Inteligencia Artificial

Practica 2. (5 semanas)

Rubén Cárdenes Almeida

Redes neuronales.

Introducción

Las Redes Neuronales Artificiales (RNA) están inspiradas en la biología, esto significa que están formadas por elementos que se comportan de manera análoga a las neuronas (en las funciones más elementales) y están organizadas de una forma similar a la del cerebro, pero las analogías no son muchas más.

Las características fundamentales de las RNA son:

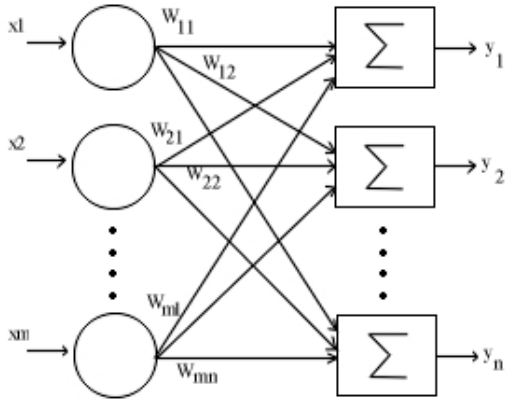
- **Aprenden de la experiencia:** Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas (quizá con las salidas deseadas), las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se han desarrollado, cada uno con sus propias ventajas e inconvenientes.
- **Generalizan de ejemplos anteriores a los ejemplos nuevos:** Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.
- **Abstracción de la esencia de las entradas:** Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento de patrones, una red puede ser entrenada en una secuencia de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto.

Redes de capa simple

A pesar de que una sola neurona puede realizar modelos simples de funciones, su mayor productividad viene dada cuando se organizan en redes. La red más simple es la formada por un conjunto de perceptrones a los que entra un patrón de entradas y proporcionan la salida correspondiente. Por cada perceptrón que tengamos en la red vamos a tener una salida, que se hallará como se hacía con un perceptrón solo, haciendo el sumatorio de todas las entradas multiplicadas por los pesos. Al representar gráficamente una red, se

añade una "capa" inicial que no es contabilizada a efectos de computación, sólamente sirve para distribuir las entradas entre los perceptrones. La denominaremos la capa 0.

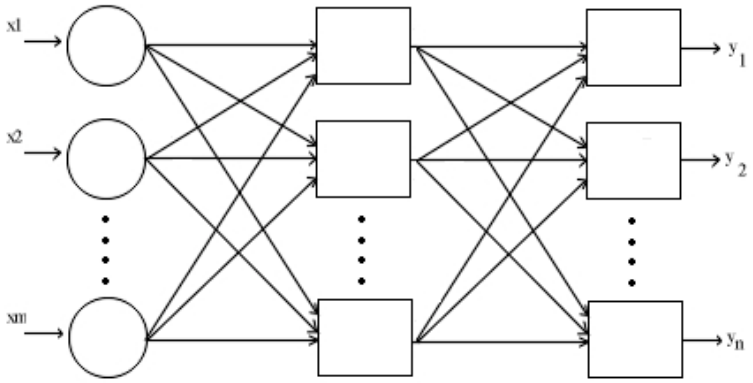
De esta manera, la representación gráfica de una red de capa simple sería la siguiente:



Redes multicapa

Las redes multicapa se forman por un conjunto de redes de capa simple en cascada unidas por pesos, donde la salida de una capa es la entrada de la siguiente capa. Generalmente son capaces de aprender funciones que una red de capa simple no puede aprender, por lo que ofrecen mejores capacidades computacionales. Para que este incremento en poder computacional sea tal, tiene que existir una función de activación no lineal entre las capas, por lo que generalmente se utilizará una función de activación sigmoidea en detrimento de la lineal o umbral.

Para calcular la salida de una red multicapa se debe hacer de la misma manera que en las redes de capa simple, teniendo en cuenta que las salidas de una capa son las entradas de la siguiente capa.:



Redes recurrentes: Las redes consideradas hasta ahora no tienen conexiones entre pesos de la salida de una capa a la entrada de la misma capa o anteriores. Las redes que poseen esta característica son conocidas como redes recurrentes. Las redes recurrentes no tienen memoria, es decir, la salida solamente está determinada por las entradas y los pesos. Las capas recurrentes redireccionan previas salidas a entradas. Su salida es determinada por su entrada y sus salidas previas, por lo que se puede asemejar a la memoria a corto plazo de los seres humanos.

Software SNNS.

El SNNS (*Stuttgart Neural Network Simulator*) es un software de simulación para redes neuronales desarrollado en el *Institute for Parallel and Distributed High Performance Systems* de la Universidad de Stuggart. El objetivo del proyecto SNNS fue la creación de un entorno de simulación eficiente y flexible para la investigación y aplicación de redes neuronales.
El simulador consta de dos componentes principales:

- núcleo escrito en C
- una interfaz gráfica.

El núcleo opera con las estructuras de datos de las redes neuronales y realiza todas las operaciones de aprendizaje y prueba. También puede ser utilizado como un programa en C. Permite trabajar con topologías arbitrarias de redes. Adicionalmente, permite que el usuario extienda los programas añadiendo funciones de activación, funciones de salida y funciones de aprendizaje, que se escriben como programas en C y se enlazan con el núcleo del simulador.
La interfaz gráfica XGUI (X Graphical User Interface) permite la representación gráfica en 2D y 3D de las redes neuronales y controla el núcleo durante una ejecución.
El paquete SNNS se puede obtener via ftp anónimo de:

ftp informatik.uni-tuebingen.de

en

/pub/SNNS/SNNSv4.2.tar.gz (2.18 MB).

En esta practica sólo utilizaremos la interfaz grafica.

Ejercicio 1. Ejemplo de funcionamiento del programa.

En este ejercicio veremos como cargar una red neuronal desde archivo en el programa de simulación SNNS, como crear una red neuronal, visualizarla y entrenarla, y visualizar el error. Para ello usaremos un ejemplo muy sencillo como es la función XOR. Para ello son necesarios los siguientes ficheros disponibles desde la página web de la signatura:

- Programa SNSS para linux: snns.bin.
- red neuronal en formato SNNS: xor.net
- Datos de entrenamiento: xor.pat.

Una vez descargados los ficheros seguiremos los siguientes apartados:

Ejecución del programa.

1. Abrimos una terminal en Linux, y nos aseguramos de que estamos en el mismo directorio donde hemos descargado el fichero snns.bin. Para ejecutarlo escribimos:
   ./snns.bin
2. Si existen problemas a la hora de ejecutar el comando, es posible que el fichero no tenga permisos de ejecución, por lo que habría que darle permiso de ejecución así:
   chmod +x snns.bin
   y entonces ya podremos ejecutar el programa.

Cargando ficheros de red y datos

Vamos a cargar desde ficheros, una red neuronal y un fichero de datos.

1. Selecciona FILE en el Manager panel. Esto genera otra caja de diálogo (Gestor de ficheros) que gestiona los ficheros (de tipo red (NETWORK), de tipo datos (PAT), de tipo resultados (RES), de tipo configuración (CFG) y de tipo traza (TXT)).
2. Pulsa sobre “NET" para indicar que quieres cargar un fichero de tipo red.
3. Teclea el nombre del fichero (en este caso, “xor")
4. Pulsa LOAD
5. Pulsa sobre “PAT" para indicar que quieres cargar un fichero de datos
6. Teclea el nombre del fichero (en este caso, “xor")
7. Pulsa LOAD
8. Cierra la caja de diálogo pulsando DONE

Visualizando la red

1. Selecciona DISPLAY (Visualizador de redes) del Manager panel. Esto genera una visualización de la red que acabas de cargar.
2. Pulsa el botón SETUP y sobre todos los botones a la derecha de “links". Termina con “DONE”. Ahora deberás ver la red con las conexiones (y sus pesos).

Entrenando la red y visualizando el error

1. Primero, selecciona GRAPH (Gráfica de error) del Manager panel. Verás una ventana donde se mostrará el error del aprendizaje de la red.
2. Selecciona CONTROL del Manager panel. Se abre una caja de diálogo (Panel de control) para controlar el aprendizaje:
   - Selecciona la función de aprendizaje *Std Backpropagation* de las opciones LEARN-SEL.FUNC.
   - En la caja CYCLES introduce, por ejemplo, 1000. (La red se entrenará durante 1000 ciclos.)
   - Pulsa sobre ALL para entrenar. Mira la ventana de la gráfica de error. ¿Se va reduciendo el error? (Lo que acabas de hacer es entrenar con la función de aprendizaje Std Backpropagation 1000 ciclos la red cargada utilizando el fichero de datos “xor".)

3. Para volver a inicializar los pesos de la red pulsa sobre INIT en el panel de control.

Probando la red entrenada

1. Fíjate, en la caja CONTROL, que el fichero que vas a utilizar es “xor" y que vas a testear la red con la muestra 1 (indicado en la caja de PATTERN).
2. Pulsa sobre el botón TEST. Mira la ventana donde se visualiza la red. ¿Qué ha ocurrido? ¿Ha sido la red capaz de clasificar correctamente esta nueva muestra?
3. Sigue pulsando el botón TEST para ir viendo cómo se van clasificando el resto de muestras.

Creando una red multicapa.

Vamos a crear una red nosotros mismos, para resolver el mismo problema de XOR. En este caso, haremos una red 2-2-1. Es decir, con dos neuronas en la capa de entrada, 2 neuronas en la capa oculta, y una neurona en la capa de salida. Para ello:

1. Selecciona BIGNET (Creación de redes) del Manager panel
2. Selecciona ”general”, y entrarás en la caja de diálogo de BIGNET.
   Esta caja te permite crear una red en términos de planos (vectores 2-D de nodos). Para una red multicapa, es suficiente especificar las capas. Una capa es un tipo especial de plano con anchura 1 (1 nodo en la dirección del eje x).
3. Introduce las unidades de cada capa:
   - Para crear la primera capa (tipo input)
     - Introduce 1 como No. of units in x-direction
     - Introduce 2 como No. of units in y-direction
     - Pulsa el botón “ENTER"
   - Para crear la segunda capa (tipo hidden)
     - Pulsa en el botón “TYPE" hasta obtener tipo hidden
     - Introduce 1 como No. of units in x-direction
     - Introduce 2 como No. of units in y-direction
     - Pulsa el botón “ENTER"
   - Para crear la tercera capa (tipo output)
     - Pulsa en el botón \TYPE" hasta obtener tipo output
     - Introduce 1 como No. of units in x-direction
     - Introduce 1 como No. of units in y-direction
     - Pulsa el botón “ENTER"

   Si quieres corregir algún dato, utiliza el botón ”PLANE TO EDIT" y luego edita.
4. Ahora pulsa sobre “FULL CONNECTION" al final de la caja de diálogo
5. A continuación pulsa sobre “CREATE NET"
6. Ahora ya puedes cerrar la caja de diálogo (pulsa “DONE")

Ahora visualiza la red, vuelve a entrenar la con el archivo xor.pat de la misma manera que antes, y fijándote en el error. Guarda la red en un archivo llamado xor221.net.

Ahora crea una red del tipo 2-3-1, (con una neurona más en la capa oculta), visualizala, vuelve a entrenarla de la mima manera, y fíjate de nuevo en el error. Guardala como xor231.net.

Ejercicio 2. Realización de una red neuronal que haga clasificación de patrones sencillos.

En este ejercicio vamos a realizar la clasificación de unos patrones (los dados en el fichero **ej2-test.pat,** con un total de 20 patrones), a partir de la información de entrenamiento dada en los patrones del fichero **ej2-train.pat,** que tiene un total de 80 patrones. Cada uno de estos patrones corresponde a las características de un billete, representadas con 4 números reales, y el objetivo es determinar si el billete es falso (=0) o no lo es (=1).
Para ello utilizaremos una red neuronal como las anteriores, con tres capas: capa de entrada, capa oculta (con 2 neuronas) y capa de salida.

Usando el SNNS, construir la red neuronal, entrenarla, ver el error, Finalmente clasificar los 20 patrones de test dados en el fichero **ej2-test.pat**. Aunque este fichero tenga todos sus patrones con salida igual a cero, es decir, todos lo billetes falsos, en realidad existen billetes verdaderos y falsos. El objetivo del ejercicio es averiguar cuáles son unos y otros.