Programación genética gramatical para el descubrimiento de reglas de clasificación

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

Resumen

En este trabajo se analiza la idoneidad de la programación genética gramatical para llevar a cabo tareas de minería de datos, concretamente para clasificación. Esta técnica evolutiva se compara con varios algoritmos clásicos de inducción de árboles de decisión y de reglas. El criterio de comparación se basa en la precisión de la clasificación. Los experimentos realizados nos han permitido comprobar que la programación genética gramatical puede ser una técnica provechosa para llevar a cabo tareas de clasificación, ya que resulta competitiva en términos de precisión y además el uso de gramáticas permite definir con facilidad diferentes características de los clasificadores.

1. Introducción

La minería de datos (MD) [11] se define como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos.

La tarea del descubrimiento de reglas ha sido abordada desde diversos paradigmas: construcción de árboles de decisión, aprendizaje inductivo, aprendizaje basado en instancias y, más recientemente redes neuronales y algoritmos evolutivos [11].

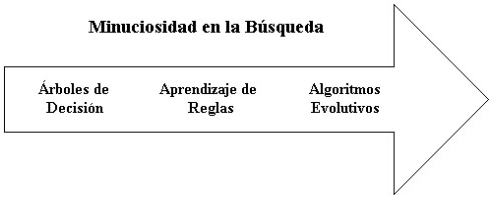

Figura 1. Algoritmos de descubrimiento de reglas.

En la Figura 1 se muestra el espectro de las técnicas de búsqueda en términos de la minuciosidad de la búsqueda que realizan. En primer lugar tenemos los algoritmos de inducción de reglas mediante árboles de decisión, que utilizan heurísticas altamente voraces y realizan una búsqueda irrevocable. Estos algoritmos son muy rápidos y sorprendentemente efectivos para encontrar clasificadores precisos, además de clasificar completamente los datos. La mayoría utilizan técnicas de particionado recursivo que van partiendo el conjunto de datos utilizando heurísticas voraces que pueden pasar por alto relaciones multivariantes que no aparecen si se tratan las variables individuales aisladamente. A continuación se encuentran los algoritmos convencionales de aprendizaje de reglas, que abarcan una amplia variedad cuya característica común es la de ser más minuciosos que los anteriores. Finalmente tenemos los algoritmos evolutivos, que son capaces de conducir muchas búsquedas minuciosas y realizar un retroceso implícito en la búsqueda del espacio de reglas que va a permitir encontrar interacciones complejas que los otros tipos de algoritmos no son capaces de hallar.

<ReglaClasificación> ::= “**SI**” (<antecedente>) **“ENTONCES”** <consecuente>
<antecedente> ::= <condición> |
<condición> “**Y**” <antecedente>
<consecuente> ::= “**PERTENECE A**” *etiqueta clase*
<condición> ::= <atributo> <operador> <valor>
<atributo> ::= *Cada uno de los atributos del conjunto*
<valor> ::= *Valor del dominio correspondiente*
<operador> ::= “=” | “≠” | “<” | “>” | “≤”| “≥”

Figura 2. Formato genérico de una regla de clasificación

Un clasificador puede expresarse como un conjunto de reglas de tipo Si-Entonces, en las que el antecedente de cada regla está formado por una serie de condiciones que debe cumplir un objeto

para que se considere que pertenece a la clase indicada en el consecuente. Un árbol de decisión puede convertirse fácilmente en un conjunto de reglas. En la figura 2 se describe el tipo de reglas que usamos en nuestro trabajo.

El algoritmo evolutivo que analizamos en el presente trabajo es 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. Si bien los algoritmos evolutivos están siendo empleados cada vez más en MD [3], 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 (no obstante, ver [12]).

El resto de este trabajo se estructura de la siguiente manera. La sección 2 se dedica a describir brevemente los algoritmos clásicos de inducción de árboles de decisión y de reglas de clasificación, mientras que la sección 3 presenta la programación genética gramatical. En la sección 4 se presenta de manera detallada el sistema de MD basado en programación genética gramatical 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. Algoritmos para el descubrimiento de reglas de clasificación

En esta sección se presentan los algoritmos clásicos para la inducción de clasificadores, primero los basados en árboles de decisión y a continuación los de inducción de reglas.

2.1. Árboles de decisión

Los sistemas de aprendizaje basados en árboles de decisión son quizás el método más fácil de utilizar y de entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Los árboles de decisión se utilizan desde hace siglos, y son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos, etc. Una de las grandes ventajas de los árboles de decisión es que, en su forma más general, las opciones posibles a partir de una determinada condición son excluyentes. Esto permite analizar una situación y, siguiendo el árbol de decisión apropiadamente, llegar a una sola acción o decisión a tomar.

Estos algoritmos se llaman algoritmos de *partición* o algoritmos de "divide y vencerás". Otra característica importante de los primeros algoritmos de aprendizaje de árboles de decisión es que una vez elegida la partición, esta decisión es irrevocable. Por tanto, uno de los aspectos más importantes en los sistemas de aprendizaje de árboles de decisión es el denominado *criterio de partición*, ya que una mala elección de la partición (especialmente en las partes superiores del árbol) generará un árbol menos óptimo.

El algoritmo C4.5 [8] forma parte de la familia de los árboles de decisión, al igual que su antecesor ID3, con algunas mejoras como permitir trabajar con valores continuos, evitar favorecer a los atributos con muchos valores y un proceso de poda del árbol.

2.2. Inducción de reglas

Este tipo de algoritmos generan directamente un clasificador representado como un conjunto de reglas. Las reglas inducidas cubren regiones de espacios en los datos que se pueden solapar, de forma que una instancia de los datos puede estar cubierta por más de una regla. Existen diversos algoritmos que inducen conjuntos de reglas de tipo Si-Entonces directamente a partir de un conjunto de datos. Algunos de estos algoritmos realizan una búsqueda dirigida en el espacio de posibles hipótesis. La búsqueda dirigida, una generalización del algoritmo "primero el mejor", se caracteriza por mantener una lista limitada de soluciones parciales durante el proceso de exploración del espacio de búsqueda. Algunos ejemplos de algoritmos que utilizan variantes de la búsqueda dirigida en el proceso de construcción de reglas son: la metodología STAR y las listas de decisión.

El algoritmo One-R [5] es un algoritmo sencillo que, sin embargo, constituye una buena base para hacer comparaciones. La idea es hacer reglas que prueban todos los pares atributo-valor y se selecciona el que ocasione el menor número de errores.

El algoritmo AQ [6] puede considerarse como el padre de una familia de algoritmos STAR que cuenta con otros miembros destacados como AQ20 y CN2. Descubre un conjunto de reglas llamado *cover* para cada uno de los valores de la clase, que clasifican correctamente todos los ejemplos. El algoritmo CN2 [2] fue desarrollado con el objetivo de mejorar el algoritmo AQ eliminando su dependencia de ejemplos específicos e incrementando el espacio de búsqueda de reglas para incluir reglas que no produzcan un ajuste exacto.

3. Programación genética gramatical

El paradigma de los algoritmos evolutivos (AE) consiste en la utilización de algoritmos de búsqueda estocástica (probabilística) que se basan en la abstracción de ciertos procesos de la teoría de la evolución darwiniana [9]. 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 estocásticos 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) [1] 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 [1]. 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 del conjunto de 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 [10], lo cual da lugar a la programación genética gramatical (PGG). 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. Además de servir para solucionar el problema de la clausura, la utilización de una gramática permite sesgar el proceso evolutivo, de manera que los individuos tengan ciertas características. También puede ser beneficioso desde el punto de vista de la eficiencia, al restringir el espacio de búsqueda.

4. Descripción del sistema

Se ha desarrollado un sistema de PGG destinado a evolucionar clasificadores basados en reglas. Pero hay que tener en cuenta que existen distintas maneras de interpretar un conjunto de reglas. Cuando se enfrenta un ejemplo al conjunto de reglas para decidir a qué clase pertenece pueden seguirse distintas filosofías. En nuestro trabajo hemos considerado dos:

- Se evalúan las reglas secuencialmente y se elige la primera cuyo antecedente se cumpla, asignando el ejemplo a la clase correspondiente. Es recomendable ordenar las reglas según su consecuente colocando primero las correspondientes a las clases mayoritarias (que más ejemplos tienen), para maximizar la tasa de aciertos.
- Se evalúan todas las reglas, y se cuenta el número de antecedentes satisfechos por cada clase, y se elige la clase que más tenga. Si se produce un empate entre varias clases se elige la mayoritaria.

Se han usado gramáticas para especificar qué operadores relacionales son adecuados según el tipo de datos de cada atributo.

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 variable, pero se obliga a que haya al menos una regla por cada clase. El antecedente de cada regla es de la forma indicada en la figura 2. 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 muestra como ejemplo la gramática que se ha 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.

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

$$\Sigma = \begin{Bmatrix} x_1, x_2, x_3, x_4, \\ 0,1,2,3,4,5,6,7,8,9,10, \\ verdadero, falso \end{Bmatrix}$$

S → REGLA1
REGLA2
REGLA3

REGLA1 →
si COND entonces 'Iris-setosa' |
REGLA1
REGLA1

REGLA2 →
si COND entonces 'Iris-versicolor' |
REGLA2
REGLA2

REGLA3 →
si COND entonces 'Iris-virginica' |
REGLA3
REGLA3

COND → VAR > NUM |
VAR <= NUM |
COND y COND |
verdadero | falso

VAR → x1 | x2 | x3 | x4

NUM → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

El conjunto de símbolos terminales estará 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 operadores relacionales incluidos en el conjunto de funciones dependen de los tipos de datos de los atributos (tal y como se indica en el apartado 4.1). El único operador lógico empleado es AND.

Tal como puede apreciarse en la gramática, las reglas 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.

Nombre	Nº ejemplos	Nº clases	Nº atributos	Nº at. numéricos	Nº at. categóricos
Breast-wisconsin	699	2	9	9	0
Iris	150	3	4	4	0
Lymphography	148	4	18	0	18
Mushroom	8124	2	22	0	22
Pima	768	2	8	8	0
Satimage	6435	6	39	39	0
Tic-tac-toe	958	2	9	0	9

Tabla 1. Conjuntos de datos usados

5. Experimentos

Hemos utilizado 7 conjuntos de datos para comparar el rendimiento de los algoritmos ID3, C4.5, One-R, AQ, CN2 y PGG. ID3 presenta la restricción de que no puede manejar atributos continuos, por lo que no se ha podido aplicar a todos los conjuntos de datos. Para el caso de la PGG se han utilizado los dos esquemas de evaluación del conjunto de reglas descritos en la sección 4, lo que da lugar a dos variantes a las que llamamos PGG-1r (una regla, la primera que cumpla el antecedente) y PGG-vr (varias reglas, se cuenta el número de antecedentes satisfechos por cada clase). El planteamiento de los experimentos consiste en estimar la precisión obtenida por cada una de las dos variantes para cada conjunto de datos.

5.1. Conjuntos de datos

Se han llevado a cabo experimentos sobre 7 conjuntos de datos de dominio público, procedentes todos ellos del repositorio para aprendizaje automático de la UCI [7]. Estos conjuntos de datos presentan una buena 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.

5.2. Parámetros PGG

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

Los conjuntos de terminales y de funciones utilizados son los indicados en el apartado 4.4. Los valores empleados para el resto de parámetros que controlan la ejecución de la PGG son los que se indican en la tabla 2.

5.3. Resultados

Para cada conjunto de datos y cada algoritmo se ha hecho una validación cruzada con 10 particiones. El conjunto de datos se divide en 10 partes de igual tamaño de manera estratificada (respetando el porcentaje de ejemplos por clase con respecto al conjunto de datos original), en base a las cuales se forman 10 particiones de entrenamiento/prueba en las que se dedica un 90% de los ejemplos para entrenamiento y un 10% para prueba. Se utilizan las mismas 10 particiones para todos los algoritmos. Para cada algoritmo, se hace una ejecución por cada partición. La tasa de aciertos se estima obteniendo la media del porcentaje de aciertos sobre los datos de prueba de las 10 ejecuciones. En la tabla 3 se muestran estos resultados medios.

Para saber qué algoritmos son mejores se ha hecho un análisis de la varianza, que permite saber si existen diferencias entre un conjunto de medias, y después se han aplicado unos métodos de comparación múltiple (la prueba de Bonferroni y

Nombre	ID3	C4.5	One-R	AQ	CN2	PGG-1r	PGG-vr
Breast-wisconsin	93,99	94,56	91,56 -	93,57	92,57	94,99	*96,56*+
Iris		*94,00*+	92,00	81,99 -	57,33 -	*94,00*+	*94,00*+
Lymphography		36,67	39,33	44,00	3,33	*76,33**	*76,95**
Mushroom	*100,00*+	*100,00*+	98,52	*100,00*+	95,78 -	94,74 -	98,08
Pima		*74,48*+	72,27	66,67 -	65,61 -	67,96	68,48
Satimage		*85,77**	59,71	81,34	48,20	66,00	72,79
Tic-tac-toe	94,57	84,54 -	69,92 -	*95,62*+	86,54 -	74,21 -	75,66 -

Tabla 3. Resultados

la prueba T2 de Tamhane) para descubrir qué medias son diferentes. Estos análisis se han llevado a cabo sobre las tasas medias de aciertos, lo cual nos permite saber qué algoritmos ofrecen mejores resultados que otros. Todas las pruebas se han hecho con un nivel de significación de 0,05.

En la tabla 3 nos hemos fijado en cuáles son los mejores algoritmos. El algoritmo que ha obtenido mejores resultados para un determinado conjunto de datos se indica en cursiva y con un símbolo '+'. Los algoritmos frente a los cuales es estadísticamente superior se indican con un signo '-'. Hay algunos casos en los que un algoritmo es mejor que todos los demás. En este caso se emplea un símbolo '*' para señalar el mejor. Como puede comprobarse, hay ocasiones en que varios algoritmos quedan en primer lugar. Así pues, en el caso del conjunto de datos Breast-wisconsin, por ejemplo, el mejor algoritmo es PGG-vr, que es mejor estadísticamente que One-R, pero que no se diferencia de los demás algoritmos. En el caso del conjunto de datos Lymphography, los algoritmos PGG-1r y PGG-vr presentan distintas tasas de acierto medias, pero las diferencias no son significativas. Ambos métodos son mejores que todos los demás.

A modo de resumen, podemos decir que en tres casos (Breast-wisconsin, Iris y Lymphography) la PGG es el mejor algoritmo. En otros dos casos (Pima y Satimage) no es significativamente peor que el mejor. Para el conjunto de datos Mushroom, PGG-1r es significativamente peor que el mejor algoritmo, pero PGG-vr no. Sólo en el caso del conjunto de datos Tic-tac-toe la PGG es peor que el mejor de los algoritmos.

Otra información que no queda totalmente reflejada en la tabla y que merece la pena resaltar es la referente a la comparación de los dos métodos de evaluar el conjunto de reglas probados con PGG (1r y vr). Se puede observar que PGG-vr siempre obtiene resultados mejores o iguales que PGG-1r, sin embargo, estos resultados por regla general no difieren de manera significativa. Sólo para el conjunto de datos Satimage el resultado obtenido por PGG-vr es estadísticamente mejor que el de PGG-1r. De paso también queremos señalar que aunque en este conjunto de datos la PGG no es el mejor algoritmo, es significativamente mejor que One-R y CN2.

6. Conclusiones

La principal conclusión que podemos obtener del trabajo realizado es que la PGG es una técnica capaz de lograr resultados competitivos en cuanto a precisión para llevar a cabo tareas de clasificación, ya que en la mayoría de los casos logra resultados mejores o no peores que los demás algoritmos con que ha sido comparada en los experimentos computacionales que hemos descrito.

Otra conclusión que hemos obtenido es que la ventaja de evaluar todos los antecedentes (variante vr) frente a elegir la primera regla cuyo antecedente se cumpla (variante 1r) es mínima, ya que no llega a resultar estadísticamente significativa por regla general.

Además, creemos que el uso de gramáticas resulta muy prometedor, ya que ofrece una flexibilidad que puede resultar muy útil para desarrollar sistemas de MD. En este trabajo, hemos utilizado la gramática para definir el tipo de antecedente a utilizar, ya que los operadores relacionales permitidos dependen de los tipos de datos de los atributos del conjunto de datos (como se describe en los apartados 4.1 y 4.4). No obstante, mediante una gramática podemos determinar fácilmente distintas características que queremos que tengan los clasificadores. Además, al utilizar una gramática se limita el espacio de

búsqueda, lo cual pude resultar beneficiosos desde el punto de vista de la eficiencia.

Este trabajo es sólo un primer paso en el estudio de la aplicación de la PGG a MD. Para seguir avanzando en este sentido, nuestros objetivos inmediatos se centran en ampliar el tipo de estudio planteado en este trabajo, aumentando el número de conjuntos de datos usados y teniendo en cuenta otros criterios para medir la calidad de los resultados, como el tamaño de los clasificadores y el tiempo de ejecución. Otra vía de interés para profundizar en este tipo de estudio consiste en aplicar la PGG a otras tareas de minería como por ejemplo la obtención de reglas de asociación o el clustering.

Agradecimientos

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

Referencias

[1] 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.

[2] Clark, P.; Niblett, T. "The CN2 Induction Algorithm". *Machine Learning*, vol. 3, no. 4, pp. 261-283, 1989.

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

[4] Gilbert, R. G.; Goodacre, R.; Shann, B.; Kell, D. B.; Taylor, J.; Rowland, J. J. "Genetic Programming-based variable selection for high-dimensional data". *Proceedings of the 3rd Conference on Genetic Programming*, pp. 109-115, 1998.

[5] 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.

[6] Michalski, R. S. **"**On the quasi-minimal solution of the general covering problem". *Proceedings of the Fifth International Symposium on Information Processing*, pp. 125-128, 1969.

[7] 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]

[8] Quinlan, R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[9] 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, 1993.

[10] 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, 1996.

[11] Witten, I. H.; Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[12] Wong, M. L.; Leung, K. S. *Data Mining Using Grammar Based Genetic Programming and Applications*. Serie Genetic Programming, vol. 3. Springer, 2000.