Reconocimiento de patrones mediante redes neuronales artificiales

Hugo Vega Huerta, Augusto Cortez Vásquez, Ana María Huayna,
Luis Alarcón Loayza, Pablo Romero Naupari

Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos, FISI

hugovegahuerta@hotmail.com, acortezv@unmsm.edu.pe,
ahuaynad@unmsm.edu.pe, lalarcon1052@yahoo.es, pjromeron@yahoo.es

RESUMEN

En los últimos años la inteligencia artificial viene brindando soluciones en muchos aspectos del quehacer humano, por ejemplo en economía, administración, medicina, física, arte, etc.

Uno de los temas de la IA que ha alcanzado trascendental importancia es redes neuronales artificiales y en dicho tema, una de las aplicaciones más importantes es el reconocimiento de patrones.

En el presente artículo presentaremos un modelo de una red neuronal artificial para el reconocimiento de patrones. Para probar la funcionalidad del modelo, lo aplicaremos al reconocimiento de los números naturales 3, 4 y 5.

Palabras clave: Redes neuronales artificiales, reconocimiento de patrones, inteligencia artificial

ABSTRACT

In recent years the artificial intelligence has been providing solutions in many areas of human endeavor, such as in economics, management, medicine, physics, art and so on. One topic of the Artificial Intelligence, which is reaching special importance is “artificial neural networks”, and in that topic, one of the most important applications is Pattern Recognition. In this article we present a model of an artificial neural network for pattern recognition. To test the functionality of the model, we will apply the recognition of natural numbers 3, 4 and 5.

Keywords: Artificial neural networks, pattern recognition, artificial intelligence

1. INTRODUCCIÓN

El reconocimiento de patrones es el reconocimiento de características únicas que identifican un sujeto de los demás de la misma especie.

Por ejemplo podemos afirmar que las huellas digitales en los humanos poseen PATRONES o características indiscutibles que identifican la huella digital de una persona respecto a la de otras personas.

Se han hecho muchos estudios de la anatomía humana y se ha llegado a la conclusión de que los odontogramas y las formas de las orejas también poseen patrones que identifican a una persona respecto a las otras.

De modo similar se pueden encontrar patrones para diferenciar y reconocer cada uno de los billetes de las diferentes denominaciones.

En este artículo explicaremos en qué consiste y como se realiza dicho reconocimiento de patrones.

2. FUNDAMENTACIÓN TEÓRICA

2.1. Definición de redes neuronales artificiales (RNA) [1]

Son modelos matemáticos construidos basándose en el funcionamiento de las redes neuronales biológicas (sistema nervioso), por consiguiente, las unidades de procesamiento fundamental de una RNA, serán las neuronas artificiales.

Una red neuronal artificial la definiremos como un conjunto de unidades de procesamiento llamados neuronas, células o nodos, interconectados entre sí por varias ligaduras de comunicación directa llamadas conexiones, con la finalidad de recibir señales de entrada, procesarlas y emitir señales de salida. Cada conexión está asociada a un peso, que representan la información utilizada por las neuronas para resolver un problema. Como se puede apreciar en la Figura N.º 1.

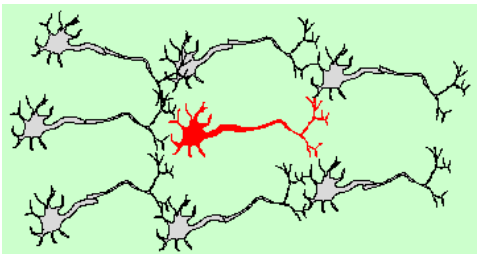
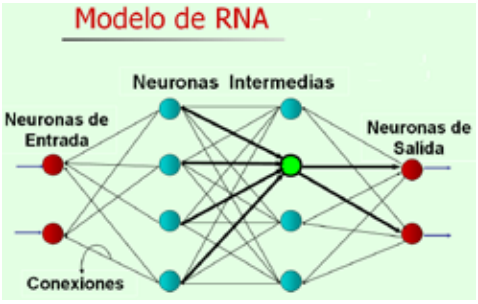


Figura N.º 1. Representación comparativa de una red neuronal natural y de una RNA.

Las entradas a una neurona pueden provenir de fuentes externas o de otras neuronas en la red. Asimismo la salida de una neurona es enviada a otras neuronas o al entorno. El conocimiento en una red neuronal está distribuido a lo largo de todo el sistema, debido a esto, se utilizan muchas interconexiones para obtener la solución de un problema en particular.



2.2. Características de las redes neuronales artificiales [2]

Las características de las RNA son, como apreciaremos, muy semejantes a las de las redes neuronales biológicas, entre las principales podemos mencionar las siguientes:

- Aprenden a través de ejemplos
- Inferencia estadística
- Adaptabilidad
- Dilema plasticidades y estabilidad
- Capacidades de generalización
- Tolerante a fallas.
- Rápida implantación

2.3. Principales aplicaciones de las RNA [3]

Las principales aplicaciones de las RNA son:

- Clasificación de Patrones
- Agrupamiento Categorización
- Aproximación de funciones
- Memoria direccionable por contenido

2.4. Funcionamiento de una neurona artificial [2]

De igual modo que las neuronas biológicas, una neurona artificial, tiene varias entradas y una sola salida, la misma que a su vez se puede aplicar a muchos otras neuronas. En la figura N.º 2 presentamos el esquema de una Neurona Artificial.

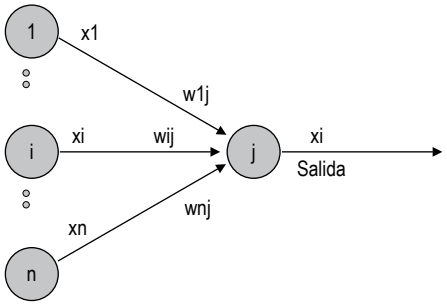


Figura N.º 2. Esquema de una neurona artificial.

Si comparamos las neuronas artificiales del presente gráfico con las neuronas biológicas, entonces (x_1,... xi,…xn) vendrían a ser los impulsos de entrada para la neurona j que provienen de las neuronas (1,..i,…n). Los pesos (w1j, …,wij,….wnj) representarían el potencial excitador o inhibidor de las conexiones sinápticas respectivas.

Cada elemento de procesamiento obtiene un valor de entrada neto basándose en todas las conexiones de entrada. La forma típica es calcular el valor de entrada neto sumando los valores de entrada, ponderados (multiplicados) con sus pesos correspondientes. La entrada neta del j-ésimo nodo se escribe:

Neto j = Σs xij * wij

2.5. Clasificación de las RNA según sus conexiones [2]

Las neuronas de una RNA se organizan en niveles (conocidos también como camada, capas o estratos). Dependiendo de las conexiones que unen las capas, las RNA se pueden clasificar en recurrentes (Feedback) y no recurrentes (Feedforward)

- No Recurrentes (Feedforward)
- Recurrentes (Feedback)

2.6. Principales modelos de redes neuronales artificiales [1] [2]

Veremos solamente el modelo Perceptrón y el modelo Retropropagación.

2.6.1. Modelo Perceptrón

El primer modelo de red neuronal artificial fue desarrollado por Rosenblatt en 1958, este modelo despertó un enorme interés en los años 60, debido a su capacidad para aprender a reconocer patrones sencillos: un Perceptrón, formado por varias neuronas lineales para recibir las entradas a la red y una neurona de salida, es capaz de decidir cuándo una entrada presentada a la red pertenece a una de las dos clases que es capaz de reconocer. A continuación en la figura N.º 3 podemos apreciar la arquitectura del perceptrón.

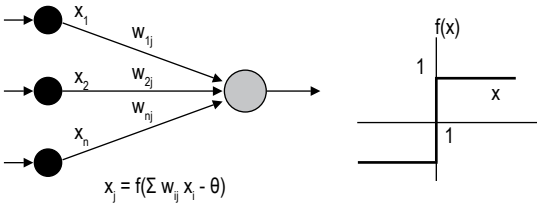


Figura N.º 3. Arquitectura del Perceptrón

La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de trasferencia de tipo escalón. La regla de decisión es responder +1 si el patrón presentado pertenece a la clase A y con –1 si el patrón pertenece a la clase B. La salida dependerá de la entrada neta (suma de las entradas Xi ponderadas) y del valor umbral 0.

Este modelo sólo es capaz de discriminar patrones muy sencillos, linealmente separables. La separabilidad lineal limita a las redes con sólo dos capas a la resolución de problemas en los cuáles el conjunto de puntos (correspondientes a los valores de entrada) sean separables geométricamente. En el caso de dos entradas, la separación se lleva a cabo mediante una línea recta, tal como puede apreciarse en la figura N.º 4. Para tres entradas, la separación se realiza mediante un plano en el espacio tridimensional, y así sucesivamente hasta el caso de N entradas, en el cual el espacio N-dimensional es dividido en un hiperplano.

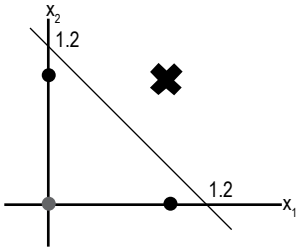


Figura N.º 4. Un problema linealmente separable: la función lógica OR

El caso de la función XOR es un problema que no puede resolver el perceptrón ya que no es linealmente separable. Sin embargo, es posible resolver correcta-

mente este problema usando una red de perceptrones (multiperceptrón). Tal como puede apreciarse en la figura N.º 5.

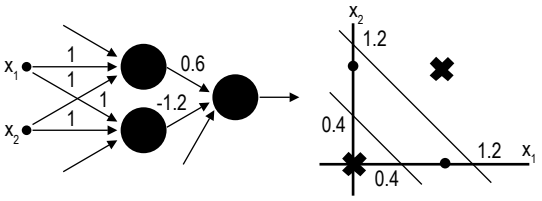


Figura N.º 5. Solución del XOR con una red Multiperceptrón

2.6.2. Modelo Retropropagación [2]

Rumelhart (1986) formalizó un método para que una red neuronal aprendiera una asociación que existe entre sus patrones de entrada y las clases correspondientes. Este método es conocido como backpropagation, propagación del error hacia atrás o retropropagación, y está basado en la regla de aprendizaje que es posible aplicar solo a modelos de redes multicapa. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en la capa o capas intermedias, para conseguir cualquier correspondencia entre la entrada y la salida de la red. En la figura N.º 6 puede apreciarse el RNA modelo retroprogamación.

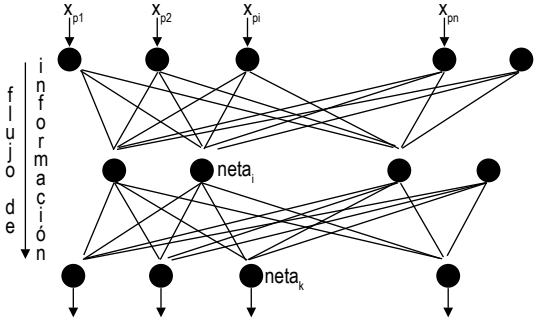


Figura N.º 6. RNA Modelo Retropropagación

Un punto importante en la red de retropropagación es su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes. Y después utilizar esa misma relación a nuevos vectores de entrada con ruido o incompletos, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje.

2.7. El aprendizaje en las redes neuronales artificiales [1] [2]

El problema principal al trabajar con RNA es programar el aprendizaje: ¿cómo escoger los pesos en las conexiones para que la red realice una tarea específica?

En algunos casos, es posible encontrar algunos problemas cuyos pesos se tienen de manera a priori y esta información es considerada para el diseño de la red, pero estos problemas son más bien raros ya que en la mayoría de los casos se debe enseñar a la red a ejecutar los cálculos por ajustes iterativos de los pesos wij. Esto puede tomar dos caminos.

2.7.1. Aprendizaje No Supervisado (Sin Maestro)

Cuando el fin del aprendizaje no es definir en términos específicos de ejemplos correctos, debido a que la información disponible solo está en correlación de datos o señales de entrada y solo se requiere, que en base a esta entrada, la red forme categorías de estas correlaciones, y produzca una señal de salida correspondiente a cada categoría de entrada. Se caracteriza porque la salida no requiere ser contrastada con algo específico ya conocido (maestro).

Hay dos formas de llevar a cabo el aprendizaje no supervisado:

- Aprendizaje hebbiano.
- Aprendizaje competitivo y cooperativo.

2.7.2. Aprendizaje Supervisado (con Maestro)

En este modo, el aprendizaje se logra basándose en la comparación directa de la salida de la red con la respuesta correcta (maestro) ya conocida. Se basa en el reforzamiento del aprendizaje, donde la retroalimentación se realiza en base a la diferencia de salida real con la salida esperada.

A continuación presentamos algunos tipos de aprendizajes supervisados:

- Aprendizaje supervisado por corrección de error.
- Aprendizaje supervisado por refuerzo.
- Aprendizaje supervisado estocástico.

a) Aprendizaje por Corrección de Error

El entrenamiento consiste en presentar al sistema un conjunto de pares de datos, representando la entrada y la salida deseada para dicha entrada. Este conjun-

to recibe el nombre de conjunto de entrenamiento. El objetivo es tratar de minimizar el error entre la Salida Deseada y la salida actual.

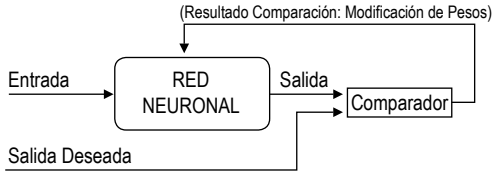


Figura N.º 7. Aprendizaje por Corrección de Error

Algoritmos de Aprendizaje por Corrección del error

Los pesos se ajustan en función de la diferencia entre los valores deseados y los obtenidos en la salida.

$\Delta w_{ij} = \alpha\, x_i\, (d_j - x_j)$

donde:

- Δw_{ij} Variación en el peso de la conexión entre el i-ésimo nodo y el j-ésimo.
- α Umbral en el aprendizaje que regula velocidad y precisión ($0 < \alpha \leq 1$)
- x_i Salida del i-ésimo nodo
- x_j Salida del j-ésimo nodo
- d_j Valor de salida deseado del j-ésima unidad de procesamiento

Regla de Mínimo Error Cuadrado.

Widrow y Hoff definieron una función que permitía cuantificar el error global cometido en cualquier momento durante el proceso de entrenamiento, agilizando este proceso

$$\frac{1}{2p} \sum_{i=1}^{n} (x_i^{(k)} - d_i^{(k)})^2$$

La fórmula calcula

n = número de nodos de salida

p = número de tramas de entrenamiento

Error cometido en el aprendizaje de la k-ésima trama.

Se trata de modificar los pesos para que las conexiones de la red minimicen esta función de error, se puede hacer de manera proporcionada a la variación relativa del error.

$\Delta w_{ij} = k\, (\partial \text{Error}_{global}) / \partial w_{ij}$

3. METODOLOGÍA PARA MODELAR UNA RNA PARA RECONOCER PATRONES

3.1. Planteamientos previos

Al margen de la estructura interna de una RNA, para trabajar en el reconocimiento de patrones debemos preocuparnos primeramente por establecer el número de neuronas en la capa de entrada y el número de neuronas en la capa de salida.

Considerando a una RNA como una caja negra podremos representar su interacción funcional con el entorno de la siguiente manera. (Figura N.º 8)

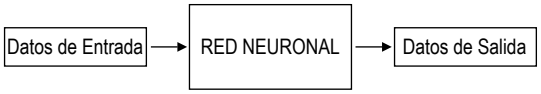


Figura N.º 8. Estructura Funcional de una Red Neuronal

Podemos concluir que los Datos de Entrada serán recibidas (leídas) por las Neuronas de Entrada las mismas que serán procesadas por la RNA y los resultados, serán entregados a los Datos de Salida, a través de las Neuronas de Salida.

Por lo tanto, los Datos de Entrada estarán en relación biunívoca con las Neuronas de Entrada y los Datos de Salida con las Neuronas de Salida lo que significa que a una RNA de n neuronas en la capa de entrada y m neuronas en la capa de salida le corresponderá como Datos de Entrada un vector X de tamaño n [X1, X2, …. Xn] y como Datos de Salida un vector Y de tamaño m [Y1, Y2, …. Ym], estableciéndose entre ellos una dependencia funcional que la podemos llamar RN, y que la podemos expresar de la siguiente manera: [Y1, Y2, …. Ym] = RN ([X1, X2, …. Xn]) (Figura N.º 9)

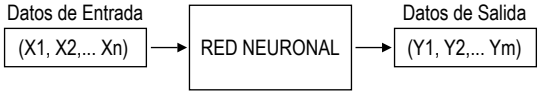


Figura N.º 9. Dependencia funcional de los datos de entrada y salida

3.2. Estructura de datos para representar los caracteres numéricos 3, 4 y 5 como datos de entrada

Para trabajar en el reconocimiento de patrones debemos encontrar una única estructura de datos que nos permita representar todos los elementos que desea-

mos reconocer. Para lograrlo, en nuestro caso, hemos seguido los siguientes pasos:

3.2.1. Representar los Patrones en una Matriz

Un artificio muy interesante para representar tanto letras del abecedario como los dígitos numéricos (A..Z, a..z, 0..9) es usar una matriz de 7 x 6 del modo mostrado en la figura N.º 10.

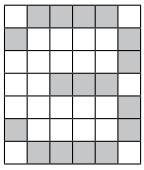


Figura N.º 10. Representar los Patrones en una Matriz

El artificio para representar un número en dicha matriz, será poner el valor uno (1) en las celdas por donde pasa la marca del número y el valor cero (0) en caso contrario. Por ejemplo el número tres se podría representar tal como muestra la figura N.º 11.

	1	2	3	4	5	6
1	0	1	1	1	1	0
2	1	0	0	0	0	1
3	0	0	0	0	0	1
4	0	0	1	1	1	0
5	0	0	0	0	0	1
6	1	0	0	0	0	1
7	0	1	1	1	1	0

0	1	1	1	1	0
1	0	0	0	0	1
0	0	0	0	0	1
0	0	1	1	1	0
0	0	0	0	0	1
1	0	0	0	0	1
0	1	1	1	1	0

Figura N.º 11. Artificio para Representar los Patrones en una Matriz

3.2.2. Convertir la Matriz en un Vector Lineal

Como se conoce los Datos de Entrada deben estar representados por un vector lineal, por esa razón en nuestro caso pasaremos de la matriz que debe ser 7 x 6 a un vector lineal X de tamaño 42, cuyos elementos serán, X1, X2, ... X42, según se precisa en la figura N.º 12.

	1	2	3	4	5	6
1	0	1	1	1	1	0
2	1	0	0	0	0	1
3	0	0	0	0	0	1
4	0	0	1	1	1	0
5	0	0	0	0	0	1
6	1	0	0	0	0	1
7	0	1	1	1	1	0

	1	2	3	4	5	6
1	X1	X2	X3	X4	X5	X6
2	X7	X8	X9	X10	X11	X12
3	X13	X14	X15	X16	X17	X18
4	X19	X20	X21	X22	X23	X24
5	X25	X26	X27	X28	X29	X30
6	X31	X32	X33	X34	X35	X36
7	X37	X38	X39	X40	X41	X42

Figura N.º 12. Convertir la Matriz en un Vector Lineal

Teniendo en cuenta dicha conversión, los 42 valores del vector X, para representar el número 3 de la figura anterior, serán mostrados en la figura N.º 13.

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	...	X42
0	1	1	1	1	0	1	0	0	0	0		0

Figura N.º 13. Representación del número 3 usando el vector X

3.2.3. Representar los caracteres que se desean reconocer usando las Estructuras de Datos

A continuación representaremos, usando las estructuras de datos, diversas presentaciones o modelos para los valores 3, 4 y 5. Aunque aquí solamente se muestran dos modelos para cada valor o patrón, en condiciones reales, para realizar el trabajo de reconocimiento de caracteres se deben buscar por lo menos siete modelos diferentes para cada elemento que se desee reconocer.

1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	1	1	1

0	1	1	1	1	0
1	0	0	0	0	1
0	0	0	0	0	1
0	0	1	1	1	0
0	0	0	0	0	1
1	0	0	0	0	1
0	1	1	1	1	0

1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1

0	0	0	1	1	0
0	0	1	0	1	0
0	1	0	0	1	0
1	0	0	0	1	0
1	1	1	1	1	1
0	0	0	0	1	0
0	0	0	0	1	0

1	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	0	0
1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	1	1	1

1	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	0	0
1	1	1	1	1	0
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	1	1	0

Figura N.º 14. Representación de Patrones usando Matrices

3.3. Estructura de datos para representar los datos de salida

Los Datos de Salida permitirán saber si un elemento de entrada ha sido reconocido o no por nuestra RNA. Por lo tanto, como en nuestro caso solamente deseamos reconocer tres valores numéricos (3, 4 y 5), entonces será suficiente un vector de dimensión 3, el cual por

corresponder a los Datos de Salida lo representaremos por el vector

Y = [Y1, Y2, Y3]

3.4. Planteamiento del modelo de una RNA para el reconocimiento de los caracteres numéricos 3, 4 y 5

De acuerdo a todo lo indicado anteriormente, el modelo funcional de la RNA para el reconocimientos de los números 3,4 y 5 podría quedar representado por la figura N.º 15, donde los Datos de Entrada estarán representados por el vector X = [X1, X2, …. X42] y los Datos de Salida estarán representados por el vector (Y = [Y1, Y2, Y3])

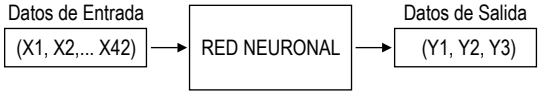


Figura N.º 15. Representación de los Datos de Entrada y Salida para el reconocimiento de Patrones

Tomando como ejemplo la imagen del número 3 mostrada en la figura 11, una representación gráfica más completa de los Datos de Entrada y los Datos de Salida seria la mostrada en la figura N.º 16.

Se puede apreciar que el vector X almacena la representación gráfica bidimensional del número 3 contenida en una matriz de 7 x 6 mientras que el vector Y tiene el valor [1, 0, 0] que indica que la imagen reconocida por la RNA corresponde al primer elemento en nuestro caso al número 3.

3.5. Proceso de entrenamiento de la RN y el reconocimiento de los caracteres numéricos 3, 4 y 5 utilizando un software

3.5.1. Preparar la Base de Conocimientos para el Entrenamiento de la RN

La base de conocimientos (experiencia) llamados también DATOS DE ENTRENAMIENTO de la RNA contiene en forma de una tabla todos los diferentes valores que se puedan dar a los vectores X e Y de modo que representen diversos modelos de los elementos que deseamos reconocer. Por ejemplo, en la figura N.º 17 hemos considerado dos modelos para cada número que deseamos reconocer, utilizando dichos ejemplos, la tabla que contiene los datos de entrada X y los datos de salida Y sería la mostrada en la figura N.º 17.

Se puede apreciar claramente que en cada línea, el vector X representa la forma gráfica de cada número mientras que el vector Y identifica el número que se está reconociendo.

Finalmente este archivo puede ser guardado en un archivo tipo texto para poderlo usar con más facilidad (digamos B_CONOCI.TXT)

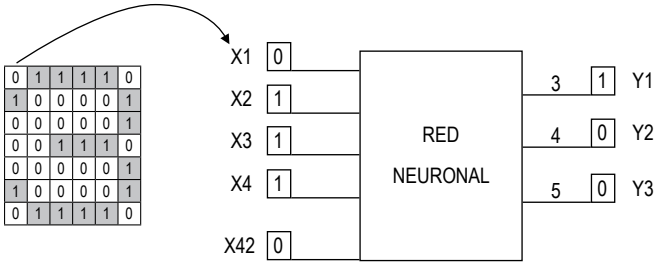


Figura N.º 16. Representación de los Datos de Entrada y Salida para el reconocimiento de Patrones para el número 3

BASE DE CONOCIMIENTOS O DATOS DE ENTRENAMIENTO

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X40	X41	X42	Y1	Y2	Y3
3	1	1	1	1	1	1	0	0	0	0	0	1	0	1	1	1	1	0	0
3	0	1	1	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0
4	1	0	0	0	0	1	1	0	0	0	0	1	1	1	0	1	0	1	0
4	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
5	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	1
5	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	1

Figura N.º 17. Base de Conocimientos o Datos de entrenamiento para el reconocimiento de los números de 3, 4 y 5

3.5.2. Proceso de Entrenamiento de la Red Neuronal

Para entrenar una RED NEURONAL se debe usar un software, nosotros usaremos el NEURONTRAINPATTERN. M que está desarrollado en MATLAB (los lectores, si desean pueden solicitar el programa fuente al correo del autor y se les enviará completamente gratis).

A continuación mostraremos con un ejemplo el proceso de entrenamiento usando el software mencionado.

a) Primer entrenamiento

En el entorno del MATLAB ejecutar el programa NEURONTRAINPATTERN

- Introducir nombre de archivo con data: B_CONOCI.TXT (Base de entrenamiento)
- Resultados de la lectura del archivo con data
- Número de entradas: 42
- Número de salidas: 11
- Número de paquetes de datos entrada-salida: 21
- Seleccionar de la Ventana Nueva: Serán generados automáticamente (SINAPSIS)
- Introducir neuronas en capa intermedia: 100
- Seleccionar de la Ventana nueva: Considera Neurona bias: SI
- Introducir ratio de aprendizaje: .05
- Introducir momento: 0
- Introducir ratio de aprendizaje de exponente a: 0
- Introducir ratio de aprendizaje del centro c: 0
- Introducir el valor máximo del error (%): 10
- Introducir el máximo etapas de aprendizaje: 200
- erreltotal = 1.9644 ...
- erreltotal = 0.0948
- Introducir nombre del archivo donde se guardará información de la red: numentre1.prn

Como podemos apreciar de la figura N.º 18, en la curva de aprendizaje, el proceso de entrenamiento ha alcanzado el objetivo esperado de tener un margen de error del 10% en 44 iteraciones. Si se desea reducir dicho margen de error se debe realizar un segundo reentrenamiento.

b) Segundo entrenamiento

En el entorno del MATLAB ejecutar el programa NEURONTRAINPATTERN

- Introducir nombre de archivo con data: numentre.prn
- Resultados de la lectura del archivo con data
- Número de entradas: 42
- Número de salidas: 11
- Número de paquetes de datos entrada-salida: 21
- Seleccionar de la Ventana nueva: Se leerán desde un archivo (SINAPSIS)
- Introducir nombre de archivo con información de la red: numentre1.prn
- Número de neuronas en capa intermedia: 100
- Neurona bias en capa de entrada (1:SI) (0:NO): 1
- Introducir ratio de aprendizaje: .02
- Introducir momento: 0
- Introducir ratio de aprendizaje de exponente a: 0
- Introducir ratio de aprendizaje del centro c: 0
- Introducir el valor máximo del error (%): 3
- Introducir el máximo número de etapas de aprendizaje: 200
- erreltotal = 0.0961
- erreltotal = 0.0297
- Introducir nombre del archivo donde se guardará información de la red: numentre2.prn

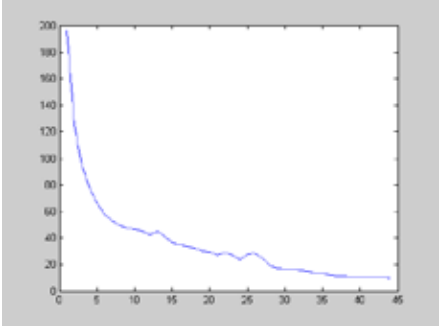


Figura N.º 18. Curva del Margen de Error (llegó al 10% en 43 iteraciones).

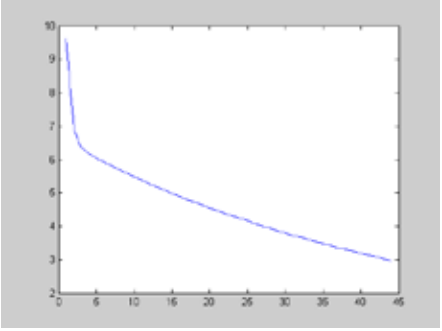


Figura N.º 19. Curva del Margen de Error (llegó al 3% en 44 iteraciones).

Como podemos apreciar en la curva de aprendizaje, de la figura N.º 19 podemos concluir que el proceso de entrenamiento ha alcanzado el objetivo esperado de tener un margen de error del 3% y por lo tanto podemos decir que la red neuronal está bien entrenada.

Por otro lado, el conocimiento almacenado de este entrenamiento ha sido guardado en el archivo numentre2.prn

3.5.3 Preparar la Base de Conocimientos para el Reconocimiento de los Caracteres Numéricos 3, 4 y 5

Ahora utilizaremos una tabla similar a la del entrenamiento. Tal como se puede apreciar en las figuras 20 y 21, los valores para el vector X, serán generados a partir de una nueva imagen por cada número que deseamos reconocer y respecto a los valores del vector Y, al iniciar el proceso de reconocimiento, todos serán 0 (ceros) y al finalizar dicho proceso la red neuronal mostrará el número que ha sido reconocido.

0	1	1	1	1	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
0	1	1	1	1	0

0	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0

0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	0	0
1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	1	1	0

Figura N.º 20. Imágenes a ser Reconocidas por la RN Entrenada

Se puede apreciar claramente que en cada línea, el vector X representa la forma gráfica de cada número que se desea reconocer, mientras que el vector Y, en este caso, está en 0 (cero) para todos los casos lo que representa que a la red neuronal no se le dice de que valor se trata, sino que el objetivo es justamente que la red neuronal, usando dicho vector Y pueda precisar que valor ha sido reconocido.

Este archivo puede ser guardado en un archivo tipo texto para poderlo usar con más facilidad (digamos B_RECONO.TXT)

3.5.4. Proceso para el Reconocimiento de los Caracteres

Con el mismo software, debemos seguir los siguientes pasos:

a) Reconocimiento

En el entorno del MATLAB ejecutar el programa NEURONTRAINPATTERN

- Introducir nombre de archivo con data: numreco.prn (MAESTRO)
- Resultados de la lectura del archivo con data
- Número de entradas: 42
- Número de salidas: 11
- Número de paquetes de datos entrada-salida: 31
- Seleccionar de la Ventana nueva: Se leerán desde un archivo (SINAPSIS)
- Introducir nombre de archivo con información de la red: numentre2.prn
- Número de neuronas en capa intermedia: 100
- Neurona bias en capa de entrada (1:SI) (0:NO) : 1
- Introducir ratio de aprendizaje: 0
- Introducir momento: 0
- Introducir ratio de aprendizaje de exponente a: 0
- Introducir ratio de aprendizaje del centro c: 0
- Introducir el valor máximo del error (%): 5
- Introducir el máximo número de etapas de aprendizaje: 1
- Introducir nombre del archivo donde se guardará información de la red: x

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X41	X42	Y1	Y2	Y3
3	0	1	1	1	1	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	1	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1	0	0	1	1	0	0	0

Figura N.º 21. Base de Conocimientos o Datos de entrenamiento para el reconocimiento de los números de 3, 4 y 5.

b) Interpretación de la Matriz de Salida

Al concluir el proceso de reconocimiento, el software tiene preparado una matriz de salida que la obtendremos e interpretaremos de la siguiente manera:

>> output (Valores obtenidos y pasados al Excell)

	Y1	Y2	Y3
3	1	0	0
4	0	1	0
5	0	-1	1

	Y1	Y2	Y3
3	0,7	0,1	0,2
4	-0	0,6	0,1
5	-0	-1	0,9

Figura 22: Matriz de Salida con Valores Reales y Valores Redondeados

Como se puede apreciar en la matriz de salida (OUTPUT) que se muestra en la figura N.º 22 las filas correspondientes a los patrones de entrada que son las formas de los números o patrones que se desean reconocer, mientras que las columnas representan los resultados del reconocimiento en este caso los valores de la figura 22 se explican de la siguiente manera:

- En la primera fila hay un 1 en la columna Y1 por lo tanto diremos que el contenido del vector X ha sido reconocido como el primer patrón, es decir como un 3.
- En la segunda fila hay un 1 en la columna Y2 por lo tanto diremos que el contenido del vector X ha sido reconocido como el segundo patrón, es decir como un 4.
- En la tercera fila, hay un 1 en la columna Y3 por lo tanto diremos que el contenido del vector X ha sido reconocido como el tercer patrón, es decir como un 5.

Los valores nulos o negativos simplemente deben ser ignorados.

CONCLUSIONES

1. Como resultado de este trabajo podemos concluir que el uso de las Redes Neuronales es una gran alternativa para la solución de muchos problemas.
2. Usando como base los resultados mostrados podemos resolver problemas más complejos como por ejemplo, reconocimiento de los patrones de la conducta humana, reconocimiento de enfermedades cardiacas, etc.
3. La debilidad que presenta este trabajo es que es una solución muy básica, pero al mismo tiempo creo que permite apreciar en detalle los pasos para realizar el reconocimiento de patrones que es la base de la visión artificial.

REFERENCIAS BIBLIOGRÁFICAS

[1] Hilera J, Martínez V. 1995. *Redes neuronales artificiales: fundamentos, modelos y aplicaciones.* Madrid: Addison-Wesley Iberoamericana. RA-MA. 390 p.

[2] Sánchez E, Alanis A. 2006**.** *Redes neuronales: conceptos fundamentales y aplicaciones a control automático.* Madrid. Prentice-Hall. 210.

[3] Dowla F, Rogers L. 1996. *Solving problems in environmental engineering and geosciences with artificial neural networks.* Cambridge. MIT Press. 310 p.

[4] Bishop C, 2006. *Neural Networks for Pattern Recognition.* New York. Oxford University Press. 504 p.