Data Mining utilizando Redes Neuronales

Romina Laura Bot rbot@fi.uba.ar
Juan M. Ale ale@acm.org
Facultad de Ingeniería
Universidad de Buenos Aires

Resumen: Las Redes Neuronales son ampliamente utilizadas para tareas relacionadas con reconocimiento de patrones y clasificación. Aunque son clasificadores muy precisos, no son comúnmente utilizadas para Data Mining porque producen modelos de aprendizaje inexplicables. El algoritmo TREPAN extrae hipótesis explicables de una Red Neuronal entrenada. Las hipótesis producidas por el algoritmo se representan con un árbol de decisión que aproxima a la red. Los árboles de decisión extraídos por TREPAN no pueden describir predicciones de Redes Neuronales entrenadas para realizar tareas de regresión y, algunas veces, los árboles de decisión extraídos no son lo suficientemente concisos. En el presente trabajo se presentan dos nuevos algoritmos de extracción de reglas que construyen árboles difusos y árboles modelo a partir de una Red Neuronal entrenada, para ser utilizados en aquellos dominios en los que el árbol de decisión no es lo suficientemente conciso, o no describe correctamente el método de clasificación de la Red Neuronal que realiza tareas de regresión. Con estos nuevos métodos es posible comprender el método de clasificación de una Red Neuronal entrenada para un rango de dominios no cubierto por TREPAN.

Palabras clave: data mining, redes neuronales, árboles de decisión, árboles modelo, árboles difusos.

1. Introducción

El objetivo de hacer Data Mining es la búsqueda de información nueva al explorar un gran volumen de datos. A menudo, para alcanzar este objetivo se aplican métodos de aprendizaje supervisado y no supervisado que construyen modelos sobre los datos que se están manejando.

Las Redes Neuronales se han utilizado en una gran variedad de aplicaciones de aprendizaje supervisado y no supervisado, para tareas relacionadas con clasificación y reconocimientos de patrones. Sin embargo, aunque son clasificadores muy precisos, no son comúnmente utilizadas para Data Mining porque producen modelos de aprendizaje inexplicables y requieren mucho tiempo de entrenamiento (Craven et al., [3]). Las Redes Neuronales pueden predecir con gran precisión pero son como una "caja negra". En la mayoría de las tareas de aprendizaje, es importante construir clasificadores que no solo sean precisos sino también fáciles de entender por los humanos (Craven et al., [4]).

Existen varios enfoques para realizar Data Mining utilizando Redes Neuronales que son capaces de producir modelos explicables y no requieren tiempos de entrenamiento excesivos. En uno de los enfoques, llamado '*Extracción de Reglas*', se extraen modelos simbólicos de Redes Neuronales ya entrenadas.

En (Craven & Shavlik, [4]) los autores han desarrollado un algoritmo que extrae reglas a partir de una Red Neuronal entrenada. El algoritmo TREPAN construye un árbol de decisión con reglas *m-*

de-*n* que modela el método de clasificación de la Red Neuronal. Los árboles de decisión extraídos por TREPAN no pueden describir predicciones de Redes Neuronales entrenadas para realizar tareas de regresión y, algunas veces, los árboles de decisión extraídos no son lo suficientemente concisos. En el presente trabajo se presentan dos nuevos algoritmos de extracción de reglas que construyen árboles difusos y árboles modelo a partir de una Red Neuronal entrenada.

La contribución de este trabajo es que con estos nuevos métodos es posible comprender el método de clasificación de una Red Neuronal entrenada para un rango de dominios no cubierto por TREPAN:

- en aquellos dominios con valores continuos para los que el árbol de decisión no es lo suficientemente conciso, se extraerán árboles difusos.

- en los dominios donde en lugar de asignar una clase como resultado de la clasificación se asignan valores numéricos, se extraerán árboles modelo.

El presente trabajo se organiza de la siguiente forma: en la Sección 2 se presentan trabajos relacionados con la extracción de reglas utilizando Redes Neuronales. En la Sección 3 se detallan los antecedentes y conocimientos previos para el desarrollo de los nuevos algoritmos. En la Sección 4 se presenta una nueva implementación del algoritmo TREPAN. Los nuevos algoritmos de extracción de árboles difusos y árboles modelo se explican en la Sección 5. Los resultados experimentales se detallan en la Sección 6. Por último, la Sección 7 corresponde a la conclusión.

2. Trabajos Relacionados

Se han desarrollado varios algoritmos de extracción de reglas a partir de Redes Neuronales. Estos algoritmos difieren en el método utilizado para la extracción de las reglas de decisión.

Muchos algoritmos de extracción de reglas realizan esta tarea como un problema de búsqueda, explorando un espacio de reglas candidatas y testeando cada una de ellas contra la red para verificar si son reglas válidas. La búsqueda continúa hasta que todas o la mayoría de las reglas generales hayan sido encontradas. El método llamado análisis de intervalos válidos (VIA – validity interval analysis) es una versión de esta técnica (Thrun, [18]). VIA testea las reglas propagando intervalos de activación a través de una red luego de poner restricciones sobre algunas de las neuronas de entrada y de salida, determinando intervalos de validez (p.e. rangos de activación válidos para cada neurona).

Los métodos basados en la búsqueda extraen reglas que describen el comportamiento de las neuronas de salida en términos de las neuronas de entrada. Otro enfoque al problema de extracción de reglas descompone la red en una colección de redes, y luego extrae un conjunto de reglas describiendo cada subred. Las reglas de cada subred pueden combinarse en un único conjunto de reglas que describen a toda la red (Setiono, [17]).

Un enfoque alternativo para hacer Data Mining con Redes Neuronales es utilizar métodos de aprendizaje que directamente aprenden hipótesis explicables produciendo Redes Neuronales simples. Estas son redes de una sola capa. Las hipótesis representadas por una Red Neuronal de una única capa son usualmente mucho más simples de entender porque cada parámetro describe una

relación simple (p.e. lineal) entre un atributo de entrada y una clase de salida. Ejemplos de este enfoque son el algoritmo BBP (Craven, [3]) y el método de Tom Bylander (Bylander, [2]).

Los métodos basados en la topología de la red son aquellos que requieren Redes Neuronales con una topología específica (cantidad de neuronas, cantidad de capas ocultas), o realizan una poda (pruning) en la red después de entrenarla, o utilizan funciones umbral determinadas, o construyen la red durante el proceso de entrenamiento. C-MLP2NL (Duch et al., [8]) transforma una Red Neuronal del tipo perceptrón multi-capa (MLP) en una red que realiza operaciones lógicas (red lógica, LN). Esta transformación se logra modificando los pesos de la red y detectando las conexiones relevantes durante el entrenamiento. El algoritmo KBANN (Towell et al., [19]) utiliza una base de conocimiento con reglas de inferencia del dominio a estudiar, para determinar la topología y los pesos iniciales de una Red Neuronal basada en el conocimiento (KNN). Luego de entrenar y refinar la red, se extraen dos tipos de reglas: versiones modificadas de las reglas utilizadas para inicializar la red, y reglas que describen nuevas relaciones entre los atributos.

Otro de los enfoques para comprender la representación aprendida por una Red Neuronal entrenada consiste en extraer la descripción de los conceptos aprendidos utilizando la misma red: dada una Red Neuronal y los datos con los que fue entrenada, producir una descripción conceptual que sea comprensible y que clasifique a las instancias de la misma forma en que lo hace la red. Un ejemplo de este enfoque es el algoritmo TREPAN, presentado por Craven y Shavlik ([3]-[6]), que trata el problema de extracción de hipótesis explicables de una red entrenada como una tarea de aprendizaje inductivo. Este método será explicado en profundidad en la próxima sección, dado que es el método tomado como base para el desarrollo del presente trabajo.

Nuestro trabajo es otro ejemplo de este último enfoque, pero difiere con el método TREPAN en el lenguaje utilizado para describir el modelo aprendido por la red. En lugar de extraer árboles de decisión, nuestros métodos construyen árboles difusos y árboles modelo. De esta forma es posible comprender el método de clasificación de una Red Neuronal entrenada para dominios con valores continuos.

3. Antecedentes

3.1 Data Mining y Clasificación

Uno de los objetivos fundamentales de Data Mining es poder predecir el valor de una variable predictiva o dependiente en función de los valores de otras variables llamadas independientes, existentes en una base de datos. Los clasificadores asignan ‘clases’ a las instancias de un dataset. Para un conjunto de registros con sus correspondientes atributos y una clase asignada a cada uno de ellos, un clasificador produce descripciones de las características de los registros de cada clase.

El aprendizaje supervisado es muy útil en Data Mining ya que al construir modelos descriptivos de un conjunto de datos, se provee una forma de explorarlos. En el aprendizaje supervisado, se da al clasificador un conjunto de instancias de la forma (x, y), donde y representa la variable que el sistema debe predecir y x es el vector de valores de los atributos relevantes para determinar y. El objetivo del aprendizaje supervisado es inducir un mapeo general de vectores x a valores y. O sea, el clasificador debe construir un modelo $y = f(x)$ de la función f desconocida, que permita predecir los valores y para ejemplos no conocidos previamente.

3.2 Redes Neuronales

Las Redes Neuronales se utilizan para clasificación y para reconocimiento de patrones. Son, en muchos casos, el método de aprendizaje preferido porque inducen hipótesis que generalizan mejor que los otros algoritmos. Varios estudios indican que en algunos dominios las Redes Neuronales proveen mayor precisión predictiva que los algoritmos de aprendizaje simbólicos comúnmente utilizados (por ejemplo: árboles de decisión). Aún así, las Redes Neuronales no son muy utilizadas para Data Mining. Hay dos razones principales: los métodos de clasificación de las Redes Neuronales ya entrenadas no son explicables, y los tiempos de aprendizaje son lentos, imprácticos para grandes volúmenes de datos (Craven et al., [3]).

Las Redes Neuronales son colecciones de nodos conectados, con entradas, salidas, y procesamiento en cada nodo. Entre las entradas y las salidas de la red existe un número de capas ocultas de procesamiento. La Red Neuronal debe ser entrenada con un conjunto de patrones de entrenamiento (aprendizaje supervisado). Una vez entrenada es utilizada para hacer predicciones.

Las hipótesis representadas por una Red Neuronal entrenada dependen de la topología de la red, de las funciones de transferencia utilizadas para las neuronas ocultas y de salida, y de los parámetros asociados con las conexiones en la red (p.e. los pesos) y las neuronas (p.e. el valor umbral). Estas hipótesis son difíciles de explicar por varias razones: las Redes Neuronales típicamente utilizadas tienen cientos o miles de parámetros. Estos parámetros codifican la relación entre los atributos de entrada, x, y el valor a determinar y. Además, en redes multi-capa, los parámetros pueden representar relaciones no lineales o no monótonas entre los atributos de entrada y el valor a determinar. Esto hace que no sea posible determinar, en forma aislada, el efecto de un atributo dado en el valor de salida obtenido.

3.3 Extracción de Reglas utilizando Redes Neuronales

Una forma de entender las hipótesis representadas por una Red Neuronal entrenada es traducir estas hipótesis en lenguajes más comprensibles. Esta estrategia es la *Extracción de Reglas* (Craven et al., [3]). Existen varios motivos por los que es deseable poder extraer reglas a partir de una Red Neuronal (Boz, [1]):

- Las reglas extraídas de la red entrenada pueden utilizarse en otros sistemas, por ejemplo, en Sistemas Expertos.

- Las reglas pueden usarse para descubrir características previamente desconocidas en los datos (Data Mining).

- Las reglas explican el razonamiento con el que se llega a un resultado.

- Las reglas pueden utilizarse para refinar dominios de conocimiento incompletos.

Hay varios métodos de extracción de reglas. Estos métodos difieren en tres características:

- **Lenguajes de Representación**: es el lenguaje utilizado por el método de extracción para describir el modelo aprendido por la red. Los lenguajes utilizados incluyen reglas de inferencias conjuntivas (*if-then*), reglas *m*-de-*n*, reglas difusas, árboles de decisión, etc.

- **Estrategias de extracción**: es la estrategia utilizada por el método de extracción para mapear el modelo representado por la red entrenada en un nuevo modelo.

- **Requerimientos de la Red**: son los requerimientos de arquitectura y de entrenamiento que los métodos de extracción imponen a las Redes Neuronales. En otras palabras, el rango de redes al que se puede aplicar el método.

Siempre que una Red Neuronal se utilice para resolver algún problema de clasificación, existirá un procedimiento de decisión implícito para determinar la clase a predecir para un caso dado. En general, una regla extraída describe (aproximadamente) un conjunto de condiciones bajo las cuales la red predice una clase.

3.4 Algoritmo TREPAN

El algoritmo TREPAN, presentado por Mark W. Craven ([3]-[6]), trata el problema de extracción de hipótesis explicables de una red entrenada como una tarea de aprendizaje inductivo. En esta tarea de aprendizaje, el objetivo es la función representada por la red. Las hipótesis producidas por el algoritmo de aprendizaje se representan con un árbol de decisión que aproxima a la red.

TREPAN puede producir árboles de decisión que mantienen un alto grado de fidelidad con la red, precisos y comprensibles. En Data Mining, es importante producir clasificadores que no sean solo precisos, sino también fáciles de entender. Las Redes Neuronales tienen limitaciones en este aspecto, ya que son difíciles de interpretar luego del entrenamiento.

Aunque TREPAN tiene muchas similitudes con los algoritmos convencionales de extracción de árboles de decisión, es sustancialmente diferente en varios aspectos:

Consultas de Pertenencia y Oráculo. Al inducir un árbol de decisión para describir la red dada, TREPAN hace *consultas de pertenencia*. Una consulta de pertenencia es una pregunta a un *oráculo* sobre una instancia del dataset. El rol del oráculo es devolver la clase a la que pertenece la instancia. Lo que se intenta aprender es la función representada por la red, entonces la red misma es la que se utiliza como oráculo, y para responder una consulta de pertenencia simplemente clasifica la instancia dada.

Las instancias utilizadas para hacer consultas de pertenencia provienen de dos fuentes. Primero, los ejemplos utilizados para entrenar la red. Segundo, TREPAN utiliza los datos de entrenamiento para construir modelos de la distribución de los datos, y con estos modelos genera nuevas instancias – instancias de consulta – que clasifica con el oráculo. La ventaja es que al seleccionar un criterio de partición para un nodo interno o al clasificar a una hoja, es posible decidir en base a una gran cantidad de ejemplos.

Expansión del Árbol. TREPAN construye el árbol de decisión usando el criterio de mejor expansión. El mejor nodo es aquel que tiene el mayor potencial para incrementar la *fidelidad* del árbol extraído. Se entiende por fidelidad el grado de coincidencia entre la clasificación del árbol y la de la red. Para obtener un árbol fiel a la red, este se construye utilizando las instancias de entrenamientos clasificadas por el oráculo.

Test de Partición. el algoritmo utiliza expresiones *m*-de-*n* para producir descripciones más compactas. Una expresión *m*-de-*n* es una expresión booleana, especificada por un umbral entero *m*, y un conjunto de *n* condiciones booleanas. Una expresión *m*-de-*n* se satisface cuando al menos *m* de sus *n* condiciones son satisfechas.

Criterio de Detención. TREPAN utiliza tres criterios para decidir cuando detener el crecimiento del árbol. El primero es un test estadístico para decidir si, con alta probabilidad, un nodo cubre solo las instancias de una sola clase. Si es así, el nodo pasa a ser una hoja. Segundo, TREPAN emplea un parámetro que permite al usuario establecer un límite en el tamaño del árbol. Tercero, el algoritmo permite utilizar un conjunto de validación para decidir cuando detener el crecimiento del árbol.

Ventajas del Algoritmo. TREPAN puede ser aplicado a una gran variedad de Redes Neuronales, ya que la interacción con la red consiste solo en realizar consultas de pertenencia. TREPAN no requiere una arquitectura especial para la red ni un método de entrenamiento en particular. Además, permite al usuario tener un mayor control sobre la complejidad de las hipótesis producidas por el proceso de extracción de reglas.

4. Una Implementación del Algoritmo TREPAN

En el presente trabajo se ha implementado una nueva versión del algoritmo TREPAN. Este algoritmo difiere con el método propuesto por Craven en los siguientes aspectos:

Lenguaje de Representación. El árbol de decisión utiliza reglas *if-then* en lugar de reglas *m*-de-*n* en la partición de los nodos. Los árboles de decisión con reglas *m*-de-*n* son más difíciles de entender que los árboles de decisión regulares (Boz, [1]), aunque a simple vista sean árboles más compactos.

Oráculo. Es posible elegir el método de clasificación que será utilizado como oráculo para la construcción del árbol.

Criterio de Detención. El algoritmo emplea dos parámetros ingresados por el usuario. El primero permite al usuario establecer un límite en el tamaño del árbol, y el segundo indica la proporción de instancias de una misma clase que determina si un nodo es hoja.

Test de Partición. Se utilizará la Ganancia de Información como criterio para generar la mejor partición en cada nodo. Los atributos continuos se discretizan encontrando un punto de corte T que divida el dataset S en dos intervalos, según el valor del atributo A: S_1 = [min(A), T] y S_2 = (T, max(A)]. El valor T se determina calculando la ganancia de información para cada punto de corte candidato, y seleccionando aquel punto que maximiza la ganancia de información.

Generación de nuevas instancias. TREPAN genera nuevas instancias –instancias de consulta– para alcanzar la cantidad mínima de instancias necesarias para evaluar la partición de un nodo. Para generar nuevas instancias, se tiene en cuenta la distribución marginal de cada uno de los atributos de las instancias a procesar. Se utilizan *distribuciones empíricas* para modelar atributos discretos y el *estimador de densidad kernel* para modelar atributos continuos. En esta versión del algoritmo, la cantidad mínima de instancias requeridas para evaluar un nodo es determinada por el usuario.

Instancias con valores desconocidos. El algoritmo permite generar árboles a partir de instancias con valores desconocidos para algunos de sus atributos. El algoritmo aplica un filtro a las instancias

de entrenamiento. Este filtro reemplaza los valores desconocidos por los valores mas frecuentes del atributo.

5. Nuevos Algoritmos

Para cierto tipo de dominios, el algoritmo TREPAN presenta algunas limitaciones (Craven et al., [6]):

- TREPAN extrae árboles de decisión. Los árboles de decisión no pueden describir predicciones de Redes Neuronales entrenadas para realizar tareas de regresión.

- Algunas veces el árbol de decisión extraído no es lo suficientemente conciso, lo que implica una gran cantidad de reglas de clasificación.

En el presente trabajo se presentan dos nuevos algoritmos que resuelven estas limitaciones:

- El algoritmo TREPAN M5 extrae árboles modelo a partir de una Red Neuronal entrenada. Un árbol modelo tiene una estructura similar a la de un árbol de decisión, pero sus hojas son funciones de valores reales, en lugar de identificadores de clases (Quinlan, [16]).

- El algoritmo TREPAN Fuzzy extrae árboles difusos. Un árbol de decisión difuso indica el grado de pertenencia de una instancia a cada una de las clases. De acuerdo a los resultados, los usuarios pueden tomar la decisión final. Se obtienen árboles más concisos y la tasa de error en la clasificación puede reducirse.

5.1 TREPAN M5

Se implementó el algoritmo TREPAN M5 tomando como base el algoritmo M5' (M5 Prime). TREPAN M5 construye árboles modelo, regresiones lineales y árboles de regresión utilizando una Red Neuronal (u otro clasificador a elección) como oráculo.

Algoritmo M5'. Esta técnica de aprendizaje supervisado usa la siguiente idea: dividir el espacio de datos en áreas (subespacios) y construir un modelo de regresión lineal en cada una de ellas. El modelo resultante puede verse como un modelo modular formado por modelos lineales especializados en un conjunto particular del espacio de entrada. M5' se basa en el principio de la teoría de la información. El test de partición es similar al de un árbol de decisión, pero en lugar de clases asigna funciones de regresión lineal a las hojas del árbol. Los árboles modelo generalizan el concepto de los árboles de regresión, que tienen valores constantes en sus hojas.

Implementación del algoritmo TREPAN M5. La implementación consistió en agregar al algoritmo M5', el oráculo. Previo a la construcción del modelo de regresión, se clasifican las instancias de entrenamiento con el oráculo. Además, en cada test de partición de instancias, es posible determinar la cantidad mínima de instancias requeridas para realizar la evaluación. De no llegar a esta cantidad mínima, el algoritmo crea las instancias faltantes realizando consultas de pertenencia al oráculo, tal como lo hace TREPAN.

Es posible construir tres tipos de modelos:

- **Regresión Lineal**: devuelve una función de regresión lineal que modela la salida como una combinación lineal de los atributos de entrada.
- **Árbol Modelo**: devuelve un árbol de decisión, en el que cada nodo hoja está asociado a una función de regresión lineal.
- **Árbol de Regresión**: devuelve un árbol de decisión, en el que cada nodo hoja está asociado a un valor numérico constante.

TREPAN M5 clasifica datasets con clases continuas. El algoritmo acepta instancias con valores desconocidos.

5.2 TREPAN Fuzzy

TREPAN Fuzzy construye árboles difusos, utilizando una Red Neuronal (u otro clasificador a elección) como oráculo. La construcción del árbol difuso se basa en el método propuesto por Yonghohg Peng y Peter Flash (Peng et al., [15]). Este método de inducción se basa en la '*Discretización Suave*' (Soft Discretization) de los atributos continuos en conjuntos difusos.

Discretización Suave. Los métodos de discretización convencionales dividen el espacio de decisión en un conjunto de subespacios excluyentes (figura 1.a). Por el contrario, un árbol de decisión difuso da resultados dentro del rango [0,1], indicando el grado de posibilidad de un objeto de pertenecer a cada clase. De acuerdo a estos resultados, los usuarios pueden tomar la decisión final o realizar una investigación más profunda. Como resultado, la tasa de error en la clasificación puede reducirse.

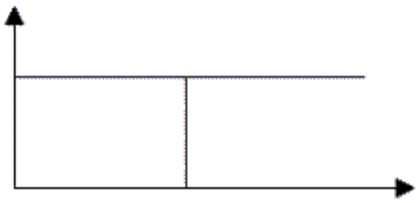

Fig. 1.a Discretización convencional

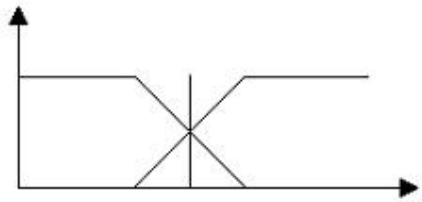

Fig. 1.b Discretización suave

La discretización suave se define con un par de conjuntos difusos que forman una partición difusa, como se muestra en la figura 1.b. En contraste con el método convencional de particiones excluyentes, los conjuntos se superponen para un rango de valores cercanos al punto de corte T. Las funciones de '*membership*' del par de conjuntos difusos A_1 y A_2 son:

$$A_1(x) = \frac{\left(T + \frac{d}{2}\right) - x}{d} \qquad A_2(x) = \frac{x - \left(T - \frac{d}{2}\right)}{d}$$

El punto de corte T es determinado por el valor que maximiza la ganancia de información difusa. Se utiliza la distancia promedio de los valores del atributo como el ancho de superposición *d*.

Implementación del Algoritmo TREPAN Fuzzy. Para la implementación, se modificó el algoritmo TREPAN para adaptarlo al método de construcción de árboles difusos basado en la discretización suave de los atributos continuos en conjuntos difusos.

6. Resultados Experimentales

En esta sección se detallan los resultados obtenidos al clasificar los siguientes datasets con los nuevos algoritmos. Estos datasets forman parte del repositorio de bases de datos de la UCI.

Dataset	#Instancias	#Atributos	#At. Nominales	#At. Numéricos	Valores Desconocidos	# Clases
heart-c	303	14	7	6	si	5
ionosphere	351	35	0	34	no	2
waveform	5000	41	0	40	no	3
cpu	209	8	0	7	no	continua
autoHorse	205	26	8	17	si	continua
pollution	60	16	0	15	no	continua

Se seleccionaron aquellos datasets para los cuales las Redes Neuronales clasifican con mayor precisión que el resto de los algoritmos. Los datasets con clases discretas se clasificaron con la Red Neuronal y con los métodos TREPAN Fuzzy y J48. Los datasets con clases continuas se clasificaron con la Red Neuronal y con los métodos TREPAN M5 y M5'. Para todos los casos, se eligió como oráculo una Red Neuronal Back Propagation. Los tiempos de ejecución de los algoritmos TREPAN M5 y TREPAN Fuzzy incluyen el tiempo que insume el entrenamiento de la Red Neuronal utilizada como oráculo.

6.1 TREPAN M5

A continuación se detallan los resultados obtenidos al clasificar los datasets con clases continuas utilizando el algoritmo TREPAN M5 para generar árboles modelo. Cada hoja del árbol modelo esta asociada a una regla, que en este caso es una función lineal. Por lo tanto, el número de reglas obtenidas es equivalente a la cantidad de hojas del árbol resultante.

CPU

Clasificador	Tiempo (seg.)	Coef. Correl.	Mean abs. error	Root mean sq error	Relative abs. error	Root Relat. sq error	# Reglas
Red Neuronal	19,98	0,9936	7,1281	17,9644	8,16%	11,64%	-
M5'	0,36	0,9763	13,3864	35,6149	15,32%	23,07%	2
TREPAN M5	83.88	0,9852	11,2033	28.1359	12,82%	18.22%	6

AutoHorse

Clasificador	Tiempo (seg.)	Coef. Correl.	Mean abs. error	Root mean sq error	Relative abs. error	Root Relat. sq error	# Reglas
Red Neuronal	7.18	0,9479	7,1875	12.626	23.56%	31.87%	-
M5'	0,26	0,9412	8,819	13,431	28,91%	33,90%	27
TREPAN M5	46.09	0,9385	7.9235	13.7037	25.97%	34.59%	6

Pollution

Clasificador	Tiempo (seg.)	Coef. Correl.	Mean abs. error	Root mean sq error	Relative abs. error	Root Relat. sq error	# Reglas
Red Neuronal	0,23	0,755	29,9736	40,9585	60,33%	66,40%	-
M5'	0,07	0,6847	35,7613	46,4166	71,98%	75,25%	1
TREPAN M5	0.32	0,7552	30,1949	40.8559	60,78%	66.23%	1

Se observa que para los casos en que la clasificación de la Red Neuronal es mas precisa, TREPAN M5 genera un clasificador que se aproxima en precisión a la red. Los tiempos de ejecución del algoritmo TREPAN M5 son elevados, aún así se justifica su uso debido a que permite obtener una descripción del método de clasificación de la Red Neuronal. Las reglas extraídas explican el razonamiento de la red.

6.2 TREPAN Fuzzy

A continuación se detallan los resultados obtenidos al clasificar los datasets utilizando el algoritmo TREPAN Fuzzy para generar árboles difusos.

Heart-C

Clasificador	Tiempo (seg.)	Precisión	Fidelidad	#Nodos	#Hojas
Red Neuronal	24,22	82,51 %	-	-	-
J48	0,02	74,59 %	-	7	14
TREPAN Fuzzy	31.68	81.18 %	86.17 %	5	12

Ionosphere

Clasificador	Tiempo (seg.)	Precisión	Fidelidad	#Nodos	#Hojas
Red Neuronal	31,28	92,02 %	-	-	-
J48	0,26	88,60 %	-	17	18
TREPAN Fuzzy	38.19	89.74 %	90.32 %	7	8

Waveform

Clasificador	Tiempo (seg.)	Precisión	Fidelidad	#Nodos	#Hojas
Red Neuronal	775,03	85,29 %	-	-	-
J48	4,01	75,72 %	-	93	94
TREPAN Fuzzy	1157.05	76.54 %	77.64 %	36	37

Se observa que al aplicar TREPAN Fuzzy a datasets clasificados con la Red Neuronal, se obtienen clasificadores precisos, y con alta fidelidad. Aunque los tiempos de ejecución del algoritmo TREPAN Fuzzy son mayores con respecto a J48, es posible obtener árboles más concisos, que modelan el método de clasificación de la Red Neuronal en un lenguaje de representación comprensible para el ser humano. Esta diferencia se debe al tiempo que insume el entrenamiento de la red utilizada como oráculo, y la creación de nuevas instancias de consulta.

7. Conclusiones y Trabajo Futuro

Se presentaron dos nuevas técnicas de extracción de reglas a partir de redes neuronales entrenadas, que producen árboles de decisión difusos y árboles modelo, basadas en el método TREPAN.

Se observó que para aquellos dominios donde las redes neuronales clasifican con mayor precisión que los métodos convencionales, es posible obtener una descripción del método de clasificación de la red, con gran precisión y con un alto grado de fidelidad.

Cabe destacar la generalidad de los nuevos algoritmos, que no requieren regímenes especiales de entrenamiento o restricciones en la arquitectura de la red. Además, es posible seleccionar cualquier método de clasificación como oráculo.

Como trabajo futuro, se analizará la reducción de los tiempos de ejecución para mejorar la escalabilidad de los algoritmos. Por otra parte, se optimizarán las técnicas de construcción de árboles modelo y árboles difusos, como así también se utilizarán nuevos lenguajes de representación del método de clasificación de la Red Neuronal.

Los algoritmos TREPAN, TREPAN M5 y TREPAN Fuzzy se han implementado en el lenguaje Java integrándolos al sistema Weka [20].

8. Referencias

[1] Boz, O. Converting A Trained Neural Network To A Decision Tree. DecText – Decision Tree Extractor. ICMLA 2002, pp. 110-116, Junio 2002.

[2] Bylander, T. Learning Linear Threshold Approximations Using Perceptrons. Neural Computation, vol. 7, no. 2, pp. 370-379, 1995.

[3] Craven, M. W. & Shavlik, J. W. Using Neural Networks for Data Mining. Future Generation Computer Systems, 13, pp. 211-229, Noviembre 1997.

[4] Craven, M. W. & Shavlik, J. W. Extracting Tree-Structured Representations of Trained Networks. Advances in Neural Information Processing Systems, pp. 24-30, Denver, CO. MIT Press, 1996

[5] Craven, M. W. & Shavlik, J. W. Rule Extraction: Where Do We Go from Here?. Department of Computer Sciences, University of Wisconsin, Machine Learning Research Group Working Paper 99-1, 1999.

[6] Craven, M. W. & Shavlik, J. W. Understanding Time-Series Networks: A Case Study in Rule Extraction. International Journal of Neural Systems, 8, pp. 373-384, Agosto 1997

[7] Dong, M. Look-Ahead Based Fuzzy Decision Tree Induction. IEEE Transactions on Fuzzy System, vol. 9, no. 3, pp. 461-468, Junio 2001.

[8] Duch W., Adamczak R. & Grabczewski K. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks, vol. 11(2), pp. 1-31, Marzo 2000.

[9] Duch W. et al. Neural Methods of knowledge extraction. Control & Cybernetics vol. 29, no. 4, 2000.

[10] Grzymala-Busse, J. & Hu, M. A Comparison of Several Approaches to Missing Attribute Values in Data Mining. Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing RSCTC'2000, Banff, Canada, pp. 340–347, Octubre 2000.

[11] Ianova, I. & Kubat, M. Initialization of Neural Networks by Means of Decision Trees. Knowledge-Based Systems8, pp. 333-344, 1995.

[12] Janikow, C. Z. Exemplar Learning in Fuzzy Decision Trees. Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, vol. 2, pp. 1500 –1505, 1996.

[13] Kohavi, R. A Study of Cross-Validation and Boostrap for Accurancy Estimation and Model Selection. International Joint Conference on Artificial Intelligence IJCAI-95, 1995.

[14] Novak, S. Generalized kernel density estimator. Theory of Probability & Its Applications, vol. 44, no. 3, pp. 570-583, 2000.

[15] Peng, Y & Flach P. A. Soft Discretization to Enhance the Continuous Decision Tree Induction. *Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, Christophe Giraud-Carrier, Nada Lavrac and Steve Moyle, editors, pp. 109-118. ECML/PKDD'01 workshop notes, Septiembre 2001.

[16] Quinlan, J. R. Learning with Continuous Classes. Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343-348, 1992.

[17] Setiono, R. & Leow, W. K. FERNN: An Algorithm form Fast Extraction of Rules from Neural Networks Applied Intelligence, vol.12, no.1-2, pp.15-25, Enero - Abril 2000.

[18] Thrun, S. Extracting Symbolic Knowledge from Artificial Neural Networks. Advances in Neural Information Processing Systems, vol. 7, 1995.

[19] Towel, G. G., Craven M. W. & Shavlik, J. W. Constructive Induction in Knowledge-Based Neural Networks. Proceedings of the Eighth International Machine Learning Workshop, Morgan Kaufmann, 1991.

[20] Witten, I. H. & Frank E. *WEKA. Machine Learning Algorithms in Java*. Morgan Kaufmann, 2000.