Introducción a Redes Neuronales Artificiales

Eduardo Rivera*

Resumen

Este artículo muestra una breve reseña histórica sobre cómo surgieron las redes neuronales, para luego hacer énfasis en los conceptos básicos que son necesarios para que se pueda comprender cómo interactúan las diferentes partes que componen una neurona básica y a una red neuronal. También se realiza una comparación entre el proceso de aprendizaje de neuronas entrenadas con "La ley de Hebb" y regla de entrenamiento para un perceptrón, ambas usadas en una aplicación de reconocimiento simple

Palabras clave—Redes neuronales, conceptos básicos , arquitectura típica de una red neuronal, Perceptrón, aprendizaje de Hebb, comparación entre aprendizajes Hebb y Perceptrón, reconocimiento de caracteres

Introducción

En la actualidad existen muchas aplicaciones que involucran inteligencia artificial, pero hablar del tema es demasiado amplio, ya que éste involucra varias áreas: algoritmos genéticos, lógica difusa, redes bayesianas, redes neuronales, etc.,

Aunque se han realizado muchos avances, se sigue estando limitado en el aspecto de lograr inteligencia artificial comparada con la del ser humano. Uno de los aspectos en el que se ha logrado avances significativos es el campo de las redes neuronales, el cual permite aprender patrones, permitiendo cierto grado de flexibilidad, basándose en procesos simples.

Antecedentes

Muchos autores coinciden en que el modelo neuronal artificial, como se conoce actualmente, fue planteado por primera vez por Alan Turing en 1936 y que la primera implementación de una red fue lanzada por Warren McCulloch y Walter Pitts en 1943. Dicho sistema consistía en una red neuronal simple basada en circuitos eléctricos.

Otro hecho relevante fue la publicación, en 1949 por Donald Hebb, del libro "La organización del comportamiento", en el que se establece una conexión entre psicología y fisiología. Un aporte adicional de este libro fue la regla de aprendizaje básica para neuronas simples.

Durante el verano de 1951, Marvin Minsky y Dean Edmonds montaron la primera máquina de redes neuronales, compuesta básicamente por 300 tubos al vacío y un piloto automático de un bombardero B-24 (Herrera 1998).

Después de su Red Neuronal, Minsky escribió su tesis doctoral acerca de ésta. En ella describía "cerebros mucho mayores", exponiendo que si se realizaba este proyecto a gran escala, con miles o millones de neuronas más y con diferentes censores y tipos de retroalimentación la máquina podría ser capaz de razonar. Mas él sabía que la realización de esta Red Neuronal era imposible y

* Ingeniero en Electrónica. Profesor de la Escuela de Electrónica de la Universidad Don Bosco.

decidió buscar otra forma de crear inteligencia.

En 1957 (Hilera 1995), Frank Rosenblatt empezó a desarrollar un modelo de red neuronal que contenía una sola neurona y tenía un aprendizaje basado en la regla de Hebb, siendo considerada ésta la red más antigua y que todavía sigue siendo utilizada en su forma evolucionada que involucra capas múltiples. Entre sus características básicas se pueden mencionar que era capaz de generalizar, es decir reconocer o identificar patrones que no se le hubieran enseñado, utilizar "n" número de entradas y una sola salida. Esta red, aunque era un gran avance para la emulación del comportamiento de las neuronas humanas, tenía ciertas limitaciones, entre las que sobresale el hecho que no podía clasificar patrones que no tuvieran una separabilidad lineal, es decir que si se pudieran ubicar los patrones a reconocer en un plano cartesiano, éstos podrían ser separados por una línea recta, por lo que el plano quedaría dividido en dos partes que representarían las dos clasificaciones que podría realizar el Perceptrón.

Conceptos básicos.

¿Qué es una red neuronal?

- Existen diferentes definiciones del término red neuronal:
- Una forma de computación inspirada en modelos biológicos. (Hilera 1995, Freeman 1993, Príncipe 1994)
- Elemento computacional de procesamiento, como unidades similares a las del cerebro humano, que tienen la capacidad de procesar información y aprender de ella. (Hilera 1995)
- Un sistema de computación que consiste en un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas. (Hilera 1995)

Cada uno de estos conceptos es válido y hace referencia a elementos de proceso individuales y a formas de algoritmos, esto es debido a que la mayoría de tipos de redes neuronales han sido creadas con el objetivo de emular en cierto grado la inteligencia artificial, lo que provoca que la mayoría de los diseños generados, sea orientada a algoritmos que puedan ser fácilmente implementados en cualquier lenguaje de programación, aunque también pueden ser puestos en funcionamiento en forma de hardware.

¿Qué partes componen una neurona?

Como puede verse en la Figura 1, una neurona artificial esta compuesta de tres partes: Entradas, núcleo y salidas.

Las entradas reciben los datos o parámetros que le permiten decidir a la neurona si estará activa o no, normalmente se representan como x1...xn.

Entre la entrada y el núcleo aparecen los pesos(w1, w2…wn), que representan la memoria de la red.

Las salidas devuelven la respuesta de la neurona, es decir si esta activa o no, representadas comúnmente como y1…yn.

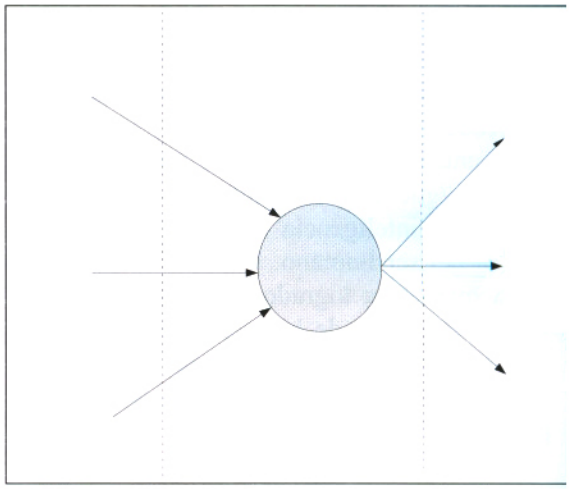

Figura 1 Partes de una neurona simple.

En el núcleo se realizan todas las operaciones necesarias para determinar la salida de la neurona; el proceso que se realiza en el núcleo varía dependiendo de la red neuronal que se este trabajando. Es importante mencionar que la salida del núcleo es pasada por

una función de transferencia; entre las más usadas tenemos:

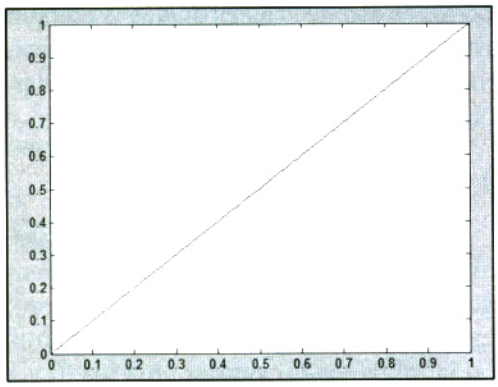

Figura 2 Representación de la función identidad

Identidad

Está representada por la fórmula

(1) $f(x) = x$

Simplemente representa que la salida calculada de la neurona será directamente lo que tenga la red como respuesta (ver Figura 1).

Escalón unitario

Esta definida por las siguientes ecuaciones

$$(2) \quad \begin{cases} f(x) = 1 & x \geq \theta \\ f(x) = 0 & x < \theta \end{cases}$$

En este caso se tienen dos posibilidades: para valores menores que θ la salida de la red será 0 y en caso contrario 1. Esto se puede comparar con un control todo o nada, es decir la neurona está inactiva, lo que significa que no contribuye con la respuesta de la red, o está activa, que tiene el 100% de influencia en la salida de la red.

Esta función esta representada en la Figura 3.

Escalón Unitario Bipolar

Su comportamiento está determinado por las ecuaciones

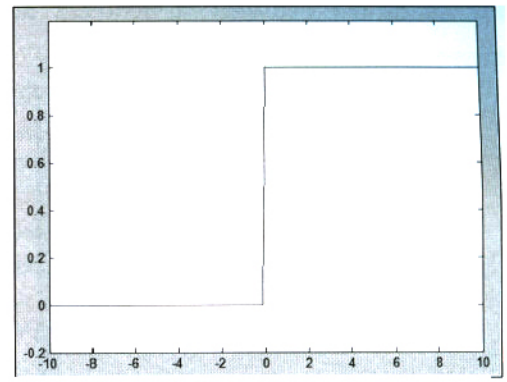

Figura 3 Representación de la función escalón unitario

$$(3) \quad \begin{cases} f(x) = 1 & x \geq \theta \\ f(x) = -1 & x < \theta \end{cases}$$

Es similar al escalón unitario, con la diferencia que toma el valor de -1 en lugar de 0 para valores menores que θ. Tiende a ser más usada que el escalón unitario debido a que no genera confusión cuando la salida es 0. Ver Figura 4.

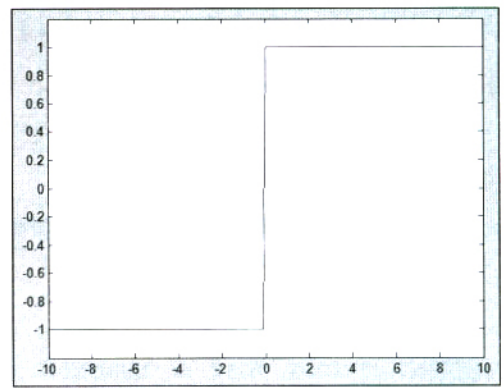

Figura 4 Representación de la función escalón unitario bipolar

Sigmoidal Unipolar

En este caso la ecuación (4), nos ayuda a representar valores más parecidos a los de una neurona real, ya que permite valores entre 0 y 1.

$$(4) \qquad f(x) = \frac{1}{1+e^{-\sigma x}}$$

Esta función suele aplicarse cuando a la salida de la red neuronal se requiera de

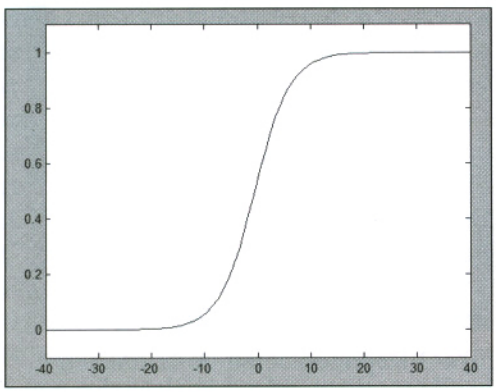

Figura 5. Representación de la función sigmoidal

valores análogos, es decir diferentes de 0 y 1. El valor de σ indica el valor de la pendiente de la curva. En la figura 5 se muestra la función sigmoidal para un valor de σ =3.

Sigmoidal bipolar

Es similar a la sigmoidal unipolar, con la diferencia de que toma valores entre -1 y 1.

$$(5) \qquad f(x) = -1 + \frac{2}{1+e^{-\sigma x}}$$

Solo tiende a ser una mejora de la sigmoidal original.

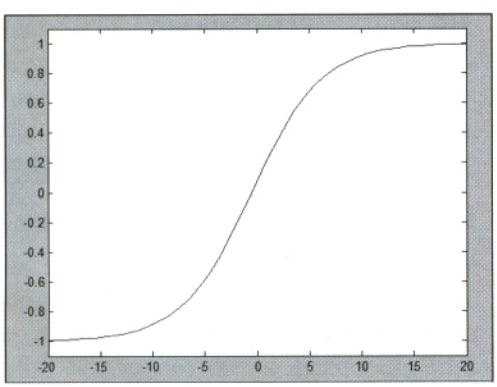

Figura 6 Representación de la función sigmoidal bipolar

Tangencial

El comportamiento de esta función es parecido a la sigmoidal bipolar y mantiene la representación de señales análogas comprendidas entre -1 y 1, con la diferencia que no tiene un valor σ que le ayude a controlar la pendiente de la función.

$$(6) \qquad f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

La función de transferencia a utilizar es definida en cada uno de los diferentes algoritmos de redes neuronales, por el creador del mismo, y lo único que determina es si la salida tendrá valores binarios (0 y 1 ó -1 y 1) o comprendidos dentro de un rango (entre 0 y 1 o entre -1 y 1).

Entre la entrada y el núcleo existe un valor que determina el aprendizaje de la neurona. Dicho número es llamado peso y representa la memorización de los datos que han sido aprendidos. Este valor junto con las entradas es el que determina si una entrada influirá o no en la salida.

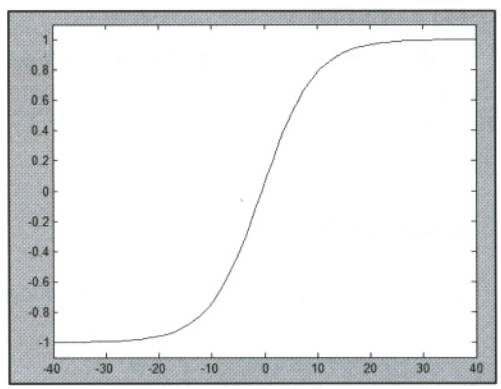

Figura 7 Representación de la función tangencial

Estructura de una red neuronal

¿Qué partes componen a una red neuronal?

Según las definiciones mencionadas anteriormente, está compuesta por varios elementos procesales (neuronas), que forman un todo. Las partes principales se muestran en la Figura 8, en la cual podemos ver que tenemos: neuronas de entradas, ocultas,

salidas y las interconexiones. (Hilera 1995, Freeman 1993, Príncipe 1994)

Las neuronas de entradas son transparentes, es decir no realizan ningún proceso, sólo dejan pasar la información que se quiere manejar en la red.

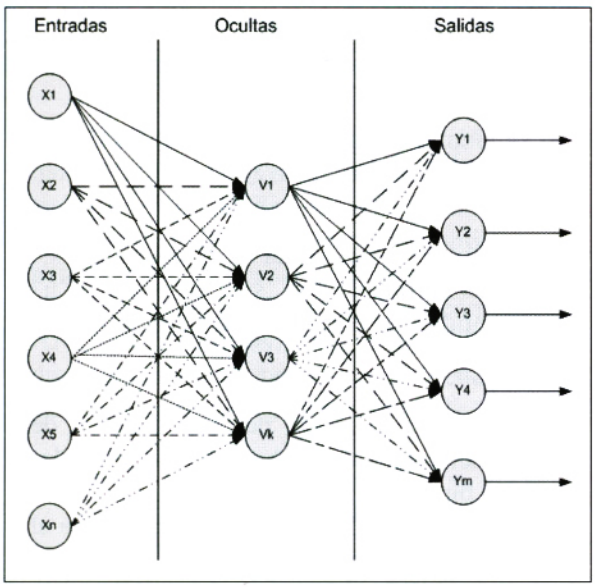

Figura 8 Capas de una red neuronal

Las neuronas ocultas reciben las entradas y tienen la función de proporcionar un mejor aprendizaje o separación de lo aprendido en categorías. Las neuronas ocultas pueden o no estar presentes en una red y su incorporación depende de dos factores:

Primero de la topología con la que se esté trabajando y segundo de la complejidad de los patrones que deben ser aprendidos por la red.

Si se determina que se requiere una capa de neuronas ocultas, debe surgir la siguiente pregunta:

¿Cuántas neuronas debe tener?

El número de neuronas de la capa oculta es variable para cada aplicación y no existe ninguna fórmula que permita definir su cantidad. La mayoría de autores coinciden en que sólo la experiencia en el entrenamiento puede proporcionar una idea de cuántas neuronas ocultas es necesario poner en una aplicación específica. En algunos casos puede ser necesario poner más de una capa de neuronas ocultas para poder realizar la aplicación deseada. Por lo general dos capas son suficientes. (Hilera 1995, Freeman 1993)

Las neuronas de salida se encargan de proporcionar la salida del sistema indicando, según su aprendizaje, una respuesta correcta o incorrecta a lo aprendido.

Las capas de neuronas ocultas y de salida están compuestas de elementos procesales como los mostrados en la Figura 1.

Ventajas y Desventajas

Ventajas de las redes neuronales

- El procesado de la información es local, es decir que al estar compuesto por unidades individuales de procesamiento cada neurona realiza su activación, dependiendo de sus entradas y pesos y de que todas las neuronas de una capa trabajan en forma paralela y proporcionan una respuesta al mismo tiempo. (Hilera 1995, Freeman 1993)
- Los pesos son ajustados basándose en la experiencia, lo que significa que se le tiene que enseñar a la red lo que necesita saber antes de ponerla en funcionamiento. (Hilera 1995, Freeman 1993)
- Los pesos pueden tener la función de activación o inhibición, es decir que son parte importante en la determinación de si una neurona está activa o no. (Hilera 1995, Freeman 1993)
- Las neuronas son tolerantes a fallos, si parte de la red no trabaja, sólo dejará de funcionar la parte para la que dicha neurona sea significativa; el resto tendrá su comportamiento normal. (Hilera 1995, Freeman 1993)
- Las neuronas pueden reconocer patrones que no han sido aprendidos, sólo deben tener cierto parecido con el conocimiento previo que tenga la red. Dicho de otra forma: si la entrada presenta alguna alteración la red podrá identificarla siempre y cuando se mantenga cierto

grado de similitud entre lo aprendido y lo mostrado en la entrada a la red. (Hilera 1995, Freeman 1993)
- Operación en tiempo real, los algoritmos de las redes neuronales son bastante eficientes una vez que la red ha sido entrenada, por lo que su respuesta es casi inmediata. (Hilera 1995, Freeman 1993)

Desventajas de las redes neuronales

- Complejidad de aprendizaje para grandes tareas, cuanto más cosas se necesite que aprenda una red, más complicado será enseñarle. (Hilera 1995, Freeman 1993)
- Tiempo de aprendizaje elevado. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar y segundo si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir más tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera enseñar. (Hilera 1995, Freeman 1993)
- No permite interpretar lo que se ha aprendido, la red por sí sola proporciona una salida, un número, que no puede ser interpretado por ella misma, sino que se requiere de la intervención del programador y de la aplicación en sí para encontrarle un significado a la salida proporcionada. (Hilera 1995, Freeman 1993)
- Elevada cantidad de datos para el entrenamiento, cuanto más flexible se requiera que sea la red neuronal, más información tendrá que enseñársele para que realice de forma adecuada la identificación. (Hilera 1995, Freeman 1993)

Red con aprendizaje de Hebb

Regla de aprendizaje de Hebb.

Se basa en el modelo de la figura 1. y está definido por los siguientes pasos(Hilera 1995, Freeman 1993, Príncipe 1994):

1. definir los patrones a enseñar a la red
2. definir la cantidad de entradas
3. definir las salidas que se desea en la red
4. los pesos comienzan con un valor de cero
5. poner en las entradas de la red un patrón a ser aprendido
6. calcular la salida de la neurona usando la fórmula

$$(7) \qquad y = \sum_{i=0}^{n} x_i * w_i$$

donde: "y" es la salida de la neurona, "x_i" son las entradas, "w_i" son los pesos y "n" la cantidad de entradas que tiene la neurona.

7. se pasa la salida por una función de transferencia escalón unitaria unipolar o bipolar
8. se iguala "y" a la salida deseada para el patrón de entrada
9. se modifican los pesos basándose en la formula

$$(8) \qquad w_i(\text{nuevo}) = w_i(\text{viejo}) + x_i y$$

10. se verifica si han pasado todos los patrones que se requiere la red aprenda. Si la respuesta es no, se retrocede al numeral 5.
11. si la respuesta es sí se termina el entrenamiento.

El ejemplo más típico para ilustrar este entrenamiento es tratar de hacer que el sistema aprenda el comportamiento de una compuerta AND (ver Tabla 1).

Es de hacer notar que en Tabla 1 la variable x_0 no cambia debido a que tiene la función de ajuste de la red y permanece constante durante todo el aprendizaje y funcionamiento.

Tabla 1
Comportamiento de una compuerta AND de dos entradas

x_0	x_1	x_2	T
1	1	1	1
1	1	-1	-1
1	-1	1	-1
1	-1	-1	-1

Resulta evidente que con esta tabla ya se han definido algunos de los parámetros que se requieren de la red: la cantidad de entradas (3), la salida (1) y la cantidad de patrones a reconocer (4, que es el número de combinaciones posibles para una compuerta AND de 2 entradas), por lo tanto lo único que faltaría es proceder a generar el entrenamiento de la red(ver Tabla 2).

Tabla 2
Entrenamiento de una neurona usando la regla de aprendizaje de Hebb.

w_0	w_1	w_2	x_0	x_1	x_2	y
0	0	0	1	1	1	1
1	1	1	1	1	-1	-1
0	0	2	1	-1	1	-1
-1	1	1	1	-1	-1	-1
-2	2	2				

La primera fila de Tabla 2 representa los valores iniciales de los pesos, así como también los valores de la primera secuencia que debe aprender la neurona. La Tabla 2 en general muestra los cambios que sufren los pesos a lo largo del entrenamiento y los valores finales con los que se puede verificar el proceso de aprendizaje simplemente aplicando los pasos 5 y 6 de la regla de aprendizaje.

Otro ejemplo típico suele ser el reconocimiento de patrones como los de la Figura 9.

En este caso las entradas serían 10, tomando en cuenta el ajuste, la salida 1 y 2 los patrones a reconocer. Para poder ingresar los valores a una neurona, se puede adoptar que un punto es 1 y un asterisco es –1, esto resuelve entonces que las entradas serían las mostradas en Tabla 3. Las salidas deseadas serían 1 para la "x" y –1 para la "o".

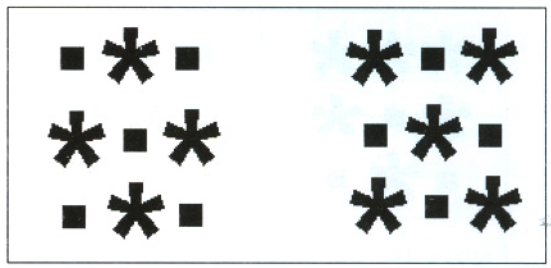

Figura 9. Matriz representando dos patrones, la "o" y la "x"

Tabla 3
Patrones de entrada

x_0	x_1	x_2	x_3	x_4	x_5	X_6	x7	x_8	x_9	t
1	1	-1	1	-1	1	-1	1	-1	1	1
1	-1	1	-1	1	-1	1	-1	1	-1	-1

Tabla 4
Entrenamiento de pesos para aprender los patrones "x" y "o"

W_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	y
0	0	0	0	0	0	0	0	0	0	1
1	1	-1	1	-1	1	-1	1	-1	1	-1
0	2	-2	2	-2	2	-2	2	-2	2	

En la última fila de Tabla 4, tenemos los pesos resultantes del entrenamiento.

Es de hacer notar que el peso w_0, de la tabla 4, se mantuvo en su valor original(0), lo que se puede interpretar que no es influyente en el resultado de esta aplicación y que bastan las entradas para realizar el reconocimiento. También se puede observar en Tabla 4, que los valores de los pesos alternan los signos, es decir que todos los pesos impares tienen signo positivo y los pesos pares signo negativo, representando la influencia de cada neurona en la salida de la red.

Una vez encontrados los pesos finales es recomendable probar patrones diferentes para definir el comportamiento de la neurona, es decir si es factible que identifique como válido algún patrón que no

sea exactamente igual a los originales que se le enseñaron.

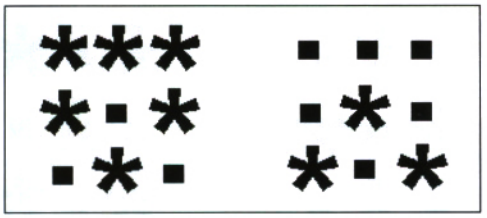

Figura 10. Variantes de los patrones de entrenamiento de la Figura 9

Para probar la red, se toman los últimos pesos de Tabla 4, que son los que quedan como resultado del entrenamiento y se procede a calcular la salida de la red, aplicando los pasos 6 y 7 del aprendizaje, con lo que se puede determinar si la respuesta es válida de acuerdo a lo que se le quería enseñar(ver Tabla3).

Tabla 5
Patrones aleatorios de prueba.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y
1	-1	-1	-1	-1	1	-1	1	-1	1	1
1	1	1	1	1	-1	1	-1	1	-1	-1

Los patrones de Tabla 5 corresponden a los mostrados en la Figura 10, tomando en cuenta que para el primero la salida obtenida con los pesos de Tabla 4 es 1 y para el segundo la salida es -1.

Lo interesante es que la red acepta, dentro de cierto rango, variaciones en la entrada, es decir que puede generalizar a pesar de que la información de entrada ha sido alterada. A éste fenómeno se le denomina plasticidad de la neurona. Esta característica suele perderse a medida que el patrón que se ponga en la entrada tienda a ser demasiado diferente al original.

Como se puede observar en ambos ejemplos, con el aprendizaje de Hebb, el entrenamiento es simple y no repetitivo, se realiza el entrenamiento exactamente un número de veces igual a la cantidad de patrones que se quiere clasificar.

También se puede dar el caso que w_0 pierda su valor de ajuste, a medida que el sistema tiende a tener más entradas que le permitan diferenciar lo que se le quiere enseñar, con lo que cuantas más entradas tenga la red menos necesario será el valor de ajuste w_0 para una correcta identificación.

Se puede dar el caso que la red no aprenda lo que se le quiere enseñar, debido a que sean señales demasiado parecidas entre si, con lo que la red tendería a confundirse.

Por otro lado este tipo de red tiene la característica de tener una sola salida, lo que limita sus posibles aplicaciones. Aunque no le quita el valor que tiene desde el punto de vista pedagógico para comprender los conceptos necesarios para explicar el funcionamiento de una red neuronal.

Aprendizaje con el Perceptrón

Al igual que Hebb, se basa en la estructura de la Figura 7 y su algoritmo de entrenamiento esta definido como (Hilera 1995, Príncipe 1994):

1. Asignar valores iniciales a los pesos, ajuste y umbral (θ), por simplicidad se sugiere que inicien en cero.
2. Establecer un valor al factor de aprendizaje α ($0 < \alpha \leq 1$). Se recomienda iniciar en uno y si es necesario decrementarlo.
3. Presentar un dato de entrada, por lo general esta compuesto del patrón a entrenar junto con la salida esperada.
4. Calcular la salida basándose en la ecuación:

$$y\_in = \sum_{i=0}^{n} x_i w_i - \theta \quad (9)$$

Donde:

xi es la iésima entrada, wi es el iésimo peso, θ es el umbral , n es la cantidad de entradas que tiene la red

5. Se pasa la salida por una función escalón unitaria unipolar o bipolar.
6. Se compara la salida obtenida con la deseada. Si son iguales se continúa al paso 8.

7. Si las salidas no son iguales se modifican los pesos con la ecuación (10)

(10) $$w_i(nuevo) = w_i(viejo) + \alpha * t * x_i$$

Donde:
α es el factor de aprendizaje, t es la salida deseada y x_i es la entrada correspondiente.

8. Se verifica si ya pasaron todos los patrones, si la respuesta es no se procede al paso 3.
9. Si han ocurrido cambios en los pesos en alguno de los patrones de entrada se reinicia el contador de patrones, se incrementa el contador de épocas y se procede al paso 3.
10. Se termina el proceso, quedando los valores de los pesos, que representan el aprendizaje realizado.

Para probar el sistema se efectúan los pasos del 3 al 6, pero con la diferencia de que en lugar de saltar al paso 8, se debería mostrar un mensaje dependiendo de la aplicación que se esteéimplementando, por ejemplo: "patrón identificado"; "identificación adecuada", identificación errónea", dependiendo del caso.

Basándose en la Tabla 1 se entrenará un perceptrón para observar su comportamiento. Se tomará e cuenta que los valores de umbral (θ) son 0 y de factor de aprendizaje (α) de 1.

Tabla 6
Entrenamiento de una neurona usando la regla de aprendizaje de Perceptrón.

w_0	w_1	w_2	x_0	x_1	x_2	y
Primera iteración						
0	0	0	1	1	1	1
0	0	0	1	1	-1	-1
-1	-1	1	1	-1	1	-1
-2	0	0	1	-1	-1	-1
-3	1	1				
Segunda iteración						
-3	1	1	1	1	1	1
-2	2	2	1	1	-1	-1
-2	2	2	1	-1	1	-1
-2	2	2	1	-1	-1	-1
-2	2	2				
Tercera iteración						
-2	2	2	1	1	1	1
-2	2	2	1	1	-1	-1
-2	2	2	1	-1	1	-1
-2	2	2	1	-1	-1	-1
-2	2	2				

En Tabla 6, se muestra en la última fila los pesos finales a los que se llegaron después de tres iteraciones, los cuales indican los valores con los que la red aprendió a comportarse como una compuerta and.

El otro entrenamiento es para los patrones de la Figura 9, que están representados en Tabla 3, siempre tomando en consideración que los valores de umbral (θ) son cero y de factor de aprendizaje (α) de 1.

Tabla 7
Entrenamiento de pesos para aprender los patrones "x" y "o"

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	y
0	0	0	0	0	0	0	0	0	0	1
1	1	-1	1	-1	1	-1	1	-1	1	-1
1	1	-1	1	-1	1	-1	1	-1	1	
Segunda iteración										
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	

Aunque w_0 mantiene un valor diferente de cero, siempre resulta poco influyente para los patrones que aprendió la red y podría no ser tomado en cuenta, debido a que a medida existan más entradas el ajuste pierde su valor.

Los pesos finales de la segunda iteración de Tabla 7, tienden a tener signos alternados, es decir un valor es positivo y otro negativo.

Similar a lo que se realizó con Hebb, es recomendable probar la red con valores que presenten ciertas variaciones a los originales, Para lo que se tomará la Figura 10 y los patrones presentados en Tabla 5.

Tabla 8
Patrones aleatorios de prueba.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y
1	-1	-1	-1	-1	1	-1	1	-1	1	1
1	1	1	1	1	-1	1	-1	1	-1	-1

Los patrones de Tabla 8, representan el cambio de 2 de las entradas de la red y el sistema sigue dando una respuesta válida, a pesar de los cambios realizados, es decir que para la primera de las entradas la reconoce como una "o" y la segunda como una "x".

Este tipo de red tiene una única salida. Entre más patrones se tengan que enseñar a la red más difícil será entrenarla; por otro lado si en algún momento la red no logra aprender, se recomienda variar el factor de aprendizaje, reduciendo este valor se puede disminuir o incrementar la cantidad de iteraciones que se requieran para lograr un correcto aprendizaje de la red neuronal. Cabe mencionar que aunque ambos tienen un proceso de aprendizaje fácil de implementar en cualquier lenguaje de programación o incluso en una simple hoja de cálculo de Excel.

Suelen ser tomados como punto de partida, de cualquier curso introductorio de redes neuronales, debido a su simplicidad y a la facilidad para explicar los conceptos necesarios para entender el funcionamiento de otras redes.

Conclusiones

La característica principal de las redes neuronales es su capacidad para aprender y retener la información aprendida, sin necesidad de mantener una base de datos cuando está en funcionamiento.

Esta red tiene el factor de aprendizaje como parámetro de control. Esto no es una garantía para que la red aprenda y puede darse el caso de que nunca se llegue a valores de pesos que permitan dar por finalizado el entrenamiento, por lo que es recomendable tener un control sobre el número de iteraciones permitido, para que éste no tienda al infinito.

Tabla 9
Características de las redes de Hebb y perceptrón

Característica	Hebb	Perceptrón
Modelo de neurona	Neurona típica	Neurona típica
Entradas	Las requeridas por la aplicación	Las requeridas por la aplicación
Salidas	Una	Una
Cálculo de la salida	$y = \sum_{i=0}^{n} x_i * w$	$y\_in = \sum_{i=0}^{n} x$
Modificación de pesos	$w_i(nuevo) =$	$w_i(nuevo) =$
Factor de aprendizaje	No tiene	A
Función de salida	Escalón unitario unipolar o bipolar	Escalón unitario unipolar o bipolar
Iteraciones	No existen	Las necesarias para lograr el aprendizaje

Comparación

En Tabla 9, se muestra la comparación de los dos algoritmos de aprendizaje.

Ambos algoritmos nos brindan un aprendizaje satisfactorio para los ejemplos utilizados. Sus diferencias radican solamente en las fórmulas para calcular la salida, el cálculo de los pesos nuevos, cuando existe un error en la salida deseada, en las iteraciones necesarias para el aprendizaje y en el factor de aprendizaje que brinda cierta flexibilidad si no se encuentran pesos adecuados que cumplan las expectativas de la aplicación que se este realizando.

Al ser similares, tienen la misma dificultad, respecto al tipo de problemas que pueden resolver, lo cual los limita a aplicaciones que tengan una separación lineal.

Las redes neuronales no están diseñadas para resolver todos los problemas de reconocimiento, pero brindan una aproximación a la capacidad de reconocimiento del ser humano.

Tanto el entrenamiento de Hebb como el del Perceptrón, proporcionan un aprendizaje rápido y aunque ambas son redes neuronales

no necesariamente llegan a valores de pesos iguales, pero sí a valores que mantengan la misma tendencia.

La flexibilidad en el comportamiento de las redes neuronales, permite decidir si una entrada presente se parece lo suficiente a lo que tiene aprendido, siempre y cuando el patrón de entrada mantenga la cantidad de información necesaria para poder ser identificado.

El factor de aprendizaje permite acelerar el proceso de entrenamiento de la red neuronal.

Sólo la experiencia puede permitir el conocer valores adecuados para inicializar las variables de un Perceptrón.

Existen casos en los que tanto Hebb como Perceptrón no son capaces de aprender, como por ejemplo el comportamiento de una compuerta lógica Xor.

La entrada de ajuste x_0 es usada por el entrenamiento de Hebb y Perceptrón, como un valor que ayuda en la salida cuando la red tiene pocas entradas, por lo que a medida que en las aplicaciones se tengan más entradas, ya no se hace necesario y se puede omitir en el proceso de entrenamiento y funcionamiento.

Se recomienda tener cuidado con las iteraciones, a la hora de entrenar un Perceptrón, debido a que si el factor de aprendizaje no es el adecuado la red puede tardar mucho en aprender o nunca llegar a pesos que satisfagan las salidas, por lo que se debería poner un límite en las iteraciones y permitir que se modifique el factor de aprendizaje o los valores iniciales de los pesos para reiniciar el entrenamiento

En el Perceptrón el número de patrones que se quiera enseñar no tiene ninguna relación con el número de iteraciones que tardará la red en realizar el aprendizaje.

Aunque el entrenamiento de Hebb no lo especifica, siempre es recomendable probar el funcionamiento de los pesos finales, con todos los patrones que se le han enseñado a la red.

Los pesos cumplen la función de memorizar lo que se le ha enseñado a la red.

Bibliografía

Fausett Laurene,Fundamentals of neural networks:Architectures, algorithms and applications, New Jersey, Prentice Hall, 1994

Freeman James, Redes neuronales: Algoritmos, aplicaciones y técnicas de programación, Delaware, Addison Wesley,1993

Gail A Carpenter and Stephen Grossberg, Pattern Recognition by Self-Organizing Neural Networks, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, 1991.

Gómez Efrén, Sistema de reconocimiento de notas musicales para flauta dulce soprano, Universidad Don Bosco, El Salvador, 2004.

Herrera Herbert Arnoldo, Aplicación de redes neuronales al pronóstico de potencia eléctrica, tesis Universidad Centroamericana José Simeón Cañas, 1998.

Hilera José, Redes Neuronales Artificiales: Fundamentos, Modelos y aplicaciones, Delaware, Addison Wesley, 1995.

Ifeachor Emmanuel C, Artificial Neural Networks in Biomedicine,New York, Springer-Verlag, 2000.

Principe José, Neural and adaptative systems: Fundamentals through simulations, New York, John Wiley, 2000

Rao Valluru B, C++ Neural Networks and fuzzy logia, Segunda edición, M and T Books, 1995.

Páginas Web Consultadas

http://www.monografias.com/trabajos/redesneuro/redesneuro.shtml Junio 2005, Gustavo Luis Pavia

http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/redes.htm. Junio 2005

http://www.domotica.net/ir'www.monografias.com/trabajos/redesneuro/redesneuro.shtml. Junio 2005, Gustavo Luis Pavia

http://cruzrojaguayas.org/inteligencia/Historia%20de%20IA.htm. Julio 2005.

http://club.telepolis.com/alimaya/intro.htm Julio 2005.

http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html septiembre 2005.