbitrario escogido al azar, que se propaga hacia delante y atrás a través de las conexiones bidireccionales, hasta que la red se estabiliza en un punto encontrando el correspondiente vector complementario y_i. Las redes BAM normalmente se entrenan utilizando la regla de *Hebb*.

Una característica muy importante de las redes BAM es que la manera en como se estabiliza la red al final de la presentación de un vector se puede representar como la minimización de una función de energía o función de Lyapunov, por lo que los pesos pueden expresarse como un mapa de energía [23].

2.3.2.2 Redes Hopfield.

Las redes de Hopfield son un tipo de memorias auto-asociativas, propuesta por John Hopfield, en los que los valores de los nodos son actualizados iterativamente en base un principio de cómputo local: el nuevo estado de cada nodo depende únicamente de sus entradas pesadas en un tiempo determinado. Es una red completamente conectada que se presentan en la Figura 2.5b. y sus pesos se actualizan utilizando la ecuación (2.30) de la regla de *Hebb* [20].

Existen diferentes versiones de las redes de Hopfield, aquellas que utilizan unidades con funciones de activación binarias y aquellas que tienen salidas reales.

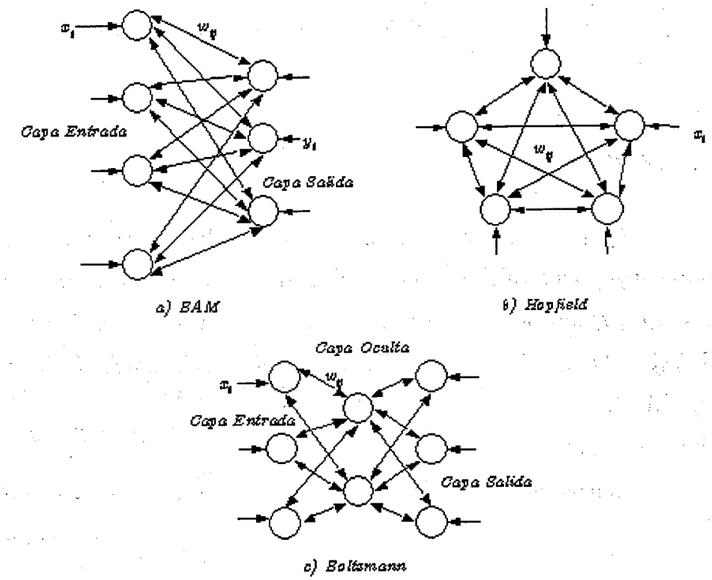


Figura 2.5. Ejemplos de redes asociativas. a) Memoria asociativa bidireccional. Las conexiones entre los nodos pueden ser simétricas o no dependiendo de la aplicación. b) Red de Hopfield con 5 unidades, como se puede observar no existen conexiones de las unidades consigo mismas. c) Máquina de Boltzmann para el problema de codificación 3-2-3 aquí el vector de entrada es el mismo que el vector de salida esperado.

El funcionamiento de una red Hopfield es asíncrona, es decir solo un nodo actualiza su estado en una iteración determinada. Normalmente el nodo a actualizar es escogido al azar.

La matriz de pesos de una red Hopfield se pueden representar como una matriz simétrica de $n \times n$, donde n es el número de nodos en la red, con los valores de la diagonal iguales a cero, lo que garantiza que al dejar correr una red Hopfield esta llegará a un estado estable.

Al igual que las redes BAM la forma en como se estabiliza una red Hopfield da lugar a una función de energía, en este caso cuadrática, la cual puede utilizarse para realizar minimizaciones de funciones que tengan la misma estructura como por ejemplo el problema del agente viajero. La ecuación de energía de una red Hopfield es la siguiente:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n}\theta_i x_i \qquad (2.31)$$

2.3.2.3 Máquinas de Boltzmann.

En 1985, Ackley, Hinton y Sejnowski generalizaron la arquitectura propuestas por Hopfield agregando unidades ocultas y utilizando una distribución conocida en el área de física estadística como la distribución de *Boltzmann* que permiten los cambios dentro de las unidades de manera estocástica. Un ejemplo de esta arquitectura se presenta en la Figura 2.5c.

Las máquinas de Boltzmann puede ser construidas con unidades binarias o bipolares, y tienen la misma función de energía que las redes Hopfield, la mayor diferencia radica en que cada una de las unidades se actualizan de manera estocástica de acuerdo a la siguiente probabilidad:

$$p_i = \frac{1}{1+\exp\left(-\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right)/T\right)} \qquad (2.32)$$

Si observamos la ecuación (2.32) podemos ver que si T es arbitrariamente pequeño una maquina de Boltzmann se comporta como una red Hopfield.

Analizando un poco la manera en como se comporta una máquina de Boltzmann podemos decir que éstas son ejemplos de procesos de Markov de primer orden, por que las probabilidades de transición solo dependen del actual estado y no de la historia del sistema[20].

Las máquinas de Boltzmann pueden ser utilizadas en casos donde otro tipo de redes asociativas quedan atrapadas en mínimos locales.

Aún teniendo estas capacidades superiores, las máquinas de Boltzmann no se comportan de mejor manera que las redes de Hopfield si la función de error tiene una forma desfavorable, por lo que son más importantes desde el punto de vista teórico que el práctico.

Capítulo 3

Redes Neuronales y su Aplicación al Procesamiento de Imágenes.

Una vez conocidos los aspectos teóricos más importantes y las arquitecturas de las redes neuronales artificiales, en este capítulo se presentan algunas de las aplicaciones que existen dentro del procesamiento de imágenes que utilizan redes neuronales.

Inicialmente se presentan algunos conceptos básicos de procesamiento de imágenes y algunas definiciones de estos conceptos. A continuación se presentan algunos de los modelos y trabajos que se han venido desarrollando en la aplicación de redes neuronales al procesamiento de imágenes, que permiten poner en perspectiva el trabajo realizado en esta tesis, así como abrir el camino para nuestra estructura y algoritmos propuestos, desarrollados en el siguiente capítulo.

3.1 Conceptos de Procesamiento de Imágenes.

Procesamiento de Imágenes, es un término utilizado para englobar diversos procesos y heurísticas capaces de extraer información comprensible de una serie de datos que forman una imagen. Entre estos procesos podemos contar aquellos que extraen bordes, encuentran líneas, segmentan, reconocen formas, eliminan ruido y reconstruyen la información contenida en una imagen o en una serie de imágenes. Además también se incluyen dentro del campo de procesamiento de imágenes

aquellos procesos que encuentran cantidad de movimiento, encuentran texturas y todos aquellos métodos capaces de extraer características interesantes de los datos.

A continuación se presentan algunas definiciones importantes dentro del ámbito del procesamiento de imágenes que son utilizadas en este trabajo.

a) Imagen.

Una imagen constituye una distribución espacial de la intensidad luminosa en un plano. Dado que las computadoras sólo pueden manejar arreglos de números digitales es necesario representar una imagen como un arreglo bidimensional de puntos en vez de una función continua de dos variables como sería en el caso general, matemáticamente hablando.

Una imagen digital, y por tanto utilizable por una computadora secuencial, está definida entonces como un arreglo bidimensional de $M \times N$ píxeles.

b) Píxel.

Se le denomina píxel a un punto de coordenadas únicas x,y incluido en la malla bidimensional de una imagen digital y representa la intensidad media en dicha posición. La posición de los píxeles se da generalmente utilizando la notación normal para matrices, con el primer valor como la posición del renglón y el segundo como la de las columnas. En el caso de esta tesis se han invertido los índices para hacer coincidir las x con las columnas y las y con los renglones.

c) Operador de Ventana.

Un operador de ventana transforma una imagen al utilizar los píxeles incluidos dentro de una región definida para calcular cada píxel de salida.

d) Segmentación.

Segmentar una imagen digital es la operación de revisar cada píxel individual para ver si pertenece a un objeto o clase en particular de interés.

Teniendo una serie de etiquetas que describen quizás los valores de intensidad medios de cada clase o el nombre de la clase, la operación de segmentación devolverá una imagen etiquetada apropiadamente de una imagen original dada. La segmentación se encuentra en la frontera entre el procesamiento de imágenes a bajo nivel y el análisis de imágenes. Existen varios tipos de segmentación entre ellos

segmentación basada en píxeles, en vecindades, en bordes, en modelos y más recientemente se ha desarrollado la segmentación paramétrica. Estos últimos modelos han demostrado ser muy robustos a problemas de ruido y dar muy buenos resultados[12].

3.2 Redes Neuronales utilizadas para segmentación.

La gran capacidad de modelado y la gran variedad de arquitecturas con propiedades diversas han hecho que las redes neuronales sean grandes candidatos para crear herramientas tendientes a resolver problemas de segmentación de imágenes en estos últimos años.

La capacidad de una red neuronal del tipo de retropropagación (BP) de ser un muy buen clasificador ha permitido el nacimiento de arquitecturas y metodologías que utilicen esto para obtener buenas clasificaciones.

La utilización de las redes BP para clasificación se ha venido utilizando en conjunción con heurísticas de pre-procesamiento y de obtención de características (features) de una imagen para obtener una serie de datos linealmente separables para utilizar entonces la red y obtener clasificaciones con tazas de error muy bajas. Todas estas implementaciones no utilizan como entrada a la red la imagen original, sino varias características de ella, como por ejemplo medidas de coherencia espacial, transformadas, etc. La gran mayoría de estas aplicaciones se han realizado dentro del ámbito de segmentación de imágenes médicas aunque también se han dado en el problema de reconocimiento de imágenes de apertura de radar (SAR). Muy buenos ejemplos de ello se pueden ver en libro de Skapura (capítulo 6) [23] y en el trabajo de Cox, et. al [3].

Otro enfoque de utilización de redes BP es el presente en el trabajo de Ossen et al [15]. En él propone una manera de ver el problema de segmentación desde el punto de vista del contexto, utilizando los valores de gris de los vecinos para inferir la clase a la que pertenece un píxel, esto es realizar una segmentación basada en texturas. Uno de los resultados más interesantes dentro de las redes neuronales es

que pueden aproximar distribuciones, más aún, que también pueden aproximar probabilidades *a posteriori*. Esto es utilizado entonces para construir una red neuronal BP que utiliza como información de entrada los píxeles dentro de una vecindad, para calcular la probabilidad *a posteriori* y obtener una clasificación para cada píxel. El usuario propone una topología básica (número de neuronas ocultas) y el número de representantes de una imagen que serán utilizados para el entrenamiento. De manera interactiva se seleccionan los representantes y se realiza el aprendizaje. Este trabajo se implementó con éxito en un sistema de visión médica.

Uno de los problemas dentro de las redes neuronales es su lento proceso de entrenamiento. Una solución a este problema, utilizando redes neuronales con unidades binarias, es el trabajo de Austin[1]. En él propone una nueva arquitectura de red que combina una red binaria con una red del tipo ADAM, que es una extensión de la red asociativa vista en la sección 2.3.2. Se alimenta esta estructura con un conjunto de patrones representativos de aquello que se requiere segmentar y se deja iterar. La parte binaria convierte el espacio de pertenencia de los patrones a uno más sencillo, esto es extrayendo las características de cada patrón. Entonces la parte asociativa "guarda" dichos patrones para obtener la clasificación, aprovechando las características de las redes asociativas de responder a patrones ruidosos. Este trabajo se probó con imágenes RS (remotely sensed data) de satélite de línea infrarroja. Aún cuando la utilización de redes binarias compromete la exactitud de la extracción de características, la rapidez de entrenamiento (pocos segundos) para grandes volúmenes de información, lo hace un muy buen candidato para aplicaciones que requieran una rápida aproximación a la segmentación para utilizarse después con métodos más robustos, tal y como apunta el autor.

La presencia de información no relevante a las clases a buscar dentro del problema de segmentación hace que el proceso sea lento. Una manera de dar la vuelta a este problema es el utilizar distintas resoluciones de la imagen, esto es conocido como segmentación de multiresolución. Este enfoque de segmentación es el utilizado por el trabajo de Osman y Blostein [14]. En el proponen una variante de

BP que va eliminando las conexiones de los píxeles redundantes a través de una modificación a la parte de entrenamiento. La red se construye con entradas correspondientes a cada píxel para cada resolución. Las unidades de entrada de píxeles que comparten un mismo espacio de imagen en sucesivas resoluciones contienen conexiones de inhibición que permiten su eliminación durante el período de entrenamiento. Esto va reduciendo la complejidad de la red hasta tener una arquitectura que incluye únicamente las partes de cada resolución que intervienen en la construcción de algún objeto que se requiera clasificar. Al utilizarlo en imágenes SAR probó ser muy eficiente, reduciendo considerablemente el número de entradas a la red y acelerando el proceso de entrenamiento para este problema.

La utilización de redes recurrentes con neuronas de código de pulso (Pulse-Coding) a sido aplicada con éxito a la segmentación por contornos en el trabajo de Weitzel et al. [27]. Las neuronas de código de pulso son unidades que modulan su salida hacia otras neuronas y ellas mismas a través de una función de potencial no lineal dependiente del tiempo. Cada neurona es una unidad dedicada a detectar las diferentes direcciones que un borde puede llegar a tener dependiendo de la información local preprocesada de los píxeles y los impulsos que recibe de sus vecinas, sincronizando así su actividad. Al final del entrenamiento cada unidad se saturará en presencia de un borde y además ayudará a sus vecinas en la dirección correcta a saturarse también, aún cuando la información que ellas reciben directamente de los píxeles no hubiera hecho que detectaran borde alguno. Los resultados de esta arquitectura novedosa han probado ser satisfactorios.

Por último, la introducción recientemente de métodos probabilísticos dentro del aprendizaje de redes neuronales ha comenzado a producir trabajos que utilizan éstas redes neuronales probabilísticas o Bayesianas en el procesamiento de imágenes, principalmente en los campos de cuantificación y segmentación. La popularidad de métodos estocásticos para el procesamiento de imágenes comienza a crecer a partir de los trabajos de Geman et al. en 1984. Dos ejemplos interesantes de redes neuronales con entrenamiento y construcción probabilística que producen

muy buenos resultados se pueden ver en los trabajos de Vehtari et al.[24] y Wang et al.[25].

3.3 Otras Aplicaciones.

La aplicación de las redes neuronales al procesamiento de imágenes no ha sido únicamente en el ámbito de problemas de segmentación.

La motivación inicial de la aplicación de las redes neuronales al procesamiento de imágenes es la solución del problema del reconocimiento de patrones para la detección de caracteres dentro de una imagen binaria o de objetos en una escena.

Uno de los primeros trabajos en este campo es el de Fukushima y su "cognitron" desarrollado inicialmente en 1975 y mejorado en trabajos posteriores hasta la introducción del neocognitron en 1988.

El neocognitron es una estructura jerárquica de capas de procesamiento, con capas internas organizadas en pares de sub-capas. Cada sub-capa se compone de series de planos de procesamiento bidimensionales, cada plano conteniendo a su vez un arreglo de elementos de procesamiento neuronales. La estructura se organiza en capas alternadas de células simples o células-S y células complejas o células-C. Cada unidad S es en esencia un detector de características (líneas, bordes, etc.) que responde en la presencia de alguna de ellas en específico en el espacio de entrada. Diferentes unidades S en la misma capa son sensitivas a la misma característica en distintas posiciones en el espacio de entrada, las posiciones son determinadas por las interconexiones existentes entre las entradas y la capa S. Las unidades C colectan las salidas determinadas por las unidades S y responden si al menos una de ellas se activa. El campo de visión de las unidades C es determinado por el constructor de la red [23].

Otros modelos desarrollados a partir del neocognitron son los modelos de atención selectiva y las redes de representación interna[23].

Algunas arquitecturas interesantes para el reconocimiento de objetos en escenas se presentan en [23]. Muchas de las aplicaciones al reconocimiento de patrones en general pueden encontrarse también en [19].

Una estructura muy interesante que no requiere la introducción de vectores de características de las imágenes para obtener un reconocimiento de los caracteres presentes en ella es el trabajo de Le Cun et al.[11]. En él presenta los resultados de aplicar una estructura de red neuronal entrenada con retropropagación directamente a los datos de las imágenes, extrayendo en el camino las características que requiere para obtener reconocimientos con altos grados de exactitud.

Otro tipo de estructuras con una base común a las redes neuronales son las máquinas de soporte vectorial que han sido aplicadas con mucho éxito a la solución del problema de reconocimiento de caracteres.

Otro tipo de redes neuronales, las redes neuronales de organización propia (self-organizing) entrenadas con retropropagación pueden ser aplicadas al problema de compresión de imágenes. También para resolver este problema se ha propuesto la implementación de redes de cálculo de componentes principales[10].

La utilización de redes Hopfield como medio para la optimización de problemas complejos ha llevado a la implementación de estas arquitecturas para solucionar algunos problemas muy complejos de procesamiento de imágenes como el encontrar la profundidad a partir de pares de imágenes estéreo y reconstrucción de imágenes[10].

Otras arquitecturas se han probado para la reconstrucción de imágenes médicas en 3D. La más utilizada hasta el momento es el mapa de organización propia (SOM) de Kohonen. Un trabajo representativo es el llevado a cabo por Reyes-Aldasoro y Aldeco [18] en el que utilizan una red neuronal SOM tanto para realizar la segmentación en 3D de una serie de imágenes médicas de resonancia magnética como la compresión de las mismas a tazas altas y relativamente poca pérdida de información.

Capítulo 4

Redes Recurrentes para Segmentación de Imágenes.

Los aspectos teóricos y el estado del arte presentados en los anteriores capítulos permitirán ahora el desarrollo de la parte central de esta tesis: la estructura y algoritmos para redes recurrentes con retropropagación para segmentar imágenes.

El capítulo comienza con una descripción de la estructura general básica que se propone para realizar el procesamiento de una imagen, después se explican a detalle los algoritmos desarrollados así como algunos métodos de inicialización utilizados.

4.1 Estructura.

Nuestra propuesta, la cual consideramos novedosa, es utilizar una sola red neuronal con múltiples capas entrenada con el algoritmo de retropropagación, incluyendo cierto grado de retroalimentación y aplicarla como un operador de ventana sobre la imagen.

Dicha red básica, del tipo retropropagación, consta de tres capas, una capa de entrada cuyo tamaño se explica más adelante, una capa oculta con n unidades y una capa de salida con un número de unidades o igual al número de clases a clasificar.

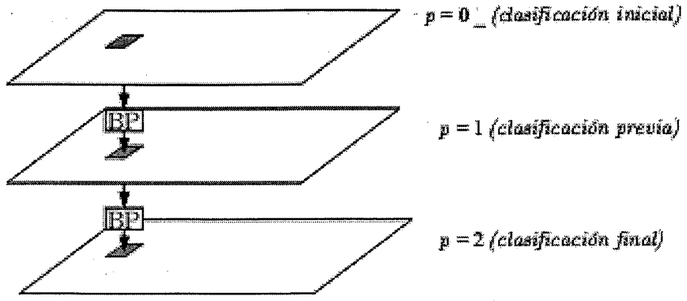


Figura 4.1. Esquema general de operación de la estructura propuesta. Cada cuadro BP es la misma red neuronal básica entrenada con retropropagación. La imagen de clasificación final se procesa como se explica más adelante en el texto para obtener la imagen de salida a presentar por el programa.

Las unidades ocultas y de salida utilizan funciones de activación sigmoideas simétricas y reglas de propagación sigma. Las unidades de entrada son simples unidades de paso.

Definiremos una *pasada* como la acción de recorrer el operador de ventana a través de la imagen y obtener un resultado de esta operación, es decir una clasificación tentativa para cada píxel.

La estructura completa da un número definido de pasadas P. Para cada pasada, se guardan las salidas $O^p_{x,y}$ correspondientes de la red neuronal, como un vector de tamaño igual al número de clases, en cada posición x,y, formando una imagen de clases I_p, para ser utilizadas como entradas en la siguiente pasada, incluyendo así un nivel de retroalimentación basado en clasificaciones sucesivas de la imagen original. La operación de la estructura general se puede observar en la Figura 4.1.

La red en una pasada p cualquiera recibe como entrada un vector de dimensión te compuesto por el valor del píxel i en la posición x,y; las coordenadas espaciales (esto es opcional) y la salidas obtenida por la red de la imagen I_{p-1} en una pasada anterior para cada píxel vecino de i dentro de una vecindad definida, incluyendo al

mismo i. Cada salida $O^p_{x,y}$ en una pasada será función, entonces, del valor del píxel en las coordenadas x,y, su posición espacial y las salidas anteriores $O^{(p-1)}_{(l,s)\in N_{x,y}}$ contenidas en la imagen I_{p-1}, donde $N_{x,y}$ es la vecindad del punto x,y.

En la estructura implementada se utilizó una vecindad de orden 1, es decir, los píxeles vecinos a x,y dentro de un cuadro de 3x3 píxeles excepto diagonales.

El número de unidades en la capa de entrada de la red básica es, por tanto, sencillo de obtener. Por ejemplo para la vecindad expuesta, utilizando las coordenadas espaciales y un problema de clasificación con tres clases tendremos que el vector de entrada te tendrá un tamaño que se calcula:

$$te = vp + coords + (4+1)o = 1 + 2 + 5(3) = 18 \qquad (4.1)$$

donde vp es el número de unidades correspondientes a el valor del píxel, $coords$ el número de unidades correspondientes a las coordenadas espaciales y o el número de clases a clasificar.

En la Figura 4.2 se muestra el esquema de las entradas de la red básica de la estructura propuesta en esta tesis para una pasada p cualquiera y un problema de segmentar una imagen con 3 clases distintas.

Una característica importante es que dado que es una única red operando sobre toda la imagen, ésta utiliza un único conjunto de pesos no importando la posición en la que se encuentre.

4.2 Algoritmos de Entrenamiento propuestos.

La anterior estructura modifica la imagen al realizar una operación de ventana tomando información tanto del valor del píxel de la imagen como cierta información de sus vecinos.

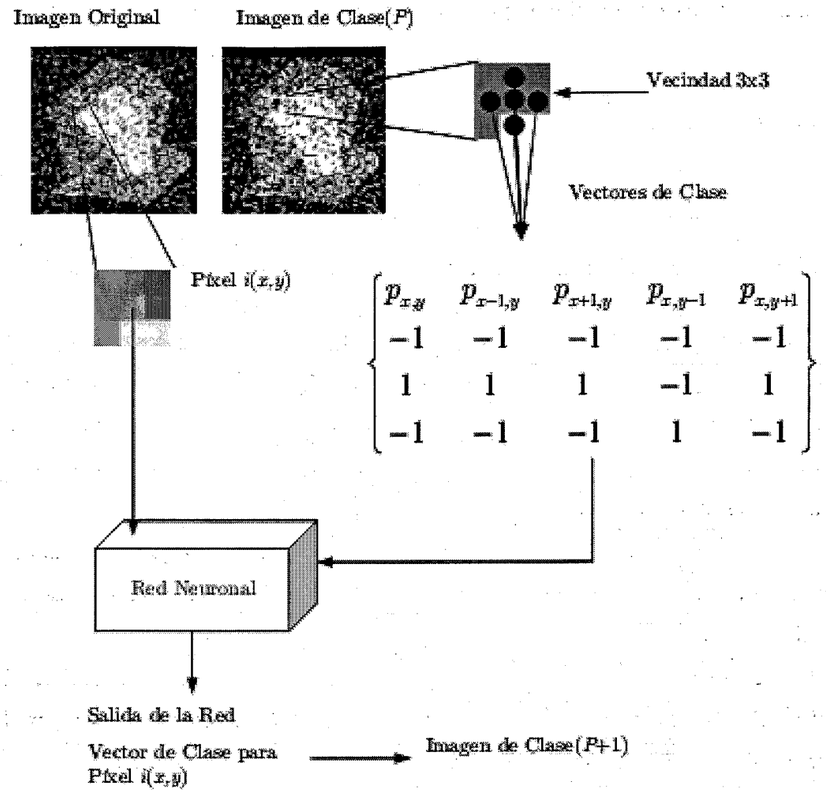


Figura 4.2. Esquema de las entradas recibidas por la red básica en la estructura propuesta para una *pasada* P cualquiera con una imagen que contiene 3 clases. Los vectores de clase no necesariamente tienen que ser bipolares, aunque pueden fijarse para que lo sean. Los algoritmos propuestos utilizan vectores de clase reales.

Si las salidas estuvieran también en el espacio de los valores de los píxeles, la estructura propuesta pudiera utilizarse para realizar segmentación no supervisada. La utilización de esta estructura en tales casos sale del ámbito de competencia de esta tesis.

La pregunta sería entonces: ¿Cómo entrenar la estructura anterior de tal manera que pueda realizar la segmentación de una imagen teniendo como información

algunas parejas de soluciones previamente validadas? A continuación se presentan los algoritmos desarrollados en esta tesis para realizar tal trabajo.

4.2.1 Algoritmo Recurrente.

Una aproximación inicial al problema de entrenamiento supervisado de la estructura anterior es utilizar un algoritmo semejante al propuesto por Jordan para su red recurrente[8]

Este algoritmo toma la base de que en cada estado de tiempo t determinado la red recurrente llega a un estado lo suficientemente estable para que el algoritmo de retropropagación pueda ser llevado a cabo dentro de los pesos de la red. Esta aseveración se cumple si tenemos que la red no recibe ningún tipo de entrada o retroalimentación en ningún período de tiempo $t+\Delta t$.

Entonces en las redes Jordan el aprendizaje se realiza retropropagando el error y modificando los pesos excepto en las conexiones entre las unidades de estado y las unidades de salida.

Si se analiza la estructura propuesta al nivel de la red básica descrita anteriormente podemos ver que es muy semejante a la red propuesta por Jordan, por lo que se utilizó su enfoque para obtener este primer algoritmo.

Inicialmente escogemos el número de pasadas P con los que la red va a operar.

Los pesos de la red básica se inicializan de manera aleatoria en el intervalo $[-1,1]$ como se recomienda para la mayoría de los problemas de aproximación en las redes neuronales [20]

Operamos la red para el número de pasadas determinado iniciando para la pasada $p=0$ con una imagen obtenida de procesar la imagen de entrada original o simplemente con una imagen de ruido.

Al final de cada pasada obtenemos una clasificación tentativa dada por la red, como una matriz de vectores de salida $O^p_{x,y}$, por lo que comparamos tal clasificación con aquella previamente validada obteniendo un error, dado por la siguiente expresión, modificada a partir de la ecuación (2.11):

$$E^p(w) = \frac{1}{2}\sum_{x=1}^{NC}\sum_{p=1}^{NR}\left\|O_{x,y}^p - t_{x,y}\right\|^2 \tag{4.2}$$

En la expresión anterior, $t_{x,y}$ corresponde al vector de salida esperado en el píxel x,y, con el valor de 1 en la posición correspondiente a la clase representada por dicho píxel y -1 en el resto, además NC y NR corresponden al número de columnas y renglones dentro de la imagen.

Derivando $E^p(w)$ con respecto a $O_{x,y}^p$ obtenemos el error por píxel x,y que será retropropagado hacia atrás dentro de la red básica utilizando las expresiones matriciales (2.24) a (2.27), que en este caso tomarán la forma:

$$\boldsymbol{\delta}_{x,y}^{(2)} = \mathbf{D}_{x,y}^2 \frac{\partial E^p}{\partial O_{x,y}^p} \tag{4.3}$$

para el error retropropagado a la capa de salida de la red básica con $\mathbf{D}_{x,y}^2$ la matriz de derivadas de las unidades en la capa oculta dada la entrada en el píxel x,y. Similarmente el error hasta la capa oculta es:

$$\boldsymbol{\delta}_{x,y}^{(1)} = \mathbf{D}_{x,y}^1 \mathbf{W}_2 \boldsymbol{\delta}_{x,y}^{(2)} \tag{4.4}$$

y las correcciones a los pesos en ambas capas de la red básica estarán dadas por:

$$\Delta\overline{\mathbf{W}}_2^T = -\gamma\boldsymbol{\delta}_{x,y}^{(2)}\delta_{x,y}^{(1)} \tag{4.5}$$

y

$$\Delta\overline{\mathbf{W}}_1^T = -\gamma\boldsymbol{\delta}_{x,y}^{(1)}\delta_{x,y} \tag{4.6}$$

obteniendo así los valores de actualización de los pesos Δw para cada píxel x,y.

Ahora la actualización global total para la pasada p se calcula con la siguiente expresión:

$$\Delta w_{(n)} = \sum_{x=1}^{NC}\sum_{y=1}^{NR}\Delta w_{(n)}^{x,y} \tag{4.7}$$

para cada conjunto de pesos dentro de la red.

En la ecuación anterior n corresponde al número de capa dentro de la red básica.

La modificación global para los pesos en una pasada p se realiza utilizando los valores de la ecuación anterior y el algoritmo Rprop comentado en la sección 2.2.3.1

Una iteración dentro de este algoritmo termina cuando realizamos el anterior proceso para cada pasada p hasta que $p = P$.

Este proceso se repite hasta que el error cuadrático medio de la última capa sea cero, se estabilice (esto es el error alcance un mínimo local) o se hayan realizado un número de iteraciones fijo.

Al final del período de entrenamiento la red nos devolverá un conjunto de pesos w que nos segmentarán una imagen con las mismas características de la imagen original con que fue entrenada.

El anterior algoritmo se resume en el cuadro 4.1.

4.2.2 Algoritmo Recurrente Modificado.

Otro enfoque hacia el entrenamiento de la estructura es considerar que el error de clasificación se da únicamente en la última pasada y que este error es propagado hacia las pasadas anteriores utilizando las mismas ecuaciones del método de retropropagación.

Esto es semejante a considerar que una red básica (tal y como se describió anteriormente) se encuentra exactamente debajo de cada píxel x,y en la imagen y que envía su salida a otras redes básicas diferentes en pasadas subsecuentes hasta la pasada final. Entonces las salidas de cada red conectan efectivamente unidades básicas de proceso como si se tratara de una red neuronal *feed-forward* más grande.

Un enfoque muy semejante es aquel que engloba las topologías modulares de redes neuronales tal y como se describen en [13]. La diferencia en este algoritmo es que es, como se dijo anteriormente, una única red neuronal en todo el proceso.

Algoritmo 1:

1. Inicializa pesos, número de iteraciones máximo K y número de pasadas P a realizar.
2. Obtención de una imagen de clasificación inicial $I(0)$. $k \leftarrow 0$.
3. Si $k<K$ realiza:
   a. $p \leftarrow 1$.
   b. Si $p<=P$ realiza:
      i. Obtener la imagen salida I(p) operando la red utilizando la imagen original y la imagen I(p-1).
      ii. Calcular el error y obtener los incrementos según la ecuación **(4.7)**.
      iii. Utilizar el algoritmo Rprop para actualizar los pesos de la red básica.
      iv. P++
   c. Calcula el error para la última imagen y devuelve como error total de la red.
4. La clasificación final para la pasada P es la salida de la red. Guarda pesos.

Cuadro 4.1. Algoritmo Recurrente.

Iniciamos el algoritmo de la misma manera que el anterior, inicializando los pesos y las constantes de aprendizaje, así como la imagen de inicio para la pasada $p=0$.

En la primera etapa operamos la red el número de pasadas P definido obteniendo en cada caso matrices I_p de vectores de salida $O^p_{x,y}$. Al igual que en el algoritmo recurrente, cada matriz de salida anterior nos servirá para obtener la siguiente.

Obtenemos el error en la última pasada con la expresión:

$$E^P(w) = \frac{1}{2}\sum_{x=1}^{NC}\sum_{y=1}^{NR}\left\|O^P_{x,y} - t_{x,y}\right\|^2 \tag{4.8}$$

y de la misma manera derivamos y retropropagamos el error resultante por píxel x,y con las expresiones de la retropropagación matricial modificadas, poniendo $p=P$:

$$\boldsymbol{\delta}^{(2)}_{p_{x,y}} = \mathbf{D}^{\,2}_{p_{x,y}} \frac{\partial E^p}{\partial O^p_{x,y}} \tag{4.9}$$

para el error en la capa de salida y similarmente para el error en la capa oculta:

$$\boldsymbol{\delta}^{(1)}_{p_{x,y}} = \mathbf{D}^{\,1}_{p_{x,y}} \mathbf{W}_2 \boldsymbol{\delta}^{(2)}_{p_{x,y}} \tag{4.10}$$

Para la actualización de los pesos se utilizan de igual manera las expresiones:

$$\Delta \overline{\mathbf{W}}_2^T = -\gamma \boldsymbol{\delta}^{(2)}_{p_{x,y}} o^{(1)}_{x,y} \tag{4.11}$$

y

$$\Delta \overline{\mathbf{W}}_1^T = -\gamma \boldsymbol{\delta}^{(1)}_{x,y} o_{x,y} \tag{4.12}$$

obteniendo así una modificación a los pesos con la ecuación:

$$\Delta w^p_{(n)} = \sum_{x=1}^{NC}\sum_{y=1}^{NR} \Delta w^{x,y}_{(n)} \tag{4.13}$$

para la pasada final. Este paso es semejante al utilizado en el anterior algoritmo.

En la segunda etapa retropropagamos dicho error hacia las pasadas anteriores.

El error total a la última pasada obtenido con (4.2), siguiendo con el enfoque descrito anteriormente, es la acumulación de los errores individuales a través de cada pasada, por lo que para obtener una mejor aproximación al incremento de los pesos por pasada será necesario retropropagar dicho error utilizando las mismas ecuaciones del método clásico de retropropagación.

Entonces, siguiendo con la misma metodología expresada en la sección 2.2.2, el error a retropropagar por cada posición x,y para cualquier pasada $p \in [1, P-1]$ intermedia estará dado por la siguiente expresión:

$$e^{p-1}_{x,y}(w) = W_1 \boldsymbol{\delta}^{(1)}_{p_{x,y}} \tag{4.14}$$

Aquí W_1 son los pesos entre la capa de entrada y la oculta de la red básica. $\boldsymbol{\delta}^{(1)}_{p_{x,y}}$ es el valor del error de la red retropropagado hasta la capa oculta de la red básica al ser estimulada con el vector de entrada correspondiente al píxel x,y en la pasada p.

Si recordamos la estructura propuesta veremos que cada salida de la red básica en una pasada influye en z entradas de una pasada posterior, donde z dependerá

de la vecindad utilizada, entonces la derivada del error total acumulado por píxel para una pasada intermedia el cual será utilizado para encontrar la modificación de los pesos en esa pasada es:

$$\boldsymbol{\delta}^{p}_{x,y}(w) = \frac{1}{4}\sum_{r,s\in N_{x,y}}\left[A^{T}_{x,y}M_{r-x.s-y}e^{p}_{r,s}(w)\right] \tag{4.15}$$

En la ecuación anterior A es una matriz diagonal compuesta por las derivadas de los elementos de las salidas obtenidas por la red en el píxel x,y. Es decir que un elemento estará dado por:

$$A_{i,i} = F'\left(O^{p}_{x,y}(i)\right) \tag{4.16}$$

En la anterior ecuación $F'(x)$ es la derivada de la función de activación, en este caso de la función sigmoidea simétrica.

$M_{r-z,s-y}$, de la ecuación (4.15), es una matriz binaria de [Número de clases]x[Número de unidades de entrada en la red básica] que enmascara los valores de $e^{p}_{r,s}(w)$ correspondientes al píxel x,y. Es decir dada la vecindad de r,s como píxel central, la matriz M tendrá unos en las posiciones correspondientes al píxel x,y y cero en el resto. Por ejemplo si $x,y = r-1,s$; es decir que el píxel x,y se encuentra un píxel arriba del centro; utilizando la vecindad de 3x3 y un tamaño de vector de entrada para 2 clases (13 posiciones), tendremos que M será igual a:

$$M_{r-z,s-y} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.17}$$

Si realizamos la operación de $M \times e$ veremos que efectivamente se devuelven únicamente los valores que dentro del vector $e^{p}_{r,s}(w)$ buscamos.

Al final es necesario retropropagar internamente en la red básica, el error obtenido con la ecuación (4.15), por lo que utilizamos nuevamente las expresiones matriciales. Entonces el error en la capa de salida de la red básica estará dado ahora por:

$$\boldsymbol{\delta}^{(2)}_{p_{x,y}} = \mathbf{D}^{2}_{p_{x,y}}\boldsymbol{\delta}^{p}_{x,y} \tag{4.18}$$

y para la capa oculta y los incrementos de los pesos, las mismas expresiones, (4.10) a (4.12), sin cambio. La actualización de los pesos en la pasada intermedia también estará dada por la ecuación (4.13).

Ahora la actualización global de los pesos una vez que obtuvimos las actualizaciones por pasada se obtiene con:

$$\Delta w^{T}_{(n)} = \sum_{p=1}^{P}\Delta w^{p}_{(n)} \tag{4.19}$$

Finalmente se utiliza el algoritmo Rprop para modificar los pesos de la red básica de la misma manera que en el algoritmo recurrente anterior.

Una iteración del algoritmo se realiza al operar la red un número de pasada P y retropropagar el error desde la pasada final hasta la inicial como se comentó anteriormente.

Este proceso, al igual que en el algoritmo recurrente, se repite hasta que el error cuadrático medio de la última capa sea cero, se estabilice (esto es el error alcance un mínimo local) o se hayan realizado un número de iteraciones fijo.

El algoritmo total se resume en el cuadro 4.2.

4.2.2.1 Notas de la implementación.

El algoritmo recurrente modificado mostrado en el cuadro 4.2 requiere que se guarden las derivadas de las unidades de la red básica en cada posición x,y, esto llevaría a la creación de matrices de derivadas para cada unidad en la red por pasada, así como las salidas en la capa oculta para cada punto.

Si se cuenta con una imagen de 32x32 píxeles y se entrena la red 4 pasadas con 10 unidades ocultas y 2 clases (semejante a las pruebas con imágenes binarias) tendríamos que guardar alrededor de 491,520 valores reales de 8 bytes. Esto no es mucho para los estándares actuales, ya que ocupan "únicamente" 3.75 Mb, pero al crecer el tamaño de las imágenes y la complejidad de la red el guardar todos esos valores si se convierte en un problema.

Ya que no era realmente importante el tiempo de entrenamiento de la estructura en general se optó por un mecanismo sencillo para darle la vuelta a este problema. En vez de guardar las derivadas y las salidas de la capa oculta de la red en cada píxel en cada pasada, se vuelve a procesar las entradas en cada punto x,y en cada pasada para obtener estos valores en la red básica antes de realizar la retropropagación interna del error. No se pierde nada en este paso ya que se tienen guardadas las salidas en cada pasada y estas son las que se utilizan para obtener las derivadas y las salidas en la capa oculta de la red en cada píxel.

El ahorro en memoria fue grande al utilizar este método pues fue posible correr sin problema el programa y entrenar una imagen con 3 clases de 128x128 píxeles, 4 pasadas y una red con 10 unidades ocultas, que sin este truco hubiera necesitado el guardar alrededor de 65 Mb extras de datos.

4.2.3 Inicialización de los Algoritmos.

Como se describió anteriormente ambos algoritmos requieren una imagen de clasificación inicial para comenzar tanto el proceso de entrenamiento como el proceso de operación. Esto es, se requiere una clasificación tentativa para $p=0$.

Como primera aproximación durante el desarrollo de la tesis se propuso utilizar una imagen aleatoria de clases para tal efecto. Tal imagen se construyó con la siguiente expresión:

$$I_c(x,y) = random[1, clases] \qquad (4.20)$$

Esta imagen proporcionaría los vectores de entrada para la red neuronal.

Algoritmo 2:

1. Inicializa pesos, número de iteraciones máximo K y número de pasadas P a realizar.
2. Obtención de una imagen de clasificación inicial $I(0)$.
   $k \leftarrow 0$.
3. Si $k<K$ realiza:
   a. $p \leftarrow 1$ (Etapa 1)
   b. Mientras $p<=P$ realiza:
      i. Obtener y guardar las matrices de salida O^p a partir de la imagen original y la imagen $I(p-1)$, además las derivadas $D^1_{p_{x,y}}$ y $D^2_{p_{x,y}}$ de las unidades ocultas y de salida de la red en cada posición x,y.
      ii. p++
   c. $p \leftarrow P$ (Etapa 2)
   d. Calcular el error E^P dadas las salidas O^P con la ecuación **(4.8)**
   e. Retropropaga el error E^P internamente en la red básica para cada píxel x,y obteniendo $\Delta w(x,y)$ con las ecuaciones **(4.9)** a **(4.13)**.
   f. Mientras $p>1$ realiza:
      i. Calcula e^{p-1} usando **(4.14)**
      ii. p--.
      iii. Calcula δ^p usando **(4.15)**
      iv. Retropropagar el error internamente en la red básica utilizando las ecuaciones **(4.18)** y **(4.10)** a **(4.13)**.
   g. Calcula los incrementos globales con la ecuación **(4.19)**.
   h. Actualiza los pesos utilizando Rprop.
   i. k++.
4. Fin

Cuadro 4.2. Algoritmo recurrente modificado.

Algunas pruebas con esta aproximación nos mostraron que no era muy conveniente utilizarla por lo que se optó por recurrir a la mejor clasificación posible de la imagen ruidosa sin tomar en cuenta ningún tipo de conocimiento a priori. Esta clasificación puede ser obtenida al colocarle a cada píxel el valor de la clase, conociendo los valores medios de gris de cada clase, más cercano al valor real del píxel.

Entonces para encontrar dicha imagen dado un conjunto de medias de clase es necesario aplicar la siguiente expresión:

$$I_c(x,y) = \min_k |I(x,y) - \mu_k| \tag{4.21}$$

Aquí μ_k es la media correspondiente a la clase k y I_c es una imagen de las mismas dimensiones que la original con valores dentro del conjunto $C = \{c_1, c_2, ..., c_n\}$, con $\|C\|$ el número de clases presentes en la imagen original.

La ecuación anterior encuentra la clase más probable de pertenencia de un píxel específico x,y utilizando únicamente la información de intensidad de dicho píxel. Esta operación puede ser vista también como una operación de cuantizar con niveles con separación desigual.

Para el trabajo de implementación, el conjunto C fue tomado simplemente como dígitos correspondientes a cada clase, aunque puede ser cualquier valor representativo.

A partir de esta imagen de clases visibles se construye la imagen requerida por la red en $p = 0$.

Como se describe en la estructura, la imagen de clases I_p utilizada en cada pasada requiere que cada píxel sea un vector de clasificación $O^p_{x,y}$ de dimensión igual a el número de clases presentes. En el caso de la imagen inicial requerida para $p = 0$ y teniendo, dada la ecuación (4.21), la clasificación correspondiente llenamos cada vector $O^p_{x,y}$ con 1 en la posición correspondiente al valor de $I_c(x,y)$ y -1 en el resto. Una interesante variante a esta operación sería el llenar cada vector $O^p_{x,y}$ de I_p con $p = 0$, con los valores de cercanía encontrados con la ecuación (4.21), esto no fue probado pues se consideró suficiente la anterior aproximación.

En la Figura 4.3 se presentan varias imágenes de escala gris y sus correspondientes imágenes obtenidas con la ecuación (4.21) para tres clases utilizando para la representación valores de gris representativos de cada clase.

Como se verá en el capítulo siguiente la utilización de este operador fue muy satisfactoria.

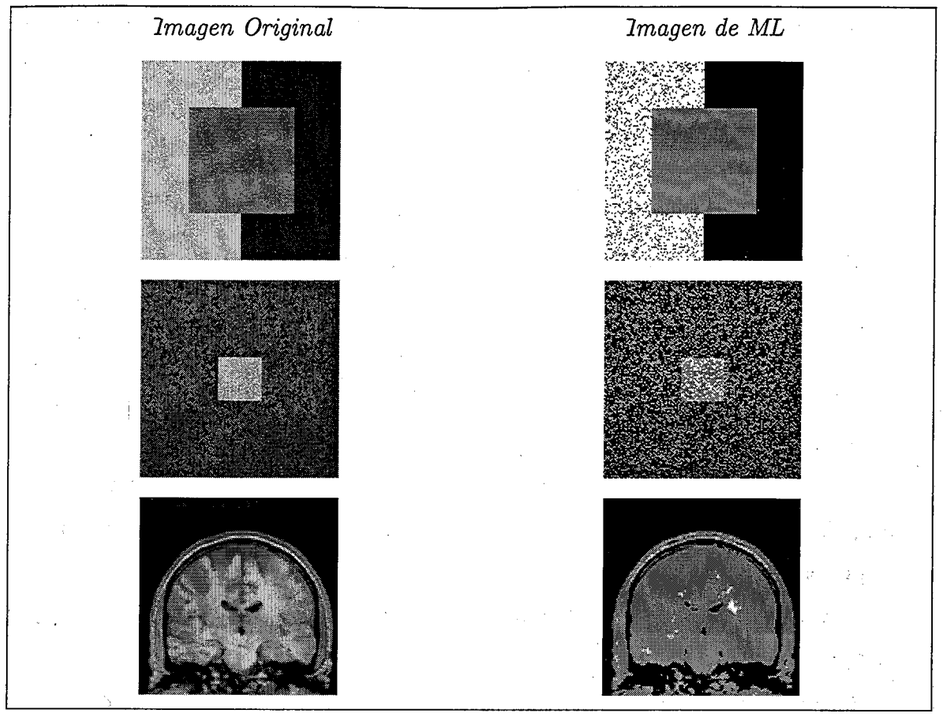


Figura 4.3. Imágenes de ML para 3 clases obtenidas con el operador mostrado en la ecuación (4.21). Los valores para cada clase se escogieron como 0, 127 y 255 para las clases 1,2 y 3 respectivamente.

Capítulo 5

Experimentos.

El siguiente paso dentro del proyecto de tesis es realizar pruebas tanto a la estructura propuesta como a los algoritmos de entrenamiento para verificar la validez de la utilización de la retroalimentación o recurrencia dentro una red neuronal de aplicación local.

Iniciamos el capítulo con una breve explicación de las consideraciones tomadas tanto para el cálculo del error cometido por la red en las pruebas realizadas así como para la obtención de las imágenes de salida presentadas.

A continuación mostramos los resultados obtenidos para un problema de clasificación con 2 clases utilizando imágenes binarias y finalizamos con un problema de clasificación de 3 clases en imágenes de escala de gris.

5.1 Consideraciones iniciales.

5.1.1 Obtención de las imágenes de salida de la estructura.

Como se recordará la estructura obtiene a la salida de cada pasada un vector por cada píxel de la imagen formando una matriz o imagen I_p de vectores $O^p_{x,y}$ que habrá de ser analizada para encontrar la imagen resultante. Cada vector contiene la relevancia encontrada de cada clase a partir de las entradas dadas de la red para cada píxel, dicha relevancia normalmente se asocia con una probabilidad [20]Es

decir, por ejemplo, que el valor de $\left(O_{x,y}^{p}\right)_1$ me dirá la relevancia de la clase 1. En un caso óptimo y sabiendo que las salidas esperadas tienen valores únicos 1 y –1 sería sencillo encontrar la clase devuelta por la red al buscar la posición cuyo valor sea cercano a 1, pero dado que es muy difícil llegar a saturaciones de los vectores de salida en todos los casos, es más sensato devolver el máximo valor.

Teniendo en cuenta esto entonces es trivial encontrar cuál es la clase, según la red, a la que pertenece un píxel dado. La clase estará dada por la siguiente expresión:

$$I_c(x,y) = \arg\max_k \left\{\left(O_{x,y}^{p}\right)_k\right\} \tag{5.1}$$

Recordando que el índice k denota la clase.

En nuestro caso, dado que conocemos el nivel medio de gris correspondiente a cada clase, fue también sencillo visualizar la imagen de clase resultante I_c al asignarle a cada píxel el valor de gris correspondiente al número de clase k devuelto por la expresión (5.1).

5.1.2 Métrica del Error.

Para obtener resultados globales dentro de los conjuntos de prueba se creó una métrica para el error de simulación de la red. El error de simulación estará dado por el promedio de los errores individuales de cada algoritmo al presentársele una imagen. Este error individual está dado por el número de píxeles erróneamente clasificados, al compararlos con la imagen que contiene la clasificación correcta. Entonces el error total de simulación para un nivel de entrenamiento de la red estará dado por:

$$Error_{sim} = \frac{1}{N}\sum_{i=1}^{N} error_i \tag{5.2}$$

donde N es el número de imágenes de prueba y

$$error_i = \frac{1}{nr \times nc}\sum_{x=1}^{nr}\sum_{y=1}^{nc}\left(1 - \delta\left(I_{c_i}^{NET(p)}(x,y) - I_{c_i}^{VERDADERA}(x,y)\right)\right) \tag{5.3}$$

con $\delta(x) = 1$ si $x = 0$ y $\delta(x) = 0$ si $x \neq 0$ y nr, nc son el número de renglones y columnas de la imagen i, $I_{c_i}^{NET(p)}$ y $I_{c_i}^{VERDADERA}$ son las imágenes de clases obtenidas por la red al final de las *pasadas p* propuestas de simulación y la imagen de clases correcta respectivamente.

Es lógico pensar que el error obtenido con las siguientes expresiones será aplicable únicamente cuando se tengan clasificaciones dadas de las imágenes a probar, algo que nunca se cumplirá en casos de utilización en situaciones que podríamos denominar "reales". Este error es pues obtenido para realizar comparaciones entre los algoritmos.

5.2 Experimentos con Imágenes Binarias.

Los primeros experimentos realizados con la nueva estructura involucraron la utilización de imágenes binarias. El problema planteado es ver si la estructura propuesta podría eliminar ruido de una imagen binaria.

Este problema cae en el ámbito de procesamiento de imágenes, pero también puede ser visto como un problema de clasificación. Aquí cada píxel en una imagen ruidosa tiene que corresponderse con un píxel clasificado como parte de la figura que contiene la imagen (clase 1) o como parte del fondo (clase 2). Como se ve este problema cuenta con una gran cantidad de *patrones conflictivos*. Esto lo hace un candidato ideal para nuestro algoritmo como se explicó en el capítulo 4.

5.2.1 Formato de las imágenes utilizadas.

Para estas pruebas se utilizaron un total de 96 imágenes de mapa de bits binarios de 32 x 32 píxeles representando distintas formas del número 2.

Dichas imágenes forman parte de un conjunto obtenido como parte de un proyecto de reconocimiento de dígitos manuscritos. Las imágenes se rastrearon a una resolución de 200 dpi a partir de hojas de prueba obtenidas para distintos sujetos a los que se les pidió que escribieran los dígitos con tinta negra.

Las imágenes originales variaban de resolución de entre 30 x 30 píxeles hasta 90 x 90 píxeles. Como parte de dicho proyecto las imágenes fueron escaladas y / o cortadas hasta obtener el tamaño de 32 x 32 píxeles con paquetería comercial de procesamiento de imágenes.

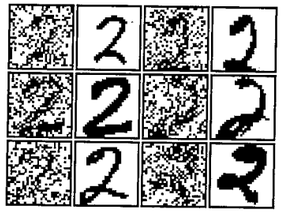


Figura 5.1. Ejemplos de imágenes binarias utilizadas en los experimentos. Las columnas 1 y 3 representan las mismas imágenes con ruido binario al 25%. Las columnas restantes son las imágenes originales utilizadas de 32 x 32 píxeles.

En las imágenes los píxeles pertenecientes al dígito se visualizan en negro sobre blanco.

Ejemplos de las imágenes se pueden ver en la Figura 5.1.

5.2.2 Características de los experimentos.

Se dividió el conjunto de imágenes en conjuntos independientes de entrenamiento y prueba. Cada conjunto contó con 48 imágenes escogidas al azar de entre las 96 imágenes originales.

A cada conjunto se le agregó ruido al 40% de los píxeles, cambiando al azar un píxel de blanco a negro y viceversa, formando así parejas de imágenes ruidosas y no ruidosas, esto es semejante a utilizar una probabilidad para el ruido aleatorio binario del 25%.

Para los experimentos se utilizó un programa realizado en C++ Builder (ver sección 5.4) que permite introducir y modificar los parámetros de la red, guardar y cargar pesos de la misma, así como visualizar el comportamiento de entrenamiento y prueba.

El entrenamiento consistió en operar la red un número dado de iteraciones por imagen y por conjunto definido dentro del programa, para pasadas desde 1 a 6.

Para mantener una homogeneidad dentro de las pruebas se optó por los siguientes parámetros para la red: un número de iteraciones totales de 500 y un número de iteraciones por imagen fijo a 10. Los valores máximo, mínimo e inicial de la constante de aprendizaje para el paso Rprop del algoritmo se tomaron fijos en 0.5, 1e-7 y 0.05 respectivamente. Estos valores se eligieron por que durante el proceso de implementación y pruebas preliminares fueron los que presentaron mejor resultado.

Dos distintos juegos de pruebas utilizando las coordenadas del píxel como entradas a la red y sin utilizarlas fueron también realizados.

La red básica para las pruebas con coordenadas quedó con 13 unidades en la capa de entrada (obtenidas como se explica en el capítulo 4), 10 unidades ocultas y 2 unidades de salida (una unidad por clase), para las pruebas sin coordenadas el número de unidades de entrada disminuyó a 11, el número de unidades ocultas se mantuvo.

Dichos parámetros fueron usados para ambos algoritmos.

5.2.3 Resultados.

Una vez realizadas las pruebas era necesario observar si los algoritmos presentaban buenos resultados en las imágenes de prueba, independientemente de su desempeño global en todo el conjunto.

En la Figura 5.2 se presentan los resultados obtenidos por la red entrenada con el algoritmo recurrente (algoritmo 1) y con el recurrente modificado (algoritmo 2), utilizando coordenadas, al aplicarla a una imagen perteneciente al conjunto de prueba un número determinado de pasadas $P = \{1,2,...,n\}$, además se incluye para comparación el resultado de entrenar la estructura únicamente una pasada (es decir sin utilizar recurrencia) e iterarla esas mismas n pasadas.

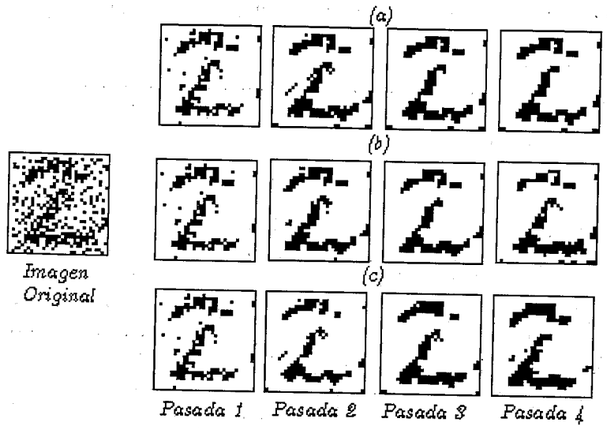


Figura 5.2. En (a) se observa la red entrenada con 1 pasada e iterada hasta 4 pasadas. (b) Soluciones mostradas por la red utilizando el algoritmo 1, iteradas con el mismo número de pasadas P con el que fueron entrenadas, para $P = 1, 2, 3, 4$. (c) Soluciones del algoritmo 2. La imagen de la izquierda es la original con 25% de ruido binario.

Como se puede observar los resultados de la segmentación son muy buenos en ambos algoritmos. Además se puede observar que el algoritmo 2 presenta un mejor desempeño en esta imagen que al utilizar la estructura entrenada con 1 pasada e iterada más pasadas, y que al utilizar el algoritmo 1.

Aún cuando estos resultados parecen alentadores, pues la imagen presentada a la estructura nunca antes fue utilizada, era necesario realizar una prueba global con el total del conjunto de las imágenes de prueba y comparar el desempeño, ya no visual, si no cuantificado a través del error sobre todo el conjunto de prueba.

En la Figura 5.3 se presentan los resultados del error total sobre el conjunto de prueba. Como se puede observar, tanto el algoritmo 1 como el algoritmo 2 presentan una mejora sustancial a la aplicación local de una red neuronal de múltiples capas sin utilizar recurrencia.

Otro resultado interesante es el que se presentan en la Figura 5.4. En ella se puede observar un patrón de comportamiento que nos hace suponer que algo de la

información de la cantidad de pasadas utilizadas queda registrada en los pesos al utilizar el algoritmo 2.

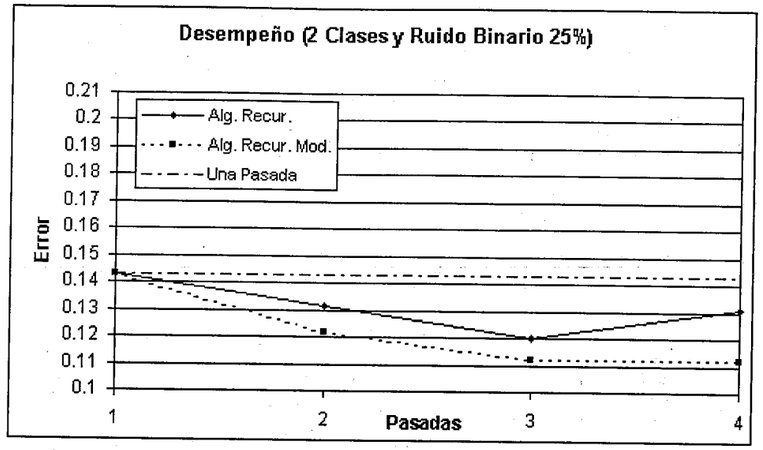


Figura 5.3. Gráfica del error obtenido por los algoritmos en la etapa de prueba para distinto número de pasadas de entrenamiento. El error se calculó como se detalla en la sección 5.1.2. Se agrega el desempeño al utilizar la estructura entrenada 1 pasada.

El comportamiento anterior no se presentó al utilizar el algoritmo 1 en ningún momento, lo que hace suponer que dicho comportamiento es producto de la manera en que el algoritmo 2 actualiza los pesos utilizando el error parcial retropropagado en vez del error total por pasada. Además como se puede observar tanto al analizar la Figura 5.4 y la gráfica que se presenta en la Figura 5.5, los resultados con el menor error, tanto al procesar una sola imagen como procesando todo el conjunto de prueba, se dan en pasadas múltiplos del número de pasadas con el que la red fue entrenada. En el caso de la prueba que presentamos en este trabajo el mejor resultado se daba en la pasada 8, pero en otras pruebas, principalmente en las que se realizaron sobre todo el conjunto, el mejor resultado se presentaba mayormente en la pasada con la que la red fue entrenada. Este comportamiento fue observado hasta en redes utilizando 2 pasadas y un número de unidades en la capa oculta tan pequeño como 5 o 6.

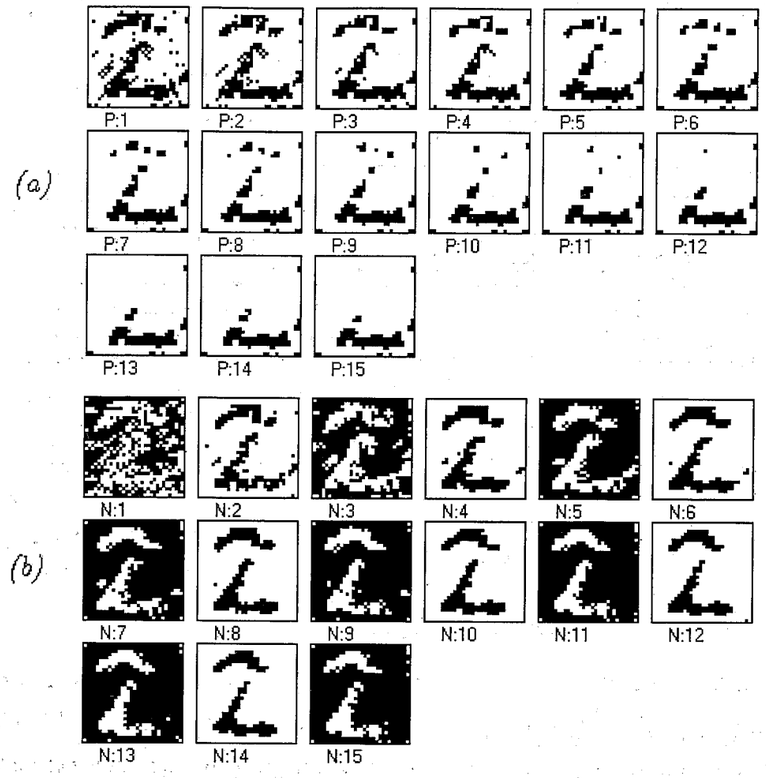


Figura 5.4. En (a) se presentan los resultados de procesar una imagen con pesos obtenidos con el algoritmo 1 para 4 pasadas, y probada un número de pasadas desde 1 a 15. En (b) los mismos resultados utilizando el algoritmo 2. La imagen procesada original es la misma que aparece en la Figura 5.3.

Todos los comportamientos presentados anteriormente también fueron observados al utilizar la estructura sin contar con las entradas correspondientes a las coordenadas, lo que hace suponer que en este caso y por el tipo de imágenes utilizadas no son relevantes.

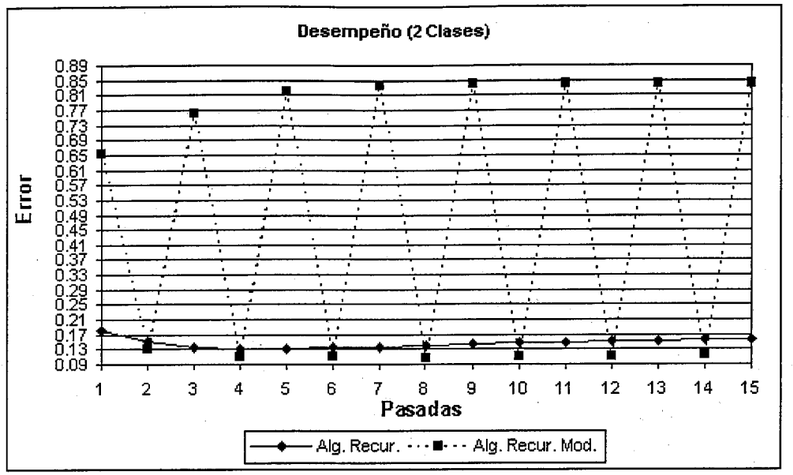


Figura 5.5. Desempeño de los algoritmos al entrenarse con 4 pasadas y probarse 15 pasadas para todo el conjunto de prueba. Como se puede observar, la curva del algoritmo 2 presenta un comportamiento cíclico y obtiene los mejores resultados en las iteraciones múltiplo de 4.

Por último se realizaron unas pruebas con los mismos conjuntos de imágenes binarias pero distintos niveles de ruido, en este caso 10 y 15% de ruido aleatorio binario. Los conjuntos fueron probados utilizando los pesos de entrenamiento para ruido al 25% y únicamente utilizando múltiplos de pasadas. Los resultados se muestran en la en la que se observa como el error es más estable en los casos donde se utilizó recurrencia.

5.3 Experimentos con Imágenes en escala de gris.

El paso natural después de haber realizado pruebas en imágenes binarias es el utilizar imágenes en escala de gris. La gran mayoría de los problemas dentro del procesamiento de imágenes utilizan imágenes en escala de gris e incluso problemas que incluyen imágenes en color pueden transformarse en problemas de imágenes de escala de gris con el empleo de la técnica de separación de canales.

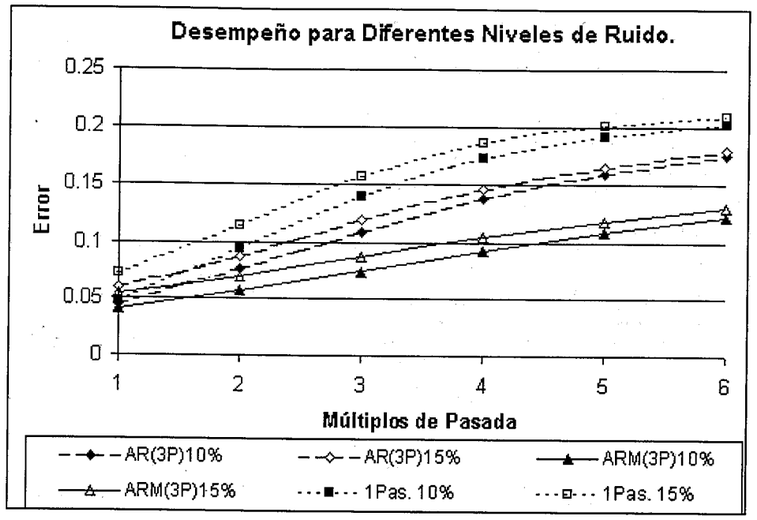


Figura 5.6. Desempeño de los algoritmos para valores de ruido de 10% y 15% al procesar la red múltiplos del número de pasada de entrenamiento. Los pesos utilizados fueron aquellos que se obtuvieron con 25% de ruido.

Nuestra principal meta era tratar de observar si el comportamiento y desempeño de nuestra propuesta se mantenía al pasar a un problema mucho más complejo, por lo que los experimentos realizados, no obstante sencillos desde el punto de vista del procesamiento de imágenes, nos parecieron suficientemente complejos y a la vez computacionalmente no prohibitivos como para comprobar esto.

5.3.1 Formato de las imágenes utilizadas.

Dentro de estas pruebas se utilizaron un total de 10 imágenes sintéticas de mapa de bits de 64 x 64 píxeles con valores en el intervalo [0, 255] , esta vez representando figuras geométricas caprichosas: poliedros, círculos, elipses y polígonos irregulares.

Cada imagen cuenta con 3 clases definidas: fondo, figura intermedia y figura exterior. Cada clase tiene asociada una intensidad de gris, así se tiene que el fondo tiene asociado la intensidad 0, la figura intermedia tiene asociada la intensidad 127 y la figura exterior tiene asociada la intensidad 255.

Los valores de las imágenes al ser incluidos dentro de la red neuronal fueron transformados al espacio [0, 1] a través de una sencilla operación lineal:

$$V_{nuevo} = \frac{V_{viejo} - V_{\min}}{V_{\max} - V_{\min}} \tag{5.4}$$

Dada la naturaleza de las imágenes utilizadas, la ecuación anterior se reduce a la siguiente expresión empleada en nuestra implementación:

$$V_{nuevo} = \frac{V_{viejo}}{255} \tag{5.5}$$

Esto permite contar con un vector de entrada con todos sus valores dentro del intervalo [0, 1].

Ejemplos de las imágenes se presentan en la Figura 5.8.

5.3.2 Características de los experimentos.

Al igual que en el anterior conjunto de experimentos estos fueron divididos en 2 grupos independientes, el grupo de prueba y el grupo de entrenamiento. Cada grupo contó con 5 imágenes. El conjunto completo se presenta en la Figura 5.7.

A cada imagen se le agregó ruido uniforme en el intervalo [0,255] con probabilidad de cambio $\eta = 0.60$ (60% de ruido), obteniendo entonces un conjunto de parejas de imágenes ruidosas y no ruidosas para las pruebas, ejemplos de estas imágenes se observan en la Figura 5.8.

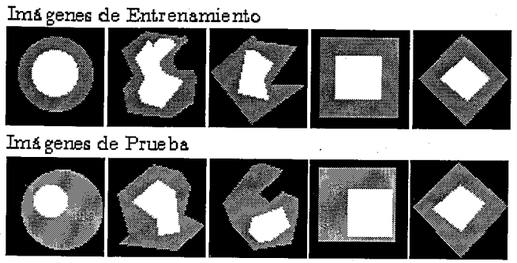


Figura 5.7. Imágenes utilizadas en los experimentos de escala de gris.

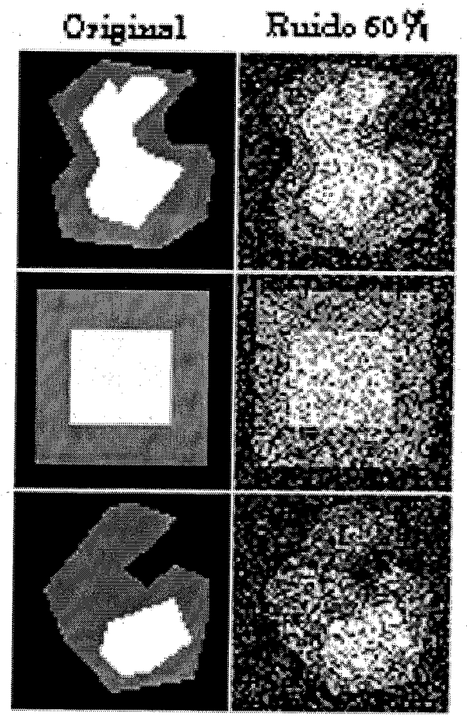


Figura 5.8. Ejemplos de imágenes sintéticas de escala de gris utilizadas. Las imágenes denotadas como *original* son aquellas sin agregarle ruido. Las otras imágenes son las originales 60% de ruido uniforme.

Las mismas condiciones de entrenamiento empleadas para las imágenes binarias fueron implementadas para estos experimentos, es decir se utilizaron un número de iteraciones a través del conjunto de 500 y un número de iteraciones por imagen de 10. Esta vez se probó únicamente el algoritmo de 1 a 4 pasadas. Los valores de las constantes de aprendizaje fueron idénticos.

Dado el incremento en el número de unidades de entrada debido al aumento en el número de clases, los experimentos fueron llevados a cabo con un número de unidades ocultas igual a 15. También se realizaron pruebas con un número de unidades ocultas igual a 3.

Cada prueba se realizó utilizando coordenadas como entradas para 15 unidades ocultas y sin coordenadas para 3 unidades ocultas.

Entonces la red básica quedó con la siguiente estructura para las pruebas: 18 unidades de entrada, 15 unidades ocultas y 3 unidades de salida en el primer caso y 16 unidades de entrada, 3 ocultas y 3 unidades de salida en el segundo caso.

5.3.3 Resultados.

Los resultados presentados a continuación utilizan las mismas ecuaciones (5.2) y (5.3) para obtener el error global en el conjunto de prueba.

Para el caso de los resultados de la estructura al aplicar una red básica con 3 unidades ocultas se realizaron un total de 12 pruebas con distintos puntos de inicio para los pesos de la red básica. Los resultados se resumen en la Tabla 5.1 y en la Figura 5.9.

En la Tabla 5.1 se muestran los resultados más importantes, aquí es significativo notar los valores mínimos que se encontraron tanto para el algoritmo 1 (recurrente) y el algoritmo 2 (recurrente modificado) al compararlos con el resultado del mínimo valor sin aplicar recurrencia. Al igual que en el caso binario los resultados muestran una mejora sustancial al utilizar recurrencia. Es en este caso de prueba en el que utilizar el enfoque del algoritmo 1 comienza a no ser muy práctico ya que la mejora

con respecto al no utilizar retroalimentación es muy poca. Por el contrario el algoritmo 2 si presenta una mejora sustancial, sobre el no utilizar recurrencia.

Pasadas	Algoritmo Recurrente				Algoritmo Recurrente Modificado			
	Media	Desv. Std.	Min.	Máx.	Media	Desv. Std.	Min.	Máx.
2	0.20791	0.02496	0.187042	0.2501	0.19416	0.03890	0.10385	0.2584
3	0.22216	0.10457	0.158723	0.50609	0.21167	0.07760	0.08413	0.3
4	0.20954	0.04441	0.180123	0.28620	0.22894	0.08096	0.07906	0.31
1	0.23442	0.025052	0.209065	0.29563				

Tabla 5.1. Tabla de resultados de las pruebas con 3 unidades ocultas, sin coordenadas para el problema de 3 clases. Se realizaron 12 pruebas con diferente inicialización de pesos.

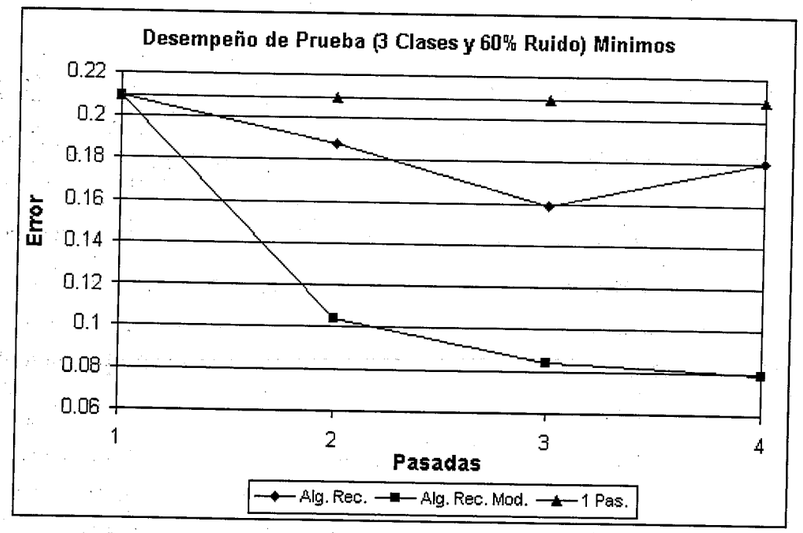


Figura 5.9. Gráfica del desempeño de los algoritmos (mínimos de las pruebas) al utilizar una red básica con 3 unidades ocultas, sin coordenadas para el problema de 3 clases.

Más interesante aún es lo que se presenta en la Figura 5.10. Como se puede ver, el comportamiento cíclico que se presentaba en el problema binario aún se mantiene al pasar a un problema más complejo con 3 clases y píxeles dentro de un intervalo de valores más grande.

En el caso de 15 unidades ocultas se realizaron un total de 6 pruebas con distintos puntos de inicio, las gráficas y resultados obtenidos muestran en este caso los valores mínimos del error en el conjunto de prueba encontrados durante el entrenamiento.

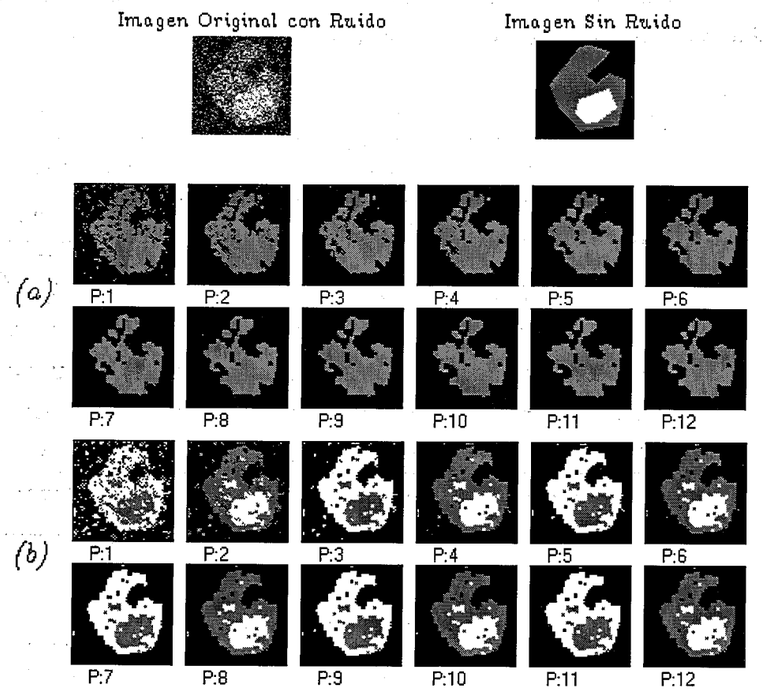


Figura 5.10. En (a) se presentan los resultados al iterar de 1 a 12 pasadas la estructura con 3 unidades ocultas entrenada con 4 pasadas, y el algoritmo 1 a una imagen con 60% de ruido. En (b) los resultados utilizando el algoritmo 2.

En la Figura 5.11 se presentan las curvas de error sobre todo el conjunto de prueba de los algoritmos para las imágenes con 60% de ruido, así como de la red entrenada sin retroalimentación (1 pasada) para el caso de 15 unidades ocultas.

La imágenes de la Figura 5.12 muestra también el mismo resultado cíclico al aplicar la red básica entrenada con el algoritmo 2, algo que no cambió al aumentar el número de unidades ocultas. Como se observa también el desempeño visual del algoritmo 1 mejoró, aunque el error total en todo el conjunto de prueba se mantuvo muy semejante al de una pasada, y en algunos casos empeoró notablemente.

También se obtuvo una gráfica de desempeño de las propiedades cíclicas en el caso de 15 unidades ocultas, semejante a la que se realizó para el problema de 2 clases, esta se muestra en la Figura 5.13.

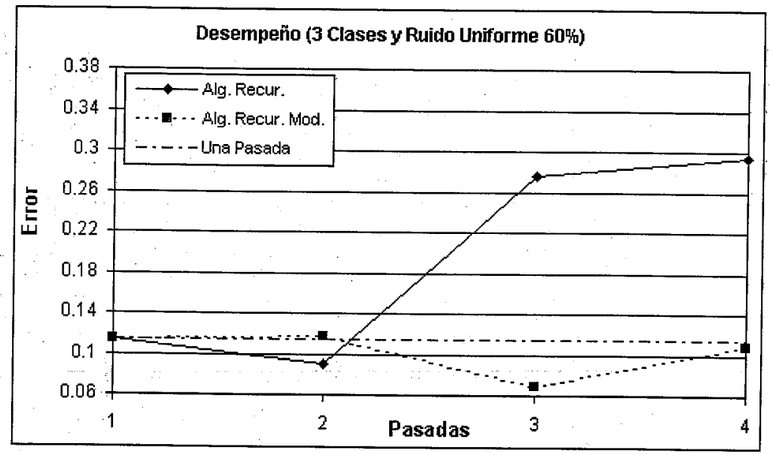


Figura 5.11.Gráfica del error obtenido en el conjunto de prueba para el problema de clasificación con 3 clases. Se entrenó y se probó con el mismo número de pasadas ambos algoritmos. Se incluye la curva de respuesta de la estructura al ser entrenada sin retroalimentación.

El observar los resultados tanto el de la Figura 5.10, el de la Figura 5.12, como el de la gráfica de la Figura 5.13, parece indicarnos que el comportamiento de presentar los mejores resultados al procesar la red un múltiplo del número con el

que fue entrenada no es propio del problema, como podría suponerse si únicamente se hubieran realizado pruebas en imágenes binarias, sino que quizás sea inherente tanto a la estructura recurrente, en menor grado, como al algoritmo de aprendizaje, más específicamente, a la consideración tomada de retropropagar el error parcial de cada pasada. Desde nuestro punto de vista esto último tiene un gran peso en este comportamiento, por no decir que es totalmente un producto de el enfoque novedoso presentado en el algoritmo modificado.

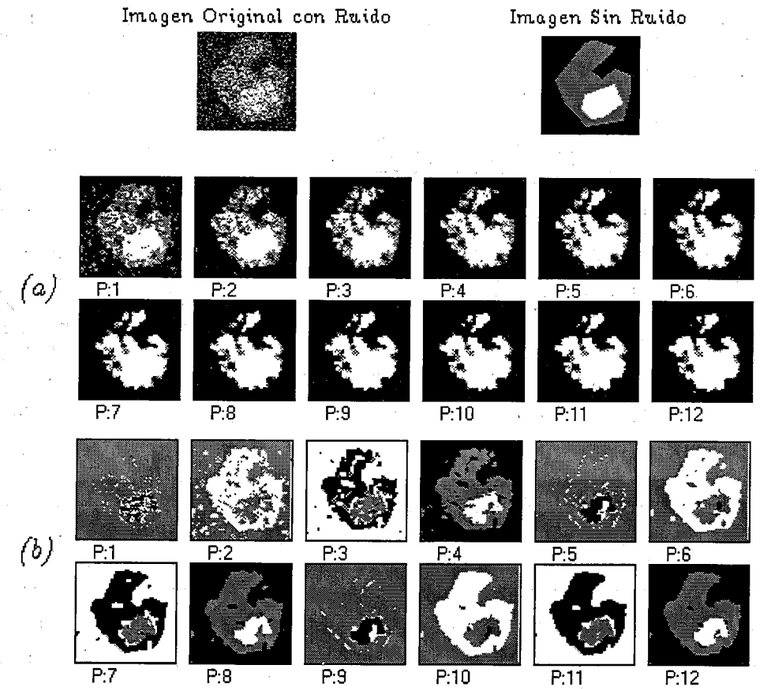


Figura 5.12. En (a) se presentan los resultados al iterar de 1 a 12 pasadas la estructura con 15 unidades ocultas entrenada con 4 pasadas y el algoritmo 1 a una imagen con 60% de ruido. En (b) los resultados utilizando el algoritmo 2.

En la Figura 5.14 se observa el comportamiento del algoritmo 2 entrenado con 3 pasadas y 15 unidades ocultas al procesarse el conjunto de prueba desde 3 hasta 30 pasadas.

Se puede observar que el algoritmo 2 presenta una gran estabilidad en los resultados y no los degrada como ocurre con una red entrenada sin retroalimentación (1 pasada) cuando se itera más de las pasadas con las que fue entrenada. Este mismo comportamiento aunque en menor grado se observó con el algoritmo 1 utilizando 2 pasadas (su mejor desempeño).

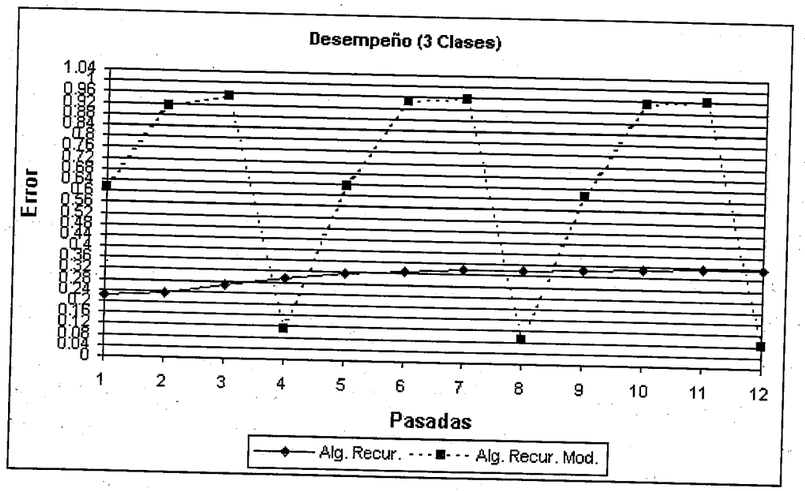


Figura 5.13. Gráficas de desempeño de los algoritmos al ser entrenados con 4 pasadas y probados de 1 a 12 pasadas. Se observa un comportamiento cíclico en la curva del algoritmo modificado, además los mejores resultados se obtuvieron al probar con múltiplos de 4 pasadas.

Dado que para el algoritmo 2 los mejores resultados se dan en múltiplos de las pasadas de entrenamiento, en la Figura 5.14 se muestran los resultados en múltiplos de las pasadas de entrenamiento para el algoritmo 1 y la red sin retroalimentación.

¿Qué tan estable es el algoritmo modificado? Para responder a esto se muestra una comparación entre el algoritmo recurrente modificado y la estructura sin utilizar recurrencia en la Figura 5.16. La gráfica muestra los mínimos encontrados para ambas estructuras utilizando 3 unidades ocultas y 15 unidades ocultas. Como se observa los resultados se mantienen más o menos estables al utilizar el algoritmo con retroalimentación.

En ellas se observa que la utilización de retroalimentación para mejorar el desempeño se vuelve mucho más notable. Esta es una prueba de la robustez al número de unidades ocultas al aplicar estos algoritmos, algo que con lo que no cuenta la red sin retroalimentación que presentó un aumento del error grande al disminuir las unidades ocultas. Más aún, la complejidad computacional entre la estructura sin retroalimentación entrenada con 15 unidades ocultas y la estructura entrenada con el algoritmo recurrente modificado con 3 unidades ocultas y 3 pasadas es comparable, pero el error es mucho menor al utilizar nuestra propuesta. Esta aseveración puede ser corroborada al observar la Figura 5.17 que muestra los tiempos de CPU necesarios para entrenar una red sin recurrencia con 15 unidades ocultas y nuestras propuestas con 3 unidades ocultas y distintos números de pasadas.

Por último se muestra un conjunto de resultados en la Figura 5.15 en las que se utilizaron los pesos obtenidos con la red para 3 pasadas y 3 unidades ocultas. En dicha figura se puede notar que no obstante que los pesos fueron obtenidos con imágenes que contenían un porcentaje de error del 60%, esos mismos pesos nos dan resultados muy buenos cuando se aplican a imágenes con menos ruido, incluso cuando se aplica a una imagen con más ruido (en este caso 80% de ruido) los resultados son aceptables. Hay que hacer notar que al aumentar el nivel de error en las imágenes aumentó también el número de múltiplos de pasada que se tenían que dar a las imágenes para obtener los mejores resultados posibles. Así para una imagen sin ruido o 20% de ruido únicamente fue necesario dar 3 pasadas, el número con que fueron entrenados los pesos, pero para la imagen con 80% de ruido se requirieron 18 pasadas para obtener el mejor resultado. El comportamiento se

refleja en la gráfica de la Figura 5.18, donde se muestra que el comportamiento mostrado en el caso binario para este mismo tipo de prueba se mantiene. La prueba únicamente se realizó para el algoritmo modificado.

5.4 Notas sobre el programa utilizado en los experimentos.

Para el desarrollo del trabajo de tesis y como parte fundamental de la investigación se requirió la implementación de la estructura propuesta y su algoritmo tal y como se describen en el capítulo 4.

Para tal efecto se decidió implementar todo en el lenguaje C++, por lo que se requirió de la ayuda del paquete Borland Builder 4 como entorno de desarrollo de la aplicación dentro del ambiente operativo MS-Windows.

Aún cuando se tenia acceso a las librerías diseñadas para la implementación de redes neuronales dentro del ambiente MATLAB, se descartó utilizarlas ya que estas aún cuando permiten un gran nivel de abstracción y libertad en la creación de redes neuronales, tienen el inconveniente de estar escritas en el lenguaje de MATLAB y por tanto introducirían una gran sobrecarga de tiempo en llamadas por lo que rutinas nativas de redes neuronales en C++ eran necesarias.

Se optó por utilizar un paradigma orientado a objetos ya que haría más fácil el desarrollo y modificación de la aplicación durante el proceso de trabajo de la tesis. No obstante no se siguió ninguna metodología de ingeniería de software para tal efecto.

5.4.1 Notas sobre el equipo.

Para todas las pruebas realizadas y descritas en el capítulo 5 se utilizaron 3 computadoras con características de capacidad distintas de acuerdo a la disponibilidad de tiempo para la realización de las pruebas.

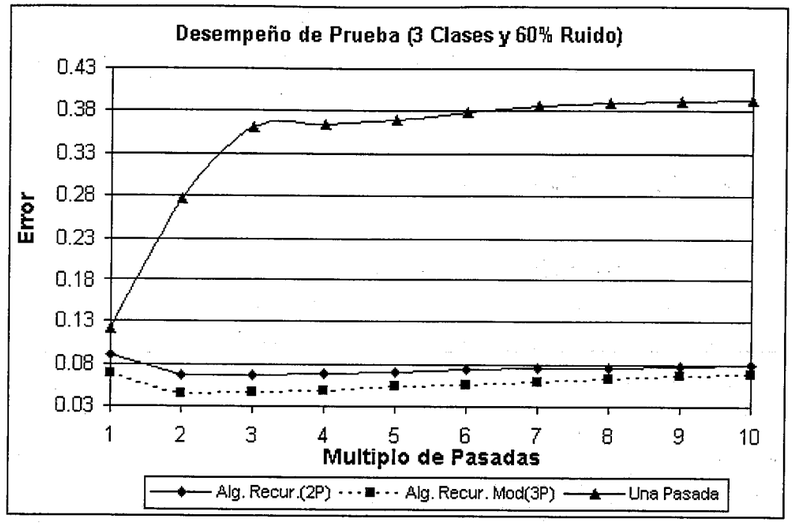


Figura 5.14. Gráficas de desempeño para mostrar la estabilidad de los algoritmos al probarlos más pasadas de las que fueron entrenadas. El eje horizontal corresponde al múltiplo de las pasadas de entrenamiento. En este el caso del algoritmo 1 los valores corresponden a las pasadas 2-20, para el algoritmo 2 pasadas 3-30 y para la red sin retroalimentación pasadas 1-10.

El desarrollo de la aplicación y las pruebas preliminares se realizaron en una computadora PC con procesador Athlon a 750Mhz y 128MB de RAM. La gran mayoría de las pruebas se corrieron en una PC con procesador Pentium III a 700Mhz con 768 MB de RAM. Las pruebas finales de desempeño con redes más sencillas se realizaron en 2 computadoras PC con procesadores Pentium III a 350Mhz y 500Mhz con 64Mb y 256MB de RAM respectivamente.

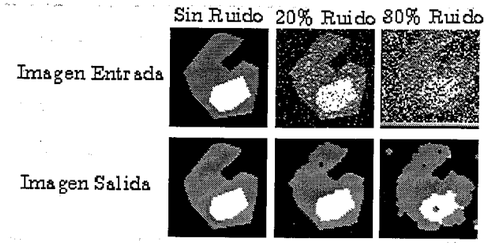


Figura 5.15. Resultados al aplicar los pesos obtenidos al entrenar la red con el algoritmo 2 y 3 pasadas con 3 unidades ocultas sin coordenadas, a diferentes versiones de una imagen de prueba. Las imágenes de salida se obtuvieron aplicando la red el número de múltiplos de 3 pasadas necesario para encontrar el mínimo error.

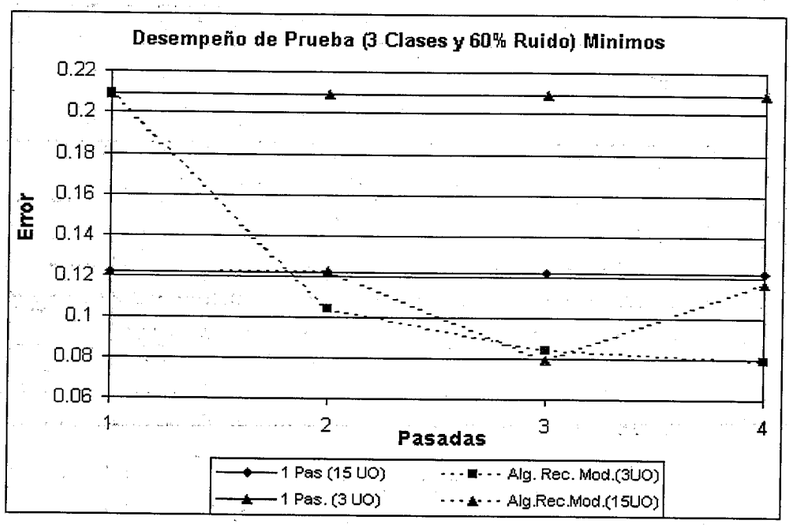


Figura 5.16. Gráfica de comparación entre los algoritmos con recurrencia (algoritmo recurrente modificado) y sin recurrencia utilizando 15 unidades ocultas y 3 unidades ocultas para el mismo conjunto de entrenamiento y prueba anteriores. Los valores son los mínimos encontrados.

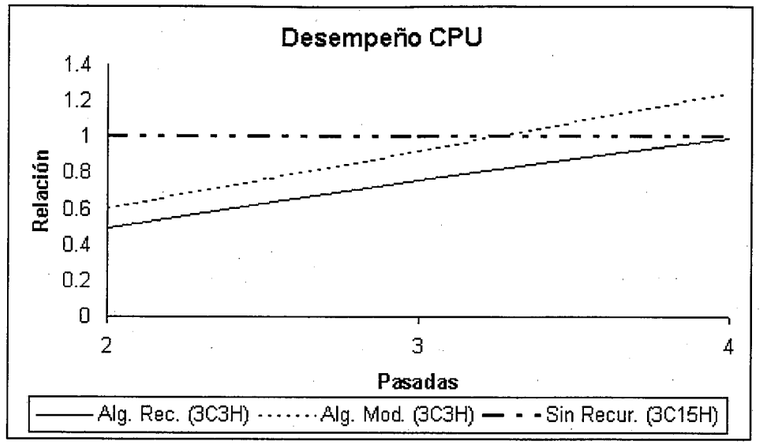


Figura 5.17. Desempeño de la utilización del CPU de los algoritmos. El valor unitario corresponde a los tiempos obtenidos por la red sin recurrencia.

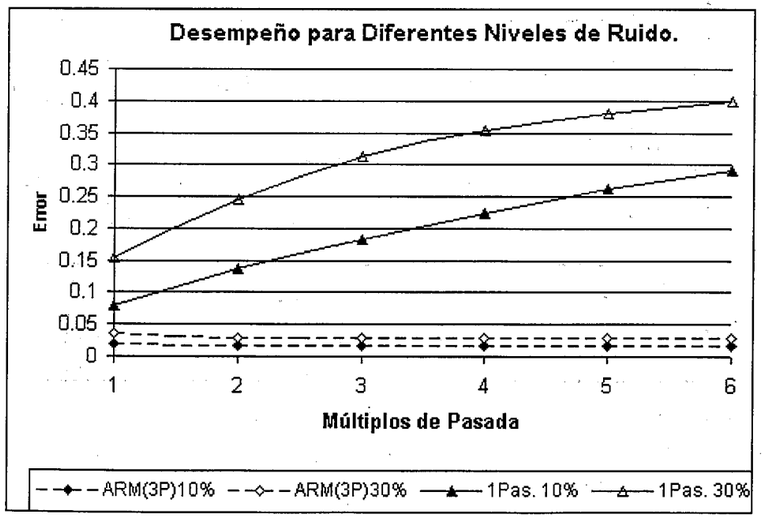


Figura 5.18. Desempeño de los algoritmos para valores de ruido de 10% y 30% al procesar la red múltiplos del número de pasada de entrenamiento. Los pesos utilizados fueron aquellos que se obtuvieron con 60% de ruido.

Capítulo 6

Conclusiones

Este trabajo de tesis presentó una nueva estructura de una red recurrente entrenada con retropropagación para la segmentación de imágenes con ruido y demostró la valía del uso de la recurrencia en este tipo de estructuras. Se desarrollaron dos algoritmos de entrenamiento y procesamiento de esta estructura que obtuvieron buenos resultados al utilizarlos en experimentos con imágenes sintéticas binarias y de escala de gris.

A continuación se presentan algunas conclusiones interesantes que se obtuvieron, así como el trabajo futuro que se considera para continuar la investigación dentro de este tipo de estructuras de redes recurrentes.

6.1 Conclusiones del trabajo.

Se enumeran algunas conclusiones muy interesantes encontradas al utilizar esta novedosa estructura así como los algoritmos de entrenamiento desarrollados.

1. Se introdujo un novedoso concepto recurrente dentro de las redes neuronales para procesamiento de imágenes y un par de algoritmos para su entrenamiento que presentan buenos resultados además de ser relativamente sencillos de implementar.
2. Aún cuando los resultados que se muestran en el capítulo 5 son alentadores, esto se circunscriben únicamente a la aplicación en imágenes

semejantes a las que se utilizaron para las pruebas. Pruebas subsecuentes en distintos conjuntos de imágenes con características diferentes quizás sean necesarias para mejorar lo que aquí se presenta.

3. Se obtuvieron resultados significativamente superiores a los obtenidos con una red convencional BP de aplicación local (no recurrente) con el mismo número de unidades y pesos, lo que demuestra la conveniencia de utilizar retroalimentación en vez de otros procesos más complicados en la segmentación de imágenes con redes de este tipo.
4. El algoritmo recurrente modificado presentó los mejores resultados tanto en las pruebas con imágenes binarias como en las de escala de gris. Este algoritmo es el que consideramos el mejor para entrenar una red que utilice recurrencia para la segmentación de imágenes.
5. Un aspecto importante que se observó en los resultados de las pruebas fue que, utilizando una red básica sencilla, únicamente con 3 unidades ocultas y sin el empleo de las coordenadas de los píxeles en la entrada, se llegó a presentar una mejora sustancial en el desempeño, principalmente al utilizar el algoritmo modificado. Esta mejora fue en el mejor de los casos de alrededor de 3 veces menos error. Esta mejora también se presentó al aumentar el número de unidades ocultas a 15, todavía por debajo, ahora en menor grado, de la red sin retroalimentación.
6. La complejidad computacional de entrenamiento encontrada de la estructura recurrente es del orden $O(NMP)$, con N las dimensiones de la imagen, M el número de unidades ocultas y P las pasadas de entrenamiento, P veces más que una red convencional. No obstante el desempeño de la estructura con recurrencia, al utilizar una red básica más sencilla, con 3 unidades ocultas, es mejor aún que el de una red sin recurrencia con 15 unidades ocultas, teniendo ambos casos una complejidad semejante. Esto demuestra que al utilizar redes más sencillas el enfoque tratado en esta tesis si representa una alternativa a la utilización de estructuras sin el empleo de retroalimentación, tanto en lo que refiere a la disminución del error como en la complejidad computacional.
7. Se presentaron una serie de comportamientos sumamente interesantes de analizar más a profundidad, principalmente aquel referente al aprendizaje de la posición temporal de la estructura, esto es la propiedad que se presentó de que la red básica devolvía los mejores resultados al aplicarla un número de pasadas múltiplo de aquellas con la que fue entrenada.
8. El algoritmo sin recurrencia (1 pasada) se deteriora rápidamente al usarse con un número de pasadas mayor, por el contrario el algoritmo recurrente modificado mejora su desempeño en múltiplos del número de pasadas de entrenamiento. El algoritmo recurrente aún cuando no deteriora las imágenes tan fuertemente no presenta este comportamiento tan peculiar.
9. Al utilizar el algoritmo recurrente modificado se encontró que es más estable con respecto al número óptimo de pasadas de entrenamiento, pues su desempeño tiende a ser similar para más de 3 pasadas. Esto no ocurre con el algoritmo recurrente, ya que el número de pasadas óptimo es difícil de encontrar.

6.2 Trabajo Futuro.

Al finalizar este trabajo quedaron algunos puntos sueltos que valdría la pena experimentar e investigar como trabajo futuro, pues consideramos que el comportamiento mostrado en las pruebas realizadas da pie a pensar que este tipo de estructuras tiene un gran camino por delante.

A continuación enumeramos algunas de ellas.

1. Al realizar únicamente pruebas con imágenes sintéticas, se dejó un gran campo abierto para la experimentación con imágenes reales. Hay que recalcar que la manera en como se preprocesen las imágenes reales dependerá de las clases que se quiera extraer de ellas. No obstante y dado que nuestra estructura no requiere la extracción de características para

dar buen resultado como en otros modelos, recomendamos que este preproceso debe aplicarse directamente a los píxeles.

2. No se probó el utilizar otros métodos de aceleración a parte del Rprop por lo que un trabajo futuro inmediato sería el utilizar métodos de aceleración de segundo orden como QuickProp, Newton y Levembert-Maquardt para tratar de ver si se mejora tanto el resultado global como el tiempo de entrenamiento.
3. Los parámetros libres del proceso, como número de iteraciones por imagen individual en un conjunto de imágenes, neuronas ocultas, constante de aprendizaje inicial, etc., merecen ser estudiados más a fondo con el fin de quizás obtener mejores resultados en menos iteraciones. Aún cuando los valores utilizados en las pruebas fueron obtenidos por prueba y error, sería muy interesante tratar de analizar el impacto de estos en los resultados.
4. La estructura propuesta tiene el potencial de aplicarse en Hardware para acelerar tanto el entrenamiento como la simulación, por lo que una vez entrenados los pesos, la segmentación de imágenes semejantes a las utilizadas para entrenar pudiera hacerse en tiempo real, teniendo esto gran aplicación en áreas como robótica y análisis médico.

Capítulo 7

Referencias

[1] Austin, J. (1996) *High Speed Image Segmentation Using a Binary Neural Network.* Tech. Rep. Advanced Computer Arquitecture Group. Department of Computer Science. University of York. UK.

[2] Cho, S. Kim, J.H. (1993) *Feedforward Neural Network Architectures for Complex Classificaction Problems.* Center of Artificial Intelligence Research and Computer Science Department. Korea Advanced Institute of Science and Technology. Korea.

[3] Cox, G. S., Hoare, F. J., Jager G. de (1992) *Experiments in Lung Cancer Nodule Detection Using Texture Analysis And Neural Network Classifiers.* Departament of Electrical Engineering. University of Cape Town. South Africa.

[4] Elman, J.A. (1990) *Finding structure in time.* Cognitive Science, 14:170-211.

[5] Gelenbe, E. Editor (1992) *Neural Networks: Advances And Applications,2.* North-Holland. NY.

[6] Hölldobler, S., Kalinke, Y. (1991) *Towards A New Massively Parallel Computational Model for Logic Programming.* Paper. FG Wissensverarbeitung, KI, Fakultät Informatik. Dresden TU. Germany.

[7] Jähne, Bernd. (1997) *Digital Image Processing: Concepts, Algorithms, and Scientific Applications.* 4th Edition. Springer-Verlag. Berlín. Germany.

[8] Jordan, M. I. (1986) *Attractor dynamics and parallelism in a connectionist sequential machine.* In Proceedings of the Annual Conference of the Cognitive Science, Págs. 531-546.

[9] Jordan, M. I. (1996) *Neural Networks.* In CRC Handbook of Computer Science, CRC Press, Boca Raton.

[10] Kröse, B., Smagt, P. van der, (1996) *An Introduction to Neural Networks.* 8th Edition. University of Amsterdam. Netherlands.

[11] Le Cun, Y., Boser, B., et al. (1990) *Handwritten Digit Recognition with Back-Propagation Network.*

[12] Marroquín, J.,L., Botello, S., Calderón, F., Vemuri, B.,C., (2000) *The MPM-MAP Algorithm for Image Segmentation.* In Proceedings of the International Congress on Pattern Recognition 2000.

[13] Mehrotra, K., Mohan, C.K., Ranka, S. (1997) *Elements of Artificial Neural Networks.* MIT Press. Massachusetts.

[14] Osman, H., Blostein, S.D. (1999) *Backpropagation Algorithm For Multiresolution Image Classification.* Tech. Rep. Department of Electrical And Computer Engineering, Queen's University. Canada.

[15] Ossen, A., et. al. (1994) *Segmentation of Medical Images Using Neural-Network Classifiers.* Tech. Rep. Dep. Of Computer Science . Technical University of Berlin.

[16] Pandya, A. S., Macy, R.B. (1996) *Pattern Recognition with Neural Networks in C++.* CRC Press IEEE Press. Florida.

[17] Pearlmutter , A.B. (1990) *Dynamic Recurrent Neural Networks.* Tech. Rep. CMU-CS-90-196, Carnegie Mellon University School of Computer Science, Pittsburgh, PA.

[18] Reyes-Aldasoro, C., C., Aldeco, A. L. (2000) *Image Segmentation and Compression using Neural Networks.* In Proceedings of the Workshop in Advances in Artificial Perception and Robotics 2000. Guanajuato. México.

[19] Ripley, B.D. (1996) *Pattern Recognition and Neural Networks.* Cambrigde Press. London.

[20] Rojas, R. (1996) *Neural Networks: A Systematic Introduction.* Springer-Verlag, Berlín.

[21] Rumelhart, D y McClelland, J.L. (1986) *Parallel Distributed Processing*; The MIT Press, Cambridge. CA.

[22] Sethi I.A. (1991) *Artificial Neural Networks and Statistical Pattern Recognition.* Machine Intelligence and Pattern Recognition Series, Volume 11, North-Holland, E.U.

[23] Skapura, D.M. (1996) *Building Neural Networks.* Acm Press. NY.

[24] Vehtari, A., Lampinen, J. (2000) *Bayesian MLP Neural Networks for Image Analysis.* Preprint to Pattern Recognition Letters.

[25] Wang, Y. Tülay, A., Kung, S., Szabo, Z. (1998) Quantification and Segmentation of Brain Tissues from MR Images: A Probabilistic Neural Network Approach. IEEE Transactions on Image Processing. Vol. 7 No. 8.

[26] Waever, S., Leemon, B., Polycarpou, M. (1998) *Preventing Unlearning During On-Line Training of Feedforward Networks.* Department of Electrical and Computer Engineering. University of Cincinnati. Wrigth-Patterson Air Force Base. USAF Office of Scientific Research. Carnegie Mellon University. Tech. Rep.

[27] Weitzel, I., Kopecz, K., *et al.* (1997) *Contour Segmentation with Recurrent Neural Networks of Pulse-Coding Neurons.* Dept. of Neurophysics. University of Marburg. Germany.