Redes Neuronales de Propagación Inversa

Jesus Alberto Delgado Rivera
Ingeniero Eléctrico y Magister en Tratamiento de Señales y Control.
Profesor Asistente del Departamento de Ingeniería Eléctrica en la Universidad Nacional de Colombia.
Investigador en Redes Neuronales y Algoritmos Genéticos Aplicados.

RESUMEN

Este artículo presenta la red neuronal por capas, su algoritmo de aprendizaje y una aplicación. Para lograr este propósito se inicia con la neurona natural y su modelo Entrada/Salida denominado neurona artificial.

ABSTRACT

This paper discusses the backpropagation neural network and its learnign law. For this purpose we begin with the biological neuron and the Imput/Output model Known as the artificial neuron.

INTRODUCCION

La red neuronal como estructura de computo, ha encontrado aplicaciones en control, sistemas de potencia, robótica, electrónica y tratamiento de señales, entre otras.

La red neuronal tiene tres aspectos notorios respecto de otros modelos computacionales.

- (i) No requiere programación. La red aprende mediante el procesamiento de una serie de ejemplos con datos de entrada y respuestas deseadas.
- (ii) Debido a su carácter no lineal, puede encontrar patrones complejos, los cuales no se pueden determinar analíticamente, en los ejemplos suministrados.
- (iii) Puede realizar generalizaciones fuera del conjunto de ejemplos presentados durante el aprendizaje, puesto que extrae las características fundamentales.

Las redes neuronales parten de un modelo simplificado de la neurona biológica y no pretenden reproducir la dinámica del cerebro humano. Son sólo una abstracción computacional.

1. NEURONA NATURAL

La figura 1 muestra una neurona natural y sus partes [1].

El Cuerpo Celular es el encargado de las actividades metabólicas de toda la neurona. Las Dendritas parten del cuerpo celular y se especializan en el recepción de del cuerpo celular y se especializan en la recepción de señales eléctricas de otras neuronas a través de conexiones o Sinápsis.

Cuando se transmite una señal eléctrica de una neurona a otra en una sinápsis, su efecto es el aumento o disminución del potencial eléctrico dentro del cuerpo de la neurona receptora. Si su potencial alcanza el umbral, se envía un pulso o potencial de acción por el Axón.

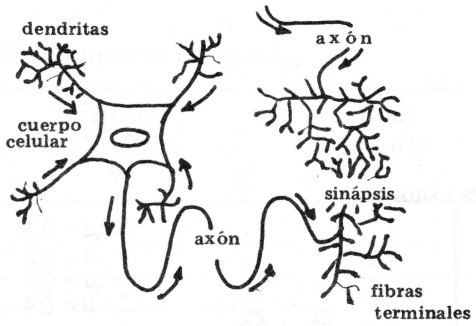

Fig. 1. Neurona natural y sus partes.

2. NEURONA ARTIFICIAL

Un modelo simple de la dinámica Entrada/Salida (E/S) de la neurona natural se observa en la figura 2.

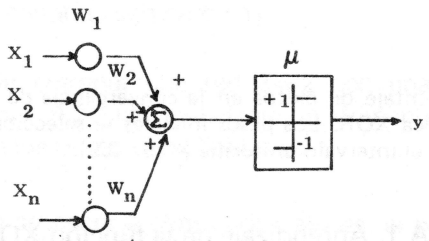

Fig. 2. Modelo E/S simple de la neurona natural.

La salida de este modelo obedece la ecuación,

$$y = f\left(\sum_{i=1}^{n} W_i X_i \mu\right)$$

donde,

$$f(a) = \begin{cases} +1, \text{ si } a \geq 0 \\ -1, \text{ si } a < 0 \end{cases} \qquad (1)$$

μ : Umbral
X_i : Señales de Entrada
W_i : Pesos de cada conexión

En palabras, cuando la suma ponderada de todas las entradas a la neurona excede el umbral, se dispara y su salida toma el valor constante de +1. En caso contrario, la salida es -1. Note que en la neurona natural la salida es un pulso no un valor constante como en el modelo.

La función f(a), se denomina función de activación. Para el modelo de la figura 2 es un limitador duro. En el caso de las redes de propagación inversa se utiliza la tangente hiperbólica (tanh) debido a la existencia de su derivada.

La ecuación (1) se puede escribir como :

$$y = f\left(\sum_{i=o}^{n} W_i . X_i\right) \qquad (2)$$

con,

Xo = + 1

Wo = $-\mu$: peso umbral

3.- RED NEURONAL POR CAPAS

La figura 3 muestra una red neuronal de propagación inversa con tres capas [2], [3] y [4]. En notación simplificada es una red 3-4-2.

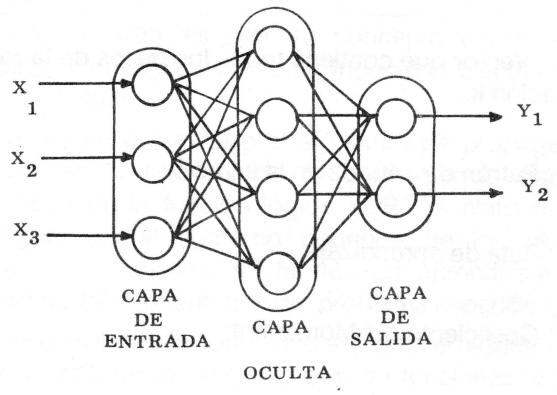

Fig. 3 - Red por Capas. Se observan tres nodos en la capa de entrada, cuatro neuronas en la capa oculta y dos neuronas en la capa de salida.

La capa de entrada no realiza procesamiento, simplemente recibe las entradas Xi para distribuirlas a la capa oculta, por lo tanto, no hay neuronas en la entrada sino nodos.

La capa oculta realiza procesamiento y está formada por neuronas similares a la de la figura 2, pero con la función de activación tanh.

La capa de salida realiza procesamiento y está formada por neuronas similares a la de la figura 2, pero con función de activación tanh.

Las capas de entrada y salida se conectan al exterior, mientras la capa del medio no. Por ello, se denomina a esta última capa oculta.

3.1.- Algoritmo de Propagación Inversa

El entrenamiento de una red neuronal consiste en modificar los pesos de las conexiones entre neuronas de tal forma que la red pueda reproducir los ejemplos presentados y generalizar.

El algoritmo de propagación inversa es el más utilizado en la actualidad. Conceptualmente funciona como sigue : Cada ejemplo o patrón se propaga hacia adelante, en la capa de salida se calcula el error cuadrático de la red y sus derivadas (deltas) respecto de las neuronas de salida. Luego estas derivadas del error cuadrático se propagan hacia atrás, para obtener las restantes derivadas (deltas) respecto de las neuronas ocultas. Debido a que las derivadas del error cuadrático se propagan en sentido inverso a la propagación de las entradas, el algoritmo se denomina de propagación inversa.

Matemáticamente, el algoritmo de propagación inversa se puede escribir como,

$$W_{k+1} = W_k + 2.\mu.\ \delta_k.X_k + \propto (W_k - W_{k-1}) \quad (3)$$

donde,

Wk : Vector que contiene todos los pesos de la red en la iteración k.

Xk : Patrón de entrada en la iteración k.

μ : Rata de aprendizaje

$\propto$: Coeficiente del Momentum

δ_k : Derivada del error cuadrático a la salida de la red respecto de la salida lineal de cada neurona.

El entrenamiento finaliza cuando el error cuadrático, sobre todos los patrones, a la salida de la red es menor que el límite dado.

La figura 4 muestra el porcentaje de éxitos en la convergencia del algoritmo (3) como función de la rata de aprendizaje y del coeficiente del momentum. Estos porcentajes se calcularon durante el aprendizaje de la función lógica XOR de dos entradas. Para cada condición se realizaron diez simulaciones desde distintas condiciones iniciales para los pesos de la red. El error cuadrático exigido sobre todos los patrones durante el aprendizaje fue del 1% y el máximo número de iteraciones permitido fue de 200.

Note que una combinación óptima para el entrenamiento de la función XOR de dos entradas es : Rata de Aprendizaje = 0.3 y Coeficiente del Momentum = 0.5.

La Tabla 1 muestra la comparación entre dos redes neuronales (μ = 0.3$\propto$ = 0.5) durante el aprendizaje de la función lógica XOR de dos entradas. El proceso finaliza cuando el error cuadrático sobre todos los patrones es menor al 1%.

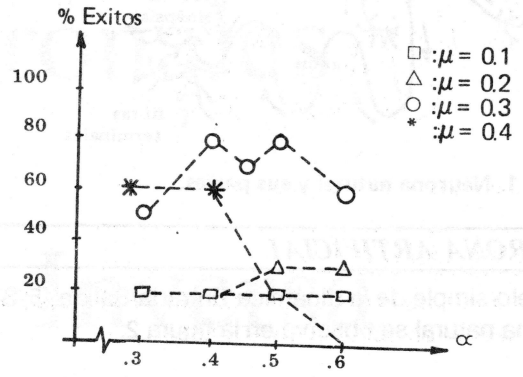

Fig. 4 Porcentaje de éxitos en la convergencia de (3) para la función lógica XOR. Los pesos iniciales se seleccionan aleatoriamente en el intervalo uniforme (-0.5+ 0.5).

TABLA 1. Aprendizaje de la función XOR en una CPU 80286 a 16 MHz.

Red	Iteraciones Promedio	Tiempo Promedio (s)
2- 2- 1	65.75	27.75
2- 3- 1	66.00	37.25

El término "Iteracion" se refiere a la lectura de todos los patrones. El promedio se realizó sobre ocho ejecuciones desde distintas condiciones iniciales, similares para las dos redes.

Vemos, Tabla 1, que las dos redes requieren un número similar de iteraciones en promedio, sin embargo, a mayor tamaño de la red mayor será el tiempo de aprendizaje.

3.2.- Aplicación

La figura 5 muestra el circuito lógico correspondiente al sumador completo que satisface la Tabla 2 [5].

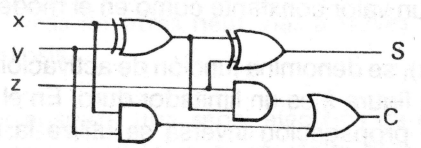

Fig. 5. Circuito lógico del sumador completo.

TABLA 2. Sumador completo.

Entradas Circuito			Salidas Circuito	
X	Y	Z	C	S
−1	−1	−1	−1	−1
−1	−1	+ 1	−1	+ 1
−1	+ 1	−1	−1	+ 1
−1	+ 1	+ 1	+ 1	−1
+ 1	−1	−1	−1	+ 1
+ 1	−1	+ 1	+ 1	−1
+ 1	+ 1	−1	+ 1	−1
+ 1	+ 1	+ 1	+ 1	+ 1

Note, Tabla 2, la utilización de los valores (-1, + 1) en lugar de (0, + 1). Esto con el propósito de que la red aprenda cuando las entradas son cero (-1).

Después de entrenar una red 3-3-2 con una Rata de Aprendizaje = 0.3 y un Coeficiente de Momentum = 0.5, se obtienen los pesos de la Tabla 3.

Cada peso se escribe como Wij, esto es, el peso de la conexión que llega a la neurona i desde la neurona j. El subíndice cero en los pesos indica la conexión a un nodo El error de entrenamiento fue menor del 1%

TABLA 3. Pesos finales de la red para un sumador completo

CAPA 1	W10 = 0.5689;	W20 = 0.0261
	W11 =− 0.9127;	W21 = 3.3076
	W12 =− 3.8331;	W22 = 2.7328
	W13 =− 0.8266;	W23 = 2.8523
CAPA 2	W30 =− 0.3800;	
	W31 =− 3,9890;	
	W32 =− 4.9363;	
	W33 =− 3.6199;	
CAPA 3	W10 =− 0.3478;	W20 = 0.4020
	W11 =− 0.1843;	W21 = 3.2794
	W12 = 0.0332;	W22 = 2.6120
	W13 =− 3.3057;	W23 = 2.9486

La Tabla 4 presenta las salidas de la red neuronal después del entrenamiento.

TABLA 4. Red neuronal 3- 3- 2 como sumador completo.

Entradas Red Neuronal			Salidas Red Neuronal	
X	Y	Z	C	S
−1	−1	−1	-0.999	-0.988
−1	−1	I+1	-0.999	0 .990
−1	+1	−1	-0.998	0.999
−1	+ 1	+1	0.997	-0.995
+1	−1	−1	-0.999	0.991
+1	−1	−1	0.991	-0.996
+1	+ 1	−1	0.997	-0.950
+1	+ 1	+ 1	0.996	0.997

CONCLUSIONES

Las redes neuronales son un concepto de computo novedoso y las aplicaciones en Ingeniería Eléctrica son objeto de estudio.

El artículo ha presentado el algoritmo de propagación inversa con un análisis de la convergencia para el aprendizaje de la función lógica XOR. Es claro que la convergencia del algoritmo depende de los valores iniciales de los pesos, de la rata de aprendizaje, del coeficiente del momentum y del problema específico.

Además de las simulaciones que exploran el algoritmo, se ilustra su aplicación en la síntesis de funciones lógicas como el sumador completo. Dada una tabla lógica la síntesis de la función se puede lograr entrenando una Red Neuronal sin recurrir a procesos de diseño complicados.

REFERENCIAS

[1] Kiloh, L.G., McComas, A.J. y Osselton, J.W.: **Clinical Electroencephalography,** Butterworths, Londres, 1972.

[2] Hertz, J., Krog, A. y Palmer, R.: **Introduction to the Theory of Neural Computation,** Addison-Wesley Publishing Company, California, 1991.

[3] Widrow, B. y Lehr, M.: **30 Years od Adaptive Neural Networks: Perceptron, Madaline and Backpropagation,** Proc. IEEE, Vol. 78, No. 9, pp. 1415-1441, Septiembre 1990.

[4] Hecht-Nielsen, R.: **Theory of the Backpropagation Neural Network,** Proc. IEEE-ICNN, I (593-605), IEEE Press, New York, a1989.

[5] Mano, Morris: **Arquitectura de Computadores,** Prentice-Hall, México, 1982.