COMPRESIÓN DE IMÁGENES DIGITALES UTILIZANDO REDES NEURONALES: MULTICAPA (Backpropagation) Y RBR's

RESUMEN
El siguiente texto tiene como objetivo plantear una aplicación de RNA's que en teoría es podría resultar obvia, pero en la practica genera cierta incertidumbre y es entonces merecedora de un poco mas de atención y estudio. Finalmente el artículo trata de dar una solución básica, pero en general abre una puerta y genera una idea digna de pulimento para tratar reducción de imágenes, con técnicas de inteligencia artificial de todo tipo, en un trabajo serio como una tesis de grado. El desarrollo programacional es implementado en C++ con librería allegro.

PALABRAS CLAVES: Redes Neuronales, C++, imágenes, almacenamiento.

JUAN DIEGO GOMEZ V.
Estudiante Ingeniería de Sistemas
Universidad Tecnológica de Pereira.
juanogo@utp.edu.co
juanogo@gmail.com

ABSTRACT
The following text has as objective to pose an application of RNA's that is of obvious and substantial results in theory, but in the practice generates certain uncertainty and it is then worthy of a little much of attention and study. Finally the paper tries to give a solution, perhaps not very good, but in general it opens a door and generates an idea worthy of polish to treat reduction of images, with artificial intelligence technical, in a serious work as a degree thesis. The development programational is create in C++ with library allegro.

KEY WORDS: *Neuronals Networks, C++, imagines, storage.*

1 INTRODUCCION

Es de innegable importancia en el saturado mundo de tecnología en que vivimos actualmente, el tratamiento o gestionamiento de la información en toda su diversidad funcional, como: su transmisión, su organización, su almacenamiento etc. Inclusive podríamos pensar que detrás de cada una de estas ramas existe una ciencia dedicada al mantenimiento, mejoramiento y optimización del recurso. Es así como en esta ocasión reducimos claramente las fronteras del análisis de la información en general y nos encasillamos netamente en un importante ítem de estudio como lo es el almacenamiento de ésta y concretamente en lo que concierne a imágenes.

Ya es para todos nosotros común y convencional el manejo de medios o dispositivos en los cuales podemos alojar de manera "constante" cierta cantidad de información; el desarrollo actual de este hardware no es de nuestra momentánea incumbencia, pero en cambio si su administración y su buen aprovechamiento, para una buena utilización de una memoria por ejemplo se requiere de un intento apresurado de aumentar su capacidad, en términos relativos, ya que en lo general estos dispositivos son fijos y por lo tanto no maleables, más no lo es la información que ellos albergaran en un momento determinado, así pues si reducimos el tamaño de la información, obviamente siempre manteniéndola; podremos mejorar el rendimiento de cualquiera de estos dispositivos independientemente de su tecnología.

Hay en el mercado miles de herramientas de software que se dedican a estos fines, estas herramientas más conocidas como compresores, son de gran utilidad y por supuesto de gran uso en el común de las personas que conviven con la informática y es precisamente pensando en ellos que surge la idea de implementar un sistema de compresión de imágenes para un mayor almacenamiento, usando técnicas no convencionales como lo son las RNA's, para así pretenciosamente, lograr un software con fines iguales pero con mejores resultados que los conocidos y renombrados winzip o winrar entre otras.

2 PLANTEAMIENTO DEL PROBLEMA

Una imagen o un mapa de bits es nada más que una matriz numérica en memoria donde cada componente de ésta, alberga un número entero que a su vez representa determinado color, así pues una foto digital de 200x200 píxeles, está lógicamente representada por una matriz numérica de igual tamaño donde cada color de los 40000 píxeles que le componen, está descrito por un código numérico finalmente si un numero X en la matriz, representa un píxel de la foto estos tendrán coincidencia de posición (fila, columna) en la matriz y la foto respectivamente, como lo vemos en la figura 1. Es esta matriz representativa la que en últimas se albergara en un dispositivo de memoria y por lo tanto sobre ella deberá recaer nuestro análisis.

Fecha de Recepción: 31 Mayo de 2005
Fecha de Aceptación: 15 Noviembre de 2005

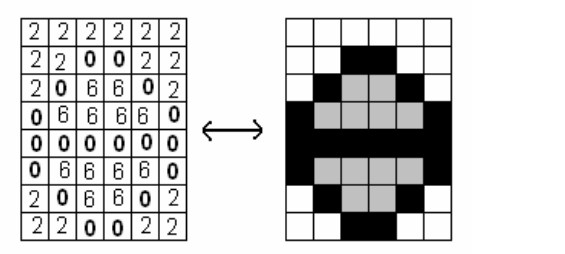

Figura 1. Relación entre imagen de píxeles y matriz numérica

Si es esta matriz la que debemos albergar; es nuestro objetivo buscar una forma astuta de llevar o "recordar" esa matriz sin necesidad de mantenerla conforme es, y así disminuir el tamaño de almacenamiento. Nuestra idea se basa en lograr encontrar una agente de tamaño muy pequeño en comparación con la matriz, que la recuerde en su "memoria" sin necesidad de tenerla escrita o fija en algún lugar tangible. Veremos más adelante que este problema será de búsqueda mas no de existencia puesto que se garantizará que para cualquier matriz siempre habrá un agente con capacidad de recordarla, pero este será difícil de hallar.

Para sumergirnos más en el problema, no solo debemos empezar a mirar la foto como una matriz representativa de ella, sino también trataremos de darle otra forma más fácil de operar matemáticamente. Si nos detenemos un momento podemos ver con claridad que esta matriz puede ser descrita como una función matemática tanto en R^2 como en R^3, de la siguiente manera:

Para R^2 podemos ver cada posición fila columna de la matriz como una posición simple, comenzando desde la primera posición superior derecha (0,0) y terminando en la posición inferior izquierda (m,n) como posición 1, 2,3… Así una matriz de m filas x n columnas, contará con m*n=k posiciones, cada una de las cuales tendrá determinado color o número, entonces podemos visualizar el color en función de la posición que este tiene en la matriz: **C(p),** como se ve en la figura 2, podemos también asumir esta función en R^3 sin necesidad de cambiar la fila, columna a una posición simple, puesto que cada par de dos dimensiones (f,c) tiene de por sí asociado un valor entero de color y así lograríamos conformar una superficie**,** como se ve en la figura 3.

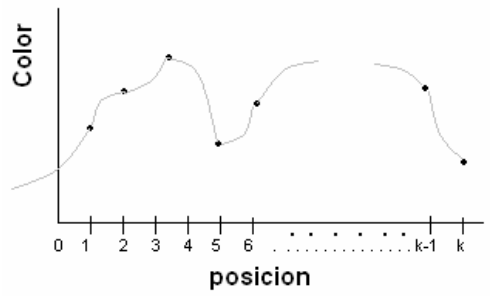

Figura 2. Curva objetivo en R^2

En cualquiera de los dos casos como superficie o como curva en este texto denominaremos dicha función con el nombre $\mathbf{C_m(p)}$ donde el subíndice m, se refiere a la discontinuidad de la función, y para efectos momentáneos de nuestro análisis trabajaremos con la función curva y no superficie; teniendo siempre presente que nuestro análisis será válido tanto para 2 como para 3 dimensiones.

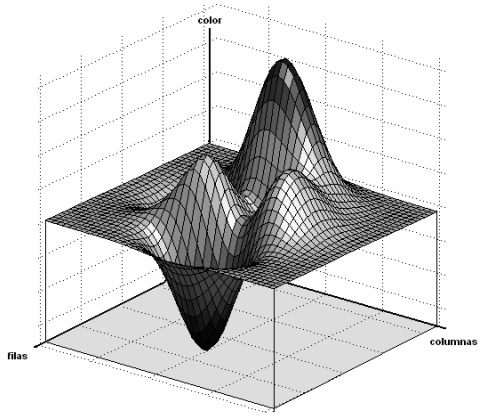

Figura 3. Superficie objetivo en R^3

Es por supuesto lógico que C(p) la función continua, necesariamente contiene a $C_m(p)$ en sus valores enteros del dominio, así que si halláramos la ecuación matemática de no mas de una línea y que describe totalmente a la curva C(p), por ejemplo: Sen(p), ln(p), 3p+2 … podríamos llevarla en un disco en lugar de llevar una inmensa matriz que consumirá exponencialmente mas memoria. Luego en el computador destino, interrogaríamos a C(p) en sus valores enteros y así recuperaríamos por medio de $C_m(p)$ aquella matriz que nos proporcionara por último la imagen deseada.

Como podemos observar en este momento, no es una tarea fácil hallar la ecuación matemática que necesitamos, al menos en la practica, pero también es inmediato el pensamiento alternativo de no encontrar dicha ecuación estrictamente, sino simplemente interpolarla o aproximarla y es aquí donde se le debe echar mano a un interpolador de alto nivel como lo es una red neuronal artificial.

3 RED NEURONAL Y LA FUNCION $C_m(p)$

Para quien tenga una pequeña noción del entrenamiento de una red neuronal, le será claro que $C_m(p)$, forma en si misma un set ya completo de entrenamiento de K entradas con K salidas deseadas, y con esto dicho podemos armar la idea concreta de lo que vamos a intentar hacer: Entrenaremos la RNA para que aprenda nuestra función, cuando este hecho se consiga, almacenaremos en disco solo la pequeña matriz de pesos final del entrenamiento, y ya en el computador destino le preguntaremos a la red por su conocimiento en valores discretos, además necesitamos un espacio adicional para

guardar en memoria el dominio de $C_m(p)$ para poder saber hasta que valor le preguntaremos a la red.

Hasta este punto hay una serie de detalles claramente deducibles y que son importantes en nuestro análisis:
Si el mapeo que realiza la función es de R→R, la RNA ya tiene fija su capa de entrada y salida en numero de neuronas (una y una), la red deberá poseer un aprendizaje reforzado y tal vez un sobre entrenamiento [1], como lo indica la figura No 5, pues es claro que nunca necesitaremos que la red generalicé a C(p) solo que memorice el set $C_m(p)$, con un error pequeño esto se debe al hecho que nunca vamos a interesarnos por valores diferentes a los del set, así que en valores intermedios no discretos (reales) o que no son del dominio de $C_m(p)$, no nos interesa como la red aproxime, así como lo muestra la figura No 4.

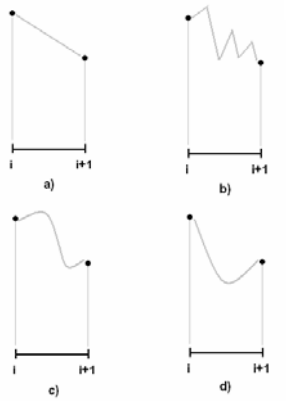

Figura 4. Cualquier 4 formas aceptables de aproximación de la red para valores no discretos entre el entero i e i+1

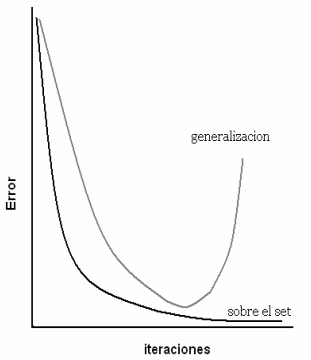

Figura 5. Comportamiento del error en una RNA en función de las iteraciones de entrenamiento

Por ultimo es consolador para nosotros saber que los errores de la red no serán fatales en la recuperación de la imagen, o sea que la red admite un apreciable margen de error. Pues debido a la implementación de la librería allegro, esta nos proporcionará una gama numérica de tonos, consecuente y suavizada, escala que en ultimas será el rango de nuestra función $C_m(p)$ (los colores). Así pues si en esta escala el numero 32000 representa un color azul y nuestra red lo aproxima como 32002, esto no será problema ya que ese numero deberá representar un mismo azul tal vez un poco mas claro u oscuro, garcías a la ya mencionada suavización de la escala. Por lo tanto a la final no se afectará la visual del usuario, cuando esto ocurra.

4 PROBLEMAS ADICIONALES

Efectivamente surge una inevitable serie de problemas en nuestro sistema hasta ahora un tanto ideal, estos giran casi todos alrededor de la idea de que nuestro sistema no puede ser estático y por el contrario deberá poseer un grado de dinamismo único. A continuación una lista de problemas con una idea de su posible solución.

4.1 Consumo de tiempo en el entrenamiento.

Hay que saber que por cada foto que procedamos a almacenar, nuestra red se deberá entrenar para aprenderla o recordarla, lo cual significa un entrenamiento on-line no común en los sistemas tradicionales de redes donde primero se entrenan y luego se ponen en marcha por "siempre". Además preocupa un poco el hecho ya enunciado y explicado en la figura No 5, que nos indica el sobreentrenamiento de la red que traduce mayor cantidad de iteraciones y a la final mayor tiempo en línea; para contrarrestar este efecto de consumo de tiempo podemos esgrimir dos ideas, la primera es el consumo de tiempo ya justificado de cualquier compresor y la segunda es el margen de error que podemos soportar en la red, o sea con mas error menos entrenamiento y con menos entrenamiento menos iteraciones y consumo de tiempo. A la final lo que nos resta es hacer un buen balance de tiempos para generar un sistema que obviamente consumirá el recurso, pero que además no abusara de este variablemente (sistema estable).

4.2 Arquitectura de la red neuronal.

Esta es la verdadera piedra angular de nuestros problemas, y en realidad la verdadera causa del inaceptable estatismo en del sistema. Puesto que para muchos es ya obvio que la arquitectura capas-neuronas de una red no es nunca constante y en ocasiones ni siquiera depende del problema, pues por lo general es dada por la experiencia del entrenador. Aún mas problemática es la idea que a lo mejor para cada foto necesitemos una red o arquitectura distinta, además sabemos que el proceso de fijación de arquitectura es un proceso de ensayo y error a veces tedioso y demorado. Si ya tenemos la idea de lo dicho, debe ser claro que este problema merece una solución de dimensiones grandes como lo es él, pero es también claro que este problema solo surge si la que utilizamos es una red multicapa de algoritmo backpropagation, ya que como veremos más adelante otras redes no presentarán de manera tan sesgada este problema de arquitectura. Por el momento se propone una posible solución en la que se está trabajando actualmente y que obviamente le da más nivel a nuestro compresor para una multicapa Bp, pues si el problema es de necesidad de experiencia para determinar el tamaño y arquitectura de la red en línea, quien mejor para realizar esta tarea que un sistema experto [3]. Un sistema experto como el que estamos desarrollando, programado en prolog, podrá contar con la suficiente experiencia para determinar la arquitectura de red para alguna foto en común.

Por el momento necesitamos una demostración para convencernos de que el sistema es realizable y para ello generaremos nuestra primera compresión de imagen con una red Bp multicapa fija, que solo nos podrá comprimir una foto también fija, así nos despreocuparemos

momentáneamente de los anteriores problemas mencionados.

5 PRUEBA PILOTO DE COMPRESION ESTATICA, CON RED Bp MULTICAPA E IMAGEN CONSTANTE.

La imagen a comprimir es una imagen de 800x600 píxeles, con buen desvanecido y profundidad de colores, El consumo inicial de memoria de esta imagen, es:
Tamaño en disco: 1.37 MB (1.441.792 bytes).
Tamaño en disco con Winzip: 432 KB (442.368 bytes).
Tamaño en disco con Winrar: 416 KB (425.984 bytes).
En este caso $C_m(p)$, es una función con dominio comprendido entre 0 y 480000 y un rango de 0 y 64000, cabe resaltar que su dominio es entero o sea que la cantidad de componentes de este son exactamente 480000.

Nuestra red neuronal después de mucha experiencia y ensayos humanos (tarea de un sistema experto), finalizo con una estructura de 4 capas 1-100-50-1, que contará con una cantidad de pesos a almacenar igual a 100+100*50+50=5150 y cantidad de bias de 1+100+50+1=152, en total para detallar esta red en otro computador debemos almacenar en disco un total de 5150+152=5302 números reales o float en C++, además de dos enteros más: el del dominio y el de cantidad de filas de la foto. Ya podemos ver que la cantidad de números a almacenar para la red (5304) es muchísimo menor que la cantidad de números a almacenar con la matriz de la foto (800*600=480000).

La red ineficientemente consumió en tiempo de entrenamiento alrededor de 70 minutos aunque obligándola a un error de magnitud pequeñísimo, lo que ahora tal vez nos inclina más a pensar que: ¿este no es el tipo de red que necesitamos, si no seguramente el que veremos más adelante u otro?.

Finalmente los resultados arrojados por la red fueron en cuanto a su almacenamiento los siguientes:

Tamaño en disco: 32.0 KB (32.768 bytes). Archivo. FIL de c++.

Cifra que se podría reducirse aun más sustancialmente si irónicamente la comprimiéramos en zip o rar. En este momento son aceptables los resultados del sistema, pero hay que recordar que este solo es un sistema estático que esta en capacidad de almacenar solo la foto de la figura No 6, y ninguna otra aunque ¿tal vez si a cualquiera del mismo tamaño?... También es necesario analizar los resultados en tiempo de compresión para cada sistema:
Zip:1 ns
Rar: 1ns
RNA: 1 hora 10 minutos (sin contar el tiempo de búsqueda de la arquitectura que fue mayor)

En las siguientes figuras veremos la imagen verdadera y la recuperación lograda por la red, podremos notar que la recuperación es excelente y que comparando el consumo de memoria de nuestra técnica, con el zip o el rar, definitivamente es animante pensar en que podemos llegar a pulir nuestro sistema hasta el punto de concebirlo tal como lo deseamos, dinámico y bajo en gasto de tiempo:

Figura 6. Imagen original

Figura 7. Imagen recuperada por la red neuronal artificial

6 LA SOLUCION: REDES NEURONALES DE BASE RADIAL Y TECNICAS DE CLAUSTERIZACION O AGRUPAMIENTO…

Y si lo que buscamos no es nada más que interpolar una función, pues por que no utilizamos entonces una red que sea hecha especialmente con ese fin, tal y como lo es la red de base radial, además sería mucho mejor si pensáramos en optimizar esta red con técnicas de agrupamiento fuzzy. Veamos:

6.1 Red de Base radial.

Sus principales características son [1]:

Facilidad para entrenar.
Rapidez para entrenar.
Aproximadores universales de funciones (potencial).
Estructura de capas fijas (3).
Y con este solo de entrada ya vemos que esta red empieza a aclarar el panorama de todos nuestros problemas, además esta red en una de sus capas no contiene bias y en la primera capa todos sus pesos son iguales a 1; tiene un aprendizaje hibrido supervisado y no supervisado fácil de manejar. En conclusión podemos visualizar casi por completo la estructura de la red para cualquier foto en la siguiente figura, a excepción del número de campanas o de neuronas de la capa oculta, parámetro que se determina en el entrenamiento de la misma.

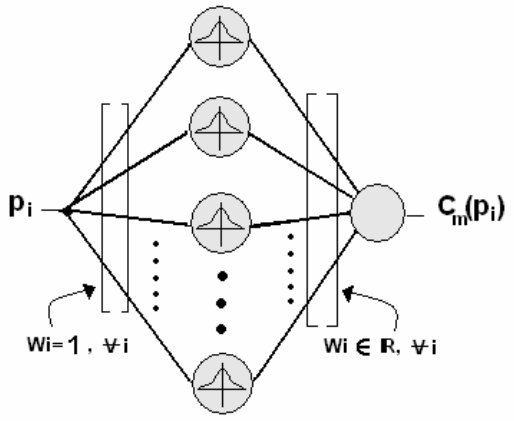

Figura 8. Arquitectura casi-fija, RBR para imágenes

Esta red aproximará nuestra función objetivo, por medio de campanas de gauss, puestas todas justo bajo la curva, de esta manera la gráfica de la función descansará sobre la serie de campanas consecutivas que tendrá la red, Figura 9, o sea que los parámetros de almacenamiento para esta red serán, los pesos de la última capa, el centro de las campanas y el ancho de cada una.

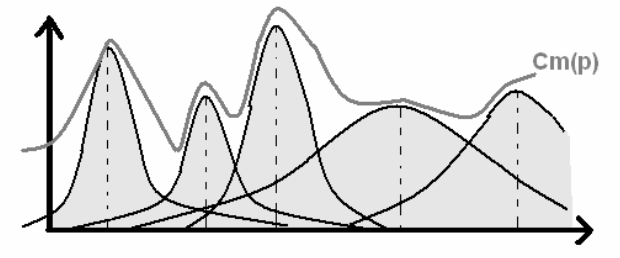

Figura 9. Manera de aproximar una función con un RBR

6.2 Entrenamiento de Base radial.

El aprendizaje de esta red consiste en determinar ciertos parámetros modificables de la red que son:

Número de campanas o neuronas de la capa oculta.
Centros de las campanas de la capa oculta.
Ancho de dichas campanas (dispersión).
Pesos de la capa de salida.

Como ya mencionamos con anterioridad esta red es de aprendizaje hibrido y por lo tanto se divide en dos etapas cada una de las cuales se dedica a hallar algunos de los parámetros antes mencionados, de la siguiente manera:

a) la etapa de aprendizaje no supervisado: Se encarga de encontrar número de neuronas, centros y anchos.
b) La etapa de aprendizaje supervisado: Se encarga solo de buscar los pesos de la capa final.

Ahora queda solo atacar cada uno de estos ítems, y en el proceso describiremos su repercusión sobre el contexto de nuestro problema.

6.2.1 Número de campanas o neuronas.

Existen dos criterios claros y matemáticos como son la familia de heurísticas y los índices de validación o análisis de datos.

En las heurísticas podemos encontrar dos técnicas como son:
Método incremental: partiendo de un numero de neuronas se debe incrementar hasta que todo el espacio de entrada este cubierto, por campanas. Si algún punto no esta cubierto por las campanas existentes, agregar una más. La ventaja de este método es la rapidez.
Método de definir cada patrón como una campana: Cada patrón de entrenamiento o sea cada uno de los elementos del dominio de $C_m(p)$, será una campana, este método es bueno ya que nos ahorrará más espacio, pues ya no debemos guardar los centros, en el disco.

Por su parte las técnicas de **validación**, son un poco más complejas y su fin es encontrar características entre patrones, para unirlos dentro de una misma campana. Estas técnicas de validación se apegan a las de clausterización [2] con el fin de hallar en el espacio de los patrones de entrada, las particiones que dependen de los rasgos de los patrones, y los centros de éstas.

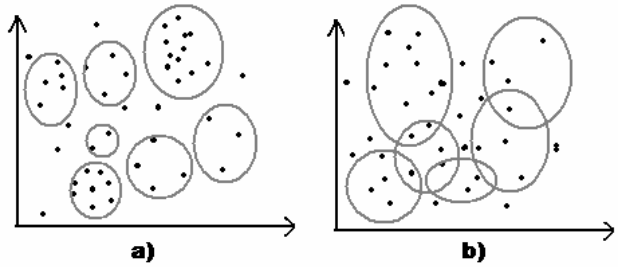

Figura 10. a) Partición dura. b) partición difusa

Estas particiones además pueden ser duras, cuando un patrón solo pertenece a una partición, o difusas cuando un patrón puede estar en varias particiones ya que comparte características de otras. Un índice de

validación bastante conocido es el de Xie-Beni [2] el cual trabaja con particiones difusas, y define una función objetivo con dominio de n/3 siendo n el número de patrones, esta función es minimizada con técnicas matemáticas para hallar el número mínimo y óptimo de particiones o campanas a generar [2].

6.2.2 Centros de las campanas o neuronas.
Los dos métodos de clausterización de lógica difusa más utilizados son:
HCM (hard clauster mean)
FCM (Fuzzy clauster mean)

HMC: para particiones duras, define la función objetivo como la distancia euclidiana entre los centros y los datos. De manera que los centros son aquellos puntos que minimizan la distancia entre ellos y cada uno de los elementos de la clase o la partición. Para este fin recurre a la siguiente formula (1).

$$\mathbf{C_j = (\sum_i \mu_j(X_i)* X_i) / (\sum_i \mu_j(X_i))} \quad (1)$$

FCM: cumple el mismo objetivo para particiones difusas, además lo hace con la misma técnica, y con la siguiente formula (2).

$$\mathbf{C_j = (\sum_i \mu_j(X_i)^m* X_i) / (\sum_i \mu_j(X_i)^m)} \quad (2)$$

Donde para cada una de las dos técnicas:
$\mu_j(X_i)$: Es el valor de pertenencia de X_i a la partición j
X_i: Es el valor del patrón i-esimo de entrenamiento.
m: Factor de fusificacion o traslape.

6.2.3 Los anchos de las campanas.
Existen también varias técnicas para este criterio, contando con algunas heurísticas [2], pero un criterio muy utilizado y que nos proporcionaría mejor capacidad para nuestro problema de almacenamiento, será el de mantener el ancho de todas las campanas fijo y pequeño.

6.2.4 Pesos de la última capa.
Este problema es ya de la fase de entrenamiento supervisado, por lo tanto requerimos de nuestro set de entrenamiento ya adquirido, $C_m(p)$, y luego aplicar una de las técnicas comúnmente usadas para el entrenamiento de una red [1]. Esta es la del gradiente descendiente empleando específicamente la regla del ADALINE (3) ó con la pseudo-inversa [2].

$$\mathbf{W_{ji}(t+1) = W_{ji}(t) + \alpha * (C_m(p_i) * p_i)} \quad (3)$$

Siendo α la rata de aprendizaje.

7 CONCLUSIONES

Se puede concluir fácilmente del siguiente escrito, que el problema atacado con redes multicapa tipo backprogation (Bp), efectivamente será solucionado de una manera eficaz, hablando de consumo de memoria que en ultimas es nuestro principal objetivo, mas sin embargo no es irrelevante, el consumo desmesurado de tiempo para la solución, y quizás tengamos que echar mano de otras técnicas de IA [3], muy complejas como los sistemas expertos, para subsanar en algo dicho consumo. Por lo tanto se describe una solución alternativa muy apropiada como es la de la inclusión de una red RBR optimizada con técnicas de agrupamiento, solución que talvez si miramos con detalle no reduzca de manera tan impresionante el consumo de memoria como la anterior técnica, pero en cambio bajara la complejidad del camino a la solución optima, creando claro esta, un sistema de inteligencia artificial que consumirá menos tiempo.

Queda entonces planteado, para quien se interese más en el tema, el siguiente set de interrogantes:

Cual sistema es verdaderamente mejor?

Existen otras alternativas para pulir aun más cada uno de los sistemas?

Podríamos crear un hibrido entre los dos sistemas, para balancear el consumo de tiempo y memoria?

Es correcto concluir la falta de experimentación del método ante más casos de comprensión para dictaminar con precisión la bondad del mismo. Esto debido al carácter casi de puro planteamiento que posee el texto, en el cual se pretende más que demostrar, motivar a los estudiosos del tema, con alguna prueba piloto.

8 AGRADECIMIENTOS

El autor brinda sus más emotivos agradecimientos a Diego Gómez López y Consuelo Valencia por el desmesurado esfuerzo que sin interés realizan a diario para contribuir en la gestión de su vida. También a Johanna Muñoz por el acompañamiento y amor brindado a cada momento.

9 BIBLIOGRAFIA

[1]. B.M del Brio, “Redes Neuronales y sistemas difusos” 2da ed. Vol 1. Ed. Zaragoza España: Alfaomega Ra-Ma 2002.

[2]. J.R Hilera “Redes Neuronales Artificiales” Ira ed. Vol 1. Ed Madrid España: Alfaomega Ra-Ma 2000.

[3]. Nilson “Inteligencia artificial” Ira ed. Vol 1. Ed Madrid España: Alfaomega Ra-Ma 2000.