HERRAMIENTA SOFTWARE CON INTERFAZ WEB PARA LA INTERPRETACIÓN SIMBÓLICA DE MODELOS NEURONALES

Denis Rincón
Escuela de Computación. Facultad de Ciencias.
Universidad Central de Venezuela. Caracas, Venezuela

Ely Rozo
Escuela de Computación. Facultad de Ciencias.
Universidad Central de Venezuela. Caracas, Venezuela

Haydemar Núñez
Laboratorio de Inteligencia Artificial
Escuela de Computación. Facultad de Ciencias.
Universidad Central de Venezuela. Caracas, Venezuela
hnunez@strix.ciens.ucv.ve

Resumen.
En este trabajo se presenta el Sistema Interpretador de Modelos Neuronales WebTREPAN. Esta herramienta Web permite extraer, de manera automática, una representación simbólica aproximada en forma de un árbol de decisión, del conocimiento que ha capturado una red neuronal durante su aprendizaje. La herramienta está basada en el algoritmo de extracción de reglas TREPAN y mediante una interfaz guiada tipo "Wizard", el usuario introduce los datos relacionados con la red neuronal que desea interpretar (tales como la topología de la red neuronal). De esta forma, WebTREPAN puede ser utilizado para facilitar la comprensión y el análisis del proceso que está bajo investigación, servir de apoyo para la toma de decisiones por parte de expertos y complementar otros sistemas utilizados en la minería de datos.
Palabras claves: Inteligencia Artificial, Redes Neuronales, Extracción de Reglas, Minería de Datos

Abstract
In this work WebTREPAN a Neural Networks Interpreter Systems is presented. This Web tool allows extract a approximate symbolic representation (in form of a decision tree) from trained neural networks automatically. WebTREPAN is based on TREPAN, a global rule extraction algorithm. It have a wizard interface, where the user can introduce the neural network data (as such the topology). In this way, it is possible to give an interpretation to the knowledge acquired by a neural networks during its learning. Therefore, WebTREPAN can be used for data exploration and for data mining applications.
Keywords: Artificial Intelligence, Neural Networks, Rule Extraction, Data Mining

1. INTRODUCCIÓN

Las redes neuronales se han utilizado ampliamente en la resolución de problemas complejos en diferentes dominios, donde han mostrado buenas capacidades de generalización [4],[10],[13]. Sin embargo, una posible limitación de estas máquinas de aprendizaje es que generan modelos del tipo *caja negra*; es decir, la solución suministrada por ellas es difícil de interpretar desde el punto de vista del usuario [24]. Convertir en interpretable estos modelos, ha sido el objetivo de diferentes métodos desarrollados en los últimos años que permiten extraer reglas a partir de redes neuronales entrenadas [1],[8],[21],[22],[23].

Además de dotar a las redes neuronales de la *capacidad de explicación*, estos *métodos de extracción de reglas* posibilitan que los modelos obtenidos sean utilizados para la adquisición automática de conocimiento en sistemas expertos [1],[17]. También, la extracción de reglas mejora la adecuación de las redes neuronales para resolver problemas de minería de datos ("data mining") [2],[8],[16], cuando el objetivo principal es descubrir relaciones desconocidas e implícitas en grandes bases de datos que, en muchos casos, es necesario expresarlas en un formato comprensible.

Uno de los métodos de extracción de reglas actuales es el algoritmo TREPAN desarrollado por Craven [7], [9]. Este algoritmo realiza la extracción desde redes neuronales entrenadas y obtiene representaciones simbólicas en forma de árboles de decisión. Una característica resaltante de TREPAN es que al ser un método de extracción global, puede aplicarse a diferentes tipos de redes neuronales y en general, a cualquier paradigma de aprendizaje supervisado en entornos de clasificación.

En este trabajo se presenta el Sistema Interpretador de Modelos Neuronales WebTREPAN, una herramienta automatizada basada en Web que incorpora como núcleo el método de extracción de reglas TREPAN. Este sistema puede ser utilizado para trasladar de manera automática un alto porcentaje del conocimiento sintetizado en una red neuronal entrenada del tal forma que ese conocimiento, expresado en la nueva representación en forma de un árbol de decisión, pueda ser utilizado para facilitar la comprensión y el análisis del problema que está bajo investigación.

Este artículo está organizado de la siguiente forma: En la sección 2 se presentan brevemente los fundamentos de los métodos de extracción de reglas a partir de redes neuronales entrenadas. En la Sección 3 se describe el método TREPAN para la extracción de reglas. A continuación, en la Sección 4, se presenta el sistema WebTREPAN, donde se describen los módulos de interfaz y de interpretación. En la Sección 5 se muestran las pruebas realizadas para verificar la funcionalidad de WebTREPAN. Por último, se presentan las conclusiones y se comentan los trabajos futuros.

2. REDES NEURONALES Y LA EXTRACCIÓN DE REGLAS

Las redes neuronales artificiales [4],[10],[13] representan, más que una sola técnica de aprendizaje, una familia de modelos que han sido utilizados con éxito en una amplia variedad de problemas de clasificación y de regresión, donde han exhibido las siguientes prestaciones: un alto rendimiento de generalización ante ejemplos no vistos durante el aprendizaje, robustez de la solución ante la presencia de imprecisión, ruido o incertidumbre en los datos de entrada, y habilidad para aprender relaciones no lineales, la cual es una característica importante para manipular bases de datos reales. Sin embargo, en los modelos neuronales el conocimiento derivado del proceso de aprendizaje se encuentra sintetizado en una forma sub-simbólica (matrices de pesos) que hace difícil su interpretación y por lo tanto, son mecanismos de resolución de problemas tratados como una *caja negra*.

Para superar la limitación de las redes neuronales relacionada con su *falta de transparencia*, la hipótesis generada por estos modelos podría trasladarse a una forma interpretable; los métodos que realizan esta transformación se conocen como *algoritmos de extracción de reglas* [1],[8],[21],[22],[23]. Y, aunque el requerimiento de una solución comprensible dependerá de las necesidades de la aplicación y del usuario final del sistema, las ventajas que se obtienen con esta transformación son:

- *Provisión de la capacidad de explicación*, lo cual redundará en un mejor entendimiento del problema bajo estudio y de su solución. Esto es especialmente importante cuando la solución final será utilizada para procesos de toma de decisiones.
- *Exploración de datos*, al *revelar* las relaciones entre las variables del sistema que hayan sido descubiertas durante el aprendizaje y que se encuentran codificadas en un formato incomprensible [3].

La extracción de reglas también posibilita que las redes neuronales puedan utilizarse como una herramienta para la adquisición automática de conocimiento en sistemas basados en reglas [1],[17]. El mayor inconveniente a la hora de diseñar un sistema experto está relacionado con la construcción y depuración de su base de conocimiento. Las redes neuronales, junto con la extracción de reglas, podrían ayudar en este sentido al realizar un aprendizaje a partir de ejemplos representativos del problema y luego, las reglas extraídas pasarían a formar parte de la base de conocimiento.

También estos métodos de extracción de reglas mejoran la adecuación de las redes neuronales para aplicaciones de minería de datos ("data mining") [1],[8],[16] cuando en muchos casos se requiere que las relaciones que hayan sido descubiertas en grandes bases de datos estén expresadas en un lenguaje comprensible.

2.1 Características de los Métodos de Extracción de Reglas

El conocimiento capturado por una red neuronal durante su aprendizaje está representado en términos de la topología de la red, las funciones de activación utilizadas por las neuronas, y los parámetros asociados con las conexiones (pesos) y las unidades (umbrales) [1],[8]. La técnica de extracción debe entonces interpretar estos tres elementos y, a partir de ellos, generar una descripción más comprensible que sea funcionalmente equivalente a la red neuronal (es decir, que suministre las mismas respuestas de predicción).
Los diversos métodos que han sido desarrollados para extraer y representar el conocimiento integrado en una red neuronal, pueden distinguirse de acuerdo a las siguientes características:

- *Lenguaje de representación* utilizado por el método para describir el modelo aprendido por la red neuronal. Los lenguajes que han sido utilizados son: reglas del tipo *si-entonces*, reglas *m-de-n*, árboles de decisión, autómatas de estado finito, reglas oblicuas (funciones lineales), reglas difusas ("fuzzy), entre otras.
- *La estrategia de extracción*, la cual define la manera en que el método utiliza a la red para realizar la extracción. En este sentido se distinguen métodos *locale*s, que analizan la estructura de la red a nivel de las unidades ocultas y de salida para extraer las reglas, y métodos *globales*, que extraen reglas sobre la base de la función entrada-salida implantada por la red, sin analizar su estructura interna. También se han propuesto técnicas *híbridas* que se encuentran entre estos dos extremos.
- *Requerimientos sobre la red*. Algunos algoritmos sólo son aplicables a ciertas arquitecturas de redes y/o pueden requerir de características específicas tales como ciertos tipos de neuronas o patrones de interconexión. Además, pueden necesitar de regímenes especiales de entrenamiento con el fin de facilitar el proceso de extracción. Otros métodos son independientes de la arquitectura o pueden ser aplicados a diferentes redes. De alguna manera, esta característica define la *portabilidad* del algoritmo de extracción.

Para finalizar, el rendimiento y calidad del conjunto de reglas extraído puede valorarse utilizando las siguientes medidas [1],[16]: la *consistencia* o *equivalencia*, que representa la capacidad de las reglas para imitar el comportamiento de la red a partir de la cual fueron generadas; la *exactitud*, la cual es la habilidad del conjunto de reglas para clasificar correctamente los datos; y la *complejidad*, medida en términos del número de reglas extraídas y del número de antecedentes por regla.

3. EL ALGORITMO DE EXTRACCIÓN DE REGLAS TREPAN

El algoritmo TREPAN es un método global de extracción de reglas desarrollado por Craven [7],[9], que permite obtener una representación simbólica aproximada en forma de un árbol de decisión del concepto representado por una red neuronal entrenada. TREPAN utiliza un *Oráculo*, un sistema a quien dirigir preguntas y que sea capaz de clasificarlas de acuerdo al conocimiento que posea (en este caso la red neuronal). De esta forma, TREPAN puede aplicarse a diferentes paradigmas de aprendizaje supervisados.
Existen diversos algoritmos de inducción de árboles de decisión a partir de un conjunto de ejemplos, como el ID3 o C4.5 [15]. Básicamente, TREPAN es similar a ID3 (ver Figura 1) con algunas diferencias que se especifican brevemente a continuación:

<u>El Oráculo</u>

El objetivo del algoritmo TREPAN es extraer y representar el concepto que aprendió la red neuronal, por esta razón utiliza a la red como un Oráculo para clasificar cada dato que se le presenta. Es así que lo primero que hace TREPAN es re-clasificar con el Oráculo a todos los datos que serán utilizados para derivar el árbol de decisión, buscando que la representación obtenida sea lo más fiel posible a ese concepto.

<u>Generación de nuevas instancias</u>

Otra característica de TREPAN es su capacidad para generar nuevas instancias, las cuales son clasificadas por el Oráculo. De esta forma, TREPAN asegura que se mantenga siempre una *cantidad mínima* de datos en el nodo antes de obtener la etiqueta de clase (si es una hoja) o seleccionar un test de partición (si es un nodo interno del árbol).
Para llevar a cabo este proceso, TREPAN construye un modelo de la distribución de los datos que llegan al nodo y luego usa este modelo para generar las nuevas instancias. Estos modelos toman en cuenta el conjunto de restricciones que causan que un dato siga un camino desde el nodo raíz hasta el nodo donde se desea generar las instancias. Estas restricciones pueden ser de dos tipos: no-disyuntivas y expresiones *m-de-n*.

<u>Expansión del Árbol</u>

Para realizar la expansión de un nodo, TREPAN hace uso del criterio *primero el mejor,* en donde primero se realiza la expansión del nodo con mayor potencial de incrementar la fidelidad del árbol que es extraído de la red. La función usada para evaluar un nodo viene dada por la siguiente expresión:

$$Evaluar(Nodo) = Alcanzar(Nodo)*(1 - Fidelidad(Nodo)) \quad (1)$$

ALGORITMO TREPAN (**Entradas** : *datos de entrenamiento **S**, variables **F**, criterios de parada, Parámetro mínimo_datos)*

Cola := { vacía }; /* se mantiene una cola de nodos para posibles expansiones */

PARA cada dato **E** del conjunto **S** /* usar la red para clasificar los datos de entrenamiento.*/
1.- Asignar una etiqueta de clase para el dato **E** := **CONSULTAR_AL_ORÁCULO**(**E**)
FPARA

2.- Inicializar la raíz del árbol **T**, como un nodo hoja
3.- Construir el Modelo de Distribución **M** de los datos que alcanzan el nodo **T**
4.- **query_instances** := **TRAZAR_EJEMPLO** ({ }, mínimo_datos - |**S**|, **M**)
5.- Usar **S** y las **query_instances** para determinar la etiqueta de clase para el nodo **T**
6.- Inicializar la Cola con la Tupla (**T**, **S**, $\textbf{query\_instances}_T$, { })

MIENTRAS (la Cola **no** este vacía) **Y** **(**el Criterio de Parada Global **no** este satisfecho)
/* Se procede a Expandir el mejor nodo en base al criterio de evaluación*/
7.- Sacar de la Cola el nodo **N** con (**N**, $\textbf{S}_N$, $\textbf{query\_instances}_N$, restricciones_n)
8.- **Test**:= **CONSTRUIR_TEST**(**F,** $\textbf{S}_N$ U $\textbf{query\_instances}_N$) /* mejor partición binaria */
9.- Hacer de **N** un nodo interno con el Test obtenido

10.- **PARA** cada salida **t** del **Test** /* hacer los nodos hijos */

11.- Hacer a **C**, un nuevo nodo hijo de **N**
12.- Restricciones**C** := restricciones**N** U { **Test** = t }
13.- **Sc** :=miembros de $\textbf{S}_N$ con salida **t** en el Test
14.- Construir el Modelo de Distribución **M** de los datos cubiertos por el nodo **C**
15.- query_ instancias:= **TRAZAR_EJEMPLO**(restricciones**C**, mínimo_datos-|**Sc**|, **M**)
16.- Usar **Sc** y las **query_ instances**c para deter inar la etiqueta de clase para el nodo **C**

17.- **SI** (el Criterio de Parada Local **no** está satisfecho) **ENTONCES**
18.- Poner {**C**, examples**C**, constraints**C**} dentro de la Cola, para futuras expansiones
FSI

FPARA
FMIENTRAS

Figura 1. Algoritmo TREPAN para la extracción de reglas

donde, *Alcanzar(Nodo)* se refiere a la fracción estimada de datos que llegan al nodo, y *Fidelidad(Nodo)* se refiere a la fracción de instancias para las cuales el árbol y la red coinciden en sus predicciones.
TREPAN mantiene una cola de nodos para futuras expansiones y en base a esta función de evaluación, determina cuál es el mejor nodo que será sacado de la cola para su expansión. Es importante mencionar que por cada nodo en la cola, TREPAN mantiene los datos de entrenamiento que llegaron a él, las nuevas instancias que fueron creadas y las restricciones del mismo.

Selección de los Tests de Partición

TREPAN sólo utiliza tests binarios para realizar la partición del árbol. Estos tests pueden ser de dos tipos: no-disyuntivos y expresiones *m-de-n*. Para construir un test se consideran todos los posibles valores que tomen las variables tanto en los datos de entrenamiento que llegan al nodo como en las instancias creadas y clasificadas por el Oráculo. Por otra parte, TREPAN utiliza una medida de evaluación heurística llamada *criterio de ganancia de información* [15] para seleccionar el mejor test de partición (el que maximice este criterio de evaluación) sobre un conjunto de candidatos.

Criterios de Parada

Otro aspecto clave de los algoritmos de inducción de árboles de decisión, es determinar cuándo se debe detener el crecimiento del árbol. Existen varios criterios para decidir si se hace hoja a un nodo; TREPAN utiliza un criterio de parada local y otro global. El primero toma en cuenta la pureza de los datos que llegan al nodo, haciéndolo una hoja si todos los datos que llegan a él pertenecen a la misma clase. Por otro lado, el criterio global se refiere al tamaño máximo que debe tener el árbol que TREPAN va a retornar. Este parámetro puede ser proporcionado por el usuario y está especificado en términos de la cantidad de nodos internos, lo cual le da un control sobre la comprensibilidad que tendrán los árboles producidos por el algoritmo. Asimismo, TREPAN es capaz de usar el conjunto de datos de test (en caso de no especificarse un límite para el tamaño del árbol), para tomar la decisión sobre el tamaño el árbol que será devuelto. Este conjunto se utiliza para calcular la fidelidad de cada subárbol generado por el algoritmo a medida que el árbol va creciendo.

4. INTERPRETACIÓN AUTOMÁTICA DE REDES NEURONALES ENTRENADAS

El objetivo de este trabajo fue el desarrollo de un sistema automatizado para la interpretación de modelos neuronales basado en el algoritmo de extracción de reglas TREPAN. Esta herramienta, llamada WebTREPAN, permite extraer una representación explícita del conocimiento sintetizado en una red neuronal entrenada de forma automática. De esta forma, WEbTREPAN puede facilitar la labor de análisis del proceso o sistema que está bajo investigación, así como ayudar en la toma de decisiones por parte de una persona experta. También, esta herramienta podría complementar a los sistemas utilizados en la minería de datos. En la Figura 2 se muestra el modelo general del sistema WebTREPAN.

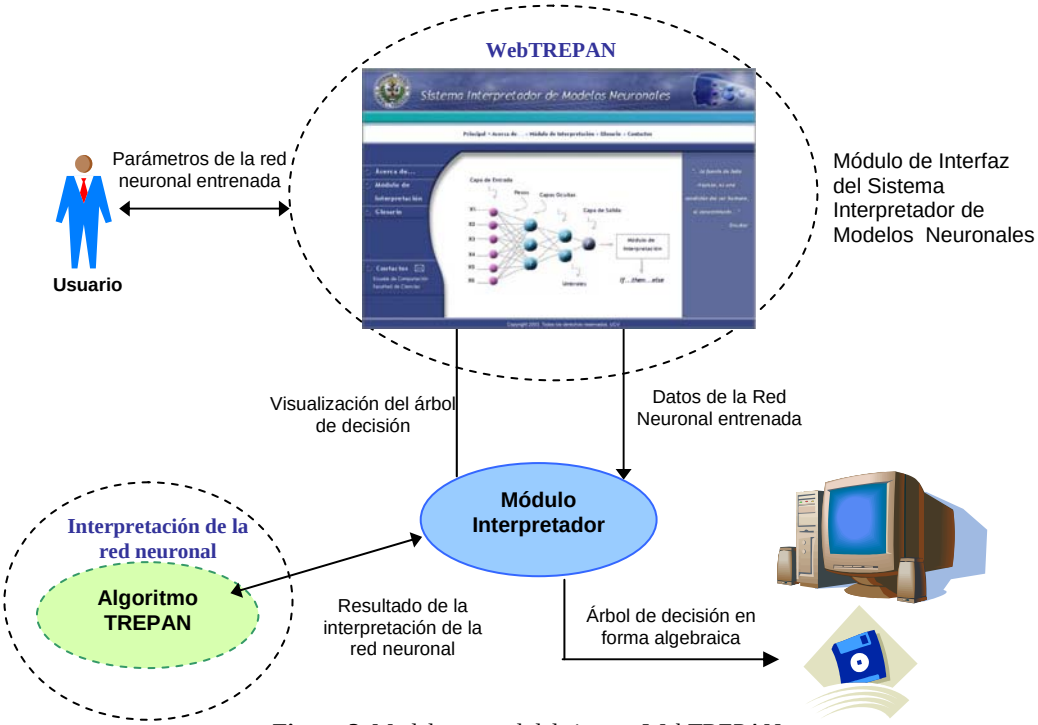

Figura 2. Modelo general del sistema WebTREPAN

La herramienta computacional está basada en Web, y mediante una interfaz humano-computador en forma de "Wizard" se pueden introducir los datos relacionados a la red neuronal entrenada que el usuario desea interpretar (tales como topología de la red, matrices de pesos y umbrales, entre otros). El "Wizard" forma parte de un módulo interpretador dentro del sistema, donde una vez que los datos han sido proporcionados por el usuario, se procede a ejecutar el método de extracción de reglas TREPAN. Posteriormente, se procede a mostrarle los resultados del proceso de interpretación al usuario a través de la interfaz del sistema; estos resultados también podrán ser almacenados en el disco duro de la PC o bien en una unidad de almacenamiento portable.

Además del módulo de interpretación, el sistema también cuenta con otras funcionalidades, como una sección que describe la justificación y los objetivos planteados, una ayuda que guía al usuario en el proceso de interpretación y de las consideraciones que debe tener en cuenta para que éste sea satisfactorio, así como una sección que suministra información del estado del proceso.

4.1 Desarrollo del Módulo de Interfaz

Para el análisis y diseño del sistema se utilizó la metodología OOSE extendida junto con el Lenguaje de Modelado Unificado UML [11],[20]. A partir del modelo de casos de uso y del modelo objeto del dominio se desarrolló un prototipo de la interfaz de usuario, que posteriormente fue refinado hasta lograr obtener la interfaz definitiva de WebTREPAN. En particular, para el diseño y desarrollo de la interfaz del sitio Web, se empleó la herramienta Macromedia Dreamweaver MX, por ser muy fácil y práctica de usar. Como programas para la edición de imágenes, se usaron las herramientas Adobe Photoshop 6.0 y 7.0; así como Macromedia Flash 5.0 y SWISH v.2 ESP en los efectos Flash que se diseñaron. A continuación, se describe el modelo de casos de uso del sistema.

4.1.1 Modelo de Casos de Uso

El diagrama de casos de uso muestra los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relacionan con su entorno (usuarios u otras aplicaciones) [12]. En la Figura 3 se muestran los dos casos de uso principales del Sistema Interpretador de Modelos Neuronales. El actor identificado es el *Usuario del Sistema,* que se refiere al usuario que utilizará el sistema interpretador y que podrá acceder a cualquiera de las funcionalidades que éste ofrece.

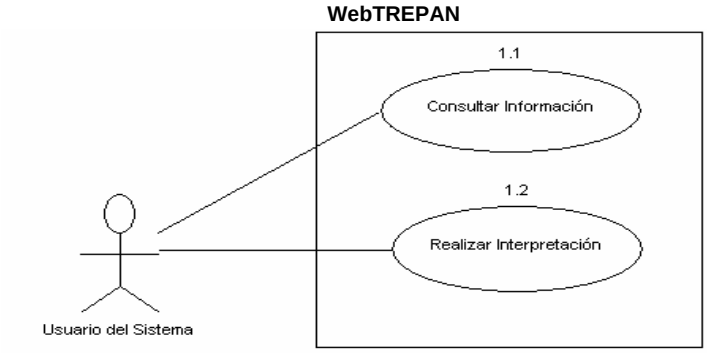

Figura 3. Interacción de los actores con los Casos de Uso del Sistema

La Figura 4 muestra el refinamiento del caso de uso *Consultar Información*. Como puede observarse, el caso de uso *Acerca de…* incluye tres funcionalidades que el usuario puede seleccionar de manera opcional para consultar información referente al sistema, como son: *Justificación*, que le brinda al usuario la posibilidad de consultar información relacionada con la justificación para el desarrollo del sistema interpretador; *Referencias*, el cual le proporciona al usuario la posibilidad de consultar información relacionada con las referencias más importantes que fueron consultadas; y *Objetivos***,** que le brinda al usuario la opción de consultar acerca del objetivo planteado en la investigación.

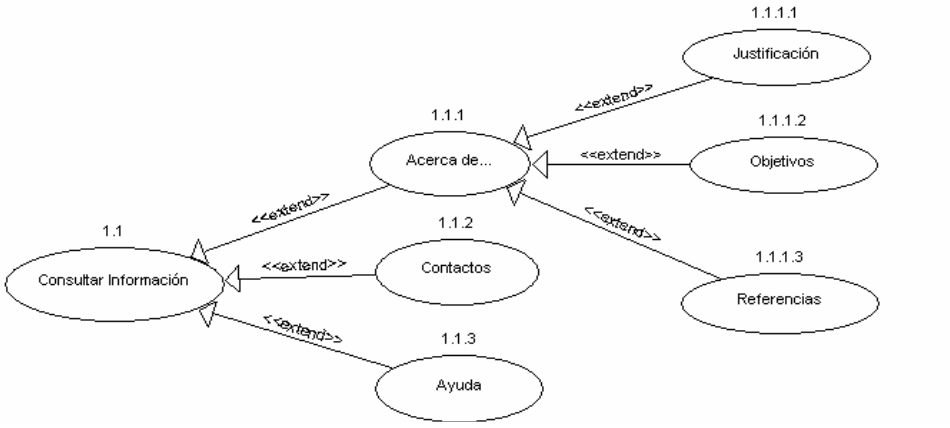

Figura 4. Refinamiento del caso de uso *Consultar información*

El caso de uso *Ayuda* es una funcionalidad donde el usuario podrá encontrar información de cómo utilizar el sistema, guiar al usuario en el proceso de interpretación y ayudarle a resolver las dudas que se le presenten para que éste sea satisfactorio. Por otro lado, el caso de uso *Contactos* le brinda información al usuario, relacionada con las personas involucradas en el desarrollo del sistema interpretador WebTREPAN.
La Figura 5 muestra el caso de uso *Realizar Interpretación,* que le permite al usuario realizar la interpretación de la red neuronal entrenada. Para realizar esta tarea, el usuario debe proporcionar todos los datos necesarios que requiere el algoritmo TREPAN. Este caso de uso incluye seis funcionalidades, cuatro de las cuales se realizan de manera obligatoria y dos que se puede realizar de manera opcional, como son:

- *Proporcionar datos de la topología*: Donde el usuario introduce los datos relacionados con la topología de la red entrenada que quiere interpretar, a saber: el número de entrada, el número de capas ocultas, el número de neuronas de las capas oculta y de salida, el tipo de función de activación por capa oculta y de salida, los pesos y umbrales de cada una de las capas.
- *Proporcionar datos de entrenamiento y prueba*: Donde el usuario selecciona los datos de entrenamiento y de test que utilizó para la creación del modelo de red neuronal.
- *Proporcionar criterio de parada*: Donde el usuario introduce de manera opcional, el número de nodos internos que desea tenga el árbol de decisión asociado a la interpretación de la red (criterio de parada global).
- *Verificar información proporcionada*: El usuario decide si modifica o no, algunos de los parámetros que introdujo como entrada al proceso de interpretación de la red.
- *Generar el árbol de decisión*: Una vez que se han proporcionados todos los datos necesarios, el usuario da inicio al proceso de interpretación de la red. Este caso de uso incluye el caso de uso que ejecuta el método de extracción TREPAN para la interpretación de la red.

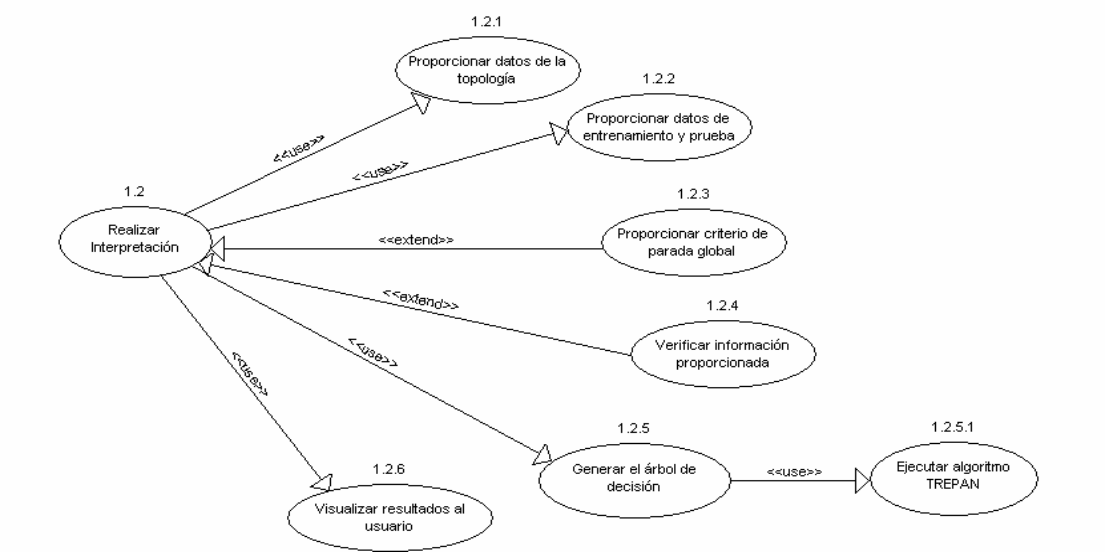

Figura 5. Refinamiento del caso de uso *Realizar interpretación*

- *Visualizar resultados al usuario*: Finalizado el proceso de interpretación, el sistema muestra los resultados obtenidos al usuario.

Para llevar a cabo la interpretación, la interfaz guiada ("Wizard"), le indica al usuario cuáles son los parámetros que éste debe introducir. Si el usuario decide realizar una nueva interpretación debe dirigirse primero a la página de inicio y desde allí acceder al módulo de interpretación, para proporcionar nuevos datos en el "Wizard".
Por otra parte, el sistema ofrece ayuda al usuario en forma de *Notas* relativas a la acción que efectúa dentro del proceso de interpretación, para orientarlo a no cometer errores o para que pueda corregirlos. Éstas se encuentran situadas en cada una de las páginas del "Wizard" en el módulo de interpretación. En caso de necesitar un entrenamiento más específico, el usuario puede consultar la ayuda global del sistema, la cual le facilita el aprendizaje del mismo. A continuación, se presentan algunas páginas que conforman la interfaz de WebTREPAN.
En la Figura 6 se muestra la página principal. En esta se puede observar el área donde se encuentran ubicados los botones que permiten acceder a cada funcionalidad del sistema y el área de presentación, donde se mostrará el contenido del "Wizard" dentro del módulo de interpretación. También hay un área destinadas a las *Notas* con información que debe conocer el usuario para realizar el proceso de interpretación. Estas notas están ubicadas a lo largo del recorrido por el módulo de interpretación y son presentadas en forma de enlaces específicos a la Ayuda del sistema donde el usuario podrá encontrar la información correspondiente.

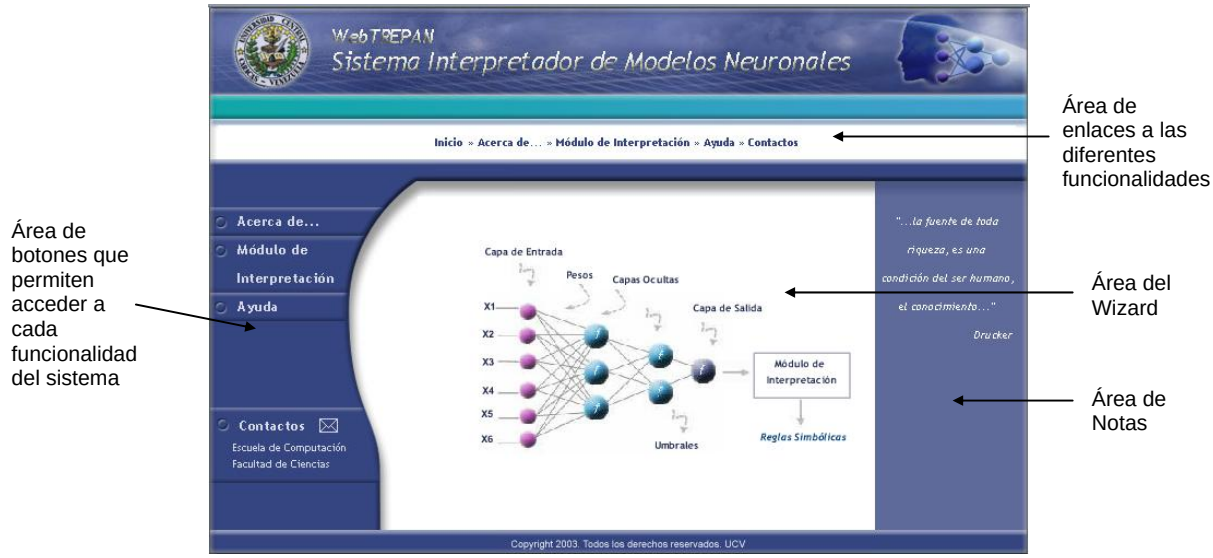

Figura 6. Página principal de WebTREPAN

La Figura 7 muestra la página principal del módulo de interpretación. Se observa el área donde se encuentran ubicadas las *Notas* a tener en cuenta para realizar el proceso de interpretación. A partir de esta página, el usuario comienza su recorrido por el "Wizard" para la interpretación de la red neuronal entrenada, según los datos que éste proporcione. Como ejemplo, en la Figura 8 se presenta la página correspondiente a la información de la capa de entrada.

Figura 7. Página principal del Módulo de Interpretación

En esta página del "Wizard", el usuario especifica el número de neuronas de la capa de entrada, selecciona la ruta del archivo de etiquetas de las variables de entrada (en caso de que haya seleccionado la opción **Si**) y escribe el número de capas ocultas que posee la red. En el área derecha de la página pueden observarse las *Notas* correspondientes, con un enlace hacia la *Ayuda* donde el usuario encontrará información sobre cómo escribir los datos de la capa de entrada y cómo escribir el archivo de nombre de las variables de entrada.

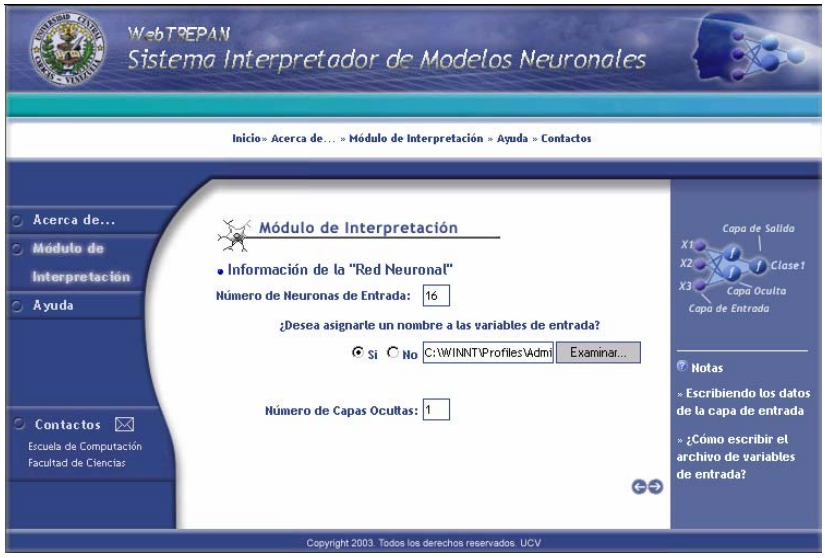

Figura 8. Módulo de Interpretación. Primera página de datos de la red neuronal

4.2 Desarrollo del Módulo de Interpretación

El proceso de desarrollo del módulo de interpretación se dividió en tres partes: la estructuración y formato de los archivos de entrada a la herramienta, la implantación de cada una de las rutinas del algoritmo TREPAN y por último, la visualización del árbol generado.

4.2.1 Estructura y Formato de los Archivos de Entrada

Los datos relacionados a la red neuronal entrenada que serán la entrada al sistema, vienen dados en 5 archivos, donde el usuario debe almacenar la siguiente información: los *datos de entrenamiento*, los *datos de test,* los *pesos y umbrales* de las capas ocultas y de salida, un archivo donde el usuario colocará los *nombres de las variables de*

entrada y otro archivo de clases donde colocará los *nombres de las variables de salida.* Esto último, sólo en el caso de que desee asociarle un nombre a cada variable, por ejemplo: Variable1 = *Temperatura,* Variable2 = *Humedad,* etc. Estos archivos podrán ser elaborados por el usuario, en un editor de texto como *Bloc de Notas* y podrán guardarse con cualquier nombre más la extensión ***.txt***.

4.2.2.Implantación del Algoritmo de Extracción de Reglas TREPAN

La implementación del algoritmo requirió un análisis previo de su estructura para lograr su descomposición en una serie de rutinas específicas. Particularmente, para la construcción del oráculo se diseñó una rutina que simula la red neuronal, lo cual requirió la implantación de las funciones de activación que podrían ser utilizadas en las capas ocultas y de salida de la misma. En particular, fueron implantadas las siguientes ocho funciones: *Escalón, Escalón Bipolar, Gaussiana, Lineal, Lineal Saturada, Lineal Saturada Bipolar, Sigmoide y Tangente Hiperbólica* [10].

Como se describió anteriormente, en la generación de nuevas instancias se debe tomar en cuenta que éstas deben seguir la distribución de los datos en el dominio del problema. Para calcular el modelo de distribución local en un nodo, se implantaron las rutinas para modelar individualmente cada variable de entrada según sea su tipo, discreta o continua.

Para una variable discreta, la distribución viene dada por los valores que puede tomar la variable, y por cada posible valor se indica la frecuencia relativa con que éste ocurre en el conjunto de datos de entrenamiento que llegan al nodo [9]. Para la generación de los números aleatorios que siguieran la distribución discreta, se aplicó el método de la transformada inversa [19] para valores discretos, donde se especifica que si la variable aleatoria **X** tiene una FDA F(x), entonces la variable u = F(x) está distribuida uniformemente entre 0 y 1. Por lo tanto, **X** se puede obtener generando números uniformes y calculando $x = F^{-1}(u)$.

Para una variable continua, se modeló la función de densidad a través del método de estimación de *densidad del núcleo ("kernel") gaussiano* [6],[14], cuya función viene dada por la expresión:

$$f(x) = \frac{1}{m}\sum_{j}^{m}\frac{1}{\sqrt{2\pi\theta}}e^{-\left(\frac{x-u_j}{2\theta}\right)^2} \qquad (2)$$

Donde m representa el número de ejemplos de entrenamiento usados en la estimación, μ_j representa el valor de la variable para el j-ésimo ejemplo y θ representa la amplitud del *kernel gaussiano*, que TREPAN fija al valor de $1/\sqrt{m}$. Para la generación de números aleatorios que siguen una distribución normal, fueron aplicados el método polar, que permite generar variables aleatorias normales estandarizadas e independientes [19]; y el método de la transformada inversa que permite transformar al número aleatorio normal generado con media μ=0 y varianza θ=1, y llevarlo al espacio original con $\mu=\mu_j$ y $\theta=1/\sqrt{m}$ [18].

Otras rutinas implantadas tienen que ver con la expansión del árbol a través de una *Cola de Prioridades*, la construcción y selección de los *test de partición* y los *criterios de parada*, entre otras.

4.2.3. Resultados de la Aplicación

La visualización del árbol generado se realizó utilizando una representación algebraica en forma de tripleta, donde para los nodos internos el primer elemento representa el nodo padre, el segundo elemento el hijo izquierdo y el tercer elemento el hijo derecho. En el sistema, se indican los nodos con letras del abecedario y se muestra una leyenda que especifica para cada letra (nodo) el test de partición asociado (si es un nodo interno) o la etiqueta de clase (para un nodo hoja); de esta forma, el usuario puede inferir el árbol obtenido. En la Figura 9 se muestra la página de resultados del Módulo Interpretador donde se observa un ejemplo de esta representación. Esta página también incluye un enlace a la *Ayuda* para asistir al usuario en la traslación de una forma a otra.

4.3 Aspectos Técnicos de WebTREPAN

El sistema se desarrolló para plataformas PC, bajo Sistema Operativo Windows 9x/2000/ME/XP/NT y en ambiente Web, específicamente para el cliente. Los requerimientos mínimos para el uso de la aplicación son los siguientes: Internet Explorer 5.0 o superior, Netscape 7.0 con soporte para la Máquina Virtual de Java (J.V.M), PC Pentium III con 256 MB de memoria RAM, Java 2 SDK, Standard Edition Versión 1.4.1, y Servidor Web Resin 2.0 o superior con soporte para Servlets.

Por otra parte, para la implantación del sistema se utilizó la tecnología JAVA; en particular, se utilizó la herramienta JCreator Pro Release V2.50 build 9 para la programación, Java Server Pages (JSP) para generar contenido Web de una manera dinámica, y JavaScript como lenguaje para la creación de los scripts. Para el manejo de los archivos de texto, como la lectura desde el cliente y la transferencia hacia el servidor para su procesamiento, se utilizó un Java Bean. También se usaron Servlets que son componentes Web, definidos como clases de Java compiladas que pueden ser cargadas de manera dinámica y ejecutadas en un Servidor Web que soporte Servlets.

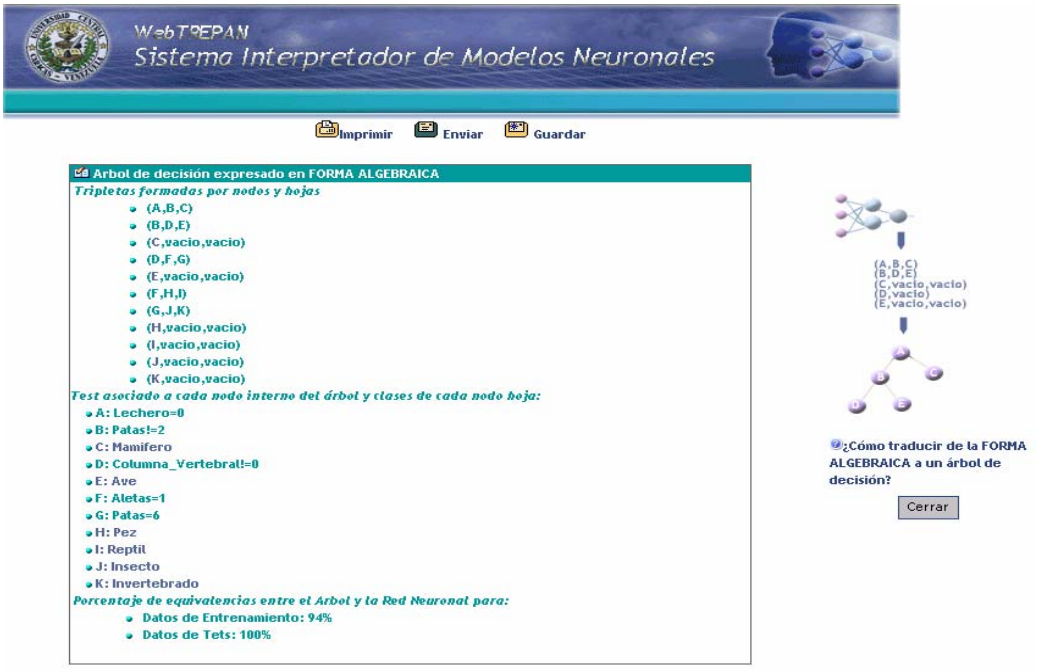

Figura 9. Módulo de Interpretación. Página de resultados

5. EXPERIMENTOS

Los experimentos que se presentan a continuación, ilustran la aplicación del sistema WebTREPAN a redes neuronales entrenadas sobre seis Bases de Datos del Repositorio UCI [5], el cual representa un estándar dentro de la comunidad del aprendizaje automático. En estos experimentos se evaluó la *fidelidad* de los árboles de decisión extraídos con respecto a la red neuronal desde la cual se realizó la extracción; esta medida viene dada por la siguiente expresión:

$$Porcentaje\ de\ Fidelidad = \frac{N°\ de\ acuerdos}{N°\ total\ de\ datos} x100 \quad (3)$$

Las características de las bases de datos utilizadas se muestran en la Tabla 1. Sobre éstas se entrenaron redes neuronales utilizando diferentes topologías y funciones de activación para evaluar la funcionalidad del sistema. Como ejemplo, en la Figura 10 se muestra el árbol obtenido (en forma algebraica y gráfica) a partir de una red neuronal entrenada con la base de datos IRIS. En esta base de datos se busca clasificar en tres clases, las cuales indican el tipo de planta IRIS (Iris-setosa, Iris-virginica e Iris-versicolor), a partir de 4 características, a saber: longitud y ancho del pétalo, y longitud y ancho del sépalo.

Tabla 1. Bases de datos y sus características

Cód.	Base de datos	No. de datos	No. de atributos	Tipo de atributos	No. de clases
1	IRIS	150	4	Numéricos (continuos)	3
2	WISCONSIN	699	9	Categóricos	2
3	WINE	178	13	Numéricos (continuos)	3
5	New-THYROID	215	5	Numéricos (continuos)	3
11	ZOO	101	16	Categóricos	7
12	HEART	270	13	Numéricos, categóricos y binarios	2

La Tabla 2 presenta los porcentajes de equivalencia de los árboles generados por WebTREPAB, con respecto a los datos entrenamiento y de test, para cada conjunto de datos.

Tabla 2. Porcentajes de equivalencia de los árboles de decisión obtenidos para las bases de datos

DATOS	IRIS	HEART	NEW THYROIDE	WISCONSIN	ZOO	WINE
Entrenamiento	94%	92%	86%	98%	94%	98%
Test	97%	93%	95%	94%	100%	94%

Árbol en forma algebraica

(A, B, C)

(B, D, E)

(E, F, C)

Leyenda:

A: 1 de 2 { Longitud del Pétalo > 4.13, Ancho sepal <= 2.96 }
B: Longitud del Pétalo > 4.77
C: Iris Setosa
D: Iris Virginica
E: Longitud del Pétalo >= 4.36
F: Iris Versicolor

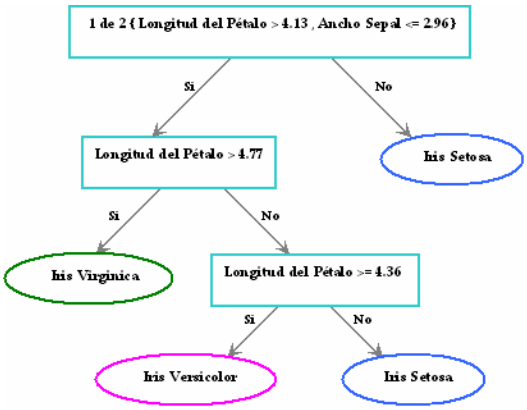

Figura 10. Árbol de decisión generado por WebTREPAN a partir de una red neuronal entrenada con la base de datos IRIS. La topología de la red es: una capa oculta con 3 neuronas, una capa de salida con 3 neuronas y función de activación sigmoide para ambas capas

6. CONCLUSIONES Y TRABAJOS FUTUROS

El Sistema Interpretador de Modelos Neuronales WebTREPAN permite extraer, de una manera automática, una representación simbólica aproximada en forma de una árbol de decisión, del conocimiento que ha capturado una red neuronal durante su aprendizaje. De esta forma, esta herramienta puede ser utilizada para facilitar la comprensión y el análisis del problema que está bajo investigación y de su solución, así como servir de apoyo en procesos de toma de decisiones por parte de expertos. Además, WebTREPAN puede complementar otras sistemas utilizados en la minería de datos.

WebTREPAN cuenta con una interfaz basada en Web sencilla y útil por medio de la cual el usuario introduce los datos de la red neuronal entrenada que desea interpretar, se le muestra al usuario el árbol de decisión en forma algebraica, y se le dan las especificaciones para que lo entienda fácilmente. Asimismo, se le brindan las consideraciones necesarias durante todo en el proceso de interpretación, lo cual reduce la posibilidad de que el usuario cometa errores. Sin embargo, se podría mejorar la visualización de los resultados de manera gráfica y dinámica, para que el usuario no tenga que realizar la traducción de la forma algebraica al árbol de decisión propiamente.

El sistema desarrollado puede ser ampliado siguiendo dos vías: una de ellas, incorporando otros paradigmas de aprendizaje considerados del tipo *caja negra*, como es el caso de las máquinas de soporte vectorial (SVM). Esto es factible ya que TREPAN es un método de extracción global; bastaría entonces con construir y añadir los Oráculos respectivos. La otra vía está relacionada con la incorporación de otros métodos de extracción de reglas para redes neuronales que produzcan otra forma de representación, como reglas de producción o autómatas de estado finito, entre otras

Referencias

[1] Andrews R., Diederich J. y Tickle A. (1995). A survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*. 8(6):373-389.

[2] Bengio, Y., Buhmann, J., Embrechts, M. y Zurada, J. (2000). Introduction to the Special Issue on Neural Networks for Data Mining and Knowledge Discovery. *IEEE Transactions on Neural Networks*. 11(3):545-549.

[3] Berthold, M. y Hand, D. (1999). *Intelligent Data Analysis*. *An Introduction*. Springer-Verlag.

[4] Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

[5] Blake, C.L. y Merz, C.J. (1998). UCI Repository of Machine Learning Data-Bases. University of California, Irvine. Dept. of Information and Computer Science. (http://www.ics.uci.edu/~mlearn/MLRepository.html).

[6] Canavos, G. (1988). *Probabilidad y Estadística: Aplicaciones y Métodos*. McGraw-Hill.

[7] Craven M. y Shavlik J. (1996). Extracting Tree-structured representations of trained networks. *Advances in Neural Information Processing Systems*. 8:24-30.

[8] Craven M. y Shavlik J. (1997). Using Neural Networks for Data Mining. *Future Generation Computer Systems*. 13:211-229.

[9] Craven, M. (1996). *Extracting Comprehensible Models from Trained Neural Networks.* PhD Thesis, Department of Computer Sciences, University of Wisconsin, Madison. USA.

[10] Haykin, S (1994). *Neural Networks. A Comprehensive Foundations*. Macmillan College Publishing Company.

[11] Jacobson, I., Booch, G. y Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software*. Addison Wesley y Pearson Educación.
[12] Jacobson, I. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley.
[13] Kecman, V. (2001). *Learning and Soft Computing. Support Vector Machines, Neural Networks and Fuzzy Logic Models*. MIT Press.
[14] Mendenhall, W., Scheaffer, R. y Wackerly, D. (1986). *Estadística Matemática con Aplicaciones*. 3ra. Edición. Grupo Editorial Iberoamérica.
[15] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
[16] Mitra, S., Pal, S. K. y Mitra, P. (2002). Data Mining in Soft Computing Framework: A survey. *IEEE Transactions on Neural Networks*. 13 (1):3-14.
[17] Negnevitsky, M. (2002). *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley.
[18] Press, W., Teukolsky, S., Vetterling, W. y Flannery, B.(1992). *Numerical Recipes in C*. 2nd Edition. Cambridge University Press.
[19] Ross Sheldon, M. (1990). *A Course in Simulation.* Macmillan Publishing Company. New York.
[20] Rumbaugh, J. (1996). *Modelado y diseño orientados a objetos: Metodología OMT.* Prentice Hall.
[21] Setiono, R., Leow, W. y Zurada, J. (2002). Extraction Rules from Artificial Neural Networks for Nonlinear Regression. *IEEE Transactions on Neural Networks*. 13(3):564-577.
[22] Tickle A., Andrews R., Mostefa G. y Diederich J. (1998). The Truth will come to light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. *IEEE Transactions on Neural Networks*. 9(6): 1057-1068.
[23] Tickle, A., Maire, F., Bologna, G., Andrews, R.; Diederich, J.: Lessons from Past, Current Issues, and Future Research Directions in Extracting the Knowledge Embedded Artificial Neural Networks. En: Wermter, S. y Sun, R. (Eds): *Hybrid Neural System*s. Springer-Verlag.
[24] Witten, I. y Frank, E. (2000). *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.