Uso de Técnicas no Supervisadas en la Construcción de Modelos de Clasificación en Ingeniería del Software

María N. Moreno García* y Vivian F. López Batista

Departamento de Informática y Automática. Universidad de Salamanca.
*E-mail: mmg@usal.es

Resumen. En minería de datos las técnicas supervisadas y no supervisadas tienen diferentes propósitos. Las primeras se utilizan para construir modelos que serán utilizados para realizar predicciones mientras que las no supervisadas, o algoritmos de descubrimiento del conocimiento, se usan generalmente para la extracción de información útil a partir de grandes volúmenes de datos. Recientemente se han propuesto modelos de clasificación para realizar predicciones basados en técnicas de descubrimiento del conocimiento tales como el análisis de asociación. Los algoritmos de generación de reglas de asociación presentan numerosos problemas que han sido objeto de muchos trabajos en la literatura, sin embargo, apenas se ha tenido en cuenta su uso con fines de clasificación. En este trabajo se trata este aspecto de dichos algoritmos proponiéndose un método para la generación de un reducido número de reglas de asociación que servirá de base para la construcción de un modelo de clasificación apropiado para realizar estimaciones en el ámbito de la Ingeniería del Software.

1 Introducción

Las técnicas de minería de datos no supervisadas, también conocidas con el nombre de técnicas de descubrimiento del conocimiento, se utilizan para la detección de patrones ocultos en bases de datos de gran tamaño. Dichos patrones representan por sí mismos información útil que puede ser utilizada directamente en la toma de decisiones. Por el contrario, los algoritmos de aprendizaje supervisado obtienen a partir de los datos un modelo que relaciona el valor de un atributo llamado etiqueta y los valores de otros atributos (descriptivos). Si la etiqueta es discreta el modelo será de clasificación mientras que si es continua tendremos un modelo de regresión. El modelo obtenido se utilizará posteriormente para realizar predicciones con datos no etiquetados.

Ambas categorías de técnicas se han usado tradicionalmente para sus propósitos originales. Sin embargo, trabajos recientes muestran que los algoritmos no supervisados pueden utilizarse con éxito para resolver problemas de clasificación [12] [11] [22]. La mejora de este tipo de algoritmos, especialmente los de generación de reglas de asociación, ha sido objeto de muchos trabajos de investigación, sin embargo los estudios sobre su uso en problemas de clasificación son muy reducidos. Los algoritmos de reglas de asociación descubren patrones de la forma "SI X ENTONCES Y". Si la parte consecuente (Y) de la regla es una clase, los patrones

obtenidos pueden usarse para predecir la clase de registros no clasificados. El dominio de aplicación más importante de las reglas de asociación es la gestión de mercados. La obtención de reglas de asociación a partir de una base de datos de transacciones comerciales proporciona información sobre productos que los clientes tienden a comprar juntos. Esta información puede utilizarse para dirigir campañas promocionales más efectivas. El objetivo es encontrar productos que impliquen la presencia de otros productos en la misma transacción considerando una transacción una compra de varios artículos.
El proceso de generación de reglas es bastante sencillo [8], el problema es el gran número de reglas generadas, muchas de las cuales no tienen ninguna utilidad, por lo que es necesario evaluar su validez. La mayoría de los métodos consideran los factores de *soporte* y de *confianza* para cuantificar la fuerza estadística de un patrón.

El problema de encontrar reglas interesantes ha sido ampliamente estudiado [5] [18] [19]. El objetivo de esas propuestas es la obtención de un número reducido de reglas con valores altos del factor de soporte y de confianza. Además de esos factores, para determinar el interés de una regla se pueden usar medidas subjetivas como la incertidumbre (*unexpectedness*) y la accionabilidad [18] [19] [14]. La incertidumbre (*unexpectedness*) indica que las reglas son interesantes si no son conocidas por los usuarios o contradicen el conocimiento existente. La accionabilidad significa que los usuarios obtienen ventajas de las reglas [14]. Lamentablemente, esos factores subjetivos no se pueden obtener con facilidad en algunas areas como la gestión de proyectos, especialmente cuando están implicados gran cantidad de atributos cuantitativos y, como consecuencia, es muy difícil adquirir conocimiento del dominio. Dichas aplicaciones presentan problemas adicionales tales como la discretización de atributos continuos, los cuales pueden tomar un amplio rango de valores. Para reducir el número de reglas generadas es necesario crear un número manejable de intervalos de valores.

Por otra parte, una regla que es accionable en una aplicación puede no serlo en otra. Por tanto, los métodos de resolución de problemas de las reglas de asociación deben adaptarse a las particularidades de cada caso.

Este trabajo se centra en el uso para predicción de reglas de asociación y aborda el problema de reducir el número de reglas generadas en el campo de la gestión de proyectos. Proponemos un proceso iterativo par refinar reglas de asociación, basado en la generación de patrones no esperados. El rasgo principal de nuestro enfoque es el descubrimiento incremental de conocimiento mediante una generación gradual de los patrones no esperados tomando un solo atributo en cada iteración. Aprovechamos el conocimiento de buenos atributos para la clasificación y los usamos progresivamente, comenzando con el mejor. Esto simplifica la selección de patrones y el proceso de refinamiento. Además, el algoritmo no sólo proporciona un número reducido de reglas sino que selecciona las reglas más útiles para los propósitos de la clasificación. Otro aspecto importante del método es que la generación de patrones no esperados no requiere el conocimiento del dominio. En nuestro contexto de estudio, es muy difícil adquirir experiencia debido al uso de atributos cuantitativos. Por esta razón, el método de refinamiento es muy conveniente para nuestros propósitos. El objetivo es predecir el tamaño del software expresado en las líneas de código (LOC) en las fases iniciales de un proyecto de software usando atributos que pueden obtenerse al comienzo del ciclo de vida. Esto nos permitirá estimar el esfuerzo del proyecto. La obtención de

estimaciones fiables es un factor crítico en la gestión de proyectos debido a que dan lugar a una buena planificación y reducen los costes del proyecto.

El resto del artículo se organiza de la forma siguiente. En la sección 2 se resumen los trabajos relacionados. En la sección 3 se describe el método propuesto. La sección 4 recoge el estudio experimental realizado y finalmente se presentan las conclusiones.

2 Trabajos relacionados

Las reglas de asociación han sido el objetivo de muchos trabajos de investigación desde que Agrawal et al. introdujeran el concepto de asociación entre artículos [2] [1] y propusieran el algoritmo Apriori [3]. La mayor parte de los esfuerzos se han orientado hacia la simplificación del conjunto de reglas generado y a la mejora de la eficiencia del algoritmo.

El primer paso del algoritmo Apriori consiste en contabilizar el número de ocurrencias de cada artículo para determinar los conjuntos de artículos cuyo soporte sea igual o mayor que el valor umbral especificado por el usuario. Existen algoritmos que generan reglas de asociación sin generar previamente los conjuntos de artículos frecuentes [12]. Algunos de ellos simplifican el conjunto de reglas generando un conjunto de reglas con restricciones, es decir reglas con los artículos de la parte consecuente fijados [5] [6]. El procedimiento ocasiona pérdida de información, sin embargo, esto no representa un inconveniente en problemas de clasificación en los que la clase constituye el único artículo de la parte consecuente de las reglas. Se han propuesto muchos algoritmos para obtener un número reducido de reglas interesantes, pero en este trabajo nos centramos en el uso de las reglas en tareas de clasificación. El número de trabajos que tratan este aspecto es mucho menor. Una propuesta de este tipo es el algoritmo CBA (*Classification Based on Association*) [13] que consiste en dos partes, un generador de reglas de asociación y un constructor de un clasificador basado en las reglas generadas. En [22] el peso de la evidencia y la detección de asociaciones se combinan para la realización de una predicción flexible con información parcial. La principal contribución de este método es la posibilidad de realizar predicciones de cualquier atributo de la base de datos. Además, pueden clasificarse observaciones nuevas que estén incompletas. El algoritmo usa el peso de la evidencia de la asociación de atributos en la nueva observación a favor de un valor particular del atributo que se va a predecir. Este enfoque usa todos los atributos de la observación, sin embargo, en muchos dominios algunos atributos tienen una influencia mínima en la clasificación, por lo que el proceso se complica innecesariamente si se tienen en cuenta dichos atributos. Nuestra propuesta evalúa previamente la importancia de los atributos en la clasificación.

La mayoría de los métodos comentados seleccionan las mejores reglas teniendo en cuenta los factores de soporte y de confianza, aunque éstos no son útiles para conocer la conveniencia de las reglas ni para detectar conflictos entre ellas. Para obtener patrones consistentes e interesantes, es necesario considerar otros factores. Esta es la motivación del refinamiento de reglas de asociación.

El tema del refinamiento del conocimiento ha sido objeto de numerosos estudios aunque la parcela de las reglas de asociación apenas se ha tratado. En [18] y [19] se

introduce el concepto de incertidumbre en un proceso iterativo para refinar reglas de asociación. Los autores han propuesto métodos para descubrir patrones no esperados en los datos, usando el conocimiento existente del dominio para reconciliar los patrones no esperados y obtener reglas de asociación más fuertes. El conocimiento del dominio se sustenta en la experiencia de los gestores. Éste es un inconveniente para el uso del método en el ámbito de la gestión de proyectos debido a que las reglas son correlaciones numéricas entre los atributos del proyecto y a que entran en juego muchos factores (el tamaño y complejidad del software, el lenguaje de programación, etc.), por lo que es muy difícil adquirir la experiencia en esta clase de problemas. En este trabajo presentamos un método de refinamiento más apropiado para nuestros propósitos que no requiere conocimiento del dominio. Se basa también en el descubrimiento de patrones no esperados, pero usa "los mejores atributos para la clasificación" en un proceso progresivo de refinamiento de las reglas. Los mejores atributos se obtienen por la técnica de "la importancia de columnas" [15] basada en la cantidad de información (entropía) que los atributos proporcionan en la discriminación de las clases (intervalos de valores del atributo de la etiqueta LOC).

El objetivo es proponer un procedimiento eficaz para problemas de clasificación que sea adecuado para aplicaciones que manejan atributos cuantitativos dónde el conocimiento del dominio no puede obtenerse fácilmente.

3 Modelo de clasificación

En este apartado se propone un algoritmo para generar y refinar reglas de asociación que serán utilizadas para construir un clasificador. Para ello se fija la etiqueta de la clase como el único artículo de la parte consecuente de las reglas y se generan las reglas incrementalmente agregando los artículos progresivamente a la parte antecedente. Se utiliza la información sobre los atributos más influyentes en la clasificación y se toman progresivamente comenzando con el mejor.

3.1 Selección de los mejores atributos

El atributo etiqueta es el objetivo de la predicción en los problemas de clasificación. Pueden usarse un modelo que relaciona la etiqueta y los otros atributos para hacer las predicciones sobre datos nuevos no etiquetados. La importancia de columnas es una técnica que determina el grado de utilidad de los atributos descriptivos (columnas) en la discriminación de los diferentes valores del atributo etiqueta. La medida de la pureza (un número de 0 a 100) informa sobre la influencia de las columnas en la diferenciación de las clases (valores diferentes del atributo etiqueta). Se basa en la cantidad de información (entropía) que la columna (el atributo) proporciona [15]. La expresión para esa información (I) es:

$$\mathrm{I}\,(P(c_1), \ldots, P(c_n)) = \sum_{i=1}^{n} - P(c_i) \log_n P(c_i)$$

donde $P(c_i)$ es la probabilidad de la clase i y n es el número de clases.

La información es 1 cuando las clases tienen las mismas probabilidades.

La pureza se define como 1 – I. Para un conjunto dado en la partición, la pureza es 0 si cada clase tiene el mismo peso y 100 si todos los registros en la partición pertenecen a la misma clase (máxima discriminación).

3.2 Reglas de asociación y patrones no esperados

La minería de reglas de asociación es útil en muchos dominios de aplicación, principalmente para encontrar patrones de compra en entornos comerciales. Los algoritmos de descubrimiento de reglas, como "Apriori" [2], no son complejos, sin embargo, normalmente generan un elevado número de reglas que representan patrones obvios o no pertinentes, incluso restringiéndolas con valores altos de soporte y de confianza. En este trabajo, se presenta un algoritmo de refinamiento que genera las reglas de asociación de una manera incremental y produce las mejores reglas para la predicción de clases. El proceso se basa en el concepto de patrones no esperados. Este concepto lo introdujeron Padmanabhan y Tuzhilin para generar patrones útiles usando el conocimiento de dominio que poseen los gestores. Ellos generan patrones no esperados o contrarios a la experiencia directiva y los usan para refinar reglas que reflejan las creencias de los directivos (*beliefs*) [18]. En nuestro método también se usa el concepto de patrones no esperados pero no se requiere conocimiento del dominio. Los fundamentos de las reglas de asociación y de los conceptos utilizados [19] se introduce a continuación.

Consideremos un conjunto de atributos discretos $At=\{a_1 ,a_2 ,.... ..,a_m \}$. Sea $D=\{T_1 ,T_2 ,.... ..,T_N \}$ una relación consistente en N transacciones $T_1 ,.... ..,T_N$ sobre el esquema de relación $\{a_1 ,a_2 ,.... ..,a_m \}$. Una condición atómica es una proposición de la forma $valor_1 \leq atributo \leq valor_2$ para atributos ordenados y $atributo = valor$ para atributos no ordenados, donde *valor*, $valor_1$ y $valor_2$ pertenecen al conjunto de valores diferente que el atributo puede tomar en D. Finalmente, un *itemset* es una conjunción de condiciones atómicas. En [19] reglas y *beliefs* se definen como reglas de asociación extendidas de la forma $X \rightarrow Y$, donde X es un *itemset* e Y es una condición atómica. La fuerza de la regla de asociación se cuantifica mediante los siguientes factores:

Confidencia or *previsibilidad*. Una regla tiene confidencia *c* si el *c*% de las transacciones en D que contienen X también contienen Y. Se dice que la regla tiene lugar si la confianza supera un umbral especificado por el usuario.

Soporte or *prevalencia*. La regla tiene soporte *s* en D si *s*% de las transacciones en D contienen X e Y.

Previsibilidad esperada. Es la frecuencia de la ocurrencia de Y.

En [18] se parte de un conjunto de *beliefs* que representan el conocimiento del dominio. Una regla $A \rightarrow B$ es inesperada con respecto a la *belief* $X \rightarrow Y$ en la base de datos D si las siguientes condiciones suceden:

- B e Y se contradicen lógicamente (B AND Y |= FALSE);
- A y X suceden en un gran subconjunto de tuplas de D;
- La regla $A, X \rightarrow B$ sucede.

3.3 Refinamiento de reglas de asociación

El algoritmo de refinamiento propuesto usa los atributos más influyentes en la clasificación para encontrar las reglas más adecuadas para la realización de predicciones.

El algoritmo requiere un conjunto de reglas iniciales (*beliefs*) y los correspondientes patrones no esperados. En [18] las reglas iniciales pueden obtenerse a partir del conocimiento de dominio de los gestores o ser inducidas de los datos usando métodos de aprendizaje automático. En nuestro caso se generaron de los datos disponibles de proyectos mediante un algoritmo de reglas de asociación [15]. En un trabajo anterior habíamos comprobado que los mejores atributos para clasificación dan buenos resultados en el refinamiento de reglas de asociación en el área de gestión de proyectos [17]. Tomando como referencia estos resultados, proponemos un procedimiento completo para resolver problemas de clasificación. Se comienza con un juego de *beliefs* que relacionan un solo atributo con la etiqueta de clase. Seguidamente se buscan patrones no esperados que pueden ayudar a aumentar la confianza o resolver ambigüedades o inconsistencias entre las reglas iniciales.

A continuación se describen los pasos del proceso de refinamiento [17]:

1. Obtener los mejores atributos para clasificación y crear la secuencia: seqA = < A_k >, $k = 1 \ldots t$ (t: número de atributos). Los atributos de la secuencia se ordenan de mayor a menor pureza.
2. Los valores del atributo A_k se representan como $\{V_{k,l}\}$, $l = 1 \ldots m$ (m: número de valores diferentes).
3. Asignar $k = 1$ y establecer la *confianza* mínima c_{min} y el *soporte* mínimo s_{min}.
4. Generar las reglas iniciales con confianza $c \geq c_{min}$ y soporte $s \geq s_{min}$.
5. Seleccionar la reglas con *confianza* próxima a c_{min} o con conflictos entre ellas: Sean $X_i \rightarrow Y_i$ y $X_j \rightarrow Y_j$ dos reglas iniciales, R_i y R_j respectivamente. Existe un conflicto entre R_i y R_j si $X_i = X_j$ e $Y_i \neg = Y_j$.
6. Con las reglas seleccionadas crear el conjunto setR = $\{R_i\}$, $i = 1 \ldots n$ (n: número de reglas seleccionadas)
7. Para todas las reglas $R_i \in$ setR do:
   7.1. Usar los valores $\{V_{k,l}\}$ del atributo A_k para generar los patrones no esperados que cumplan las condiciones requeridas (*unexpectedness*) y con *confianza* $\geq c_{min}$. La forma de los patrones es: $V_{k,l} \rightarrow B$.
   7.2. Refinar las reglas buscando otras reglas R' de la forma:
   X_i , $V_{k,l} \rightarrow B$
   X_i , $\neg V_{k,l} \rightarrow Y_i$
   7.3. Sea setR' el conjunto de reglas refinadas, entonces las reglas refinadas en el paso 7.2 se añadirán al conjunto:
   setR' = setR' $\cup$ { R'_u}, $u = 1 \ldots f$ (f: número de reglas refinadas obtenidas en la iteración i).
8. Asignar $k = k + 1$ y setR = setR'.
9. Repetir los pasos 7 y 8 hasta que no se puedan encontrar más patrones no esperados.

Una ventaja significativa del algoritmo es el descubrimiento incremental del conocimiento mediante la generación gradual de reglas refinadas. En el proceso

iterativo se van tomando uno a uno los mejores atributos en la discriminación de las clases. Esto simplifica la selección de patrones, el proceso de refinamiento y genera las mejores reglas para la predicción de las clases. Otros métodos consisten en la generación de un gran conjunto de reglas que se poda posteriormente sin considerar criterios de clasificación.

3.4. Construcción del clasificador

El modelo asociativo obtenido en la etapa anterior es muy apropiado para propósitos de clasificación, por lo que puede ser utilizado para predecir a qué clases pertenecen registros no clasificados. El modelo está compuesto por reglas en diferentes niveles de refinamiento. Las reglas más refinadas se usan en primer lugar. Si una observación a clasificar tiene valores de los atributos coincidentes con la parte antecedente de la regla, se le asignará la etiqueta de clase correspondiente a la parte consecuente de la regla. Si no hay reglas a ese nivel que encajen con la observación se utilizará el nivel de refinamiento anterior.

El problema de encontrar más de una regla que encaje con la observación se resuelve seleccionando la regla con mayor confianza. Pueden proporcionarse probabilidades de pertenencia a una clase en función de la confianza.

La simplicidad de los modelos obtenidos con las reglas de asociación da lugar a un eficiente proceso de predicción. Otro beneficio es el menor número de atributos descriptivos necesarios en relación con los métodos de clasificación tradicionales.

4. Estudio experimental

En el área de gestión de proyectos es fundamental realizar una buena estimación del tamaño del software que servirá de base para la predicción del esfuerzo y coste del proyecto y para la realización de una planificación óptima. La mayoría de los métodos de estimación del tamaño del software proporcionan medidas funcionales del tamaño del software tales como puntos de función [4], bloques de función [10] y puntos objeto [7]. Dichas variables funcionales se deberían corresponder con el tamaño del producto final expresado, por ejemplo, en líneas de código. En un trabajo previo [16] se han aplicado algunas técnicas supervisadas clásicas en la estimación del tamaño del software tomando como referencia un método basado en componentes [21] y un método global (Mark II) [20], encontrando una relación entre las líneas de código (LOC) y otros atributos. En este estudio se intenta optimizar el uso de esos atributos en la predicción del tamaño del software.

4.1 Descripción de los datos

Los datos utilizados para realizar el studio provienen de experimentos realizados por Dolado [9]. Se dispone de datos referentes a 42 proyectos implementados con un lenguaje de cuarta generación (Informix-4GL). Los sistemas desarrollados son aplicaciones de contabilidad con las características de sistemas comerciales. Esta

información se ha dividido en dos grupos, en el primero se ha realizado la descomposición del proyecto en módulos o componentes de tres tipos diferentes siguiendo las pautas de Verner y Tate [35] y se han obtenido atributos de cada uno de los módulos. El segundo grupo contiene datos globales de cada uno de los proyectos. En el primer caso contamos con 1537 registros con datos referentes a los módulos, mientras que en el segundo caso se trabaja con 42 registros con información correspondiente a los proyectos estudiados.

Los datos de componentes son:

TYPECOMP: Tipo de componente (1: menú, 2: entrada, 3: informe/consulta)

OPTMENU: Número de opciones (sólo para componentes de tipo 1)

DATAELEMEN: Número de elementos de datos (sólo para componentes de tipo 2 y 3)

RELATION: Número de relaciones (sólo para componentes de tipo 2 y 3)

LOC: Número de líneas de código del módulo

Los atributos globales correspondientes a los proyectos son los utilizados en el método Mark II [20]:

LOC: Número de líneas de código

NOC: Número de componentes

NTRNSMKII: Número de transacciones MKII

INPTMKII: Número total de entradas

ENTMKII: Número de entidades referenciadas

OUTMKII: Número total de salidas (elementos de datos sobre todas las transacciones)

UFPMKII: Número de puntos de función no ajustados MKII

Calculando nuevos atributos globales a partir de los atributos de los módulos se obtuvieron mejores resultados [16]. Los nuevos atributos: NOC-MENU (número total de componentes de tipo menú), NOC-INPUT (número total de componentes de entrada), NOC-RQ (número total de componentes del tipo informe/consulta) OPTMENU (número total de opciones de menú), DATAELEMENT (número total de elementos de datos), RELATION (número total de relaciones).

4.2. Aplicación del algoritmo de refinamiento

El objetivo que se persigue es la construcción de un modelo que relacione el tamaño final del software con atributos del proyecto que puedan obtenerse fácilmente al comienzo del ciclo de vida. Por tanto, LOC-bin (intervalos de números de líneas de código) puede considerarse la etiqueta de clase. En primer lugar se buscan los atributos que mejor discriminan los valores de la etiqueta LOC-bin mediante el cálculo de pureza acumulativa. Los atributos encontrados con la herramienta *Mineset* [15] fueron RELATION, NTRNSMKII, OPT_MENU y OUTMKII. Estos atributos se usaron posteriormente para refinar las reglas de asociación.

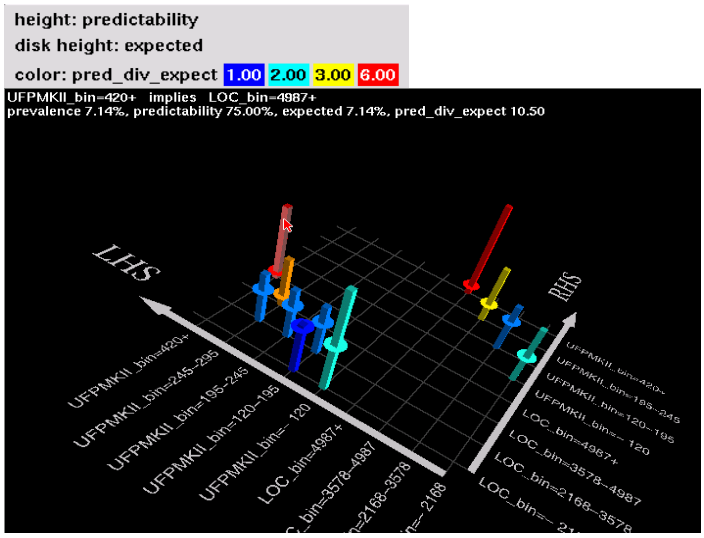

Fig. 1. Reglas iniciales

Comenzamos con un conjunto de reglas (*beliefs*) que relacionan líneas de código con UFPMKII debido a que los puntos de función MKII (UFPMKII), que reflejan el tamaño funcional del software, están relacionados con el tamaño físico [16]. Las reglas iniciales, generadas a partir de los datos disponibles de los 42 proyectos, están representadas en la figura 1. La representación gráfica, proporcionada por *Mineset* muestra en sus ejes los atributos de la parte izquierda (LHS) y derecha (RHS) de las reglas respectivamente. Mediante barras, discos y colores se muestran diferentes propiedades de una regla (LHS → RHS) en el interior del gráfico, en las celdas de intersección de LHS y RHS.

Dado que algunas de las reglas iniciales presentan conflictos entre ellas o muestran correlaciones débiles entre líneas de código y puntos de función Mark II, se buscaron patrones no esperados que nos ayudasen a resolver los conflictos y a obtener reglas más fuertes que matizasen la relación entre LOC y FPMKII. Para ello se hizo uso de los atributos que mejor discriminan las clases, encontrados previamente mediante el cálculo de la pureza. La figura 2 muestra las reglas refinadas en la primera iteración. La parte antecedente contiene ahora los atributos UFPMKII y número de relaciones (RELATION). Puede observarse que estas reglas tienen un factor de confianza superior al de las reglas iniciales. La mayoría de ellas tienen una confianza del 100%. El proceso iterativo puede continuar si se descubren nuevos conflictos o reglas con baja confianza. En este caso, para refinar las reglas se utilizaría el siguiente atributo encontrado mediante la técnica de importancia de columnas. Después del proceso completo de refinamiento incremental se obtuvieron un conjunto de reglas con alta confianza de gran utilidad para realizar la estimación del tamaño del software.

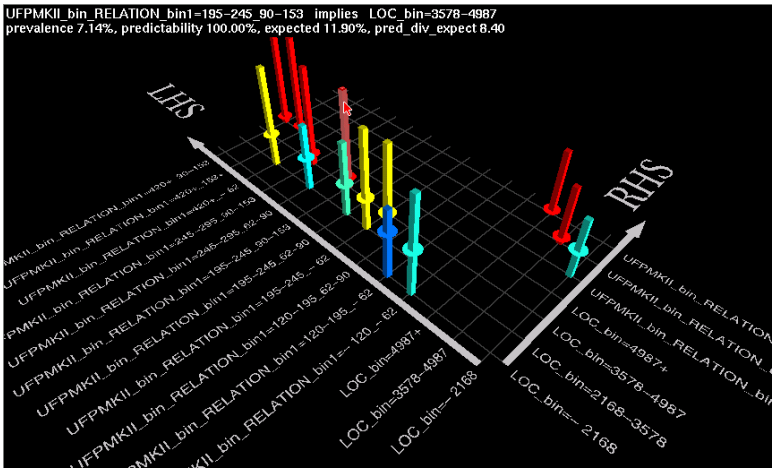

Fig. 2. Reglas refinadas en la primera iteración

5 Conclusiones

En este trabajo se muestra la forma de construir un clasificador que toma como base un modelo de asociación generado con un método no supervisado. Se propone un procedimiento de obtención y refinamiento de reglas de asociación que trata el problema de producir un número reducido de reglas útiles en el campo de gestión de proyectos. Este modelo proporciona un conjunto de reglas con alta confianza para la toma de decisiones sin necesidad de hacer uso de conocimiento del dominio. La identificación de reglas débiles y conflictos entre ellas es el punto de partida del proceso de refinamiento iterativo.

La propuesta evalúa la importancia de los atributos en la clasificación. El algoritmo produce las reglas más útiles para predecir a qué clases pertenecen registros sin clasificar, debido a que utiliza los atributos más importantes en la discriminación de los valores del atributo que representa la clase (etiqueta), que es el objetivo de la predicción. Otros métodos de refinamiento no consideran esta característica, por lo que reducen el número de reglas, pero no obtienen las más adecuadas para los propósitos de clasificación. Con respecto a las ventajas de usar asociación en lugar de clasificación, la primera de ellas es el menor número de atributos que se requieren, y la segunda, la mayor eficiencia de los algoritmos de asociación.

Referencias

1. Agrawal, R., Imielinski, T., Swami, A.: Database Mining: A performance Perspective. IEEE Trans. Knowledge and Data Engineering, vol. 5, 6 (1993b) 914-925.

2. Agrawal, R., Imielinski, T., Swami, A.: Mining associations between sets of items in large databases. Proc. of ACM SIGMOD Int. Conference on Management of Data, Washinton, D.C. (1993a) 207-216.
3. Agrawal, R., Srikant, R.: Fast Algorithms for mining association rules in large databases. Proc. of 20th Int. Conference on Very Large Databases, Santiago de Chile (1994) 487-489.
4. Albrecht, A.J. Measuring Application Development: Proc. IBM Applications Development Joint SHARE/GUIDE Symposium, Monterey, CA, (1979) 83-92.
5. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense database, Proc. 15th Int. Conference on Data Engineering (1999) 188-197.
6. Bayardo, R., Agrawal, R.: Constraint-based rule mining in large, dense database. Proc. Mining the most interesting rules. Proc. ACM SIGKDD Int. Conf. Knowledge Discovery in Databases, ACM Press, NY (1999) 145-154.
7. Boehm, B.W., Clark; B. Horowitz E. et al., Cost Models for Future Life Cycle Processes: COCOMO 2.0: Annals *Software Engineering* 1, No. 1 (1995) 1-24.
8. Cabena, P., Hadjinian, P., Stadler, R. Verhees, J. and Zanasi, A.: Discovering Data Mining. from concept to implementation, (Prentice Hall, 1998).
9. Dolado, J.J. A Validation of the Ccomponent-Based Method for Software Size Estimation: IEEE Transactions on Software Engineering 26, 10 (2000) 1006-1021.
10. Hall, B., Orr, G., Reeves, T.E. A Technique for Function Block Counting: The Journal of System and Software, 57 (2001), 217-220.
11. Hu, Y.C., Chen, R.S., Tzeng, G.H.: Mining fuzzy associative rules for classifications problems. Computers and Industrial Engineering,
12. Li, J., Shen, H., Topor, R.: Mining the smallest association rule set for predictions. Proc. IEEE International Conference on Data Mining (ICDM'01) (2001).
13. Liu, B., Hsu, W. Ma, Y.: Integration Classification and Association Rule Mining. Proc. 4th Int. Conference on Knowledge Discovery and Data Mining, (1998) 80-86.
14. Liu, B., Hsu, W., Chen, S., Ma, Y.: Analyzing the subjective Interestingness of Association Rules. IEEE Intelligent Systems, september/October (2000) 47-55.
15. Mineset user's guide, v. 007-3214-004, 5/98. (Silicon Graphics, 1998).
16. Moreno, M.N., Miguel, L.A., García, F.J., Polo, M.J.: Data mining approaches for early software size estimation. Proc. 3rd ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'02), 361-368, Madrid, Spain (2002).
17. Moreno, M.N., Miguel, L.A., García, F.J., Polo, M.J.: Building knowledge discovery-driven models for decision support in project management. Dec. Support Syst., (doi:10.1016/S0167-9236(03)00100-3).
18. Padmanabhan, B., Tuzhilin, A.: Knowledge refinement based on the discovery of unexpected patterns in data mining. Decision Support Systems 27 (1999) 303– 318.
19. Padmanabhan, B., Tuzhilin, A.: Unexpectedness as a measure of interestingness in knowledge discovery. Decision Support Systems 33 (2002) 309– 321.
20. Symons, C.R. Software Sizing and Estimating MKII FPA (John Wiley and Sons, 1991).
21. Verner, J. and Tate, G. A Software Size Model: IEEE Transaction of Software Engineering, 18 No. 4 (1992): 265-278.
22. Wang, Y., Wong, A.K.C.: From Association to Classification: Inference Using Weight of Evidence. IEEE Transactions on Knowledge and Data Engineering, 15 (2003) 764-767.