Reglas de Asociación en Datos Multi-Instancia mediante Programación Genética Gramatical

José María Luna
Depto. Informática y Análisis Numérico,
Universidad de Córdoba
Email: jmluna@uco.es

Oscar Reyes
Depto. Informática y Análisis Numérico,
Universidad de Córdoba
Email: ogreyesp@gmail.com

María José del Jesus
Depto. Informática,
Universidad de Jaén
Email: mjjesus@ujaen.es

Sebastián Ventura
Depto. Informática y Análisis Numérico,
Universidad de Córdoba
Email: sventura@uco.es

***Abstract*—El estado del arte actual en minería de reglas de asociación es bastante prometedor en cuanto a la extracción automática de relaciones de interés entre patrones o elementos de grandes bases de datos. Por lo general, los datos se encuentran representados en forma tabular, donde cada fila define inequívocamente un registro de datos u objeto en el dominio de la aplicación. En ocasiones, estos registros de datos no pueden o deben ser considerados de forma aislada (por ejemplo, datos que comprenden hábitos de compra de clientes de manera que diferentes registros pueden describir un mismo cliente) y, por tanto, el problema no puede ser abordado como un problema de minería de reglas de asociación tradicional. Un primer intento de resolver este problema consideró un enfoque determinista sobre dominios discretos y requiriendo analizar todo el espacio de búsqueda (el tiempo de computo es exponencial respecto al número de elementos). Sin embargo, cuando nos enfrentamos a un problema real, la información se encuentra generalmente definida en dominios continuos, se requieren resultados en el menor tiempo posible, y se necesita satisfacer las expectativas de los usuarios (los resultados deben coincidir con la forma deseada). En este sentido, el objetivo de este trabajo es solucionar estos desafíos mediante un algoritmo de programación genética gramatical. Tanto sus fortalezas como defectos se analizan sobre un conjunto variado de datos en el estudio experimental.**

I. Introducción

Desde que se produjo la revolución digital, la cantidad de información que se almacena en cada dominio de aplicación ha crecido a un ritmo exponencial. En este sentido, se requiere que las diferentes técnicas de minería de datos mejoren su eficiencia para poder tratar con tales conjuntos masivos de datos en un tiempo considerable. Una de las tareas de minería de datos que se ha visto claramente afectada por la dimensionalidad de los datos es la minería de reglas de asociación (ARM por su término en inglés) [1]. Considerando un conjunto de datos con k elementos, el espacio de búsqueda en el que ARM debe buscar soluciones incluye un total de $3^k - 2^{k+1} + 1$ reglas de asociación diferentes obtenidas sobre un total de $2^k - 1$ patrones o combinaciones de elementos.

Incluso cuando se trata de un problema realmente complejo, especialmente para grandes conjuntos de datos, diferentes investigadores han propuesto algoritmos de ARM realmente eficientes. Dichas propuestas fueron especialmente relevantes gracias a las arquitecturas paralelas actuales y *frameworks* [**?**]. En este sentido, el problema de ARM se puede considerar como resuelto desde el punto de vista del rendimiento [**?**]. Sin embargo, el creciente interés en el almacenamiento de datos ha causado no solo un crecimiento en la dimensionalidad de los datos, sino también en la representaciones de dichos datos. A modo de ejemplo, consideremos una cadena de supermercados que quiere extraer conocimiento útil sobre los hábitos de compra de sus clientes para tomar decisiones correctas. Supongamos que la información se almacena en una tabla tradicional (cada fila indica una transacción diferente). Aquí, cada cliente tendrá un peso específico en la descripción de datos con respecto al número de transacciones que representa y, por lo tanto, esta representación de datos desvía el conocimiento extraído según el tipo de cliente (los clientes frecuentes son más importantes que los clientes esporádicos). Consideremos ahora que cada cliente se almacena como un ejemplo que comprende un número indefinido de transacciones, y los datos se describen sobre conjunto de ejemplos en lugar de sobre el conjunto de transacciones. De esta forma, el conocimiento extraído revela información explícita sobre los hábitos de los clientes independientemente de la cantidad de transacciones que realiza cada cliente. Esto implica que la forma en que se estructuran los datos es esencial para proporcionar el conocimiento correcto.

Este almacenamiento de información está por tanto relacionado con *multiple-instance learning* (MIL) [4], una generalización del aprendizaje supervisado tradicional. En MIL, los ejemplos son ambiguos y un solo ejemplo puede tener muchas instancias alternativas que lo describen. Esta ambigüedad se analizó recientemente [5] desde una perspectiva de descripción de datos mediante ARM sobre estructuras de datos de múltiples instancias. Sin embargo, este primer intento se basó en una metodología de búsqueda exhaustiva para establecer las bases en esta nueva representación para análisis descriptivo de datos y no se prestó atención al tiempo de ejecución, la metodología, el dominio de la aplicación y las

preferencias de los usuarios.

El objetivo de este trabajo es proponer un nuevo algoritmo de ARM para trabajar sobre datos de multi-instancia. El enfoque propuesto, conocido como MIAR-GePro, se basa en una metodología de programación genética gramatical y tiene como objetivo mejorar el estado del arte en este campo. MIAR-GePro puede trabajar tanto en dominios discretos como continuos, representando una importante ventaja respecto a la propuesta existente (requiere una discretización previa al proceso de extracción). Además, el modelo evolutivo propuesto es capaz de extraer reglas de asociación en un solo paso (la propuesta existente requiere extraer primero todos los patrones frecuentes y, a posteriori, obtener reglas de asociación sobre dichos patrones). Finalmente, los resultados obtenidos mediante MIAR-GePro están restringidos de acuerdo a una gramática predefinida por el usuario, por lo que es éste usuario final quien tiene la capacidad de definir la forma de las reglas que desea obtener. Todas estas características, así como algunos inconvenientes de esta propuesta se analizan en la etapa experimental al considerar un conjunto variado de datos.

El presente documento está organizado de la siguiente manera. La sección II presenta las definiciones más relevantes y el trabajo relacionado. La sección III describe el algoritmo MIAR-GePro propuesto. La sección IV presenta los conjuntos de datos utilizados en los experimentos, la configuración seguida, así como los resultados obtenidos. Finalmente, algunas observaciones finales se describen en la sección V.

II. Preliminares

En esta sección se describe por primera vez el problema de datos multi-instancia. A continuación, se define la tarea de minería de regla de asociación (ARM por su término en inglés) y se analiza el primer intento de extraer reglas de asociación en datos multi-instancia.

A. Datos Multi-Instancia

El problema de aprendizaje sobre datos multi-instancia fue introducido por primera vez por *Dietterich et al.* [4] en 1997 en el contexto de la predicción del comportamiento de diferentes fármacos. En este dominio de aplicación, los ejemplos son ambiguos y pueden incluir múltiples instancias alternativas que lo describen. De manera formal, supongamos que cada ejemplo e_i puede tener v_i variantes $e_{i,1}$, $e_{i,2}$, ..., e_{i,v_i}. Cada una de estas variantes estará representada por un vector de características distintas $V(e_{i,j})$, por lo que un ejemplo completo e_i se representa por lo tanto como un conjunto de vectores de características en la forma $\{V(e_{i,1}), V(e_{i,2}), ..., V(e_{i,v_i})\}$. Incluso cuando este problema se ha aplicado generalmente a las tareas de aprendizaje supervisado [6], realizar un análisis descriptivo de ese tipo de datos es de gran interés en muchos dominios. Considerando el conocido problema de análisis de la cesta de la compra, un único cliente puede describir diferentes compras, por ejemplo, en días diferentes de un mes. Como resultado, cada ejemplo (un cliente específico) incluye un número indefinido de registros (diferentes compras). En este escenario, el problema se puede definir como el análisis de un conjunto de datos en el que los datos se representan como multi-instancias, y se pueden aplicar diferentes hipótesis de trabajo.

La hipótesis de trabajo más simple fue propuesta por *Dietterich* [4], y considera que un determinado ejemplo se cumple si al menos una de las instancias incluidas en dicho ejemplo es satisfecha. En la minería de patrones, el problema se modela como encontrar un patrón P en el que $\exists j \ : \ P \subset V(e_{i,j})$ para una gran cantidad de ejemplos e_i. Teniendo en cuenta esta hipótesis en el análisis de la cesta de mercado, su objetivo es encontrar artículos (o conjunto de artículos) comprados al menos una vez por un gran número de clientes. Aquí, el objetivo es descubrir cuáles son los productos más populares o qué productos generalmente compran los clientes a la vez.

Múltiples autores han propuesto diferentes hipótesis de trabajo para tratar el problema de multi-instancia. *Weidmann et al.* [7] definió dos tipos adicionales de hipótesis que son casos especiales del propuesto por *Dietterich*. En la primera de estas hipótesis adicionales, el problema se modela como el de encontrar un patrón P en el que $\sum_{j=1}^{v_i} P \subset V(e_{i,j}) \geq m$ para un número alto de ejemplos e_i. Centrándose en el análisis de la cesta de la compra, esta hipótesis es realmente útil para determinar qué artículos (o conjunto de artículos) generalmente se compran más de m veces por un gran número de clientes. Aquí, no solo es importante conocer productos populares, sino que también se requiere que tales productos sean generalmente comprados por cada uno de los clientes. Finalmente, el segundo tipo de hipótesis definido por *Weidmann et al.* [7] establece que un patrón P se satisface con un número determinado de instancias en un ejemplo, y este número está entre un valor mínimo m y un valor máximo o. En otras palabras, $o \geq \sum_{j=1}^{v_i} P \subset V(e_{i,j}) \geq m$ en una gran cantidad de ejemplos e_i. Volviendo al análisis de la cesta de la compra, esta hipótesis es esencial para conocer el conjunto de clientes que consumen un producto específico un número predefinido de veces (dando un valor umbral mínimo y máximo).

B. Minería de Reglas de Asociación

La tarea de ARM fue definida formalmente por *Agrawal et al.* [8] en los años 90. Esta tarea tiene como objetivo extraer implicaciones de la forma $Antecedente \rightarrow Consecuente$ que sean de especial interés y desconocidas. Tanto $Antecedente$ (A) and $Consecuente$ (C) representan conjuntos disjuntos de elementos. De manera formal, se define $I = \{i_1, i_2, ..., i_n\}$ como el conjunto de n elementos incluidos en un conjunto de datos Ω, y $T = \{t_1, t_2, ..., t_m\}$ como el conjunto de transacciones o registros que cumplen $\forall t \in T : t \subseteq I \in \Omega$. Una regla de asociación es una implicación de la forma $A \rightarrow C$ donde $A \subset I$, $C \subset I$, y $A \cap C = \emptyset$. La frecuencia (también conocida como soporte) de una regla $A \rightarrow C$ es definida como el porcentaje de transacciones en las que se cumple $A \cup C \subseteq t : t \in T$. Este hecho se ha definido también en términos absolutos como el número de transacciones en las que aparece $A \cup C$, es decir, $|\{\forall t \in T : A \cup C \subseteq t, t \subseteq I \in \Omega\}|$. El significado de una regla de asociación es que si el antecedente

G = $(\Sigma_N, \Sigma_T, P, S)$ con:

S = Regla
Σ_N = {Regla, Antecedente, Consecuente, Condiciones, Condición, Condición_Nominal, Condición_Numérica}
Σ_T = {'Y', 'Atributo', 'valor', '=', '$\geq$','$\leq$'}
P = {Regla = Antecedente, Consecuente ;
Antecedente = Condiciones;
Consecuente = Condiciones;
Condiciones = 'Y', Condiciones, Condición | Condición ;
Condición = Condition_Nominal | Condición_Numérica ;
Condición_Nominal = 'Atributo', '=', 'valor' ;
Condición_Numérica = 'Atributo', '$\geq$', 'valor' | 'Atributo', '$\leq$', 'valor';}

Fig. 1. Gramática de contexto libre que representa reglas de asociación expresadas en notación BNF

de la regla se cumple, entonces es bastante probable que el consecuente de dicha regla también se cumpla.

El uso de ARM sobre representaciones de datos no estándar, por ejemplo datos multi-instancia, han sido recientemente estudiados debido a su relevancia. En un conjunto de datos multi-instancia Ω, cada bolsa de instancias o ejemplos e_i $in\Omega$ comprende un número variado l de transacciones o instancias, es decir, $e_i = \{t_{i,1}, t_{i,2}, ..., t_{i,l}\}$. Cada instancia única $t_{i,j}$ se define como un subconjunto de elementos tales que $t_{i,j} \in e_i$: j $leql, t_{i,j} \subseteq I$. La principal entre ARM clásica y la basada en multi-instancias radica en las medidas de calidad utilizadas para determinar el interés de las soluciones. En la forma más simple, una regla de asociación $A \rightarrow C$ cumplirá un ejemplo e_i si y sólo si al menos una de sus instancias $t_{i,j} \in e_i$ es satisfecha por la regla, es decir, $A \cup C \subseteq t_{i,j} : t_{i,j} \in e_i$.

La métrica conocida como soporte de una regla $A \rightarrow C$ se define como el número de ejemplos que la cumplen, es decir, $|\{\forall e_i \in \Omega \ : \ A \cup C \subseteq t_{i,j} \in e_i\}|$. El soporte puede definirse también en su forma relativa al tamaño del conjunto de datos tal que $|\{\forall e_i \in \Omega \ : \ A \cup C \subseteq t_{i,j} \in e_i\}|/|\Omega|$. Una característica importante del soporte en datos multi-instancia es la posibilidad de que tanto A como C satisfagan todas las transacciones, pero si se analizan juntas no satisfacen ninguna transacción. Esta afirmación no es posible en ARM clásico ya que considera cada transacción como una sola instancia. Centrándonos en otra medida de la calidad de las reglas en problemas multi-instancia, la confianza se define como la proporción del número de ejemplos que incluyen A y C entre todos los ejemplos que comprenden A. Otra métrica ampliamente utilizada entre las muchas existentes en ARM [10] es conocida como *lift* y permite establecer cuántas veces ocurren A y C juntas más a menudo de lo que se esperaría si fueran estadísticamente independientes, es decir $Lift(A \cup C) = Soporte(A \cup C)/(Soporte(A) \times Soporte(C))$.

III. MIAR-GePro

El algoritmo propuesto, conocido como MIAR-GePro (por sus siglas en inglés de *Multiple-Instance Association Rule Genetic Programming*), incorpora una gramática libre de contexto que se adapta tanto al problema en cuestión como al conocimiento subjetivo del usuario. Esta gramática permite predefinir la forma de las soluciones y, por lo tanto, el espacio de búsqueda se reduce a aquellas soluciones que cumplen con la gramática.

a) ***Criterio de codificación****:* en programación genética gramatical [11] cada solución al problema bajo estudio está representada por un genotipo (árbol de derivación basado en el lenguaje definido por una gramática G) y un fenotipo (significado de la estructura del árbol). Una gramática libre de contexto se define como $(\Sigma_N, \Sigma_T, P, S)$ donde Σ_T representa el alfabeto de los símbolos terminales y Σ_N el alfabeto de los símbolos no terminales, siendo ambos conjuntos disjuntos, es decir, $\Sigma_N \cap \Sigma_T = \emptyset$. Además, P establece el conjunto de reglas de producción que se utilizan para codificar una solución, y estas reglas de producción deben comenzar desde el símbolo inicial S. Una regla de producción se define como una regla $\alpha \rightarrow \beta$, donde $\alpha \in \Sigma_N$, y $\beta \in \{\Sigma_T \cup \Sigma_N\}^*$. Como se describió anteriormente, se aplica una serie de reglas de producción del conjunto P para obtener una solución que se representa como un árbol donde los nodos internos contienen solo símbolos no terminales y las hojas contienen solo terminales.

La Figura 1 muestra la gramática de contexto libre propuesta en el algoritmo MIAR-GePro. En esta gramática, el símbolo inicial S se representa con el término Rule (raíz del árbol de derivación), el cual siempre tiene como nodos hijos el antecedente y el consecuente de la regla. Estos dos nodos secundarios (antecedente y consecuente) pueden ser expandidos en una sola condición o un conjunto de condiciones que incluyen características tanto discretas como continuas. Como resultado y considerando la gramática propuesta (ver Figura 1), se obtiene el siguiente lenguaje de la gramática $L(G) = \{(Y \ Condicion)^n \ Condicion \ \rightarrow (Y \ Condicion)^m \ Condicion, \ n \geq 0, m \geq 0\}$ que cualquier solución válida debe satisfacer.

b) ***Proceso de evaluación****:* El proceso clave de cualquier algoritmo evolutivo es la evaluación de las soluciones, el cual guía el proceso de búsqueda a través de soluciones prometedoras dentro del espacio de búsqueda. Aunque se pueden utilizar muchas hipótesis de trabajo para evaluar las soluciones de MIAR-GePro (varias hipótesis se describieron en la sección II-A), en este trabajo hemos considerado la hipótesis tradicional de *Dietterich* donde un ejemplo específico se satisface si y sólo si se cumple al menos una instancia dentro de dicho ejemplo. Así pues, el número máximo de registros

que puede satisfacer una regla (solución al problema) vendrá dado por el número de ejemplos y no por el número total de instancias (considerando todos los ejemplos).

c) ***Operadores genéticos****:* MIAR-GePro incluye dos operadores genéticos para generar nuevas soluciones en cada generación del proceso evolutivo. Estos dos operadores se basan en el cruce y la mutación selectiva propuestos por *Whigham* [12]. El operador de cruce permite crear nuevas soluciones intercambiando subárboles aleatorios a partir de dos padres de forma que se garantizan soluciones correctas de acuerdo al lenguaje de la gramática (ver Figure 1). Para evitar el crecimiento incontrolado, se utiliza como parámetro una profundidad máxima de árbol. Con respecto al operador de mutación, éste selecciona de forma aleatoria un nodo no terminal del árbol y genera un nuevo subárbol a partir de él.

d) ***Esquema evolutivo****:* MIAR-GePro se basa en un algoritmo evolutivo generacional clásico con elitismo (ver Algoritmo 1). La élite está formada por un número máximo predefinido de soluciones, que son las soluciones que deben devolverse al finalizar la ejecución del algoritmo (ver línea 16, Algoritmo 1). Inicialmente, se genera aleatoriamente una población de individuos (soluciones en forma de árboles) (ver línea 3, Algoritmo 1) al considerar la gramática propuesta de contexto libre G y los atributos dentro del conjunto de datos Ω. Una vez que los individuos son evaluados con respecto a algunas medidas de calidad (ver *Evaluation Procedure*, líneas 17 a 44, Algoritmo 1) se ejecuta el ciclo principal, que comprende las siguientes operaciones (ver líneas 7 a 15, Algoritmo 1): (a) se selecciona un conjunto de individuos de la población pop para actuar como padres por medio de un selector por torneo de tamaño 2; (b) el conjunto de padres se cruzan y mutan para formar nuevos individuos (incluidos en el conjunto $newPop$); (c) la población general pop se actualiza mediante un reemplazo directo con elitismo, tomando las mejores soluciones de la élite $\mathcal{S}$ y del conjunto $newPop$. El tamaño máximo de $\mathcal{S}$ siempre es igual o menor que el tamaño de población principal (representado como $popSize$).

Algorithm 1 Pseudocódigo del algoritmo MIAR-GePro

Require: $\Omega, popSize, maxGen, maxSol, \alpha, G$
Ensure: $\mathcal{S}$
1: $\mathcal{S} \leftarrow \emptyset$
2: $pop \leftarrow \emptyset$ ▷ Conjunto de soluciones de cada iteración
3: $pop \leftarrow$ createSolutions $(G, \Omega, popSize)$ ▷ Gramática G usada para generar soluciones
4: $pop \leftarrow$ evaluationProcedure $(\Omega, nExamples, pop, \alpha)$
5: $\mathcal{S} \leftarrow$ takeBestSolutions$(pop, maxSol)$ ▷ Selecciona los $maxSol$ mejores soluciones encontradas en pop
6: $generation \leftarrow 1$
7: **while** $generation \leq maxGen$ **do** ▷ El algoritmo itera un número específico de generaciones
8: $parents \leftarrow$ selectParents(pop)
9: $newPop \leftarrow$ applyGeneticOperators $(parents)$
10: $newPop \leftarrow$ evaluationProcedure $(\Omega, newPop, \alpha)$
11: $\mathcal{S} \leftarrow \mathcal{S} \cup newPop$
12: $pop \leftarrow$ takeBestSolutions$(\mathcal{S}, popSize)$ ▷ Selecciona las $popSize$ mejores soluciones encontradas en $\mathcal{S}$
13: $\mathcal{S} \leftarrow$ takeBestSolutions$(pop, maxSol)$ ▷ Selecciona las $maxSol$ mejores soluciones encontradas en pop
14: $generation \leftarrow generation + 1$
15: **end while**
16: **return** $\mathcal{S}$

17: **procedure** Evaluation Procedure(Ω, pop, α)
18: $nExamples \leftarrow$ getNumberExamples(Ω)
19: **for all** solutions $s \in pop$ **do** ▷ Soluciones en pop
20: $countS \leftarrow 0$ ▷ Ejemplos satisfechos por la regla
21: $countA \leftarrow 0$ ▷ Ejemplos satisfechos por el antecedente
22: **for all** examples $e_i \in \Omega$ **do**
23: **for all** instance $t_{i,j} \in e_i$ **do**
24: **if** $s \subseteq t_{i,j}$ **then**
25: $countS \leftarrow countS + 1$
26: **break**
27: **end if**
28: **if** antecedent$(s) \subseteq t_{i,j}$ **then**
29: $countA \leftarrow countA + 1$
30: **break**
31: **end if**
32: **end for**
33: **end for**
34: $support \leftarrow countS/nExamples$
35: $supportA \leftarrow countA/nExamples$
36: $confidence \leftarrow countS/countA$
37: **if** support $\geq \alpha$ **then**
38: assignFitness$(s, support \times confidence)$
39: **else**
40: assignFitness$(s, 0)$
41: **end if**
42: **end for**
43: **return** pop
44: **end procedure**

IV. Estudio experimental

En esta sección, analizamos el rendimiento del algoritmo MIAR-GePro considerando una serie de conjuntos de datos y teniendo en cuenta hipótesis tradicional propuesta por *Dietterich*. En primer lugar, se realizará una comparativa del rendimiento de la propuesta respecto a un algoritmo de búsqueda exhaustivo (única propuesta existente en este campo hasta la fecha). En segundo lugar, se estudiarán una serie de métricas para medir la calidad media de las soluciones. Todos estos experimentos se realizarán considerando los siguientes parámetros en MIAR-GePro: 50 individuos; 100 generaciones; 0.8 probabilidad de cruce; 0.2 probabilidad de mutación; elitismo con un número máximo de soluciones de 20; y 0.1 como umbral de soporte mínimo.

En este estudio experimental se consideran 10 conjuntos de datos multi-instancia generados artificialmente (ver descripción en la Tabla I). A pesar de que existen multitud de conjuntos de datos reales multi-instancia en la literatura, éstos han sido especialmente diseñados para tareas de clasificación y, por tanto, no proporcionan resultados interesantes para una tarea específica como ARM. Con el fin de poder ejecutar algoritmos de búsqueda exhaustiva como Apriori-MI, los conjuntos de datos que comprenden atributos numéricos se han discretizado previamente. No obstante, cabe destacar que MIAR-GePro se puede aplicar directamente tanto a atributos discretos como continuos, no requiriendo ningún paso previo de procesamiento de datos.

TABLE I
CONJUNTO DE DATOS UTILIZADOS EN EL ESTUDIO EXPERIMENTAL

Conjunto de datos	#Ejemplos	#Atributos	#Instancias	Tamaño medio ejemplos
artificial1	200	6	705	3.53
artificial2	200	8	690	3.45
artificial3	500	8	1,742	3.48
artificial4	500	8	7,406	14.81
artificial5	1,000	10	6,958	6.96
artificial6	1,000	10	12,522	12.52
artificial7	2,000	12	19,986	9.99
artificial8	2,000	16	20,213	10.11
artificial9	5,000	6	17,482	3.50
artificial10	5,000	8	37,748	7.55

A. Análisis del rendimiento

El análisis de la escalabilidad de MIAR-GePro con respecto al algoritmo Apriori-MI [5] se ha llevado a cabo por medio de dos formas diferentes (ver Figuras 2 y 3). En primer lugar, es importante destacar que la escalabilidad de los algoritmos de ARM ha sido ampliamente estudiada por diferentes investigadores [3], lo que demuestra que el tiempo de ejecución de los algoritmos de búsqueda exhaustiva aumenta exponencialmente con el número de atributos. En este estudio, queremos analizar el rendimiento de los algoritmos de búsqueda tanto evolutivos como exhaustivos cuando se aplican sobre conjuntos de datos multi-instancia, considerando diferentes números de ejemplos (Figura 2) y atributos (Figura 3). Como se muestra, el algoritmo Apriori-MI requiere un tiempo computacional más alto con el incremento del número de ejemplos (también conocidos como bolsas), obteniendo todas las posibles reglas de asociación que satisfacen los umbrales establecidos (valor mínimo de soporte de 0.1. Destacar que este valor tan bajo permite obtener casi todas las reglas de asociación existentes). En cuanto a la escalabilidad de los algoritmos para diferentes números de atributos (ver Figura 3), se obtiene que algoritmos de búsqueda exhaustivos como

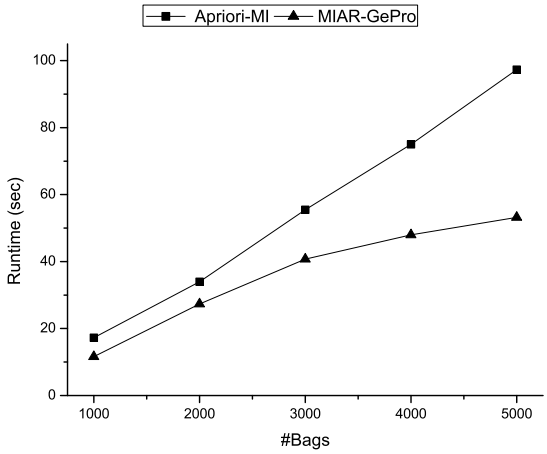

Fig. 2. Tiempo de ejecución en segundos para diferentes número de ejemplos (bolsas de instancias)

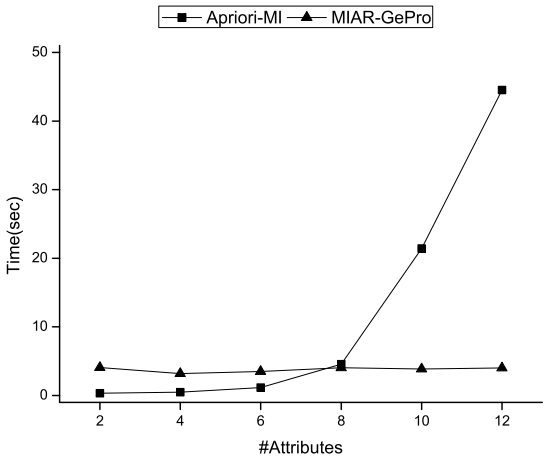

Fig. 3. Tiempo de ejecución en segundos para diferentes números de atributos

Apriori-MI aumentan exponencialmente su tiempo de cálculo con el incremento del número de atributos. Por el contrario, las propuestas evolutivas como MIAR-GePro no se ven afectadas por la cantidad de atributos.

B. Análisis de las medidas de calidad

La tabla II muestra los resultados obtenidos para diez conjuntos de datos diferentes discretizados en 5, 10 y 15 intervalos de igual anchura (la discretización es un requisito previo para poder ejecutar el enfoque de búsqueda exhaustiva existente). El algoritmo Apriori-MI se ejecuta utilizando un Soporte mínimo de 0.1, por lo que el algoritmo no tiene en cuenta ninguna regla que tenga un valor de Soporte inferior y calcula tres medidas de calidad diferentes (Soporte, Confianza y Lift). Analizando los resultados promedio representados en la Tabla II, el número de reglas descubiertas es mayor cuando el número de intervalos en los que se discretiza es menor. Esto se debe a la mayor cantidad de ejemplos que podrían satisfacerse cuando los atributos están discretizados en un pequeño número de intervalos (mayor rango de valores). Por el contrario, el mero hecho de considerar un número mayor de valores discretos (rango inferior de valores) implica que los atributos o patrones apenas se satisfacen, por lo que el número de ejemplos que comprenden patrones específicos es menor.

El mismo análisis anterior se ha realizado utilizando un enfoque evolutivo (ver Tabla III). En este caso específico, MIAR-GePro no requiere discretizar los conjuntos de datos en un número de intervalos, por lo que MIAR-GePro se ejecuta utilizando únicamente un umbral de soporte mínimo de 0.1. Además, MIAR-GePro permite extraer un número específico de reglas, por lo que hay que indicar dicho número que será el de las mejores reglas encontradas (se fija en 20 debido a que es el número de reglas generalmente utilizadas por los expertos en ARM con enfoques evolutivos [3]). El análisis de los resultados para la métrica de Confianza en el algoritmo evolutivo (no requiere preprocesado) determina mejores resultados que Apriori-MI. Es bastante interesante cómo cuanto mayor es el número de intervalos, mejores son

TABLE II
NÚMERO DE REGLAS OBTENIDAS Y VALORES MEDIOS PARA TRES MEDIDAS DE CALIDAD PARA EL ALGORITMO APRIORI-MI.

Conjunto de datos	#Reglas	Soporte	Confianza	Lift
artificial1-5	686	0.155	0.319	0.653
artificial1-10	192	0.156	0.408	1.101
artificial1-15	160	0.125	0.404	1.115
artificial2-5	1,480	0.147	0.324	0.702
artificial2-10	262	0.154	0.407	1.014
artificial2-15	208	0.128	0.406	1.112
artificial3-5	2,978	0.127	0.307	0.626
artificial3-10	140	0.240	0.548	0.994
artificial3-15	210	0.167	0.502	0.994
artificial4-5	36,788	0.147	0.244	0.365
artificial4-10	13,966	0.129	0.250	0.424
artificial4-15	430	0.313	0.560	0.987
artificial5-5	7,392	0.213	0.359	0.567
artificial5-10	294	0.294	0.542	0.993
artificial5-15	276	0.299	0.586	1.008
artificial6-5	26,670	0.172	0.263	0.372
artificial6-10	12,964	0.120	0.221	0.369
artificial6-15	538	0.289	0.538	0.998
artificial7-5	19,998	0.191	0.334	0.586
artificial7-10	35,979	0.118	0.342	0.993
artificial7-15	422	0.176	0.359	1.002
artificial8-5	15,523	0.164	0.366	0.898
artificial8-10	12,033	0.126	0.322	0.921
artificial8-15	528	0.194	0.542	1.001
artificial9-5	1,916	0.122	0.304	0.623
artificial9-10	100	0.243	0.552	1.002
artificial9-15	150	0.169	0.507	1.002
artificial10-5	4,498	0.226	0.372	0.581
artificial10-10	276	0.277	0.518	0.982
artificial10-15	224	0.309	0.592	1.024

TABLE III
NÚMERO DE REGLAS Y VALORES MEDIOS PARA TRES MÉTRICAS DE CALIDAD OBTENIDOS CON MIAR-GEPRO.

Conjunto de datis	#Reglas	Soporte	Confianza	Lift
artificial1	20	0.748	0.998	1.005
artificial2	20	0.895	1.000	1.005
artificial3	20	0.909	0.914	1.002
artificial4	20	0.975	0.999	0.999
artificial5	20	0.989	1.000	1.000
artificial6	20	0.987	0.999	0.999
artificial7	20	0.991	0.999	0.999
artificial8	20	0.995	1.000	1.000
artificial9	20	0.981	0.999	1.000
artificial10	20	0.997	0.999	0.999

los resultados obtenidos para esta medida de calidad en el enfoque de búsqueda exhaustiva. Esto explica por qué MIAR-GePro obtiene los mejores resultados, ya que no requiere un número fijo de intervalos (funciona directamente sobre atributos continuos).

V. CONCLUSIONES

El creciente interés en el almacenamiento de datos ha causado no solo un crecimiento en la dimensionalidad de los datos, sino también en la variedad de sus representaciones. Estudios recientes han propuesto soluciones para extraer reglas de asociación en conjuntos de datos donde múltiples registros describen un único objeto. Sin embargo, las soluciones existentes para la minería de reglas de asociación en este tipo de datos están restringidas a dominios discretos y requieren analizar todo el espacio de búsqueda (el tiempo de ejecución aumenta exponencialmente con el número de atributos). Así pues, el objetivo de este documento es proponer un nuevo algoritmo que permita trabajar en dominios continuos, obtener resultados lo antes posible y satisfacer las expectativas de los usuarios (los resultados deben coincidir con la forma deseada).

La propuesta MIAR-GePro utiliza programación genética gramatical para ARM sobre datos multi-instancia. MIAR-GePro puede trabajar en dominios discretos y continuos, representando una ventaja importante con respecto a la propuesta existente (Apriori-MI). Además, el modelo evolutivo propuesto es capaz de extraer las reglas de asociación en un solo paso (la propuesta existente requiere extraer primero todos los patrones frecuentes y, luego, obtener reglas de asociación de dichos patrones). El análisis experimental llevado a cabo en 10 diferentes conjuntos de datos ha demostrado que la propuesta es altamente eficiente, requiriendo un tiempo de ejecución que es casi constante para cualquier cantidad de atributos.

AGRADECIMIENTOS

Este trabajo ha sido financiado por el proyecto TIN2017-83445-P del Ministerio de Economía y Competitividad y Fondos FEDER.

REFERENCES

[1] C. C. Aggarwal and J. Han, *Frequent Pattern Mining*. Springer International Publishing, 2014.
[2] F. Padillo, J. M. Luna, F. Herrera, and S. Ventura, "Mining association rules on big data through mapreduce genetic programming," *Integrated Computer-Aided Engineering*, vol. 25, no. 1, pp. 31–48, 2018.
[3] S. Ventura and J. M. Luna, *Pattern Mining with Evolutionary Algorithms*. Springer International Publishing, 2016.
[4] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1–2, pp. 31 – 71, 1997.
[5] J. M. Luna, A. Cano, V. Sakalauskas, and S. Ventura, "Discovering useful patterns from multiple instance data," *Information Sciences*, vol. 357, pp. 23–38, 2016.
[6] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez Tarragó, and S. Vluymans, *Multiple Instance Learning - Foundations and Algorithms*. Springer, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-47759-6
[7] N. Weidmann, E. Frank, and B. Pfahringer, "A Two-level Learning Method for Generalized Multi-instance Problems," in *Proceedings of the 14th European Conference on Machine Learning*, 2003, pp. 468–479.
[8] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD Conference '93, Washington, DC, USA, 1993, pp. 207–216.
[9] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," *Data Mining and Knowledge Discovery*, vol. 8, pp. 53–87, 2004.
[10] F. Berzal, I. Blanco, D. Sánchez, and M. A. Vila, "Measuring the accuracy and interest of association rules: A new framework," *Intelligent Data Analysis*, vol. 6, no. 3, pp. 221–235, 2002.
[11] R. McKay, N. Hoai, P. Whigham, Y. Shan, and M. O'Neill, "Grammar-based Genetic Programming: a Survey," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 365–396, 2010.
[12] P. Whigham and D. O. C. Science, "Grammatically-based genetic programming," in *Proceedings of the Workshop on Genetic Programming*, Tahoe City, California, USA, 1995, pp. 33–41.