FUNDAMENTOS DE NEUROCOMPUTACION

Germán J. Hernández[1]
Luz Gloria Torres[2], Luis Fernando Niño[1]

Resumen

En este documento se realiza una introducción histórica al desarrollo de la neurocomputación, se estudian los fundamentos teóricos que describen el funcionamiento de los dispositivos computacionales neuronales, llamados también Redes Neuronales Artificiales (RNA's) y se menciona una gama de aplicaciones que tienen tales dispositivos en la solución de problemas en Ciencias e Ingeniería.

Introducción

Un dispositivo de cómputo cumple dos funciones fundamentales, la primera es memorizar o almacenar información y la segunda es calcular o procesar, ésto es, obtener un resultado específico a partir de alguna información suministrada o almacenada. La medida del desempeño de estas dos funciones se conoce como capacidad de computación del dispositivo. El estudio de la capacidad de computación de los dispositivos se denomina teoría de la computación. En la actualidad ésta se subdivide en tres ramas: la teoría "clásica" de la computación o teoría de la computación secuencial, la teoría de la computación concurrente o distribuida y la teoría de la computación masivamente paralela.

La teoría "clásica" de la computación se originó en los intentos por entender cómo se efectúan los cálculos en el cerebro humano. Existe una aparente naturaleza secuencial en la forma en que se realiza el procesamiento en el cerebro humano, esta hipótesis surge de la forma como se llevan a cabo las operaciones aritméticas (p.e. la suma). A partir de tal hipótesis se desarrolló una amplia gama de modelos de computación secuencial, entre los cuales está el introducido por Von Newmann [46], donde una computación es una secuencia de instrucciones que son ejecutadas para la solución de un problema, a esta secuencia de instrucciones se le llama *algoritmo*. Los modelos teóricos de la computación secuencial tratan de formalizar la noción de algoritmo y entre ellos tenemos la Máquina de Turing [2][31][40][45], la Teoría de la Funciones Recursivas [2][31][40], el λ-Cálculo de Church [39], las Gramáticas de Chomsky [31][39][40] y la Máquina RAM de Aho-Ullman [1][21].

La computación secuencial tiene limitaciones, i.e., existen problemas para los cuales no existe un algoritmo que permita solucionarlos y existen problemas que a pesar de tener solución algorítmica, necesitan recursos (en tiempo o espacio de almacenamiento) que exceden las posibilidades reales de solución. Los primeros son llamados problemas no-algorítmicos o no-solu-

1 Universidad Nacional de Colombia, Departamento de Ingeniería de Sistemas.

2 Universidad Nacional de Colombia, Departamento de Matemáticas y Estadística.

bles secuencialmente y los segundos son llamados problemas no-factibles. Debido a esto surge la necesidad de construir dispositivos de cómputo de mayor capacidad que los secuenciales, que permitan solucionar estos dos tipos de problemas de manera factible. El primer intento para lograrlo se hace a través de la computación distribuida o concurrente.

En la computación concurrente se interconectan varios dispositivos de cómputo secuencial que resuelven simultáneamente diferentes partes independientes de un mismo problema. En este tipo de computación se pueden resolver sólo problemas algorítmicos y tales que el algoritmo se pueda separar en partes independientes simultáneamente ejecutables, lo cual se tiene únicamente en casos muy especiales, ésto deja una gran parte de los problemas no-factibles y los problemas no-algorítmicos sin solución.

Entre los problemas no solubles por los tipos de computación, anteriormente descritos, se encuentra el reconocimiento de imágenes y voz, el cual es resuelto de manera muy eficiente por el cerebro humano. Por lo tanto se descarta la hipótesis inicial de un procesamiento secuencial en el cerebro. En realidad, los estudios en Neurofisiología han mostrado que el sistema nervioso central humano es de naturaleza masivamente paralela, es una red de células nerviosas (neuronas), altamente interconectadas que exhibe una capacidad de cómputo muy alta.

Todo ésto ha conducido al estudio y desarrollo de modelos y dispositivos de cómputo masivamente paralelo. Los más populares son las llamadas Redes Neuronales Artificiales (RNAs), las cuales pretenden emular las redes neuronales biológicas formadas por unidades simples no-lineales de procesamiento (neuronas), altamente interconectadas y que dan lugar a fenómenos dinámicos complejos. El desarrollo de tales modelos ha resultado del trabajo conjunto en diferentes disciplinas del conocimiento como Neurofisiología, Matemáticas, Física y Ciencias de la Computación. Al área de investigación que busca el establecimiento de modelos computacionales neuronales se le ha llamado Neurocomputación (NC) [1] [10] [11] [13] [18] [20] [27] [28] [30].

En la actualidad la teoría de la computación masivamente paralela incluye la Neurocomputación, el estudio de los Autómatas Celulares (AC) [7][17][19][41][44][47] y de las Redes de Autómatas (RA) [6][7][42][43], que constituyen el modelo más general conocido de este tipo de computación.

En este trabajo se tratará fundamentalmente la Neurocomputación, en 2 se hace una introducción histórica a su desarrollo, en 3 se presentan los fundamentos de redes neuronales artificiales, en 4 se estudia el aprendizaje o entrenamiento de RNAs y en 5 se enumeran algunas de sus aplicaciones.

Historia

La fuente de inspiración para el estudio de las RNAs es el sistema nervioso central humano (más exactamente el cerebro), sobre el que, aún en la actualidad, tenemos un escaso conocimiento, y del cual se están investigando ampliamente muchos aspectos. En varias de tales investigaciones se usan hipótesis y resultados obtenidos de la NC. En la medida en que la Neurofisiología y la Neurobiología nos brinden mayor información sobre el cerebro y su funcionamiento, en un futuro cercano, se podrán formular mejores modelos que reflejen su capacidad computacional.

El origen de la NC se ubica en 1943 con los trabajos de McCulloch y Pitts [32] quienes propusieron un modelo simple de una neurona artificial y probaron que un ensamblaje sincrónico de estas neuronas, bajo ciertas condiciones, tiene capacidad de computación (secuencial) universal, esto es, que puede realizar cualquier función que realice una Máquina de Turing o un computador secuencial de cualquier tipo. Durante los siguientes quince años, se hizo énfasis

en la lógica de estas redes, se consideraron como máquinas de estado finito y se analizó su capacidad de computación. Por otro lado, se desarrollaron algunas teorías en campo de la Neurociencia como la Neurodinámica o teoría neuronal de campos que trabajó en la formulación de ecuaciones diferenciales que describían los patrones de actividad electromagnética en el cerebro.

Alrededor de 1960, Rosenblat y su grupo [36] influyeron notablemente en el desarrollo de la NC. Ellos se concentraron en el desarrollo y estudio de una red llamada Perceptron capaz de resolver algunos problemas de reconocimiento de patrones relacionados con percepción visual. Más tarde, en 1969, Minsky y Papert [33] probaron que el perceptron era incapaz de realizar algunas tareas sencillas, tales como calcular la función o-exclusiva (XOR). Implícitamente, defendían la tesis de que todas las redes neuronales adolecían de la misma falla y que era mejor explorar otros campos de la Inteligencia Artificial. Debido al planteamiento de Minsky y Papert se abandonó el trabajo intenso en esta área por cerca de 20 años. Durante este lapso, sólo algunos se dedicaron al desarrollo de la teoría de las redes neuronales especialmente en lo relacionado con las memorias asociativas.

A partir de 1982 aparece en el escenario el prestigioso físico John Hopfield [22][23][24] con dos artículos que iniciaron un segundo período de intensa actividad investigativa en NC. Hoy, las redes neuronales no solo están revolucionando ideas fundamentales en ciencias de computación sino que se están convirtiendo en herramientas importantes en campos muy diversos como Ingeniería, Física, Matemáticas, Medicina, Biología, Estadística, Ciencia Cognoscitiva e Inteligencia Artificial, entre otras.

Fundamentos de redes neuronales artificiales

Las redes neuronales biológicas consisten de muchas células nerviosas comunicadas a través de un alto grado de interconexión. Por lo tanto, inicialmente se describirán las características esenciales de las neuronas biológicas lo cual conduce a un modelo formal de una neurona artificial.

Descripción de la neurona biológica

La Figura 1 muestra un dibujo esquemático de una neurona biológica simplificada. Una red de fibras nerviosas llamadas dendritas están co-

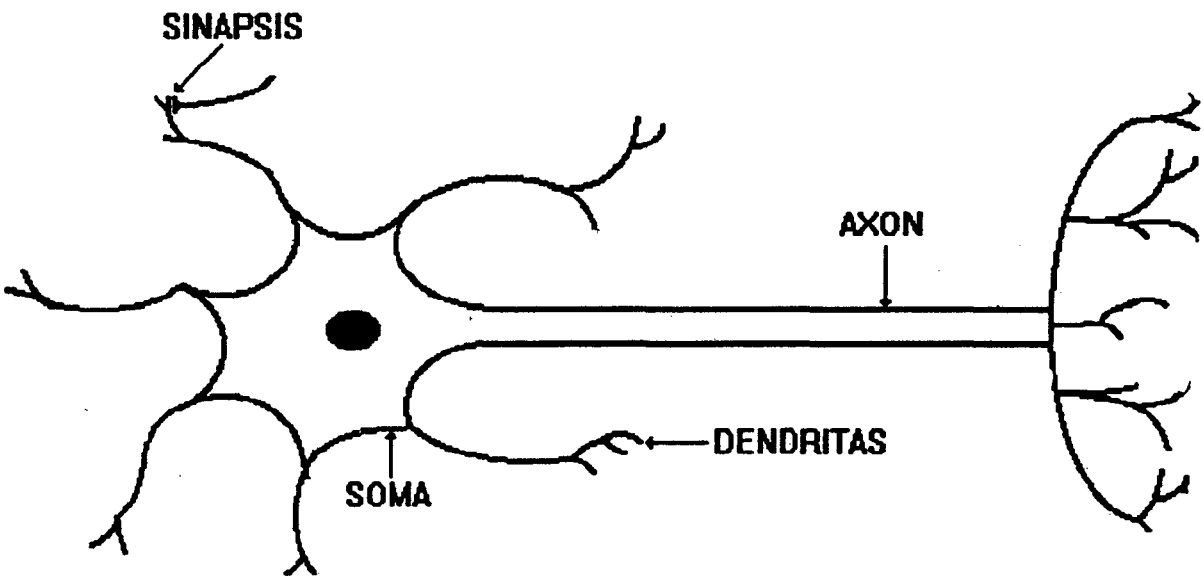


FIGURA 1. Diagrama exquemático de una neurona.

nectadas al cuerpo de la neurona llamado soma. Saliendo del soma de la neurona se encuentra una fibra sencilla larga llamada el axón, el cual se divide en una serie de fibras terminales.

Una neurona recibe impulsos de otras neuronas a través de las dendritas. Los impulsos que llegan son enviados por los terminales de los axones de las otras neuronas. A la conexión entre una de las fibras terminales del axón de una neurona y una de las dendritas de otra neurona se le llama unión sináptica o sinapsis. Cabe anotar que algunas sinapsis van de un terminal del axón de una neurona al soma de otra directamente. El axón de una neurona típica hace miles de sinapsis con otras neuronas.

La transmisión de una señal de una neurona a otra, a través de una sinapsis, es un complicado proceso químico, en el cual sustancias transmisoras específicas son enviadas del lado que transmite en la sinapsis. El efecto de tal sustancia es generar una baja en el potencial eléctrico al interior de la célula receptora. Si el nivel de potencial alcanza un nivel de umbral determinado, la célula envía un pulso o acción potencial de fuerza y duración fija a través de su axón, es decir, envía un pulso a todas las neuronas que están conectadas con ella. Cuando esto sucede, se dice que la neurona disparó. Después de disparar, la neurona se mantiene un tiempo fuera de actividad, este tiempo es llamado período refractorio. Una vez pasado este tiempo la neurona está en capacidad de volver a disparar.

McCulloch y Pitts [32] propusieron en 1943 un modelo de neurona o unidad binaria de umbral (binary threshold unit), inspirado en el comportamiento de las neuronas biológicas típicas. Tal modelo es explicado en seguida.

Modelo de neurona de McCulloch-Pitts

El modelo de neurona de McCulloch-Pitts, llamada también unidad binaria de umbral, calcula una suma ponderada (weighted sum) de las entradas que tiene desde otras neuronas y coloca en su salida uno o cero dependiendo de que tal suma supere o no un nivel de umbral asociado a la neurona. La Figura 2 muestra un diagrama esquemático de la neurona de McCulloch-Pitts.

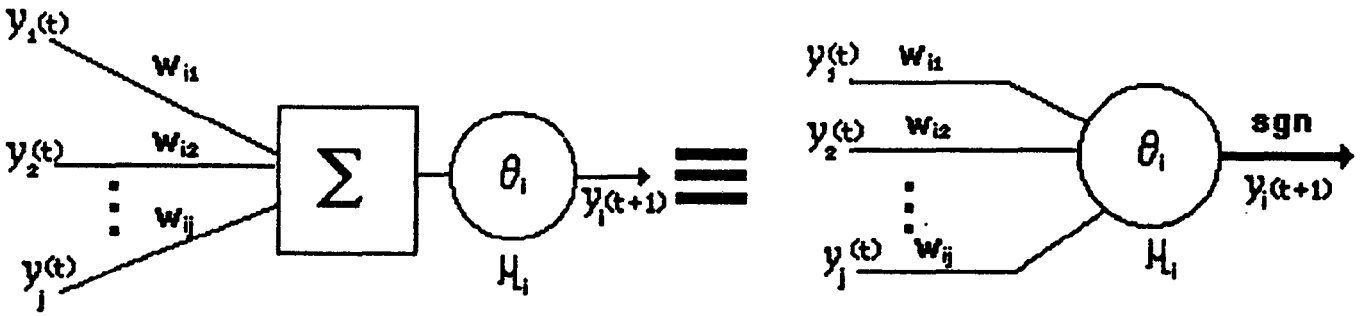


FIGURA 2. Diagrama Esquemático de la neurona de McCulloch-Pitts.

En la Figura 2, se supone que la neurona está interconectada con otras neuronas como ella y las entradas $y_1(t)$, $y_2(t)$,..., $y_j(t)$ corresponden a las salidas de tales neuronas en el instante t. Además, el valor que ella produce a su salida varía únicamente en instantes de tiempos discretos, considerando para su actualización sus entradas en el instante anterior. Se tiene que

$y_i(t+1)$: Salida de la i-ésima neurona en el instante $t+1$.

w_{ij}: Peso (fuerza) de conexión entre la salida de la j-ésima neurona y entrada de la i-ésima neurona.

μ_i: Umbral de disparo de la i-ésima neurona

$\theta_{i:R} \rightarrow R$: Función de respuesta de la i-ésima neurona

y la correspondiente ecuación del nodo

$$y_i(t+1) = \theta_i \left[\sum_i w_{ij}\, y_j(t) - \mu_i \right]$$

donde j toma todos los valores correspondientes a los números de las neuronas con las cuales está interconectada la neurona i y la función está definida así

$$\theta(x) = \begin{cases} 1 \text{ si } x \geq 0 \\ 0 \text{ en otro caso} \end{cases}$$

es llamada función paso-unitario o función de Heaviside, Figura 3.

Claramente, se observa que la función de respuesta de la neurona es no-lineal y, además, discontinua. La no-linealidad proviene del hecho que las neuronas biológicas tienen la propiedad universal de que su salida tiene una relación no-lineal respecto de su entrada.

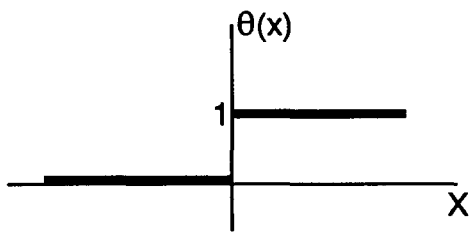


FIGURA3. Función paso-unitario o de Heaviside.

La neurona de McCulloch-Pitts es un sencillo dispositivo computacional, pero al interconectar varias de estas neuronas se tiene un poderoso dispositivo computacional, ya se mencionó que una red ensamblada con neuronas de este tipo y en la cual las neuronas se actualizan de manera sincrónica (todas se actualizan en cada instante de tiempo) tiene capacidad de computación universal. En otras palabras podría realizar cualquier computación que realice un computador secuencial digital, aunque tal computación con una red de neuronas puede no ser más rápida o conveniente.

Neuronas biológicas vs. Neuronas de McCulloch-Pitts

A continuación se hace un recuento de las características de las neuronas biológicas que recoge el modelo de McCulloch-Pitts y se hacen notar características de tales neuronas que no están reflejadas en el modelo.

(i) Las neuronas biológicas en general no son ni siquiera aproximadamente dispositivos de umbral como las describe el modelo de McCulloch Pitts. En realidad, ellas responden a su entrada de una manera continua, lo cual es llamado respuesta gradual. Existen modelos de neurona en los cuales la salida es continua y son llamadas unidades de valor continuo.

(ii) La no-linealidad de la relación entre las entradas y las salidas sí es una propiedad universal de las neuronas biológicas. Por ello, algunos autores llaman a las neuronas elementos no-lineales de procesamiento.

(iii) Las neuronas biológicas producen en su salida una secuencia de pulsos y no un simple nivel de salida n_i, y aunque este nivel de salida represente la frecuencia de disparo de una manera continua se estaría perdiendo información, p. e., con respecto a la fase de la señal. Cabe aclarar que muchos investigadores consideran que la fase de la señal no juega un papel fundamental en el comportamiento de las redes neuronales.

(iv) Las neuronas biológicas, en general, no tienen un mismo intervalo de tiempo luego del cual se actualizan, es decir, no existe un reloj central que gobierna la actualiza-

ción de todas las neuronas. En realidad, en las redes neuronales biológicas, las neuronas se actualizan de manera asincrónica.

Modelo de neurona de McCulloch-Pitts generalizado

Tratando de considerar algunas de las características no consideradas en el modelo inicial, se formuló el modelo generalizado de McCulloch-Pitts, que se resume en una modificación en la ecuación de nodo.

Ecuación del nodo McCulloch-Pitts Generalizado

$$y_i := g\left[\sum_j w_{ij} y_j - \mu_i\right]$$

donde g es una función no-lineal y continua, a la cual se le conoce como función de respuesta, función de activación, función de transferencia o función de ganancia. Nótese que en la nueva ecuación de nodo aparece ":=" que significa asignación, es decir, que la expresión de la derecha se le asigna a la variable de la izquierda cuando le corresponde a la neurona actualizar su estado, pero que la igualdad no es cierta en todo instante de tiempo.

Este modelo de neurona ya considera que la salida de la neurona es de valor continuo y que su actualización puede ser no sincrónica. En general, se supone que para todas las neuronas se tiene la misma función de activación, pero existen modelos de redes neuronales en los cuales no ocurre ésto.

A continuación se dan algunas funciones típicas que se usan como funciones de activación.

Función Rampa Unidad

$$r(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \in [0,1] \\ 1 \text{ si } x > 1 \end{cases}$$

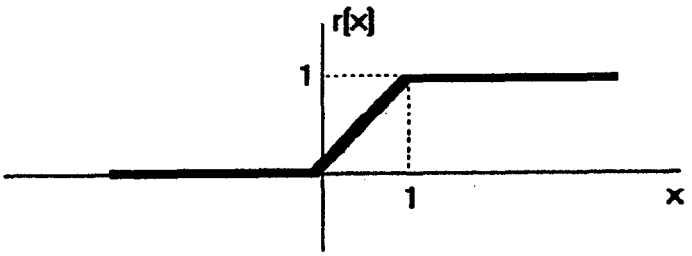


FIGURA 4.

Función Sigmoide

$$s(x) = \frac{1}{1 + e^{-x}}$$

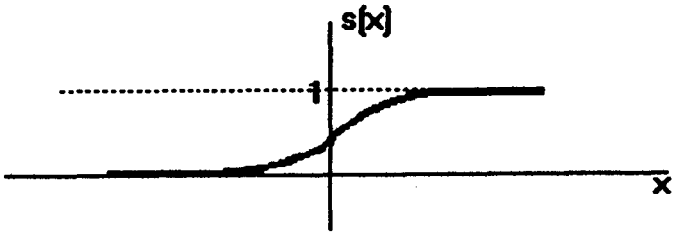


FIGURA 5.

Función Sgn (Signo)

$$sgn(x) = \begin{cases} -1 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$$

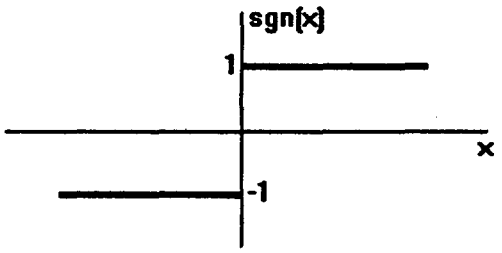


FIGURA 6.

Redes Neuronales Artificiales RNAs

Las RNAs son modelos en los cuales se tienen elementos computacionales simples (llamados neuronas, nodos, células o unidades) densamente interconectados. Tales elementos computacionales son en general no-lineales que pueden ser análogos o discretos. Estos elementos computacionales, en general, son neu-

ronas de McCulloch-Pitts o elementos que responden a modelos similares.

Componentes de una RNA

Una red neuronal queda determinada completamente describiendo tres aspectos de la misma:

(i) Nodos
La información de los nodos incluye la descripción del tipo de las entradas y salidas de los nodos o neuronas, p. e., si son discretas o continuas, enteras o decimales. También, incluye la descripción del tipo de los pesos (en general números reales), la definición de la o las funciones de activación de los nodos y la ecuación general de nodo.

(ii) Topología de la red
Descripción de la forma en que se interconectan los nodos p.e. si los nodos están organizados en niveles, si tienen conexión de retroalimentación, si es completa la red, es decir si todos los nodos están conectados con todos, etc.

(iii) Algoritmo de aprendizaje
Descripción de la forma de ajustar los pesos de interconexión entre los nodos, para que la RNA cumpla una tarea específica. Existen dos categorías de algoritmos de aprendizaje, los supervisados, los no supervisados y por refuerzo, en la sección 4 se trata con mayor profundidad este tópico.

Características de las Redes Neuronales

Las redes neuronales son robustas (tolerantes a fallas), debido a la alta conectividad en la red (para cada neurona se están sumando ponderadamente muchos términos en su entrada) se tiene que al presentarse errores en algunos de estos términos podría no tener consecuencias en el comportamiento global del sistema. Por ejemplo, en el cerebro mueren en todo momento neuronas sin que ésto afecte sus funciones, lo cual nos da una idea de lo robusta que puede ser una red neuronal. En contraposición, en un sistema convencional de computación secuencial, todo el funcionamiento del sistema puede arruinarse por un solo error o daño en el dispositivo.

Las redes neuronales pueden estar compuestas de elementos computacionales lentos, con respecto a los elementos en un sistema de computación secuencial, y sin embargo llevar a cabo tareas de procesamiento complejo de manera muy rápida: el ciclo de tiempo de las neuronas en el cerebro es del orden de milisegundos, es decir éstas son un millón de veces más lentas que los elementos de procesamiento en un computador digital, que son las compuertas lógicas construidas con semiconductores. Y como es bien sabido, el cerebro realiza rápidamente tareas complejas como visión, control motor y toma de decisiones basadas en datos incompletos o erróneos.

Las redes neuronales pueden cumplir tareas aunque la información que se les suministre esté incompleta o contenga algunos errores.

Definición formal de una RNA

Para definir formalmente una RNA se debe definir el conjunto del cual los nodos o neuronas toman sus entradas y salidas, este conjunto es el mismo del cual se toman los pesos de conexión entre nodos, se requiere que tal conjunto tenga definida una suma y un producto, que satisfacen ciertas propiedades, lo que en matemáticas se llama un anillo $A = (A, +, *)$ (en la mayoría de los casos se toma como A el conjunto de los reales R). A cada nodo v se le asocia una función de activación del anillo en el anillo $f_v : A \rightarrow A$, con ésto quedan especificadas las características de cada nodo. Por otro lado, para especificar la topología de una RNA, es decir, la forma en que están interconectados los nodos, se hace con un digrafo con arcos etiquetados, la etiqueta de un arco es el peso de conexión entre los correspondientes nodos que une el arco, este peso,como se mencionó antes, debe estar en el anillo A.

Una RNA se define formalmente como una tripla

$$RNA = (A, D, [f_i]_{i \in V})$$

con

(i) $A = (A, +, *)$ Anillo: Conjunto de donde se toman las entradas y salidas de los nodos y los pesos de interconexión.

(ii) $D = (V, \Gamma, E)$ Digrafo con arcos etiquetados: Vértices V, Conjunto de arcos Γ, y función E: $\Gamma \rightarrow A$ etiqueta de los arcos. En cada vértice $v \in V$ se encuentra un nodo o elemento de procesamiento; E asigna a cada arco en Γ el peso de conexión entre los correspondientes nodos que conecta el arco.

(iii) $\{f_i\}_{i \in V}$ conjunto de funciones: f_i: A $\rightarrow$ A de activación, una función para cada nodo, las funciones no son necesariamente distintas.

y la ecuación del nodo es

$$y_i := f_i \left[\sum_{(j,i) \in \Gamma} E[(j,i)] * y_j \right]$$

con y_i la salida de nodo $i \in V$.

En esta definición se incluyen las RNAs discretas y continuas, sincrónicas y asincrónicas, y, finitas o infinitas.

Aprendizaje o entrenamiento

Los procedimientos de aprendizaje en RNAs o aprendizaje conexionista se pueden dividir en tres clases fundamentales:

(i) Los procedimientos supervisados, los cuales requieren un "profesor" que especifique el vector de salida deseado.

(ii) Los procedimientos de refuerzo (reinforcement) que solamente requieren una evaluación escalar de la salida.

(iii) Los procedimientos no supervisados, los cuales construyen modelos internos en la red que capturan regularidades en sus vectores de entrada sin recibir ninguna información adicional.

Existen frecuentemente formas de convertir una clase de procedimiento de aprendizaje en otro. En la mayoría de las redes neuronales que tienen capacidad de aprendizaje, pero no en todas, el aprendizaje se lleva a cabo a través de la modificación de los pesos de los elementos de procesamiento, una forma de aprendizaje distinta podría ser que se modifiquen las funciones de activación. En seguida se desarrolla un modelo mental del proceso de modificación de los pesos.

Para este desarrollo se hacen las siguientes consideraciones:

(i) Los pesos pertenecen a los números reales

(ii) La RNA es finita, esto es, tiene n elementos de procesamiento con pesos que son modificados por una ley de aprendizaje. El vector de pesos de la red es el vector formado por todos los pesos de todos los elementos individuales de procesamiento.

Este vector de pesos se puede escribir así:

$$W = (w_{11}, w_{12}, \ldots, w_{1n}, w_{21}, w_{22}, \ldots, w_{2n}, \ldots, w_{n1}, w_{n2}, \ldots, w_{nn})$$

sea

$$W_i = (w_{i1}, w_{i2}, \ldots, w_{in})$$

tenemos

$$W = (W_1, W_2, \ldots, W_n)$$

donde los vectores W_1, W_2, ..., W_n son los vectores de pesos de los elementos de procesamiento 1,2,...,n respectivamente.

El valor de activación a_i del nodo i-ésimo (también notado net_i) es la suma de las entradas que recibe de otras neuronas en la red, ponderada por los pesos de conexión.

$$a_i = \sum_j w_{ij} y_j$$

El estado de activación de la red ***A*** es el vector de los estados de activación de cada una de los nodos.

$$A = (a_1, a_2, \ldots, a_n)$$

La salida de cada neurona *i*, notada c_i, se calcula como

$$c_i = f_i(a_i)$$

donde f_i es la función de activación de la i-ésima neurona. Al vector formado por las salidas de las neuronas se le llama configuración de la red.

$$C = (c_1, c_2, \ldots, c_n)$$

Si se toma el vector *F* como el vector de la funciones de activación de cada uno de las neuronas, compuesta con la proyección correspondiente, así:

$$F = (f_1 \circ p_1, f_2 \circ p_2, \ldots, f_n \circ p_n)$$

se puede calcular la configuración de la red a partir del estado de activación como

$$C = F(A)$$

Un patrón de entrada **x** es un vector de R^n que se coloca como configuración inicial de la red. A la configuración que se obtiene cuando se estabiliza la red, a partir de la configuración inicial **x**, se le llama el patrón de salida **y** correspondiente a **x**. Una red se estabiliza cuando su configuración no cambia al seguir operando. Se puede considerar también como patrón de salida aquel que se obtiene después de tiempo fijo de operación.

Dado un conjunto de parejas de patrones de entrada y patrones de salida correspondientes $\{(x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)\}$ se considera que la RNA ha aprendido si se han modificado los pesos de manera que cuando se coloca como patrón de entrada x_i la RNA produce como salida el patrón y_i. A continuación se presenta el algoritmo de aprendizaje más popular para una clase especial de RNAs.

Aprendizaje por retropropagación en RNAs multicapa

Formalmente una red neuronal *multicapa* está definida sobre un digrafo $D = (V, \Gamma)$ vértices *V* y arcos Γ acíclico (i.e. sin ciclos dirigidos o circuitos), unidireccional (i.e., todo par de vértices están conectados en por lo menos una dirección), m-partito (i.e., el conjunto de vértices tiene una partición en *m* subconjuntos $V_1, V_2, \ldots, V_m \subset V$ disyuntos $V_i \cap V_j = \emptyset$ *si* $i \neq j$, con $V = \bigcup_{i=1}^{m} V_i$ y tal que para todo arco $(u,v) \in \Gamma$, $u \in V_i$ y $v \in V_j$ con $i \neq j$. Cada uno de los V_k se llama una

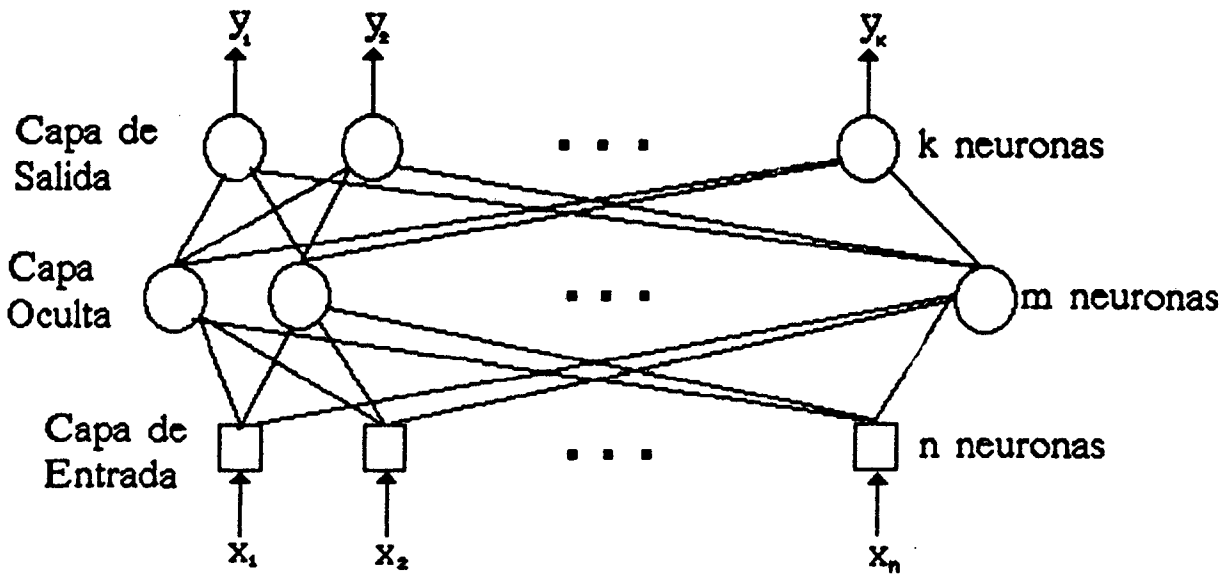


FIGURA 7. RNA multicapa típica.

capa). En la figura 7. se ilustran las conexiones en una RNA multicapa en la que cada nodo en una capa está conectado con todos los nodos de la capa precedente.

Se ha probado que solo se requieren tres capas para muchas tareas de aprendizaje de patrones, la primera capa se llama capa de entrada, la segunda capa oculta y la tercera capa de salida. Si se considera que las conexiones entre capas son totales, tenemos la matriz de pesos entre las neuronas en la capa de entrada con n neuronas, numeradas $1,2,...,n$ y la capa oculta con m neuronas, numeradas $1,2,...,m$, notada W_H (H por hidden).

$$W_H = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ \vdots & \vdots & & \vdots \\ w_{21} & w_{22} & \dots & w_{2n} \\ w_{n1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

donde w_{ji} es el peso de la conexión entre la i-ésima neurona de la capa de entrada y j-ésima neurona de la capa oculta. Si se coloca en las neuronas de la capa de entrada un vector $x \in R^n$ y se considera que $F_H = (f_1 \circ p_1, f_2 \circ p_2, ..., f_m \circ p_m)$ es el vector de las funciones de activación de las neuronas en la capa oculta compuestas con las respectivas proyecciones, se tiene que a la salida de las neuronas de la capa oculta se tiene el vector dado por

$$h = F_H (W_H x)$$

La matriz de pesos entre las neuronas en la capa oculta con m neuronas y la capa de salida con k neuronas, numeradas $1,2,...,k$, W_O (O por output) se define así

$$W_O = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{k1} & w_{k2} & \dots & w_{km} \end{bmatrix}$$

donde w_{ji} es el peso de la conexión entre la i-ésima neurona de la capa oculta y j-ésima neurona de la capa de salida. Si se coloca en las neuronas de la capa de entrada un vector $x \in R_n$ y se considera que $F_O = (f_1 \circ p_1, f_2 \circ p_2, ..., f_k \circ p_k)$ es el vector de las funciones de activación de las neuronas en la capa de salida compuestas con la respectiva proyección, se tiene que a la salida de las neuronas de la capa de salida se tiene el vector $y \in R^k$ dado por

$$y = F_O (W_O h) = F_O (W_O F_H (W_H x))$$

En una red multicapa se considera que los patrones de entrenamiento son un conjunto de parejas $\{(x_1, y_1), (x_2, y_2), ..., (x_r, y_r)\}$ donde los valores que se colocan en la capa de entrada y y_i son las salidas de las neuronas de la capa de salida al propagar la entrada x_i,.

Un aprendizaje heurístico robusto para RNA multicapa llamada la regla delta generalizada (RDG) o regla de aprendizaje por retropropagación (backpropagation learning rule) fue propuesta por Rummelhart y otros [38]. La RDG es una técnica basada en el descenso usando la dirección del gradiente para minimizar una función objetivo $E(W)$ que es diferenciable con respecto a cada uno de los pesos en la red (si la función objetivo es también una función de la salida de la red, esto implica que cada función de activación es diferenciable con respecto a los pesos correspondientes).Sin embargo, la principal contribución de Rummelhart, fue proveer una técnica de optimización distribuida basada en el descenso, usando la dirección del gradiente, tal que las expresiones de descenso que usan el gradiente aparecen como reglas discretas de actualización de los pesos,que utilizan únicamente información localmente disponible a cada nodo y una cantidad que es retropropagada desde los nodos en la penúltima capa. A continuación se describe la deducción de la regla y luego se explica cómo se aplica la regla.

Como ya se dijo atrás, la activación o entrada neta aj al nodo j es una función lineal de las salidas yi, de los nodos que están conectados al nodo j-ésimo y de los pesos wji de tales conexiones

$$(1) \qquad a_j = \sum_i y_i w_{ji}$$

Tenemos que la salida del nodo j-ésimo, y_j es de valor real y es una función no lineal de su entrada total, más exactamente se supone que la función de activación de todas las neuronas es la función sigmoide.

$$(2) \qquad y_j = \frac{1}{1 + e^{-aj}}$$

El propósito es encontrar un conjunto de pesos que asegure que para cada vector de entrada, el vector de salida producido por la red, sea igual a (o lo suficientemente cercano a) el vector de salida deseado. Si hay un conjunto fijo finito de casos entrada-salida, el error total en el desempeño de la red con un conjunto de pesos particulares se puede calcular comparando los vectores de salida deseados y los reales para todos los casos. El error total, E, es definido como

$$(3) \qquad E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2$$

donde c es un índice sobre los casos (parejas entrada-salida), j es un índice sobre los nodos de salida, y_j es el estado real de un nodo de salida y d_j es su estado deseado.

Para minimizar E usando la dirección del gradiente se necesita calcular la derivada parcial de E con respecto a cada peso en la red. Esto es simplemente la suma de las derivadas parciales para cada uno de los casos entrada-salida.

Para cada caso dado, las derivadas parciales del error con respecto a cada peso son calculadas en dos pasos. El primer paso (paso directo), ya descrito, consiste en obtener los estados de los nodos en cada capa, determinados por la entrada que reciben de los nodos en la capa precedente, usando las ecuaciones (1) y (2). El segundo paso (paso inverso) propaga las derivadas desde la última capa, hacia atrás, hasta la primera capa. El paso inverso empieza calculando

$\frac{\partial E}{\partial y}$ para cada nodo de la capa de salida.

Diferenciando (3) para un caso particular, c, y suprimiendo el índice c da

$$\frac{\partial E}{\partial y_j} = y_j - d_j$$

Se puede aplicar la regla de la cadena para calcular

$$\frac{\partial E}{\partial a_j}$$

obteniendo

$$\boxed{\frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial y_j} \frac{dy_j}{da_j}}$$

Diferenciando la ecuación (2) se obtiene

$$\frac{dy_j}{da_j} = y_j (1 - y_j)$$

y reemplazando en la ecuación anterior, se tiene:

$$\frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial y_j} y_j (1 - y_j)$$

por tanto aquí se sabe cómo un cambio en una entrada total a para un nodo de salida afectará el error.

Ahora, para un peso w_{ji}, de i a j la derivada es

$$\boxed{\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \frac{\partial E}{\partial a_j} y_i}$$

usando (1), y para la salida del i-ésimo nodo la contribución a

$$\frac{\partial E}{\partial y_i}$$

que resulta del efecto de i sobre j es simplemente

$$\frac{\partial E}{\partial y_i} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial y_i} = \frac{\partial E}{\partial a_j} w_{ji}$$

por tanto teniendo en cuenta todas las conexiones que salen de la unidad i se tiene

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial a_j} w_{ji}$$

Aquí se acaba de ver cómo calcular $\partial E/\partial y$ para cualquier nodo en la penúltima capa cuando se

tiene $\partial E/\partial y$ para todos los nodos en la última capa. Por tanto se puede repetir este procedimiento para calcular este término para las primeras capas restantes.

Se puede usar $\partial E/\partial w$ para cambiar los pesos después de cada caso entrada-salida. Un esquema alternativo es acumular $\partial E/\partial w$ sobre todos los casos entrada-salida antes de cambiar los pesos.

La versión más simple del descenso usando el gradiente, es cambiar cada peso en una cantidad proporcional al acumulado $\partial E/\partial w$.

$$\Delta w = - \varepsilon \frac{\partial E}{\partial w} \text{ para } \varepsilon > 0 \text{ (pequeño)}$$

este método no converge muy rápidamente, pero puede mejorarse significativamente introduciendo un término de aceleración, y se obtiene

$$\Delta w\,(t) = - \varepsilon \frac{\partial E}{\partial w}(t) + \propto \Delta w\,(t-1)$$

donde t se incrementa en uno después de usar todo el conjunto de casos entrada-salida, y es un factor de decaimiento exponencial entre 0 y 1 que determina la contribución relativa del gradiente actual y los anteriores al cambio en los pesos.

Aplicación de la regla delta generalizada

$$E(w) = \sum_c E_c = \sum_c \frac{1}{2} \left| y_c - d_c \right|^2$$

Para la entrada x_c la salida dada por la red es y_c y la deseada es d_c.

1. La RDG empieza inicializando todos los pesos de conexiones y sesgos (biases) a valores reales pequeños entre -0.5 y 0.5 escogidos aleatoriamente.

2. Se presenta un patrón de entrenamiento en las entradas de la red y es propagado hacia adelante para producir y_c.

3. Si el error para el patrón presentado (caso entrada-salida) es menor que una tolerancia de aprendizaje (p.e. 0.01) ningún cambio en los pesos ocurre. En otro caso, los pesos son ajustados según:

$$\Delta_c w_{ji} = \eta \delta_{cj}\, y_{cj} + \propto \Delta_{c-1} w_{ji}$$

donde, $\Delta_c w_{ji}$ es el cambio en el peso del nodo i al nodo j cuando se presenta el patrón c, η es un término de ganancia (rata de aprendizaje) $\propto$ y es un término de momento, que suaviza el efecto de cambios dramáticos en los pesos, adicionando una fracción del cambio en el peso, $\Delta_c - 1 w_{ji}$ en la presentación del anterior patrón, $c-1$. La señal de error δ_{cj} es una medida de la distancia del nivel de activación del nodo j a su nivel deseado.

4. La RDG provee dos reglas para calcular la señal de error de un nodo

(i) Para un nodo de salida j:

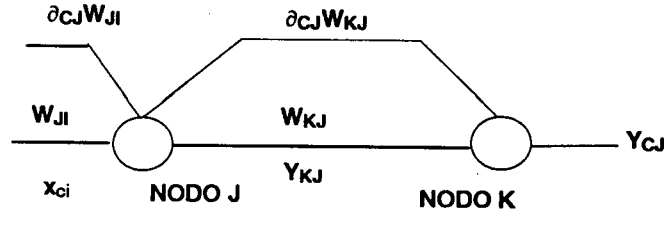
$$\delta_{cj} = (d_{cj} - y_{cj}) \frac{df_j\,(a_{cj})}{da_{cj}}$$

(ii) Para un nodo j en una capa escondida:

$$\delta_{cj} = \frac{df_j}{da_{cj}} \sum_k \delta_{ck} w_{jk}$$

la suma se hace sobre todos los nodos k a los cuales el nodo j envía su salida.

FIGURA No. 8



Al aplicar las derivadas se tiene:

$$\delta_{cj} = (d_{cj} - y_{cj})\, y_{cj}\, (1 - y_{cj})$$

si j es nodo de salida

$$\delta_{cj} = y_{cj}\, (1 - y_{cj}) \sum_k \delta_{ck}\, W_{jk}$$

en otro caso

Como el nombre retropropagación sugiere, la idea básica detrás de este cálculo de señales de error para los nodos ocultos es propagar los errores hacia atrás, basados en la discrepancia observada entre los valores de los nodos de salida y los esperados para un patrón de entrenamiento, ver figura 8. Nótese que en la RDG los pesos se cambian cuando se presenta cada patrón de entrenamiento.

Aplicaciones de las redes neuronales

En la Física, las redes neuronales constituyen una herramienta para modelar los fenómenos microscópicos de los sistemas físicos. Un tipo de red neuronal llamado autómata celular [40][43][46] provee un modelo discreto simple para modelar una amplia variedad de fenómenos naturales y complejos como los modelos Greenberg-Hasting para la reacción de Belusov-Zhabotinsky, el modelo digital de los billares para la mecánica conservativa, los modelos HPP-GAS, TM-GAS y FHP-GAS de la dinámica de fluidos para la ecuación de Navier-Stokes, los sistemas magnéticos y muchos otros.

En Ingeniería se puede utilizar una red de Hopfield para la reconstrucción de imágenes digitales[3][27], para la implementación de circuitos eléctricos, para implementaciones ópticas. Por otro lado, se puede utilizar una red de Boltzman para resolver problemas de optimización clasificados como NP en la teoría de la computación, para resolver problemas de optimización en procesamiento de imágenes.

En las Matemáticas, las redes neuronales constituyen por sí mismas un objeto de estudio: los algoritmos de aprendizaje no son otra cosa que técnicas de optimización de una función costo. En su operación básica, una red multicapa aproxima cualquier función de interés. Los problemas definidos por iteraciones locales de las funciones de activación que inducen transformaciones globales de un espacio multidimensional en otro son temas de creciente interés en sistemas dinámicos.

En Estadística, las redes neuronales se pueden tomar como método alternativo (no lineal) para clasificación, análisis discriminante, regresión y predicción con resultados satisfactorios.

En Biología y Neurofisiología se pueden modelar sistemas nerviosos de organismos primitivos, se puede modelar parte del cerebro humano, áreas corticales, mecanismos de control de movimiento de los ojos, etc.

En Ciencias de la Computación se encuentra que el poder computacional de los sistemas en paralelo es estrictamente mayor que el de la máquina de Turing. El problema de la parada, que es insoluble en la teoría de la Computabilidad clásica, tiene solución en la computabilidad neuronal, en la cual se establece, de manera análoga, la insolubilidad del problema de la estabilidad[6][7].

En Inteligencia Artificial (IA) encuentra un nuevo paradigma para la representación del conocimiento. En la red neuronal, el conocimiento es una propiedad emergente del comportamiento global de la red (visión conexionista), no de su microestructura, en contraste con la visión simbólica donde las estructuras del conocimiento se codifican explícitamente junto con algoritmos de inferencias que usan esa información. La Lógica Matemática constituye la base teórica de la IA ortodoxa, mientras que ninguna rama de las Matemáticas en especial domina una teoría formal del enfoque conexionista, aunque la Termodinámica, la Mecánica Estadística y la Teoría de Sistemas Dinámicos han servido en el aná-

lisis de su comportamiento. No se puede decir que un enfoque sea mejor que el otro. Ambas tendencias pueden sobrevivir como modelos útiles en dominios particulares. Algunas tareas inteligentes se realizan mejor con las técnicas tradicionales, mientras que otras tareas que involucran alto grado de paralelismo encuentran en las redes neuronales una enorme posibilidad para ser solucionadas eficientemente, por ejemplo, visión, control de movimiento, aprendizaje, etc.Son muchas las aplicaciones que se han implementado y en las que se está trabajando. A pesar de ésto, solo podemos decir que la neurocomputación está en las primeras etapas de desarrollo. Por lo tanto, es muy grande el panorama de investigación que se abre en todos los campos.

BIBLIOGRAFIA

1. Aho A., Hopcroft J.E. and Ullman J.D., The Design and Analysis of Computer Algorithms, Addison-Wesly, 1974.

2. Brookshear G. Teoría de la Computación, Addison-Wesley, 1993.

3. Ersoy O.K. Signal/Image Processing and Understanding with Neural Networks, Neural Networks: Concepts, Applications and Implementations Vol 1 y 2 (P. Antogneti V. Milutinovic eds.), Prentice-Hall, Engelwood Cliffs, N. J., 1992.

4. Michel A.N and Farrell J.A. Associative Memories via Artificial Neural Networks, IEEE Control Systems Magazine vol 3, (1990), 6-17.

5. Franklin S.P. y Garzón M., Global dynamics in neural networks, Complex Systems 3(1989), 29-36.

6. Franklin S.P. y Garzón M. Computation on graphs Memphis State University (preimpresion).

7. Garzón M. Analysis of cellular automata and neural networks (preimpresión) Memphis State University, 1990.

8. Garzón M. Cellular automata and discrete neural networks, Physica D 45 (1990), 431-440.

9. Garzón M. Graphical words and languages, Memphis State University (1990).

10. Garzón M. and Franklin S.P., Neural Computabilityjur Internal report Department of Mathematical Sciences and Institute for Intelligent Systems. Memphis State University, 1989.

11. Garzón M. and Franklin S., Neural Computability II, Extended Abstract, Proc. 3rd Int. Joint Conf. on Neural Networks (1989), 631-637, Washington D.C.

12. Garzón M. and Franklin S., Global Dynamics in Neural Networks II, Complex Systems(1990), 431-440.

13. Hecht-Nielsen R. Neurocomputing, Addison-Wesley, Reading MA., 1990.

14. Hedlund G.A., Endomorphism and Automorphism of the Shift Dynamical System, Math Sys. Theory 3 (1969), 320.

15. Hernández G., Construcción de memorias asociativas binarias sobre redes neuronales, Memorias del seminario de redes neuronales U. Nacional Bogotá D.C., 1992.

16. Hernández G. y Niño L.F., Transformaciones geométricas en imágenes digitales, Memorias III encuentro de geométria y sus aplicaciones, Bogotá D.C., 1992

17. Hernández G. y Niño L.F., Automatas Celulares en procesamiento de imágenes, Seminario Multimedia U. Nacional Bogotá D.C., 1993.

18. Hernández G., Torres L.G. y Niño L.F., Fundamentos de Redes Neuronales, Memorias II Congreso de Ingeniería de Sistemas, Bogotá D.C., Nov 1991.

19. Hernández G., Torres L.G., Automatas Celulares Estocasticos, Rev. Fac. Ciencias U. Valle (sometido para publicación).

20. Hertz J., Krogh A. and Palmer R.G. Introduction to the Theory of Neural Computation, Addison-Wesley 1991.

21. Hopcroft J.E. and Ullman J.D., Introduction to Automata Theory, Languages and Computation, Addison-Wesley publishing Co., 1979.

22. Hopfield J.J. Neural Networks and Physical Systems with Emergent Collective Computation Abilities, Proc. Natl. Acad. Sci. USA April (1982) vol 79, 2554-2558.

23. Hopfield J.J., Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons, Proceedings of National Academy of Science, May (1984) vol 81, 3088-3092.

24. Hopfield J.J. and Tank D.W., Computing with Neural Circuits: A Model, Science August vol 233 (1986), 625-633.

25. Johnsonbaugh R. and Murata T., Petri nets and marked graphs : Mathematical models of concurrent computation Americal Mathematical Montly. October (1982).

26. Kosko B., Constructing an Associative Memory, BYTE Magazine September (1987), pages 137-144.

27. Kosko B., Neural Networks for Signal Processing Prentice-Hall, 1992.

28. Kosko B., Neural Networks and Fuzzy Systems: A Dynamical Systems Approach, Prentice-Hall, Englewood Cliffs, N.J, 1992.

29. Lee Y.C. et al, Adaptive Stochastic Cellular Automata: Theory Physica D 45, (1990), 159-180, North-Holland.

30. Lippmann R.P., An Introduction to Computing with Neural Nets IEEE ASSP Magazine April (1987) pages 4-22.

31. Manna Z., Mathematical theory of computatio n Ed. MacGrawHill-Kogakusha, 1974.

32. McCulloch W.S. and Pitts W., A Logical Calculus of the Ideas Imminent in Nervous Activity, Bulletin Mathematical Biophysics vol 5 (1943), 115-133.

33. Minsky M., Papert S., Perceptrons: An introduction to computational geometry, MIT press (1969).

34. Narendra K. and Thathachar M.A.L., Learning Automata: An Introduction Prentice-Hall, 1989.

35. Niño L.F. AMNIAC : Red Neuronal Universal, Memorias seminario Redes Neuronales U. Nacional Bogotá D.C. (1992).

36. Rosenblatt R., Principles of Neurodynamics, NewYork Spartan Books, 1960.

37. Rujan P., Cellular Automata and Statistical Mechanical Models, Statistical Physics, vol 49 (1987), 139-232.

38. Rummelhart D.E., Hinton G.e. and Williams R.J. Learning internal representations by back-propagating errors, Nature 323 (1986), 533-536.

39. Sethi R., Lenguajes de Programación: Conceptos y constructores, Addison-Wesley, 1992.

40. Sudkamp T., Automata, Languages and Machinespubl Addison-Wesley, 1990.

41. Toffoli T. and Margolus N., Cellular Automata Machines, The MIT Press, London, England, 1986.

42. Torres L.G., Lenguajes sobre grafos, Memorias del seminario informática e imagen U. Nacional Bogotá D.C., 1992.

43. Torres L.G., Computación secuencial vs. computación masivamente paralela, Memorias del seminario redes neuronales U. Nacional Bogotá D.C., 1992.

44. Torres L.G., Hernández G. y Niño L.F., Autómatas Celulares, IX Coloquio Distrital de Matemáticas y Estadística, Bogotá D.C., 1992.

45. Turing A.M., ¿Puede pensar una máquina?, SIGMA Vol 6. (ed. Newman J.R.), Grijalbo Ed., Barcelona, 37-69.

46. Von Neumann J., Teoría general y lógica de dispositivos automáticos, SIGMA Vol 6. (ed. Newman J.R.), Grijalbo Ed., Barcelona, 8-35.

47. Wolfram S., Statistical mechanics of cellular automata, Rev. of Modern Phys. vol 55 (1983), 601-644.