WEB DIDÁCTICA PARA LA APROXIMACIÓN DE FUNCIONES MEDIANTE REDES NEURONALES

Julio Comabella López
Grup de Proc. de Senyal
Universitat de Vic
julio.comabella@bull.net

Jordi Solé i Casals
Grup de Proc. de Senyal
Universitat de Vic
jordi.sole@uvic.es

Sharam Hosseini
LIS
Institut National
Polytechnique de Grenoble
Sharam.Hosseini@inpg.fr

Christian Jutten
LIS
Institut National
Polytechnique de Grenoble
Christian.Jutten@inpg.fr

ABSTRACT

In this article we presents a project [1] developed to demonstrate the capability that Multi-Layer Perceptrons (MLP) have to approximate non-linear functions [2]. The simulation has been implemented in Java to be used in all the computers by Internet [3], with a simple operation and pleasant interface. The power of the simulations is in the possibility of the user of seeing the evolutions of the approaches, the contribution of each neuron, the control of the different parameters, etc. In addition, to guide the user during the simulation, an online help has been implemented.

1. INTRODUCCIÓN

Una red MLP [4] es una red multicapa unidireccional con conexiones totales entre dos capas. La red está constituida de una capa de entrada, una capa de salida y de una o varias capas ocultas. Cada capa esta formada por un conjunto de unidades no relacionadas entre sí. Los algoritmos que se presentan para la aproximación de funciones utilizan MLP's de una sola capa oculta.

1.1. Parámetros de la función

La página permite escoger entre diferentes funciones en un menú desplegable. La potencia de ruido que contamina estas funciones es un parámetro regulable por el usuario. El ruido se genera aleatoriamente a partir de una semilla (parámetro), para demostrar la validez de los resultados obtenidos.

1.2. Parámetros de los algoritmos

Una vez seleccionada la función a aproximar con el ruido deseado, se selecciona los parámetros relacionados con los algoritmos utilizados, el algoritmo de optimización y el criterio de parada a partir del valor de error o el número de iteraciones que limitará el número de intentos para la aproximación. También se escoge el número de neuronas a utilizar.

1.3. Aproximación de la función

Con todos los parámetros seleccionados anteriormente se lanza la simulación y por cada iteración se calculan los pesos de las neuronas y el error de salida. El criterio de parada es doble o triple según la simulación; por la superación del número de iteraciones o intentos de aproximación, por haber minimizado el error o por sobrepasar el número de neuronas a utilizar.

2. PROGRAMACION

Para hacerlo innovador y accesible a todo el mundo se optó por implementar los algoritmos en Java.

2.1. Java

Este lenguaje de programación tiene muchas ventajas respecto a sus competidores. Es accesible a partir de cualquier máquina que tenga un navegador de Internet. Es rápido de descargar y seguro, pues sólo utiliza el entorno del navegador en la máquina cliente. Es multiplataforma pues es independiente del sistema operativo de la máquina. La única restricción queda en el navegador utilizado para ejecutar el *applet* y normalmente, viene en función de la versión del compilador de Java utilizado por el navegador.

2.2. Estructura

La programación de las simulaciones se ha hecho a partir de librerías creadas especialmente para soportar el tipo de operaciones a realizar por los algoritmos. Se han implementado un conjunto de funciones para los diferentes dominios, como las operaciones entre matrices (sumas, productos, traspuestas, inversas, etc.), las funciones gráficas (gráficos normalizados, ventanas con los ejes para las señales, etc.), generador de señales (generadores de las diferentes señales de entrada), y cálculos matemáticos (derivadas, autocorrelaciones, tangentes hiperbólicas, etc.).

2.3. Restricciones

Es evidente que la velocidad de ejecución de una simulación en C es mucho más rápida que en Java pero los resultados han sido muy satisfactorios. Dadas las dificultades para debugar los programas, se optó por utilizar un entorno "privado", y el código se optimizó para el navegador Explorer, versión 4 o superiores.

3. ALGORITMOS DE LONGITUD FIJA

Estos algoritmos corresponden a las simulaciones con el título *Ordinary Least Square* y *Comparison of a few algorithms*. Con estos algoritmos se demuestra [2] que con MLP con una capa oculta y limitando el número de neuronas, los resultados obtenidos son mucho mejores en comparación a MLP con más de

una capa oculta. La figura 1 muestra ambas simulaciones ya que el diseño y la estructura es prácticamente la misma.

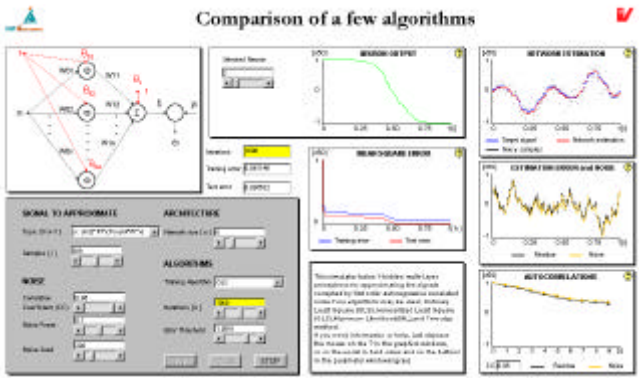

Figura 1. Comparison of a few algorithms

3.1. Ordinary Least Square

Esta simulación fue la referencia para todas las demás y con ella se comprueba que hay dos cosas muy importantes en la aproximación de funciones: (1) que no por utilizar muchas neuronas el resultado es mejor, ya que la aportación de muchas neuronas puede ser nula. La aportación de cada neurona es visible por la ventana gráfica *Neuron Output*. (2) Por contra, es muy importante el criterio de optimización. El usuario puede escoger entre la aproximación por *Gradiente Simple* o por *Gradiente Conjugado*. El Gradiente simple es mucho más sencillo de implementar pero menos efectivo que el conjugado. Con el primero es fácil oscilar alrededor de un mínimo absoluto y muy fácil quedarse en un mínimo relativo, según la tasa de aprendizaje (parámetro constante μ) [5].

3.2. Comparison of a few algorithms

La finalidad de esta simulación es mostrar que dentro de las MLP con una sola capa oculta hay diferentes métodos y los resultados también mejoran según el método utilizado. La estructura es casi la misma que la anterior. La principal diferencia es que las muestras de entrada estan ahora corrumpidas por un ruido coloreado, lo que provoca que los métodos clásicos (OLS) de minimización de error no consigan una buena aproximación. Por esto se proponen diferentes algoritmos GLS. En la misma simulación se pueden comparar los resultados obtenidos con los diferentes algoritmos. En la figura 2 se muestran dos ejemplos de funciones y ruido a estimar, con direrentes correlaciones entre las muestras de ruido.

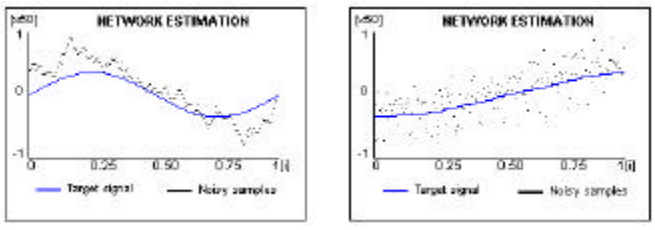

Figura 2. Factor de correlación

4. ALGORITMOS CONSTRUCTIVOS

Otro tipo de MLP con una sola capa oculta son los algoritmos constructivos [2], simulados con el título de *Constructive Networks*. La arquitectura es diferente a los anteriores, pues disponemos de redes accesorias de neuronas. Cada red accesoria tiene limitado el número de neuronas, a la vez, que el número máximo de redes accesorias está también limitado. Este algoritmo es inteligente, a partir de los parámetros de entrada es capaz de decidir si hace falta añadir una neurona o añadir una red accesoria, y además no siempre el número de neuronas será el máximo en cada red accesoria.

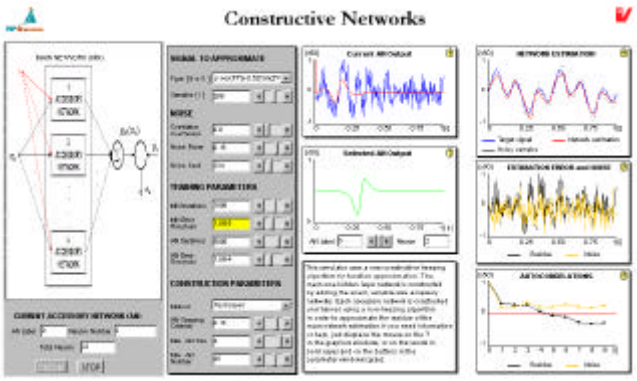

Figura 3. Constructive Networks

5. CONCLUSIONES

El objetivo de esta WEB es doble. Por un lado pretende ser una herramienta didáctica para la aproximación de funciones mediante redes neuronales. Por ello se puede escoger la función a aproximar, la potencia de ruido, el algoritmo y sus parámetros, y gráficamente se puede ver la evolución de cada neurona, el resultado final, la evolución del error, etc. Por otro lado se presentan nuevos algoritmos desarrollados en [2] para la aproximación de funciones, con lo cuál esta WEB permite comparar algoritmos clásicos y algoritmos constructivos, siendo, a nuestro conocimiento, la única WEB que permite esta posibilidad.

6. REFERENCIAS

[1] J. Comabella, “Aproximation de fonctions par reseaux neuronaux en Java”. TFC. Universidad de Vic. Grenoble, 2001.

[2] S. Hosseini, “Contribution à la régression non linéaire par les réseaux de neurones”. Thèse Doctoral. INPG, Grenoble, 2000.

[3] www.lis.inpg.fr/demos/neur_net/NNdemo/mlpacceuil.htm.

[4] J.Hérault et C.Jutten, “Réseaux Neuronaux et Traitament du Signal”. París, 1994.

[5] Martin T.Hagan, Howard B.Demuth et Mak Beale. *Neural Network Design*. PWS Publishing Company, 1996.