Influencia de la elección de operadores lógicos en la comprensibilidad del conocimiento inducido

Pedro G. Espejo, César Hervás, Sebastián Ventura
Dept. de Informática y Análisis Numérico
Universidad de Córdoba
14071 Córdoba
{pgonzalez, chervas, sventura}@uco.es

Resumen

Son varias las características que determinan la calidad del conocimiento obtenido en el proceso de minería de datos. De estas características, a la que más atención se ha dedicado tradicionalmente ha sido la precisión, relegándose a un segundo plano la comprensibilidad. En este trabajo desarrollamos un sistema de minería de datos orientado a la tarea de clasificación, utilizando reglas como formalismo de representación. El objetivo principal es analizar el balance entre precisión y comprensibilidad, centrándonos en un aspecto de la comprensibilidad poco tratado hasta la fecha: el que viene determinado por la elección de los operadores lógicos que pueden aparecer en el antecedente de las reglas. El sistema de minería desarrollado se basa en la programación genética gramatical, ya que otro objetivo de nuestro trabajo es estudiar la utilidad de esta técnica evolutiva para llevar a cabo tareas de minería.

El trabajo desarrollado nos ha llevado a conclusiones de interés para entender mejor la comprensibilidad y que pueden facilitar el diseño de sistemas de minería de datos en los que la interpretabilidad del conocimiento obtenido sea importante. Además, hemos comprobado que la programación genética gramatical puede ser una técnica provechosa para llevar a cabo tareas de minería, ya que resulta competitiva en términos de precisión y además el uso de gramáticas permite controlar de manera flexible factores que determinan la comprensibilidad de los resultados.

1. Introducción

Las tres características que determinan la calidad del conocimiento obtenido en el proceso de minería de datos (MD) son la precisión, la comprensibilidad y el interés [7]. En la mayoría de los trabajos de MD publicados hasta la fecha se ha prestado una atención mayoritaria a la precisión, relegándose a un plano muy secundario las otras dos características. No obstante, hay algunos trabajos que tratan específicamente sobre la comprensibilidad [1][9][12] y otros en los que ésta es tenida en cuenta junto con la precisión [8][11][3][5][14]. Además, la inmensa mayoría de estos trabajos sólo tienen en cuenta uno de los diversos factores que afectan a la comprensibilidad: la complejidad sintáctica del modelo obtenido, es decir, su tamaño. Son todavía muchas las cosas que necesitamos aprender acerca de la comprensibilidad y su importancia. Es en este sentido en el que se orienta nuestro trabajo.

Nuestro interés por la comprensibilidad se centra en un aspecto relativo a la representación del conocimiento. El formalismo de representación que utilizamos para el conocimiento son las reglas. No obstante, hay que tener en cuenta que la capacidad expresiva de las reglas depende de los operadores que se permita que aparezcan en los antecedentes de las mismas y la manera en que puedan combinarse. El antecedente de una regla puede ser una condición compuesta por varias condiciones elementales conectadas mediante operadores lógicos. Este es el punto sobre el cual enfocamos nuestro trabajo. Si en el antecedente de una regla pueden combinarse de manera arbitraria condiciones elementales mediante los operadores lógicos AND (Y) y OR (O), será posible formar condiciones más complejas que si sólo se permite la utilización de uno de estos operadores (o bien AND o bien OR). Una premisa de la que se parte es que una regla en cuyo antecedente intervengan ambos operadores puede ser mucho más difícil de interpretar por una persona que otra en la que

intervenga sólo uno, como podemos comprobar en al caso de las siguientes reglas:

R1: SI (x2>50 O x3>0.1) Y (x1>10 O (x3>0.5 Y x4>2)) ENTONCES c1

R2: SI x2>50 O x3>0.1 O x1>10 O x3>0.5 O x4>2 ENTONCES c2

Aunque las reglas R1 y R2 tienen las mismas condiciones elementales, la segunda es mucho más fácil de entender por una persona, y esto es debido a que en su antecedente aparece un único operador lógico, mientras que en la primera intervienen dos. Por otra parte, la utilización de ambos operadores en el antecedente permite que éstos sean más complejos, y pueden expresarse condiciones que no pueden expresarse con uno solo.

La hipótesis que se plantea en este trabajo es que la mayor capacidad expresiva ofrecida por el uso de dos operadores lógicos permitirá obtener clasificadores más precisos que aquellos que sólo empleen un operador lógico. El objetivo del presente trabajo es estudiar el balance entre precisión y comprensibilidad y la influencia que tiene en el mismo la presencia de uno o dos operadores lógicos en el antecedente de las reglas.

La metodología de trabajo consiste en resolver varios problemas de clasificación de dominio público utilizando dos variantes de clasificadores basados en reglas, una en la que se permite la aparición de los dos operadores lógicos en el antecedente de cada regla y otra en la que sólo se permite un operador lógico por regla. Una vez obtenidos los resultados, para cada problema se compara la precisión obtenida mediante cada variante, para comprobar si la utilización de dos operadores lógicos resulta en la obtención de reglas más complejas pero también más precisas. No obstante, en el caso de que así fuera, habría que ponderar si la diferencia en precisión compensa la diferencia en comprensibilidad, lo que en todo caso dependerá de las características del problema y de las preferencias del usuario.

El algoritmo de búsqueda que empleamos para construir los clasificadores es un algoritmo evolutivo, concretamente una variante de la programación genética conocida como programación genética gramatical (PGG). En PGG se define una gramática, garantizándose que todos los individuos de la población que va evolucionando a lo largo del proceso evolutivo son legales con respecto a la gramática. Además, la gramática permite sesgar el proceso evolutivo, forzando que los individuos tengan ciertas características. Nosotros empleamos esta capacidad para decidir qué operadores lógicos pueden aparecer en los antecedentes de las reglas, pero puede emplearse para determinar otras características relativas a la comprensibilidad, y también puede ser beneficioso desde el punto de vista de la eficiencia, al restringir el espacio de búsqueda. Así, con este trabajo profundizamos también en la utilización de la PGG en tareas de minería. Si bien los algoritmos evolutivos (AE) están siendo empleados cada vez más en MD [7], la mayoría de los trabajos se centran en el uso de algoritmos genéticos (AG), siendo muchos menos los que se inclinan por la PG, y aún menos los que utilizan PGG (ver [16], no obstante).

El resto de este trabajo se estructura de la siguiente manera. Las secciones 2 y 3 se dedican a describir brevemente la comprensibilidad y la PGG respectivamente, que son las áreas en las que se basa nuestro trabajo. En la sección 4 se presenta de manera detallada el sistema de MD basado en PGG que hemos implementado. En la sección 5 se describen los experimentos realizados y se analizan los resultados obtenidos. Finalmente, en la sección 6 se recogen las conclusiones obtenidas y se plantean futuras líneas de trabajo.

2. Comprensibilidad

La definición más extendida de MD es la que se da en [6]: "El proceso no trivial de identificar en unos datos patrones válidos, novedosos, potencialmente útiles y comprensibles". Esta definición ha llevado a distinguir tres factores que hoy en día son ampliamente aceptados en la comunidad de MD como los criterios para medir la calidad del conocimiento obtenido por el proceso de MD: precisión, comprensibilidad e interés [7]. Sin embargo, a pesar de la importancia que parece atribuirse a la comprensibilidad en todo momento y que se propugna en todos los textos generales acerca de MD, la realidad es bien distinta. En los trabajos que se han publicado hasta la fecha, los esfuerzos se centran de manera inmensamente predominante en la precisión de los resultados, siendo muy pocos los trabajos dedicados hasta el momento a la comprensibilidad (y al interés). Además de escasos, los trabajos relativos a la comprensibilidad en MD suelen ser

bastante limitados y faltos de un enfoque estructurado, lo cual es normal en un área en la que apenas se están dando los primeros pasos.

El de comprensibilidad es un concepto tremendamente esquivo, difícil de definir y subjetivo, que depende del área y contexto en que se utilice, que no puede ser medido de manera precisa, para el que no existe un marco común de referencia, que depende de diversos factores, algunos de los cuales seguramente están aún por comprender y definir. Los dos factores referentes a la comprensibilidad más citados en la bibliografía son la representación del conocimiento y la complejidad sintáctica.

De entre los diversos formalismos de representación del conocimiento disponibles, hay algunos que han sido tradicionalmente proscritos del terreno de juego de la MD, particularmente los modelos estadísticos y los basados en redes neuronales. Este rechazo se debe precisamente a la comprensibilidad, o más bien a la falta de la misma, ya que han sido considerados como mecanismos de caja negra que pueden lograr una elevada precisión pero que resultan difíciles de entender para el ser humano. Los formalismos preferidos por su comprensibilidad han sido tradicionalmente los árboles de decisión y las reglas. No puede afirmarse tajantemente que un formalismo de representación sea más comprensible que otro, ya que esto depende del problema en cuestión y de las preferencias del usuario. No obstante, por regla general suele considerarse que las reglas resultan más comprensibles que los árboles de decisión.

Además del formalismo de representación propiamente dicho, hay otras cuestiones relativas a la representación del conocimiento que influyen en la comprensibilidad, como por ejemplo: los tipos de datos presentes, la discretización de los atributos continuos o los operadores aritméticos, relacionales y lógicos que pueden emplearse. En este último punto es en el que enfocamos el presente trabajo.

En la inmensa mayoría de la bibliografía en la que hay una preocupación por la comprensibilidad, ésta se reduce a tener en cuenta el tamaño de los modelos obtenidos, pero hay otros aspectos relativos a la complejidad. En [12] se distinguen dos tipos de medidas de complejidad:

- Medidas sintácticas. Son medidas intensionales, que no tienen en cuenta los ejemplos usados para construir el modelo o los ejemplos a los que se aplica el modelo una vez construido, como por ejemplo el tamaño medio de regla o el número medio de variables por regla.
- Medidas semánticas. Son medidas extensionales, que sí tienen en cuenta los ejemplos usados para construir el modelo o los ejemplos a los que se aplica el modelo una vez construido, como por ejemplo la proporción ejemplos/reglas o el índice de redundancia.

3. Programación genética gramatical

El paradigma de los algoritmos evolutivos (AE) consiste en la utilización de algoritmos de búsqueda estocástica que se basan en la abstracción de ciertos procesos de la teoría de la evolución darwiniana [13]. Bajo la denominación genérica de AE se agrupa un cierto número de técnicas diferenciadas. Los cuatro tipos de AE que más atención han recibido tradicionalmente son: programación evolutiva, estrategias de evolución, algoritmos genéticos y programación genética. Las características básicas de los AE son:

- Trabajan con una población de individuos (soluciones candidatas) a la vez, en lugar de con una única solución candidata.
- Generan nuevos individuos por medio de un mecanismo de herencia. Los descendientes son generados aplicando operadores genéticos a los individuos de la generación actual. Los dos operadores fundamentales y más habitualmente usados son el cruce (recombinación) y la mutación. El cruce intercambia parte del material genético entre varios individuos (normalmente dos), mientras que la mutación cambia de manera aleatoria el valor de una pequeña parte del material genético de un individuo.
- Utilizan un método de selección sesgado en base al ajuste del individuo, es decir, en una medida de calidad que indica cómo de buena es una solución candidata. Así, cuanto mejor es el ajuste de un individuo, más probable es que sea seleccionado en el proceso evolutivo, y por lo tanto más probable será que su

material genético pase a las posteriores generaciones de individuos.

La programación genética (PG) [2] es básicamente una variante de los algoritmos genéticos (AG) en la que se hace uso un lenguaje de representación complejo para codificar los individuos de la población. El tipo de representación utilizado más frecuentemente son los árboles, aunque existen otras posibilidades [2]. El objetivo inicial de la PG, como su nombre indica, era el de evolucionar programas de ordenador. No obstante, esta técnica se emplea hoy día para evolucionar otras abstracciones de conocimiento, como por ejemplo expresiones matemáticas o sistemas basados en reglas. Aunque la principal diferencia entre los AG y la PG radica en el tipo de representación usada, existen una serie de factores y problemas propios de la PG que permiten distinguir claramente estos dos tipos de AE.

La diferencia más importante es que en la PG los individuos representan programas de ordenador (aunque las interpretaciones de lo que es un programa de ordenador pueden ser muy variadas), por lo que los individuos consisten no sólo en estructuras de datos, sino también en operaciones (operadores y funciones). Así, en un individuo árbol, las hojas corresponden con los símbolos terminales (variables y constantes), mientras que los nodos internos se corresponden con los no terminales (operadores y funciones). En general, el conjunto de no terminales debe cumplir dos propiedades: suficiencia y clausura. La suficiencia hace referencia al hecho de que el poder expresivo de los conjuntos de terminales y no terminales debe bastar para poder representar una solución para el problema en cuestión. La clausura se refiere a que una función u operador debería ser capaz de aceptar como entrada cualquier salida producida por cualquier función u operador del conjunto de no terminales. En la práctica, no es raro encontrarse con situaciones en las que los programas a evolucionar tengan que manejar variables de distinto tipo, lo cual hace que se difícil de satisfacer la propiedad de clausura.

Una de las vías recientemente propuestas para solventar el problema de la clausura se basa en la utilización de gramáticas para especificar un lenguaje al cual se atendrán los individuos de la población, lo cual da lugar a la programación genética gramatical (PGG) [15]. El uso de una gramática debe ir acompañado de unos operadores de mutación y cruce que se apoyen en la misma, de manera que se garantice que todo individuo de la población, tanto los que forman la población inicial como los que se van generando a lo largo del proceso evolutivo, son legales con respecto a la gramática.

4. Descripción del sistema

Se ha desarrollado un sistema de PGG destinado a evolucionar clasificadores basados en reglas. Para poder analizar la influencia que tiene la utilización de uno o dos operadores lógicos en el antecedente de las reglas se han construido dos tipos de clasificadores: uno en el que pueden aparecer dos operadores lógicos en el antecedente de cada regla y otro en el que sólo puede aparecer uno en cada antecedente (aunque en antecedentes distintos pueden usarse operadores diferentes). Se han utilizado gramáticas para especificar qué operadores lógicos se permiten en cada regla. Las gramáticas son también de utilidad para indicar qué operadores relacionales son adecuados según el tipo de datos de cada atributo. En los siguientes apartados se describen los detalles de nuestro sistema.

4.1. Representación de individuos

En nuestro sistema evolutivo, cada individuo es un clasificador completo, es decir, un conjunto de reglas, por lo que puede decirse que seguimos el enfoque Pittsburgh. El número de reglas es fijo, una regla por cada clase. El antecedente de cada regla está formado por varias cláusulas o condiciones elementales que se combinan entre sí mediante los operadores lógicos (opl), de la forma

$cláusula_1$ opl $cláusula_2$ opl ... opl $cláusula_n$

Cada cláusula a su vez corresponde al siguiente formato

var opr *const*

Es decir, que se compara el valor de una variable o atributo (*var*) con el de una constante (*const*) mediante un operador relacional (opr). Los operadores relacionales permitidos dependen del tipo de dato del atributo. Para los atributos numéricos hemos empleado los operadores '>' y '<=', y para los atributos categóricos hemos empleado los operadores '=' y '<>'.

4.2. Dinámica evolutiva

Se emplea un mecanismo de selección por torneo que funciona de la siguiente manera: se selecciona un conjunto de individuos de la población de manera aleatoria, entonces se calcula el grado de ajuste de cada individuo. Una vez hecho esto, los individuos se ordenan de acuerdo a su grado de ajuste. Los mejores individuos tienen la oportunidad de reproducirse y sustituir a los peores. La dinámica a la que da lugar este tipo de torneos es algo diferente a la que se emplea habitualmente, ya que utilizamos torneos en los que participa un número relativamente grande de individuos (50, en los experimentos realizados). El criterio de parada es alcanzar un determinado número de torneos.

Se tiene en cuenta la edad de los individuos. La edad es el número de torneos en que ha participado un individuo. Uno de los parámetros del sistema es la edad máxima. En cada torneo, los individuos viejos (aquellos cuya edad es mayor que la edad máxima) son apartados en un conjunto llamado "viejos". Los individuos jóvenes son ordenados según su grado de ajuste, y se forman cuádruplas de individuos (familias), cada una con dos individuos jóvenes y dos viejos. Los dos individuos jóvenes serán sometidos a las operaciones genéticas que se describen más abajo y los hijos a los que den lugar sustituirán a los dos viejos.

Los dos padres se cruzan, y a continuación ambos hijos se mutan. No obstante, si los dos padres tienen el mismo ajuste, no se cruzan, y uno de ellos se muta.

El número de cruces que tienen lugar por cada evento de reproducción es variable, y depende del número de nodos de los padres. Así, la probabilidad de cruce indica la probabilidad de cruce por nodo, en lugar de la probabilidad de que dos individuos se crucen o no. El cruce consiste, como es habitual, en intercambiar los subárboles de los dos padres.

En lo que se refiere a la mutación, también se llevan a cabo varias mutaciones por individuo, en un número directamente proporcional al número de nodos. Hay varios operadores de mutación disponibles. La aplicación de uno u otro se decide aleatoriamente en base a probabilidades. Los operadores de mutación utilizados son:

- Mutación puntual: se selecciona un nodo, ya sea interno u hoja, y se sustituye por un símbolo no terminal (una función u operador en nuestro caso) de la misma aridad o por un símbolo terminal (una variable o una constante).
- Macromutación: a su vez hay varios tipos: reemplazar subárbol; copiar subárbol; intercambiar subárboles; insertar nodo interno; borrar nodo interno.

4.3. Función de ajuste

El criterio empleado para juzgar el grado de ajuste de cada individuo es el porcentaje de aciertos de clasificación, definido como

$$PA = \frac{\#aciertos}{\#ejemplos} \qquad (1)$$

donde *#aciertos* es el número de ejemplos correctamente clasificados, y *#ejemplos* es el número de ejemplos clasificados.

4.4. Gramáticas

En nuestro sistema hacemos uso de gramáticas libres de contexto. Una gramática libre de contexto se especifica en forma de cuádrupla (N, Σ, P, S), donde N es el alfabeto de símbolos no terminales, Σ es el alfabeto de símbolos terminales, P es el conjunto de producciones y S es el símbolo inicial. Las producciones tienen la forma $x \to y$, donde $x \in N$, $y \in \{\Sigma \cup N\}^*$.

A continuación se muestran las gramáticas que se han utilizado para uno de los problemas a los que hemos aplicado nuestro sistema (ver apartado 5.1). Se trata de la conocida base de datos Iris en la que se distinguen tres clases: iris-setosa, iris-versicolor e iris-virginica. Además de la clase, existen cuatro atributos, todos de tipo numérico. Para todas las gramáticas correspondientes a este problema, el conjunto de terminales es el mismo:

$$\Sigma = \left\{ \begin{matrix} x_1, x_2, x_3, x_4, \\ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \\ \text{verdadero}, \text{falso} \end{matrix} \right\}$$

Para evolucionar clasificadores en los que en cada regla intervenga un único operador lógico, pero de manera que distintas reglas puedan utilizar

diferentes operadores lógicos, necesitaríamos una gramática como la siguiente:

$N = \{\text{COND}, \text{CONDA}, \text{CONDO}, \text{VAR}, \text{NUM}\}$

S → si COND entonces 'Iris-setosa'
si COND entonces 'Iris-versicolor'
si COND entonces 'Iris-virginica'
COND → CONDA | CONDO
CONDA → VAR > NUM |
VAR <= NUM |
COND y COND |
verdadero | falso
CONDO → VAR > NUM |
VAR <= NUM |
COND o COND |
verdadero | falso
VAR → x_1 | x_2 | x_3 | x_4
NUM → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

Para obtener clasificadores en los que puedan aparecer ambos operadores lógicos en el antecedente de cada regla usaríamos la siguiente gramática:

$N = \{\text{COND}, \text{VAR}, \text{NUM}\}$

S → si COND entonces 'Iris-setosa'
si COND entonces 'Iris-versicolor'
si COND entonces 'Iris-virginica'
COND → VAR > NUM |
VAR <= NUM |
(COND y COND) |
(COND o COND) |
verdadero | falso
VAR → x_1 | x_2 | x_3 | x_4
NUM → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

Tal como puede apreciarse en las gramáticas que se muestran, las reglas resultantes del proceso evolutivo pueden hacer referencia en sus antecedentes a cualquier número de variables, por lo que nuestro sistema lleva a cabo de manera implícita una selección de características, es decir, que elige de entre las variables disponibles las que son relevantes para obtener unas soluciones de la mejor calidad posible.

5. Experimentos

Hemos aplicado nuestro sistema a 29 conjuntos de datos de dominio público. Para cada uno hemos probado una gramática que permite un único operador lógico por regla (llamamos a este tipo de gramática *1r*) y otra que permite dos operadores lógicos (gramática *2*). El planteamiento de los experimentos consiste en estimar la precisión obtenida por cada una de las dos variantes para cada conjunto de datos. Esto permitirá comprobar si se cumple la hipótesis planteada en la sección 1, es decir, si al permitir el uso de dos operadores lógicos pueden alcanzarse mayores tasas de acierto que con uno solo.

5.1. Conjuntos de datos

Se han llevado a cabo experimentos sobre 29 conjuntos de datos de dominio público, procedentes todos ellos del repositorio para aprendizaje automático de la UCI [10], salvo la llamada Smoking, que procede de [4]. Estos conjuntos de datos presentan una gran diversidad respecto a distintas características, como número de ejemplos, número de clases, número de atributos y tipos de datos. En la tabla 1 se detallan estas características.

Algunos de estos conjuntos de datos presentan valores ausentes. En estos casos hemos optado por llevar a cabo un pequeño paso de preparación de los datos consistente en sustituir cada valor nulo por el valor más frecuente del dominio sobre el cual se encuentra definido el atributo correspondiente.

Cada uno de los conjuntos de datos se ha dividido en dos subconjuntos, uno de entrenamiento y otro de prueba. Se ha hecho una partición estratificada destinando un 75% de ejemplos para entrenamiento y un 25% para prueba, salvo para los conjuntos de datos Monks y Thyroid, que se distribuyen ya particionados y para los que se ha respetado dicha partición original.

5.2. Parámetros PGG

Por regla general, el conjunto de símbolos terminales está formado por los nombres de los atributos, un cierto número de valores tomados de los respectivos dominios sobre los que se definen los atributos y los valores lógicos verdadero y falso. El conjunto de funciones está formado por los operadores relacionales y lógicos. Los valores empleados para el resto de parámetros que controlan la ejecución de la PGG son los que se indican en la tabla 2.

Nombre	Nº ejemplos	Nº clases	Nº atributos	Nº at. numéricos	Nº at. categóricos
Australian	690	2	14	6	8
Breast-wisconsin	683	2	9	9	0
Cmc	1473	3	9	2	7
Credit-screening	690	2	15	6	9
German-original	1000	2	20	7	13
Haberman	306	2	3	3	0
Heart-cleveland	303	5	13	6	7
Heart-statlog	270	2	13	7	6
Ionosphere	351	2	34	34	0
Iris	150	3	4	4	0
Lenses	24	3	4	0	4
Lymphography	148	4	18	0	18
Monks1	432	2	6	0	6
Monks2	432	2	6	0	6
Monks3	432	2	6	0	6
Nursery	12960	5	8	0	8
Page-blocks	5473	5	10	10	0
Postoperative	90	3	8	1	7
Primary-tumor	339	21	17	0	17
Smoking	2855	3	13	3	10
Sonar	208	2	60	60	0
Spambase	4601	2	57	57	0
Thyroid	7200	3	21	6	15
Tic-tac-toe	958	2	9	0	9
Voting-records	435	2	16	0	16
Waveform-21	5000	3	21	21	0
Waveform-40	5000	3	40	40	0
Yeast	1484	10	8	7	1
Zoo	101	7	16	1	15

Tabla 1. Conjuntos de datos usados

Parámetro	Valor
Tamaño población	500
Nº torneos	1000
Tamaño torneo	50
Edad máxima	6
Criterio parada	Nº torneos
Prob. cruce por nodo	1/75
Prob. mutación por nodo	1/100

Tabla 2. Parámetros PGG

5.3. Resultados

Por cada conjunto de datos y cada una de las dos gramáticas se han hecho 30 ejecuciones utilizando la misma partición entrenamiento/prueba, para tener en cuenta la naturaleza aleatoria de la PGG. La tasa de aciertos se estima obteniendo la media del porcentaje de aciertos sobre los datos de prueba de las 30 ejecuciones. En la tabla 3 se muestran estos resultados medios. También se muestra la tasa de aciertos obtenida por el algoritmo C4.5 usando la misma partición de entrenamiento/prueba. Estos resultados se han obtenido haciendo una única ejecución, puesto que el algoritmo C4.5 es determinístico, y se proporcionan para ofrecer una aproximación al rendimiento relativo que puede ofrecer respecto a la tasa de aciertos la PGG, pero no se hace una comparación desde un punto de vista estadístico. Estos datos aportan cierta evidencia de que

nuestro sistema de PGG permite obtener tasas de acierto similares a las de C4.5.

Conjunto de datos	C4.5	PGG-1r	PGG-2
Australian	90,17	89,17	88,92
Breast-wisconsin	93,56	94,46	94,56
Cmc	51,89	49,05	49,36
Credit-screening	89,59	83,72	88,17*
German-original	72,4	70,08	70,25
Haberman	76,92	73,80	73,50
Heart-cleveland	50,64	54,94	54,98
Heart-statlog	77,94	77,25	79,66*
Ionosphere	88,76	81,61	85,43*
Iris	94,44	93,80	93,98
Lenses	85,71	92,38	90,00
Lymphography	76,92	80,00	79,74
Monks1	75,69	71,39	75,97*
Monks2	65,04	65,28	65,78*
Monks3	97,22	88,89	95,71*
Nursery	48,47	62,17	67,29*
Page-blocks	97,37	93,91	93,98
Postoperative	62,5	60,97	62,22
Primary-tumor	27,95	26,70	25,91
Smoking	64,61	69,27	69,19
Sonar	50,94	60,82	60,63
Spambase	91,48	84,49	86,81*
Thyroid	99,38	97,01	97,35
Tic-tac-toe	74,16	70,61	71,35
Voting-records	96,33	96,33	96,33
Waveform-21	77,21	67,97	69,89*
Waveform-40	75,29	67,10	68,37*
Yeast	49,19	42,34	43,38
Zoo	89,28	82,26	82,02

Tabla 3. Porcentaje de aciertos sobre los datos de prueba (%)

Nuestro objetivo es tratar de comprobar si al permitir una mayor capacidad expresiva mediante el uso de dos operadores lógicos se logra una mayor precisión que con uno solo. Por lo tanto, se ha comparado la tasa de aciertos estimada entre las gramáticas 1r y 2 para cada uno de los conjuntos de datos. Para hacer esta comparación se ha utilizado una prueba t emparejada. En la tabla 3 se utiliza un asterisco ('*') para indicar que una de las gramáticas es mejor que la otra a un nivel de significación de 0,05.

En contra de lo que cabría esperar en un principio, el uso de dos operadores lógicos no mejora por regla general la precisión. En sólo un tercio (un 34,5%) de los conjuntos de datos analizados, el uso de una gramática de tipo 2 ofrece una mejora estadísticamente significativa con respecto al uso de una gramática de tipo 1r. Además, en estos diez casos, aunque la diferencia es significativa, no parece excesivamente grande. La mayor diferencia corresponde a Monks3, con 6,82 puntos; la diferencia mínima corresponde a Monks2, con 0,5 puntos y la diferencia media es de 3,32 puntos.

A continuación se presentan como ejemplo algunos de los clasificadores obtenidos, para poder visualizar el nivel de comprensibilidad que puede introducir cada tipo de gramática. Se han seleccionado uno de los conjuntos de datos para los que no hay diferencia significativa en la tasa de aciertos (Australian) y el que presenta una diferencia significativa mayor (Monks3). Hay que tener en cuenta que estos clasificadores son sólo una muestra ilustrativa. Por cada conjunto de datos y cada gramática se ha elegido, de las 30 ejecuciones realizadas, el clasificador que mejor tasa de aciertos presenta sobre los datos de prueba.

Para el conjunto de datos Australian, el mejor clasificador obtenido con la gramática 1r es:

```
SI (x11=1 O x10>0.01 O x9<>1)
    ENTONCES clase=1
SI (x5=1 O x8<>1)
    ENTONCES clase=0
```

Mientras que el mejor clasificador obtenido con la gramática 2 es:

```
SI (x4<>5)
    ENTONCES clase=1
SI ((((x9<>1 O x8=0) Y ((x6=5 O x8<>1) O x8<>1)) Y (x8<>0 O x4<>3)))
    ENTONCES clase=0
```

Para el conjunto de datos Monks3, el mejor clasificador obtenido con la gramática 1r es:

```
SI (x3<>3 O x6=1)
    ENTONCES clase=0
SI (x2<>3 Y x5<>4)
    ENTONCES clase=1
```

Mientras que el mejor clasificador obtenido con la gramática 2 es:

```
SI (((x2<>3 O x5<>2) O (x5<>2 O (x5<>4 Y x6<>4))))
    ENTONCES clase=0
SI ((x5<>4 Y (((x4<>2 Y ((x4=1 Y x5=3) Y x5<>2)) Y (x4=1 O x6<>3)) O x2<>3)))
    ENTONCES clase=1
```

En estos casos de ejemplo se puede apreciar que los clasificadores formados utilizando un sólo operador lógico por regla pueden ser más fáciles de entender por una persona, frente a los que utilizan dos operadores lógicos. No obstante, la decisión de qué tipo de clasificador es más adecuado (con uno o dos operadores lógicos) depende del problema en cuestión y de las preferencias del usuario.

6. Conclusiones y trabajo futuro

La conclusión más importante que podemos obtener de los experimentos realizados es que la mayor capacidad expresiva que ofrece el permitir la utilización de dos operadores lógicos en el antecedente de las reglas frente al uso de un único operador lógico por regla no ofrece, por regla general, un incremento significativo de la tasa de aciertos. Por lo tanto, en situaciones en las que haya un interés explícito por la comprensibilidad de los clasificadores, puede ser conveniente sesgar el algoritmo utilizado (en este caso la PGG) para formar clasificadores en los que se permita un único operador lógico por regla, ya que de esta manera posiblemente logremos obtener clasificadores tan precisos como los que conseguiríamos utilizando dos operadores lógicos, y que además sean más fáciles de interpretar.

Sin embargo, esta es una recomendación de carácter genérico. Como ya hemos visto, el planteamiento a adoptar a la hora de buscar el clasificador más adecuado para un problema determinado depende de las características de éste y de las preferencias del usuario. Asimismo, hemos visto cómo en la mayoría de los casos no se logra una mejora de la precisión al permitir dos operadores lógicos, pero hay algunos casos en que ocurre lo contrario. Hay problemas en los que se obtiene una tasa de aciertos más elevada con dos operadores lógicos. En casos como éstos habría que decidir si la diferencia en precisión compensa o no la pérdida de comprensibilidad. El método ideal a la hora de afrontar un problema real de clasificación sería generar un clasificador (o varios) con cada una de las dos gramáticas (uno o dos operadores lógicos), y elegir el que se considere más adecuado teniendo en cuenta tanto la tasa de aciertos como la comprensibilidad. No obstante, si por falta de recursos o por algún otro motivo no es posible llevar a cabo este sondeo previo y hay un interés marcado por la comprensibilidad, lo más recomendable sería optar por generar clasificadores limitados a un único operador lógico por regla.

En lo que se refiere a la utilidad de la PGG como algoritmo de búsqueda para inducir clasificadores, la comparación con los resultados producidos por C4.5 apuntan a que la técnica evolutiva puede resultar competitiva en lo que se refiere a la tasa de aciertos. De hecho, para algunos de los conjuntos de datos probados, el sistema basado en PGG ha obtenido resultados mejores que C4.5. Pero la principal ventaja del uso de la PGG a la hora de afrontar problemas de clasificación es la que se deriva del uso de una gramática, ya que mediante ésta podemos sesgar el proceso de búsqueda y definir fácilmente el tipo de clasificador que queremos generar. Nosotros hemos utilizado esta facilidad para determinar si se permiten uno o dos operadores lógicos por regla. El principal inconveniente de la PGG frente a C4.5 es el tiempo de ejecución. Aunque no presentamos las medidas del tiempo de CPU consumido en nuestras ejecuciones, basta decir que los tiempos de ejecución son del orden de los segundos para C4.5 y del orden de los minutos para la PGG.

En este trabajo hemos arrojado algo de luz sobre uno de los diversos factores que influyen en la comprensibilidad: la manera de utilizar los operadores lógicos para expresar el conocimiento inducido. Este es sólo un primer paso en un largo camino por explorar, ya que la comprensibilidad es un área que sigue sin recibir demasiada atención en el campo de la MD. Para seguir avanzando en este sentido, nuestros objetivos inmediatos se centran en ampliar el tipo de estudio planteado en este trabajo, teniendo en cuenta el tamaño de los clasificadores obtenidos, además del porcentaje de aciertos. También creemos que es muy importante realizar estudios similares destinados a entender otros factores que también influyen en la comprensibilidad.

La PGG se muestra como una técnica flexible y poderosa para llevar a cabo este tipo de tareas. Es interesante por dos motivos: en primer lugar, apunta a que puede lograr resultados de una calidad similar o incluso superior a otros algoritmos; en segundo lugar, gracias al uso de gramáticas resulta muy cómoda para plantear estudios comparativos como los que pretendemos

llevar a cabo. Puesto que la PGG prácticamente no se ha aplicado antes a tareas de MD, esperamos poder estudiar su adecuación para tales fines a la vez que logramos entender mejor qué es la comprensibilidad y cómo podemos ajustar nuestros métodos de minería para obtener modelos fáciles de interpretar por las personas, sin que esto afecte negativamente a otros parámetros de calidad, como por ejemplo la precisión.

Agradecimientos

Este trabajo ha sido financiado por el MCYT a través del proyecto TIC2002-04036-C05-02 y de fondos FEDER.

Referencias

[1] Askira-Gelman, I. "Knowledge discovery: comprehensibility of the results". *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences*, vol. 5, pp. 247-255. IEEE, Enero 1998.

[2] Banzhaf, W.; Nordin, P.; Keller, R. E.; Francone, F. D. *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann / dpunkt.verlag, 1998.

[3] Bohanec, M.; Bratko, I. "Trading accuracy for simplicity in decision trees". *Machine Learning*, vol. 15, no. 3, pp. 223-250, Junio 1994.

[4] Bull, S. "Analysis of attitudes toward workplace smoking restrictions". *Case Studies in Biometry*, editado por Lange, N.; Ryan, L.; Billard, L.; Brillinger, D.; Conquest, L.; Greenhouse, J. pp. 249-271. John Wiley & Sons, 1994.

[5] Endou, T.; Zhao, Q. "Generation of comprehensible decision trees through evolution of training data". *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1221 -1225. IEEE, Mayo 2002.

[6] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P. "From data mining to knowledge discovery in databases". *AI Magazine*, vol. 17, no. 3, pp. 37-54, 1996.

[7] Freitas, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[8] Holte, R. C. "Very simple classification rules perform well on most commonly used datasets". *Machine Learning*, vol. 11, no. 1, pp. 63-91, Abril 1993.

[9] Kodratoff, Y. "The comprehensibility manifesto". *KDD Nugget Newsletter*, vol. 94, no. 9, Mayo 1994.

[10] Murphy, P. M.; Aha, D. W. *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California, Irvine, California, 1994. [http://www.ics.uci.edu/~mlearn/MLRepository.html]

[11] Murphy, P. M.; Pazzani, M. J. "Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction". *Journal of Artificial Intelligence Research*, vol. 1, pp. 257-275, 1994.

[12] Sommer, E. "An approach to quantifying the quality of induced theories". *Proceedings of the IJCAI 95 - Workshop on Machine Learning and Comprehensibility*, editado por Nedellec, C. Morgan Kauffman, 1995.

[13] Spears, W. M.; De Jong, K. A.; Bäck, T.; Fogel, D. B.; de Garis, H. "An overview of evolutionary computation". *European Conference on Machine Learning* - Lecture Notes in Computer Science, vol. 667, editado por Brazdil, P. pp. 442-459. Springer, Abril 1993.

[14] Tan, K. C.; Yu, Q.; Heng, C. M.; Lee, T. H. "Evolutionary computing for knowledge discovery in medical diagnosis". *Artificial Intelligence in Medicine*, vol. 27, no. 2, pp. 129-154, 2003.

[15] Whigham, P. A. "Gramatical Bias for Evolutionary Learning". Tesis doctoral. School of Computer Science - University College - University of New South Wales - Australian Defence Force Academy, Octubre 1996.

[16] Wong, M. L.; Leung, K. S. *Data Mining using Grammar-Based Genetic Programming and Applications*. Kluwer Academic Publishers, 2000.