MatLab & Redes Neuronales

Rubén Wismark Plata Cheje
benhurwk@hotmail.com

RESUMEN

En el marco de las jornadas de conferencias de ingeniería electrónica JCEE'02, se presenta un breve resumen de una de las aplicaciones típicas de las redes neuronales artificiales ANN (Artificial Neural Network).
En ella, un perceptrón multinivel MLP (Multilayer Perceptron) se usa para el reconocimiento óptico de caracteres OCR (Optical Character Recognition). Por último, se simula una red neuronal en el entorno MATLAB, entrenándola mediante el conocido algoritmo back propagation BP.

Palabras Claves

Red neuronal, entradas, pesos, bias, función de transferencia, salidas, sumación, sigmoide, algoritmo de entrenamiento, feedforward, red multicapa, salida deseada, gradiente descendente, entrenamiento por lote, tan-sigmoide, radbas, pesos de la entrada, los pesos de las capas, momentum, mínimo local, backpropagation, GUI, nntool.

1. INTRODUCCION.

Las redes neuronales son una rama de la Inteligencia Artificial. En las redes neuronales el conocimiento se incorpora mediante el aprendizaje a partir de ejemplos.

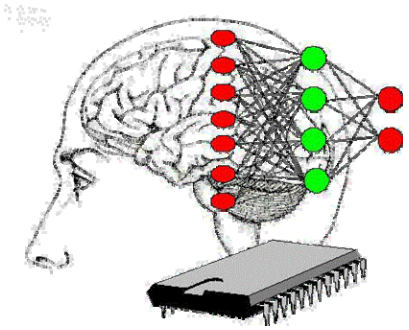

Figura1

Las redes neuronales como su nombre lo indica pretenden imitar a pequeñísima escala la forma de funcionamiento de las neuronas que forman el cerebro humano.

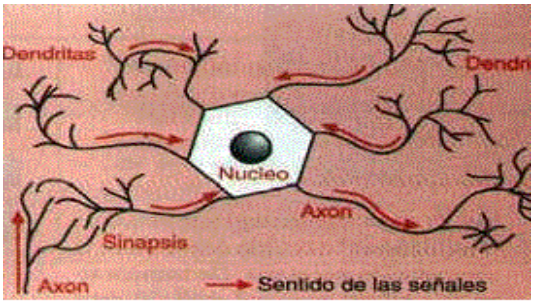

Figura 2.

Entonces en el campo de la IA(Inteligencia Artificial), las redes neuronales son dispositivos inspirados en la funcionalidad de las neuronas biológicas, aplicados al reconocimiento de patrones que las convierten aptas para modelar y efectuar predicciones en sistemas muy complejos. Las Rn son un conjunto de técnicas matemáticas para modelar las conexiones / relaciones entre un conjunto de datos.

2. MARCO TEORICO

2.1 Redes Neuronales

Una red neuronal artificial es un procesador distribuido en paralelo de forma masiva que tiene una tendencia natural para almacenar conocimiento de forma experimental y lo hace disponible para su uso.

Se parece al cerebro humano en dos aspectos:

- El conocimiento es adquirido por la red a través de un proceso de aprendizaje.
- Los pesos sinápticos o fuerza con que están interconectadas las neuronas se utilizan para almacenar la información.
- Otras definiciones son:
- Una nueva forma de computación, inspirada en modelos biológicos.
- Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.
- Un sistema de computación hecho por un gran numero de elementos simples, elementos de proceso interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.
- Redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

Aunque cada definición aporta información sobre lo que es una red neuronal, para saber como funcionan y en que consisten es necesario desarrollar una explicación extensa. Sin embargo, en este trabajo se describirá una aplicación típica de las redes neuronales multicapa, concretamente el reconocimiento de patrones.

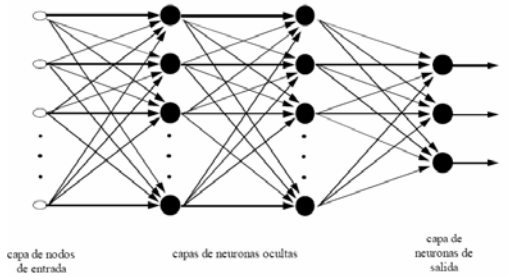

Fig. 1: Estructura de un perceptrón multinivel.

Figura 3.

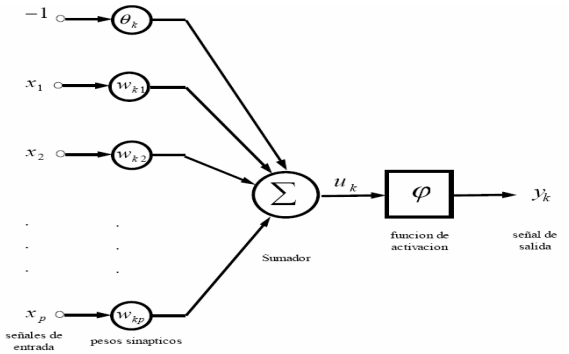

Fig. 2: Modelo de una neurona.

Figura4

En este tipo de tarea hay un número fijo de categorías en las cuales las muestras de entrada deben clasificarse. Para ello primero se requiere una fase de entrenamiento en la que se presenta a la red los patrones que debe aprender y la categoría en cual clasificarlo. Entonces se le presenta a la red un patrón nuevo y desconocido pero que pertenece a alguna de las categorías aprendidas y esta debe decidir a que categoría se parece más.

La ventaja de usar redes neuronales está en el hecho que se pueden separar regiones no lineales de decisión tan complicadas como se desee dependiendo del número de neuronas y capas. Por lo tanto, las redes neuronales artificiales sirven para resolver problemas de clasificación de alta complejidad.

2.1.1. Perceptrón Multinivel

Dentro de las redes neuronales, las que más utilizadas son las redes con múltiples capas que funcionan hacia delante. Esta red esta compuesta por un conjunto de nodos de entrada que componen la capa de entrada, un conjunto de una o más capas ocultas de neuronas y una capa de neuronas de salida. La señal de entrada se propaga hacia adelante desde la capa de entrada por la oculta hasta la salida; este tipo de configuración se conoce como MLP o "MultiLayer Perceptrons" (figura 1) [1],[2],[3].

El hecho de que este tipo de red se aplique para resolver con éxito multitud de problemas se debe a la utilización del algoritmo de aprendizaje que actualmente está más extendido, el algoritmo o regla back propagation, el cual es una generalización de la regla LMS "Least Mean Square", por lo tanto también se basa en la corrección del error.

Básicamente el proceso back propagation consiste en dos pasadas a través de las diferentes capas de la red, una pasada hacia adelante y una pasada hacia atrás. En la pasada hacia adelante, se aplica en la capa de entrada un patrón o vector de entrada, este propaga su efecto a través de las diferentes capas y como consecuencia produce un vector de salida. Durante este proceso, los pesos sinápticos de la red son fijos y no se modifican.

Durante la pasada hacia atrás en cambio, los pesos si se modifican de acuerdo con la regla de corrección del error.

La señal de salida real se compara con la señal deseada y como resultado se obtiene una señal de error, que se propaga en dirección contraria a través de la red modificando los pesos, de forma que, al volver a pasar el vector de entrada hacia adelante, la respuesta obtenida se asemeje más a la salida deseada. Concretando, se puede decir que un perceptron multicapa tiene tres características:

1. El modelo de cada neurona (figura 2) incluye una función no lineal. En este caso, a diferencia del perceptrón donde es la función escalón, y debido a la necesidad de que sea una función continua y derivable, es la función sigmoide, donde *uk* es la suma total de la actividad interna en la neurona *k* (la señal de entrada) e *yk* la salida que se produce en la neurona.

$$y_k = \frac{1}{1+\exp(-u_k)} \qquad (1)$$

2. La red contiene una o más capas ocultas de neuronas que no forman parte ni de la entrada ni de la salida. Estas neuronas ocultas capacitan a la red para aprender progresivamente cualquier correspondencia entre la entrada y la salida y almacenar internamente esta información.

3. La red posee un gran número de conexiones, estas vienen determinadas por los pesos de la red. Un cambio en la conexión entre las neuronas equivale a un cambio en los pesos.

La combinación de estas características, hace que la habilidad de esta red para aprender a partir del entrenamiento sea muy potente, por ejemplo es capaz de resolver el problema de la OR-exclusiva a diferencia del perceptrón.

De todas formas, este comportamiento hace que sea difícil conocer a priori la respuesta de la red. Esto se debe a dos motivos, el comportamiento no lineal de las neuronas, las cuales están muy interconectadas, (lo que hace difícil un análisis teórico de la red) y la existencia de neuronas ocultas, que impide poder "ver" como se produce el aprendizaje y determinar cuales son las características que mejorarían el aprendizaje.

El desarrollo del algoritmo back propagation proporciona un método eficiente para entrenar este tipo de redes. Aunque no es capaz de resolver todos los problemas, se ha demostrado como el mejor de todos. Su importancia está en su capacidad de autoadaptar los pesos de las neuronas intermedias para aprender la relación que existe entre el conjunto de vectores o patrones de entrada y su correspondiente salida, y poder aplicar esa relación después del entrenamiento a nuevos vectores de entrada imperfectos o con ruido. Esta capacidad se conoce como generalización. La red debe encontrar una representación interna que le permita generar las salidas deseadas durante la etapa de entrenamiento, y posteriormente durante el funcionamiento ser capaz de generar salidas para entradas que no le fueron mostradas durante el aprendizaje pero que se asemejan a alguna de las que si le fueron mostradas.

2.2 MatLab

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en

que se escribirían adicionalmente, sin necesidad de hacer uso de la programación tradicional.
MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyos especializados, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc. es un entorno de cálculo técnico, que se ha convertido en estándar de la industria, con capacidades no superadas en computación y visualización numérica.
De forma coherente y sin ningún tipo de fisuras, integra los requisitos claves de un sistema de computación técnico: cálculo numérico, gráficos, herramientas para aplicaciones especificas y capacidad de ejecución en múltiples plataformas. Esta familia de productos proporciona al estudiante un medio de carácter único, para resolver los problemas más complejos y difíciles.

2.2.1. Origen de MATLAB

nace como una solución a la necesidad de mejores y mas poderosas herramientas de calculo para resolver problemas de calculo complejos en los que es necesario aprovechas las amplias capacidades de proceso de datos de grandes computadores.
El nombre MATLAB viene de "matrix laboratory" (laboratorio matricial). MATLAB fue originalmente escrito para proveer acceso fácil al software matricial desarrollado por los proyectos LINPACK y EISPACK, que juntos representan el estado del arte e software para computación matricial. Hoy MATLAB es usado en una variedad de áreas de aplicación incluyendo procesamiento de señales e imágenes, diseño de sistemas de control, ingeniería financiera e investigación médica. La arquitectura abierta facilita usar MATLAB y los productos que lo acompañan para explorar datos y crear herramientas personalizadas que proveen visiones profundas tempranas y ventajas competitivas.

2.2.2. Iniciación al Matlab

El Lenguaje de Computación Técnica MATLAB es un ambiente de computación técnica integrada que combina computación numérica, gráficos y visualización avanzada y un lenguaje de programación de alto nivel.
Sea cual fuere el objetivo, un algoritmo, análisis, gráficos, informes o simulación, MATLAB lo lleva allí. El lenguaje flexible e interactivo de MATLAB permite a ingenieros y científicos expresar sus ideas técnicas con simplicidad. Los poderosos y amplios métodos de cómputo numérico y graficación permiten la prueba y exploración de ideas alternativas con facilidad, mientras que el ambiente de desarrollo integrado facilita producir resultados prácticos fácilmente.

MATLAB es la fundación numérica y gráfica para todos los productos de The MathWorks. MATLAB combina computación numérica, gráficos 2D y 3D y capacidades de lenguaje en un único ambiente fácil de usar.
Con su amplio rango de herramientas para modelar sistemas de control, análisis, simulación y procesamiento de prototipos, MATLAB es el sistema ideal para desarrollar sistemas avanzados de control. Usted puede modelar su sistema de control usando las cajas de herramientas para el diseño de controles avanzados de MATLAB - Control System, Robust Control, μ-Analysis and Synthesis, Model Predictive Control, QTF Control Design y LMI control. Posteriores análisis y refinamientos pueden ser efectuados estableciendo una simulación interactiva en Simulink, y luego sintonizar automáticamente los parámetros usando el Nonlinear Control Design Blockset. Finalmente, usted puede generar código C para correr en controladores incrustados con Real Time Workshop.
Combinando MATLAB con Signal Processing Toolbox, Wavelet Toolbox y un conjunto de herramientas complementarias - tales como Image Processing, Neural Network, Fuzzy Logic, Statistics y otras - usted puede crear un ambiente de análisis personalizado de señales y desarrollo de algoritmos DSP. Para simulación y desarrollo de prototipos usted puede agregar Simulink y el DSP Blockset para modelar y simular sus sistemas DSP, y luego usar Real-Time Workshop para generar código C para su hardware designado.

2.2.3. Características de MATLAB:

1. Cálculos intensivos desde un punto de vista numérico.
2. Gráficos y visualización avanzada.
3. Lenguaje de alto nivel basado en vectores, arrays y matrices.
4. Colección muy útil de funciones de aplicación.

Las poderosas capacidades de cálculo técnico de MATLAB se ponen a la disposición de los estudiantes, aunque limita el tamaño de las matrices a 8192 elementos, la edición de estudiante mantiene toda la potencia de la versión profesional de MATLAB 4.0, en una forma diseñada para que los estudiantes puedan ejecutarlo en sus propios ordenadores personales bajo Windows.
Toolbox especiales :
Se incluyen el Toolbox de señales y Sistemas (un conjunto de herramientas para el procesamiento de señal y para el análisis de sistemas de cuadro) y el Toolbox Symbolyc Math (herramienta de cálculo simbólico basada en Maple V).
A continuación presentamos la interfase de usuario de MATLAB 4.0 con el despliegue de una aplicación con grafica en 3D correspondiente al modelo Z=x^y-y^x su tabla de calculo y el análisis de la función.

2.2.4. Salidas o Presentaciones

MATLAB provee acceso inmediato a las características gráficas especializadas requeridas en ingeniería y ciencias. Potente graficación orientada a objetos gráficos le permite graficar los resultados de su análisis, incorporar gráficos en sus modelos de sistemas, rápidamente presentar complejos 3-D objetos, y crear resultados de presentación, entre lo cual se destaca:
Representaciones 2-D y 3-D, incluyendo datos triangulados y reticulados. Representaciones 3-D quiver, ribbon, y stem

- Control de fuentes, letras Griegas, símbolos, subíndices y superíndices

- Selección expandida de símbolos marcadores de curvas
- Gráficos de torta, de barras 3-D y gráficos de barras horizontales Gráficos 3-D y sólido modelado
- Representación de imágenes y archivos I/O
- Gráficos comentados
- Leer/Escribir archivos de datos Hierarchical Data Format (HDF)
- Presentación de OpenGL software y hardware
- Animación
- Display de buffer x rápido y exacto
- Soporte de colores verdaderos (24-bit RGB)
- Fuentes múltiples de luz para superficies coloreadas
- Vista basada en cámara y control de perspectiva
- Iluminación Plana, Gouraud y Phong
- Soporte eficiente de imagen de datos de 8-bit
- Control de eje y cámara
- Propiedades de superficie y patch
- Modelos de iluminación
- Control gráfico de objetos
- Impresión y representación de copias
- Formatos gráficos exportables
- Soporte de publicación de escritorio

2.2.5 Lista parcial de funciones especiales

Funciones Matemáticas

- Funcionales especiales y elementales
- Funciones gamma, beta y elípticas.
- Transformación de sistemas de coordenadas.
- Matriz identidad y otras matrices elementales.
- Matrices de Hilbert, Toeplitz, Vandermonde, Hadamard, etc.
- Partes reales, imaginarias y complejas conjugadas.
- Funciones trigonométricas y de potencias.

Algebra lineal numérica

- Valores propios y descomposición de matrices.
- Funciones generales de evaluación de matrices.
- Determinantes, normas, rangos, etc.
- Matrices inversas y factorización de matrices.
- Matriz exponencial, logarítmica y raíces cuadradas.

Polinomios e interpolación

- Interpolación 1-D y 2-D.
- Construcción polinomial.
- Interpolación por splines cúbicos.
- Diferenciación de polinomios.
- Evaluación de polinomios.
- Multiplicación y división de polinomios.
- Residuos de polinomios y residuos.

Métodos numéricos no lineales

- Búsqueda de ceros en funciones de una única variable.
- Minimización de funciones de una o más variables.
- Resolución numérica de integrales.
- Solución numérica de ecuaciones diferenciales ordinarias.

Estadística y análisis de Fourier

- Convolución 1-D y 2-D.
- Filtros digitales 1-D y 2-D.
- Transformadas de Fourier 1-D y 2-D y su inversa.
- Coeficientes de correlación y matrices de covarianza.

Deconvolución.

- Magnitudes y ángulos de fase.
- Funciones max, min, sum, mean y otras funciones de estadística básica.

Operaciones algebráicas y lógicas

- Suma, resta, multiplicación, división y potencias de matrices.
- Matrix traspuesta.
- Operadores lógicos AND, OR, NOT y XOR.

3 DESARROLLO

Para trabajar con redes neuronales, seguramente podremos encontrar con una simple búsqueda en Internet un gran número de API's y frameworks que implementen por nosotros la estructura de la mayor parte de los tipos de redes y la funciones necesarias para trabajar con ellas. Uno de estos frameworks es el Toolbox que matlab posee, que nos ofrece una implementación genérica de redes neuronales, así como implementaciones de redes neuronales concretas como las perceptrón, backpropagation, Som, etc.

Matlab utiliza una estructura única que nos dará acceso a todas las propiedades de la red neuronal, independientemente del tipo que esta sea, de manera que utilizando esta propiedad podremos modificar las entradas, capas, conexiones, pesos, etc. De esta manera una vez configurada la red neuronal según nuestras necesidades invocaremos las funciones de manipulación de redes neuronales disponibles en matlab, (simulación, entrenamiento, inicialización, etc.), pasándole como parámetro la estructura de la red neuronal.

Net = network;

Si ejecutamos el comando anterior y visualizamos el contenido de la variable *myNetwork* se nos vializará la estructura mencionada, la cual se puede dividir en cinco secciones:

1.-Arquitectura: Define las características básicas de la red neuronal, número de entradas, capas, conexiones de bias, etc.
2.-Subobjetos: Contiene referencias a las subestructuras de la red neuronal, que nos permitirán configurar las propiedades de los distintos componentes que forman la red (capas, entradas, salidas, etc.).
3.-Funciones: Funciones principales de la red neuronal, utilizadas para ejecutar las operaciones de inicialización, entrenamiento o simulación.
4.-Parámetros: Configuración de los parámetros asociados a las funciones seleccionadas en el bloque de funciones.
5.-Valores: Aquí se definen las matrices con los valores de los pesos de entrada, conexiones entre capas y vías.

Una vez creada la red neuronal, para trabajar con la misma, podremos utilizar las siguientes funciones para realizar las operaciones típicas:

Inicialización (net = init(net)):
Mediante la función de inicialización, obtenemos una red neuronal con los valores de los pesos y bias actualizados según las funciones de inicialización que le hayamos asociado a la red, mediante su propiedad net.initFcn, o bien net.layers{i}.initFcn y net.biases{i}.initFcn.

Entrenamiento ([net, tr, Y, E, Pf, Af] = train(net, P, T, Pi, Ai, VV, TV);):

Realiza el entrenamiento de la red neuronal, modificando los pesos asociados a las conexiones entre las diferentes capas y neuronas de la misma. Para esto, debemos indicar unos patrones de entrada a la red (P, matriz de dimenesiones MxN siendo M la suma de los tamaños de las capas de entrada de la red neuronal, y N el número de patrones que deseamos aplicar en el entrenamiento). En caso de ser un entrenamiento supervisado también indicaremos los targets (T, matriz de MxN), con estos datos la matriz de patrones se aplica a la red neuronal, y el toolbox utilizando las funciones de entrenamiento que le hemos indicado en las propiedades "*trainFcn*" se encargará de actualizar los pesos asociados a las conexiones de la red. Los resultados del entrenamiento los obtendremos en la variable de retorno Y y los errores para cada patrón de entrada respecto a la salida esperada en la variable de retorno E.

Simulación ([Y, Pf, Af, E, perf] = sim(net, P, Pi, Ai, T)):
Función parecida a la anterior pero que no actualizará los pesos de la red neuronal. Una vez que tengamos entrenada la red neuronal y esta ofrezca unos resultado válidos, utilizaremos esta función para analizar nuevos patrones de entrada.

Redes neuronales conocidas y predefinidas por matlab
Normalmente a la hora de trabajar con redes neuronales, querremos trabajar con un tipo de red neuronal concreto, el cual se ajuste mejor a nuestras necesidades. En este caso en vez de utilizar la función "network" para la creación de la estructura base, podemos utilizar funciones específicas para cada tipo de red neuronal, de manera que la estructura base que matlab nos devuelva tenga una configuración de capas de entrada, ocultas, conexiones etc apropiada para el tipo de red neuronal deseado.

Perceptron: newp(P,S)
Backpropagation: newff(P, [S1,…., Sn])
Radiales: newgrnm(P,T)
Mapas Autoorganizados: newsom(P,S)

3.1. Neural Network Toolbox

Este toolbox proporciona funciones para el diseño, inicialización, simulación y entrenamiento de los modelos neuronales de uso más extendido en la actualidad: Perceptrón, redes lineales, redes de retropropagación, redes de base radial, aprendizaje asociativo y competitivo, aplicaciones autoorganizativas, aprendizaje de cuantización vectorial, redes de Elman y redes de Hopfield.

Mediante la inclusión de un amplio abanico de funciones y procedimientos escritos para MATLAB, el usuario puede mediante el Neural Network Toolbox efectuar el diseño de arquitecturas complejas, combinando los modelos que ya estan proporcionados por defecto en el toolbox. Asimismo, el usuario puede definir sus propias funciones de transferencia e inicialización, reglas de aprendizaje, funciones de entrenamiento y estimación de error para usarlas posteriormente con las funciones básicas.

El toolbox, aporta las facilidades y prestaciones gráficas de MATLAB para el estudio del comportamiento de las redes: visualización gráfica de la matriz de pesos y vector de desplazamiento mediante diagramas de Hinton, representación de errores a lo largo del entrenamiento, mapas de superficie de error en función de pesos y vector de desplazamiento, etc. Estos gráficos resultan muy útiles en el estudio de la convergencia y estabilidad de los algoritmos de aprendizaje.

Este toolbox incluye un manual de introducción al campo de las redes neuronales (ayuda del software) junto con una colección de demostraciones y aplicaciones muy didácticas, útiles para el estudio y la profundización en las cuestiones fundamentales de los paradigmas de redes neuronales básicos. Asimismo, se proporcionan las referencias bibliográficas más significativas referidas a los distintos modelos que aparecen en la aplicación.

A pesar de que el estudio de las redes neuronales se inició ya hace algunas décadas, las primeras aplicaciones sólidas dentro de este campo no han tenido lugar hasta hace unos doce años y aun ahora constituyen un área de investigación en rápido desarrollo. Este toolbox tiene por tanto una orientación diferente a aquellos destinados a campos como el de sistemas de control u optimización donde la terminología, fundamentos matemáticos y procedimientos de diseño estan ya firmemente establecidos y se han aplicado durante años.

Este toolbox pretende que sea utilizado para la valoración y diseño de diseños neuronales en la industria y sobre todo en educación e investigación.Esta herramienta tiene el soporte de MATLAB 4.2c y SIMULINK. La librería de SIMULINK contiene modelos de capas de redes neuronales de cada tipo de neurona implementada en el toolbox de redes neuronales. Es posible por tanto diseñar sistemas SIMULINK para simular redes neuronales creadas usando esta herramienta. Simplemente, las capas se conectan de acuerdo con la arquitectura de la red y se proporcionan como entrada a la caja de diálogo de cada capa la matriz de pesos apropiada y el vector de desplazamiento. Usando el generador de código C de SIMULINK es posible generar automáticamente el código correspondiente a un diseño neuronal.

Dentro de las aplicaciones básicas de este toolbox, cabe destacar aquellas que están orientadas a aquellas que se enmarcan dentro del campo de la industria aeroespacial y automoción (simulación, sistemas de control, autopilotaje), banca, defensa (reconocimiento de patrones, procesamiento de señales, identificación de imágenes, extracción de características, compresión de datos), electrónica (control de procesos, análisis de errores, modelado no lineal, síntesis de voz, visión por ordenador), economía (análisis financiero, análisis predictivo), industria (control de procesos, identificación en tiempo real, sistemas de inspección), medicina, robótica (control de trayectorias, sistemas de visión), reconocimiento y síntesis del habla, telecomunicaciones (control de datos e imágenes, servicios de información automatizada, traducción del lenguaje hablado en tiempo real, diagnosis, sistemas de enrutamiento), etc. El toolbox contiene muchos ejemplos de algunas de estas aplicaciones.

4.- APLICACION

Para simular el funcionamiento de un perceptrón multinivel entrenado mediante el algoritmo back propagation, se plantea un sencillo problema de reconocimiento de óptico de caracteres. Su descripción es la siguiente:

Dado un panel de entrada compuesto por una matriz de 7x5 puntos, se consideran 12 clases diferentes donde se pretenden clasificar las muestras que se introducen. Los patrones que definen correctamente a cada una de las clases
son los números del 0 al 9, el punto y el guión (figura 3).
Cuando a la entrada se presente una muestra distinta de los patrones correctos, el sistema presentará a su salida la información decodificada de la clase a la que pertenece la muestra, o bien, de la clase a la cual se aproxima más.

En base a este planteamiento, la red neuronal dispone de 35 entradas que se corresponden con los puntos de la matriz numerados en la figura 4. El valor de cada entrada puede ser 0 si el punto es blanco y 1 si el punto es negro. Por otro lado, dispone de 12 salidas, una por cada clase. Cuando se introduzca una muestra a la entrada únicamente se activará la salida de la clase a la que pertenezca, permaneciendo las 11 restantes desactivadas con valores próximos a cero. Se considera que una salida está activada cuando su valor es próximo a la unidad.

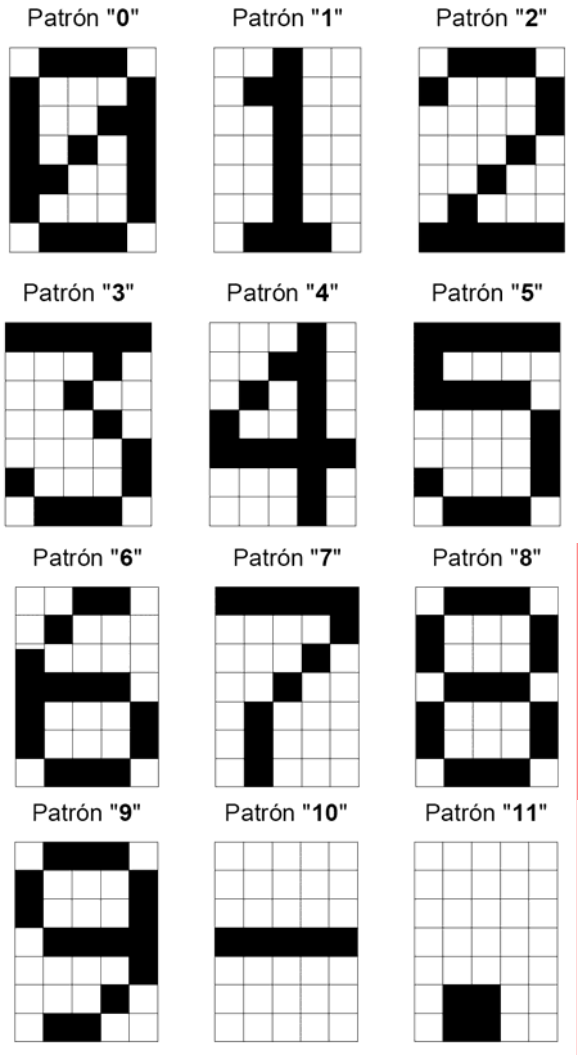

Figura 5. Representación gráfica de los patrones de los caracteres numéricos en el formato de matriz de puntos 7x5

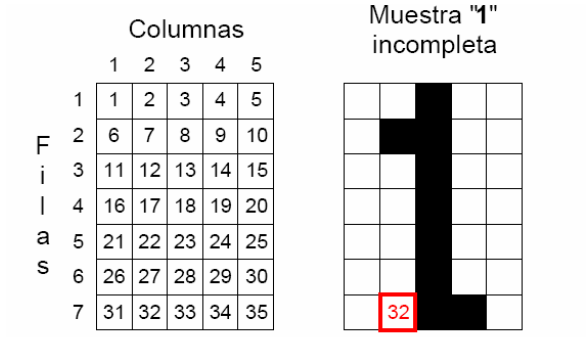

Fig. 4: *Numeración de los puntos del display 7x5 y ejemplo de una muestra del patrón "1" con un punto erróneo.*

Resultados de simulación

Se ha desarrollado el programa *BP5.m* para el entorno MATLAB 5.0 Student Edition. En él se ha programado un perceptrón multinivel con 35 entradas y 12 salidas. También dispone de dos capas ocultas a las cuales se les puede modificar el número de sus neuronas. La red neuronal se ha entrenado con el algoritmo back propagation fijando el valor del momento en 0.8 y el factor de aprendizaje en 0.2. En este proceso únicamente se han usado doce muestras diferentes, es decir, los doce patrones sin ningún punto erróneo.

En la tabla I se muestran los resultados obtenidos para una red neuronal de tamaño 35-30-20-12. Se aprecia que tras el proceso de entrenamiento, el sistema responde de forma casi ideal cuando se introduce un patrón sin error.

Salida ideal.	Patrón "1" sin error.	Patrón "1" con error en el punto 32.
0	0.00000000038246	0.00000000022760
1	0.97094634494005	0.93598496351919
0	0.00483116118854	0.00425679820665
0	0.00000007267857	0.00000003653425
0	0.00001815880785	0.00001549628867
0	0.00000000010405	0.00000000001916
0	0.00000098764700	0.00000340726166
0	0.00038787588951	0.00012976360904
0	0.00016876462031	0.00008967151863
0	0.00127029941580	0.00051591379971
0	0.02144501839329	0.01084408021869
0	0.01863962414026	0.06933193518770

Tabla I: *Resultados de simulación.*

De la misma manera, cuando se introduce una muestra con error en un punto (figura 4), la red clasifica perfectamente dicha muestra en la clase correcta. Este sencillo ejemplo sirve para confirmar que los perceptrones multinivel resuelven excelentemente el problema de clasificación de muestras, compitiendo con otros métodos como puedan ser los estadísticos.

5.- CONCLUSION

- Las redes neuronales todavía se han de desarrollar mucho. Aún se debe estudiar para que sirven realmente, conocer en que tareas pueden resultar realmente útiles, ya que por ejemplo es

difícil saber cuánto tiempo necesita una red para aprender cierta tarea, cuántas neuronas necesitamos como mínimo para realizar cierta tarea, etc...

- En la robótica, las redes neuronales también parecen prometer mucho, sobre todo en su sensorización.

6. BIBLIOGRAFIA

[1] MATLAB User's Guide, The MathWorks, Inc., Massachusetts, 1995.

[2] Joan Cabestany Moncusí, Sergi Bermejo Sánchez; "Xarxes Neuronals".

[3] Disponible en http://petrus.upc.es/~microele/neuronal/xn/J. R. Hilera; V. J. Martínez; "Redes neuronales artificiales.

[4] Fundamentos, modelos y aplicaciones". Ed. Rama, 1995.

[5] S. Haykin; "Neural networks. A comprehensive foundation". IEEE Press, 1994.

[6] Joan Cabestany Moncusí, Sergi Bermejo Sánchez; "Xarxes Neuronals". Disponible en: petrus.upc.es/ microele/ neuronal