Software para el Análisis del Método de Entrenamiento de Backpropagation de una Red Neuronal

Software for the Analysis of Neural Network Training with Backpropagation and an Evolutionary Algorithm

Jorge Cervantes Ojeda Departamento de Matemáticas Aplicadas y Sistemas. Universidad AutónomaMetropolitana - Cuajimalpa.
Avenida Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos, C.P. 05348, México D.F.
Email: jcervantes@correo.cua.uam.mx
Jonathan Morales Pérez.
Ingeniería en Computación. Universidad Autónoma Metropolitana – Cuajimalpa.
Dirección: Avenida Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos, C.P. 05348, México D.F.
Email: ing.01jonas@gmail.com

PALABRAS CLAVE:

Software Educativo; Redes Neuronales Artificiales; Métodos de Entrenamiento.

RESUMEN

Presentamos un programa para el análisis del comportamiento del método Backpropagation de entrenamiento de redes neuronales. El programa permite ejecutar diferentes casos y guardarlos en disco. En él es posible ver en pantalla la evolución del método a través de las iteraciones y ajustar en vivo algunos de los parámetros. Este programa está disponible gratuitamente sin restricción.

KEYWORDS:

Educational Software; Artificial Neural Networks; Training Methods.

ABSTRACT

We present a program for the behavior analysis of the Backpropagation training method of neural networks. The program allows running different test cases and save them to disk. It is possible to see on the screen the evolution of the method through iterations live and adjust some parameters. This program is freely available without restriction.

Recibido: 27 de julio del 2015 • Aceptado: 3 de Enero del 2016 • Publicado en línea: 29 octubre 2016

1 INTRODUCCIÓN

Entre los métodos de entrenamiento de redes neuronales sin retroalimentación (redes feedforward) está el de propagación hacia atrás del error de salida (Error Backpropagation). Este método tiene algunos inconvenientes como la posibilidad de que el aprendizaje sea lento o que llegue a óptimos locales en función de las condiciones iniciales en cada corrida. Cuando se desea explicar esto a alumnos de nivel licenciatura es fácil que éstos lo acepten pero realmente difícil que comprendan el porqué de este comportamiento. Comprender a profundidad lo que realmente está pasando con el método es sumamente importante para motivar el estudio de otros métodos de aprendizaje o el desarrollo de técnicas complementarias que ayuden a mejorar el desempeño del método (por ejemplo: [1,2,3]).

Para explicar tales sutilezas del método es necesario llevar a cabo múltiples experimentos y analizar cada uno en detalle. Esto puede fácilmente llevar a los alumnos al cansancio y/o aburrimiento, lo que reduciría su rendimiento. Es importante que los alumnos estén siempre motivados a la exploración de alternativas conforme se presentan casos tanto exitosos como no exitosos.

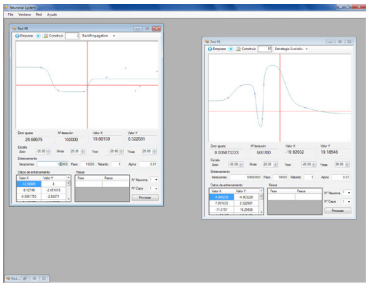
Algunos programas disponibles gratuitamente tienen la forma de bibliotecas de funciones e implementan el algoritmo de manera que se le pueda aplicar a algún problema (por ejemplo: [4]). No hay, hasta donde pudimos averiguar, ningún programa diseñado para explicar los porqués del estancamiento en óptimos locales del método. Por esto, hemos desarrollado una herramienta software con fácil interacción gráfica en la que el método es "visible" en cuanto a su comportamiento respecto al tipo de problema a resolverse y a los parámetros utilizados.

2 DESCRIPCIÓN DEL SOFTWARE

Con la versión actual del software es posible resolver problemas de regresión de funciones escalares de una variable escalar no lineal. A continuación describimos la interfaz de usuario, la variante del método de Backpropagation que se usa y algunas otras características que hacen de este software una herramienta diferente y poderosa para realizar las tareas para las que fue diseñado. El programa para Windows se encuentra disponible en el siguiente enlace: https://dl.dropboxusercontent.com/u/33761126/Neuronal System Version 0.20.exe

2.1 INTERFAZ DE USUARIO

Una vista de la interfaz de usuario del programa se muestra en la Figura 1. Se muestran dos ventanas abiertas. En cada una puede verse que en la parte superior tiene la parte gráfica en donde se muestran los datos de entrenamiento y la función generada por la red neuronal en cada iteración del método. Los datos de entrenamiento pueden introducirse fácilmente aquí mediante clicks del mouse y son señalados con una x. En la parte inferior se encuentran varios campos de captura



y controles del programa. La explicación de cada control está en el Apéndice.
Figura 1: Vista de la interfaz de usuario.

2.2 MÉTODO DE ENTRENAMIENTO

El método de entrenamiento utilizado es una variante del método de Backpropagation llamada por lotes o Batch Backpropagation. Backpropagation es un método de minimización numérica derivado del método del gradiente normalizado y aplicado a redes neuronales multicapa. La función por minimizar es la función de error cuadrático que, en el caso de una red neuronal feedforward con una sola entrada externa, una capa intermedia con n neuronas y una capa de salida con 1

$$E(\mathbf{W}(t)) = \frac{1}{2}\sum_{s}\left(y_s - f(x_s)\right)^2$$

con

$$f(x) = \sum_{i} w_{i1}^2 \tanh\left(\sum_{j} w_{ij}^1 x_j\right)$$

$$\mathbf{W} = (w_{11}^1, w_{12}^1, w_{21}^1, w_{22}^1, w_{31}^1, w_{32}^1, \ldots, w_{n1}^1, w_{n2}^1, w_{11}^2, w_{21}^2, w_{31}^2, \ldots, w_{n1}^2)$$

neurona, es:

donde W es el vector de pesos de la red neuronal con componentes wij^k (neurona i , entrada j , capa k), ys es la salida esperda para la muestra xs, f es la función de

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \alpha \frac{\nabla E(\mathbf{W}(t))}{|\nabla E(\mathbf{W}(t))|}$$

salida de las neuronas de la capa 1 (intermedia). La regla de actualización de pesos resulta:

En esta variante del método la actualización de los pesos se hace una vez por iteración tomando en cuenta todos los datos de entrenamiento a la vez.

2.3 CARACTERÍSTICAS ADICIONALES

2.3.1 HILOS

En este programa corren dos hilos: uno para la interfaz de usuario y otro para el entrenamiento de la red. Esto permite ir agregando datos de entrenamiento al mismo tiempo que el entrenamiento está corriendo. Esto es muy útil cuando se quiere explicar el entrenamiento en línea de un agente que va adquiriendo datos conforme va haciendo experimentos. Es interesante analizar la dependencia del método del orden en el que los datos son adquiridos para estos casos.

2.3.2 GUARDAR Y CARGAR DATOS DE ENTRENAMIENTO Y PESOS ACTUALES

Es posible guardar y cargar tanto los datos de entrenamiento como los pesos actuales de la red. Esto es muy importante para que un profesor pueda crear casos ilustrativos específicos para analizar en clase. También es importante para hacer comparaciones usando diferentes datos con propiedades típicas y no típicas.

3 EJEMPLOS DE USO

3.1 ESTANCAMIENTO POR FALTA DE NEURONAS EN CAPA INTERMEDIA

Cada neurona de la capa intermedia de una red feedforward produce, en el caso estudiado, una salida sigmoidea en función de las entradas recibidas y sus correspondientes pesos. Esta sigmoide es acumulada junto con las sigmoides de las demás neuronas de la capa intermedia por la neurona de salida. Cada neurona intermedia produce una sigmoide con diferente pendiente máxima y diferente traslación horizontal. Esto hace que al sumarse en la capa de salida se produzca una salida con varias sigmoides superpuestas en donde cada una de estas sigmoides proporciona una característica de dicha salida.

Para poder modelar los datos de entrenamiento dados con un error bajo, es necesario que el número de neuronas en la capa intermedia sea suficiente. Entonces, es posible que el número de neuronas sea demasiado bajo para poder reducir el error a niveles bajos. Este número debe ser suficientemente grande como para poder captar los diferentes intervalos de la variable independiente en los que la función tiene diferentes valores de pendiente. Véase por ejemplo el caso de la figura 1 en donde en el extremo inferior del eje x muestra un mal ajuste de la red neuronal con los datos de entrenamiento. En este caso ya no es posible mejorar el resultado con más iteraciones ya que las 5 neuronas de la capa intermedia están ya ocupadas modelando algún intervalo de la función.

3.2 EXCESO DE NEURONAS EN LA CAPA INTERMEDIA

También es posible que la red neuronal haga un mal ajuste si el problema es relativamente sencillo pero se usan demasiadas neuronas en la capa intermedia. Puede presentarse el caso de que las neuronas en exceso representen cualidades inexistentes en la función que se está ajustando. Véase el ejemplo de la figura 2 (izquierda) en donde el método llegó a un punto en el que el error ya es sumamente bajo porque la curva pasa casi por encima de todos los puntos de entrenamiento, sin embargo la curva real no es tan complicada sino que los datos de entrenamiento tienen un cierto grado de error que no debería modelarse. Véase en la figura 2 (derecha) cómo el mismo caso puede resolverse mejor con tan sólo 2 neuronas en la capa intermedia. Aunque la curva no pasa exactamente por los puntos de entrenamiento, la forma que tiene es mucho más razonable. La experimentación con este tipo de casos resulta muy interesante para después comprender mejor las técnicas más avanzadas para mejorar el rendimiento del método de Backpropagation.

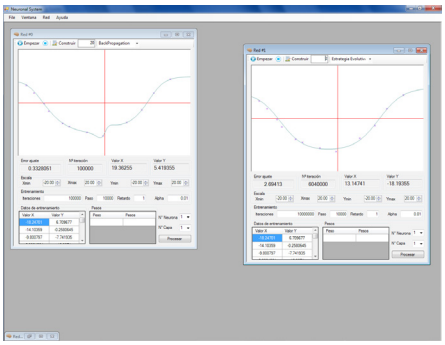


Figura 2: Ejemplo de mal ajuste por número excesivo de neuronas en la capa intermedia (izquierda) y de su solución al reducir este número (derecha).

4 CONCLUSIONES

El software presentado en este artículo no tiene similar al menos que esté disponible en línea de forma gratuita y abierta. Esta herramienta tiene propiedades que pueden motivar resultados interesantes en el área del entrenamiento de redes neuronales. También puede usarse fácilmente para explicar de una forma muy clara algunos fenómenos negativos que se dan cuando se entrena una red con el método de Backpropagation en su variante por lotes.

En la siguiente etapa de desarrollo incorporaremos diferentes técnicas de mejoramiento de la eficiencia del método para que puedan ser comparadas. Luego se implementarán otras técnicas y métodos alternativos al Backpropagation.

6 APÉNDICE

Controles de la interfaz de usuario:

•File -> Nuevo: Construye una nueva ventana.

•Ventana -> N Red #N: Selecciona la ventana activa.

•Red -> Nuevo: Borra todos los datos de entrenamiento de la ventana activa.

•Red -> Pesos - > Guardar pesos ... : Guarda los pesos actuales de la red neuronal.

•Red -> Pesos - > Cargar pesos ... : Carga la configuración y los pesos guardados a la red neuronal actual.

•Red -> Puntos - > Guardar puntos ... : Guarda los puntos de entrenamiento actuales.

•Red -> Puntos - > Cargar puntos ... : Carga los puntos de entrenamiento desde un archivo.

•Empezar: Inicia o reanuda el proceso de entrenamiento.

•Parar: Detiene el proceso de entrenamiento.

•Construir: Construye o reconstruye la red neuronal con la estructura dada por el parámetro anterior y con pesos en las conexiones aleatorios.

•N° Neuronas: Captura el número de neuronas en la capa oculta.

•Error ajuste: Muestra el error actual en el ajuste de la función.

•N° Iteración: Muestra el número de iteración de la última actualización del gráfico.

•Valor X, Valor Y: Despliega el valor X y Y al que apunta el mouse cuando está sobre el gráfico.

•Xmin Xmax: Captura el valor mínimo y máximo del rango en el eje X del gráfico que será desplegado.

•Ymin Ymax: Captura el valor mínimo y máximo del rango en el eje Y del gráfico que será desplegado.

•Iteraciones: Captura el número de iteraciones máximo del entrenamiento.

•Paso: Captura el número de iteraciones del método antes de actualizar el gráfico.

•Retardo: Captura el tiempo de retraso entre una actualización del gráfico y otra.

•Alpha: Captura la tasa de aprendizaje.

•Datos de entrenamiento: Despliega la lista de datos de entrenamiento que han sido capturados en el gráfico.

•N° Neurona: Captura el número de neurona a consultar

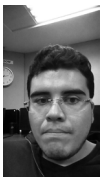
•N° Capa: Captura el número de capa a consultar.

•Procesar: Despliega los pesos de las conexiones de entrada de la neurona indicada en el campo N° Neurona y N° Capa.

REFERENCIAS

1. Wang, X. G., Tang, Z., Tamura, H., Ishii, M., Sun, W. D., 2004. An improved backpropagation algorithm to avoid the local minima problem. Neurocomputing. 2004, 56, 455-460.
2. Wang, X. G., Tang, Z., Tamura, H., Ishii, M., 2004. A modified error function for the backpropagation algorithm. Neurocomputing. 2004, 57, 477-484.
3. Magoulasa, G. D., Vrahatisb, M. N., Androulakisb, G. S., 1997. Effective Backpropagation Training with Variable Stepsize. Neural Networks. 1997, 10(1), 69-82.
4. Kirillov, Andrew. 2006 Neural Networks on C#. Neuro-demo. Recuperado el 27 de Noviembre de 2015 de http://www.codeproject.com/Articles/16447/Neural-Networks-on-C. 2006.

Acerca de los autores:



Jonathan Morales Pérez.

Adscripción: Ingeniería en Computación. Universidad Autónoma Metropolitana – Cuajimalpa.

Dirección: Avenida Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos, C.P. 05348, México D.F.

Email: ing.01jonas@gmail.com

Semblanza: Es alumno de Ingeniería en Computación en la UAM Cuajimalpa desde septiembre del 2011. Ha trabajado en proyectos SAP durante 1 año como programador ABAP en la empresa Enable, dando soporte a empresas como SAVI y DIAVAZ. Trabajó en el desarrollo de un software Web para la asignación de horarios de profesores. Actualmente se encuentra desarrollando un libro para el aprendizaje de Android. Sus intereses son la optimización, el álgebra geométrica y el desarrollo de aplicaciones móviles.

Jorge Cervantes Ojeda

Adscripción: Departamento de Matemáticas Aplicadas y Sistemas. Universidad Autónoma Metropolitana - Cuajimalpa.

Dirección: Avenida Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos, C.P. 05348, México D.F.

Email: jcervantes@correo.cua.uam.mx

Semblanza: Es Doctor y Maestro en Ciencias (Computación) por la Universidad Nacional Autónoma de México. Es también Ingeniero en Electrónica (área Sistemas Digitales y Computadoras) por Universidad Autónoma Metropolitana. Es Profesor Asociado de Tiempo Completo en la UAM Cuajimalpa desde 2007. Trabajó como ingeniero de software en la empresa Alcatel-Indetel desarrollando software para centrales telefónicas digitales. Colaboró con Alcatel Bell, en la ciudad de Amberes, Bélgica, en el departamento de Diseño y Desarrollo de Software. Su experiencia como docente abarca las áreas de desarrollo de software, programación y matemáticas. Fue profesor en la Facultad de Estudios Superiores Cuautitlán de la Universidad Nacional Autónoma de México durante 7 años y actualmente es docente en la UAM Cuajimalpa en cursos relacionados con la ingeniería de software. Sus intereses en investigación son la inteligencia artificial y la ingeniería de software.