Reglas de asociación y Market Basket Analysis © EDICIONES ROBLE, S.L.

Indice

Reglas de asociación y Market Basket Analysis

I. Introducción

La capacidad de descubrir las relaciones existentes en grandes conjuntos de datos puede ser de gran interés en una gran cantidad de aplicaciones. Por ejemplo, en un supermercado, identificar los productos que los clientes compran de forma conjunta puede suponer una gran ventaja competitiva, ya que permite adaptar la oferta a la demanda. De este modo, si se sabe que el cliente que compra leche y pan suele comprar también mantequilla, será posible ubicar estos productos de forma que el cliente pueda hallar los tres sin esfuerzo.

En minería de datos y aprendizaje automático, la identificación de las relaciones existentes dentro de los conjuntos de datos se realiza mediante las llamadas técnicas de análisis asociativo, gracias a las cuales se pueden identificar tanto los eventos que suceden de forma conjunta como las llamadas reglas de asociación.

En esta unidad se presentarán diferentes algoritmos para la construcción de reglas de asociación a partir de los conjuntos de datos y se prestará especial atención al algoritmo Apriori, uno de los más utilizados en minería de datos.

Esta unidad se complementa con un notebook Python en el que se incluye todo el código utilizado y algunos ejemplos adicionales a los que se hace referencia en el texto. Es aconsejable seguir el texto con este notebook U7_Códigos y apriori.py

II. Objetivos

Los objetivos que los alumnos alcanzarán tras el estudio de esta unidad son:

- Saber lo que es una regla de asociación.
- Comprender los conceptos de soporte, confianza y lift.
- Comprender y saber utilizar el algoritmo Apriori.
- Comprender y saber utilizar el algoritmo Eclat (*Equivalence Class Transformation*).
- Comprender y saber utilizar el algoritmo FP-Growth.

II. Soporte, Confianza y Lift

En muchas bases de datos es normal encontrar registros como los que se muestran en el ejemplo de la tabla 6.1., donde se muestran siete tiques de caja de un supermercado. En esta tabla se puede apreciar que el número de productos en cada uno de los registros puede variar desde los dos del primer registro hasta los cuatro del segundo registro, al mismo tiempo, el orden en el que aparecen los diferentes productos parece completamente aleatorio. La forma de identificar patrones en este tipo de datos es mediante la utilización de las técnicas de análisis asociativo.

Tique	Productos
1	Pan, Leche
2	Pan, Pañales, Cerveza, Huevos
3	Leche, Pañales, Cerveza, Cola
4	Leche, Pan, Pañales, Cerveza
5	Pañales, Pan, Leche, Cola
6	Pan, Leche, Pañales
7	Pan, Cola

Tabla 6.1. Conjunto de tiques de ejemplo.

El análisis asociativo se utiliza para buscar las relaciones existentes en los conjuntos de artículos (itemset) que existen en los conjuntos de datos (datasets). Mediante la utilización de este tipo de análisis se pueden identificar dos tipos de relaciones en los conjuntos de datos:

- Los conjuntos de artículos frecuentes.
- Las reglas de asociación.

Los conjuntos de artículos frecuentes hacen referencia a las colecciones de artículos que se encuentran de forma conjunta. Por ejemplo, en la tabla 6.1. se puede observar que el conjunto de leche y pan se repite cuatro veces (tiques 1, 4, 5 y 6), lo mismo que el conjunto de leche y pañales (tiques 3, 4, 5 y 6), mientras que el conjunto de pan y cola solamente se presenta dos veces (tiques 5 y 7).

Las reglas de asociación hacen referencia a las implicaciones entre dos conjuntos de artículos entre los que no existe ningún artículo común. A estas reglas se las suele denominar como antecedentes y consecuentes y se expresan de la siguiente forma:

$$X \Rightarrow Y$$

X es el antecedente e Y el consecuente. Es importante resaltar que tanto el antecedente como el consecuente no pueden compartir ningún artículo, es decir:

$$X \cap Y = \emptyset$$

Por ejemplo, siguiendo con el ejemplo de la tabla 6.2. se pueden definir múltiples reglas de asociación:

$$\{leche\} \Rightarrow \{pañales\}$$
$$\{pan\} \Rightarrow \{leche\}$$

En el ejemplo, la primera relación se observa tres veces mientras que la segunda aparece cuatro veces. En el caso de las reglas de asociación, al indicar una implicación entre los conjuntos de artículos, la regla:

$$\{leche\} \Rightarrow \{pan\}$$

no es lo mismo que la regla:

$$\{pan\} \Rightarrow \{leche\}$$

Para diferenciar entre ambas, es necesario introducir previamente los conceptos de soporte, confianza, mejora de la confianza (*lift*) y convicción (*conviction*). Gracias a ellos, se puede diferenciar el sentido de la relación en la regla de asociación entre dos conjuntos de ítems.

El soporte (*support*) de un conjunto de artículos o de una regla de asociación es la probabilidad de que este aparezca en el conjunto de datos. Se puede escribir como:

$$supp(X) = P(X)$$

Siguiendo con el ejemplo de la tabla 6.1., se puede decir que el soporte del pan es:

$$supp(\{pan\}) = \frac{6}{7} = 0,86$$

Mientras que el de la regla “pan implica leche” es:

$$supp(\{pan\} \Rightarrow \{leche\}) = \frac{4}{7} = 0,57$$

La confianza (*confidence*) de una regla de asociación es el soporte de la regla dividido por el soporte del antecedente, es decir:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} = \frac{P(X/Y)}{P(X)}$$

Este valor se puede interpretar como el cociente entre la probabilidad de que observe la regla en el conjunto de datos respecto a la probabilidad de que aparezca el antecedente, por lo que su valor se encuentra acotado entre cero y la unidad. A partir de este valor se puede identificar el sentido de la regla. Como se ha comentado antes, la regla “pan implica pañales” no es la misma regla que “leche implica pan”, en ambos casos el soporte es 0,57 pero no así la confianza.

$$conf(\{pan\} \Rightarrow \{leche\}) = \frac{4}{6} = 0,67$$

Sin embargo:

$$conf(\{leche\} \Rightarrow \{pan\}) = \frac{4}{5} = 0,80$$

A partir de lo cual se puede deducir que el sentido de la regla es que la compra de leche implica la de pan y no en sentido contrario, ya que el valor de la confianza es mayor en la segunda que en la primera.

La mejora de la confianza (*lift*) es la fracción del soporte observado en la regla respecto al soporte que se obtendría asumiendo que el antecedente y precedente que aparecen se observan de forma independiente, es decir:

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)} = \frac{conf(X \Rightarrow Y)}{supp(Y)} = \frac{P(X/Y)}{P(X)P(Y)}$$

Este valor se puede interpretar como el porcentaje de veces que se observa la regla de asociación respecto a las veces que se esperaría que apareciesen los conjuntos de artículos de forma conjunta debido solamente al azar. Un valor de uno indica que la regla se puede explicar simplemente por azar, los valores por encima de uno indican que la relación es más fuerte de lo que se puede explicar por azar y por debajo que es más débil. A diferencia de la confianza, la mejora de la confianza no depende de la implicación, por lo que las reglas “pan implica leche” y “leche implica pan” tendrán el mismo valor.

Volviendo al ejemplo empleado anteriormente, se puede calcular la mejora de la confianza para la regla de asociación “leche implica pan”:

$$lift(\{leche\} \Rightarrow \{pan\}) = \frac{4/7}{5/7 * 6/7} = 0,93$$

Esto indica que esta regla es más débil que el azar y, por lo tanto, a pesar de ser una regla con valor de la confianza alto, no señala una relación, ya que simplemente se puede explicar por azar.

Finalmente, la convicción (*conviction*) es la fracción de la frecuencia esperada para el antecedente que ocurriría sin que se produjera el antecedente, es decir:

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)} = \frac{1 - P(Y)}{1 - P(X/Y)}$$

Este valor se puede interpretar como la frecuencia con la que la regla realiza una predicción errónea. Al igual que la confianza, este indicador es diferente para la regla en un sentido y en otro, es decir, la regla “leche implica pan” y la regla “pan implica leche” no tendrán el mismo valor.

Volviendo al ejemplo, la convicción de la regla “leche implica pan” es:

$$conv(\{leche\} \Rightarrow \{pan\}) = \frac{1 - 6/7}{1 - 4/7} = 0,33$$

Esto indica que se tiene un 33% de probabilidades de que la regla sea errónea. Por otro lado, la confianza de la regla con la implicación en sencido contrario es:

$$conv(\{pan\} \Rightarrow \{leche\}) = \frac{1 - 5/7}{1 - 4/7} = 0,66$$

Aquí se señala que la regla se equivoca en un 66% de las ocasiones, por lo que la regla que realiza previsiones con mayor convicción es la primera.

IV. Algoritmo Apriori

El algoritmo Apriori, propuesto por Agrawal y Srikant en 1994, es uno de los más utilizados en minería de datos para la generación de reglas de asociación a partir de bases de datos que contengan transacciones, como el ejemplo de la tabla 6.1. Su gran popularidad se debe a la facilidad de implementación y la rapidez con la que se pueden obtener resultados en diferentes conjuntos de datos.

El algoritmo Apriori utiliza un enfoque bottom up. Partiendo de los conjuntos de artículos más pequeños, mediante un proceso iterativo, se van combinando estos conjuntos hasta que se llega a un límite fijado previamente. Posteriormente se procede a la construcción de las reglas de asociación utilizando únicamente los conjuntos de datos creados en el paso anterior.

Creación de reglas de asociación utilizando Apriori

La creación de reglas de asociación utilizando Apriori comienza al definir el soporte mínimo que se desea para los conjuntos de artículos. A partir de este punto, se identifica en la base de datos todos los conjuntos de un artículo que tengan este soporte mínimo. Este paso es el que se muestra en la figura 6.1., donde cada conjunto de datos se identifica con un círculo de diferente color y los que no alcanzan este valor mínimo se eliminan como candidatos, en la figura se representan tachados con un aspa.

Figura 6.1. Selección inicial de los conjuntos de artículos (itemsets).

Una vez que son identificados los conjuntos de un artículo que superan el soporte mínimo, se procede a localizar todos los conjuntos de dos artículos que se puedan formar a partir estos. En el nuevo conjunto de pares, al igual que en el paso anterior, se seleccionan únicamente aquellos que tengan un soporte por encima del fijado. Este paso se muestra en la figura 6.2., donde se presentan los pares seleccionados. Al igual que en el paso anterior, los pares que no alcanzan el mínimo se eliminan, marcándolos con un aspa en la figura.

Se ha de destacar que todos los conjuntos de dos artículos que se crean con el subconjunto de un artículo que supera el soporte mínimo, al igual que los de n que se crean con n-1, tienen un soporte inferior o igual al mínimo soporte de los dos conjuntos de ítems con los que se forman, es decir:

$$supp(\{X,Y\}) \leq min(supp(\{X\}), supp(\{Y\}))$$

Utilizando este proceso de poda para la creación de los candidatos, nunca se perderá ningún conjunto candidato que supere el nivel de soporte fijado. Esto es así porque el porcentaje de veces que la unión de dos elementos puede aparecer en los registros de la base de datos viene limitado por las apariciones del que tiene el menor de los dos.

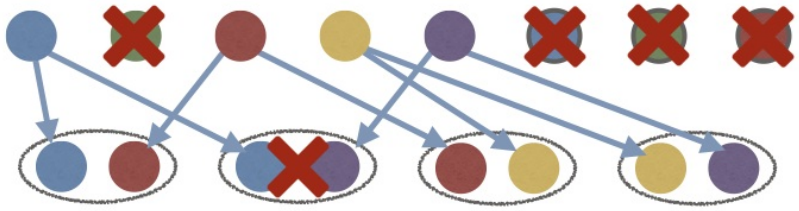

Figura 6.2. Generación de los conjuntos de artículos de dos artículos.

En el caso de que exista algún conjunto de artículos de dos artículos que tengan el soporte mínimo se continúa buscando los conjuntos de tres, en caso contrario, la fase de identificación de conjuntos de artículos finaliza en este punto. En la figura 6.3. se muestra el paso para el conjunto de artículos de tres artículos.

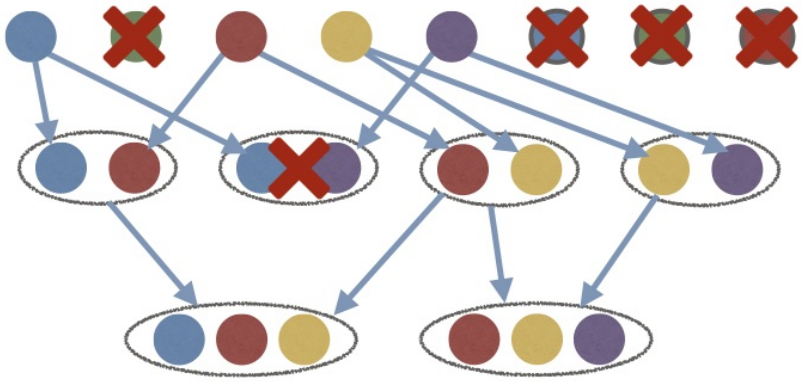

Figura 6.3. Generación de los conjuntos de artículos de tres artículos.

Siguiendo los pasos indicados hasta ahora se pueden identificar todos los conjuntos de artículos en los datos que tengan un valor mínimo de soporte. Esto se consigue sin la necesidad de comprobar el soporte de todos los posibles conjuntos de datos, debido a que, en cada paso, la lista de candidatos se reduce.

Reglas de asociación

Una vez identificados todos los conjuntos de artículos que existen en la base de datos con un soporte mínimo, se ha de definir un valor de la confianza mínima y localizar todas las reglas que cumplan este criterio, obteniéndose así el conjunto de reglas.

Por ejemplo, este algoritmo se puede evaluar sobre el conjunto de datos la tabla 6.1. El primer paso es fijar un valor mínimo de la confianza, por ejemplo, 55%, que se corresponde con todos aquellos conjuntos de artículos que aparezcan en 4 o más registros de los 7 (ya que 3/7 representa un 42% y 4/7 un 57% de los registros), estos son:

- {Leche}: 5
- {Pañales}: 5
- {Pan}: 6

El segundo paso consiste en buscar todas las veces que aparecen todos los pares de estos tres artículos y seleccionar aquellos que superan el soporte:

- {Leche, Pañales}: 4
- {Leche, Pan}: 4
- {Pañales, Pan}: 4

En este caso particular, los tres conjuntos de dos ítems pasan el criterio y por lo tanto se ha de continuar evaluando el único conjunto de ítems posible a partir del de dos:

- {Leche, Pañales, Pan}: 3

Como en esta iteración no hay ningún conjunto que supere el criterio de selección, el proceso de selección de conjuntos de ítems termina aquí. Se obtienen, entonces, 6 conjuntos de datos candidatos para la creación de reglas:

- {Leche}
- {Pañales}
- {Pan}
- {Leche, Pañales}
- {Leche, Pan}
- {Pañales, Pan}

En el último paso, se han de crear las reglas y obtener su confianza:

- {Leche} ⟹ {Pañales}: 4 de 5
- {Leche} ⟹ {Pan}: 4 de 5
- {Leche} ⟹ {Pañales, Pan}: 0 de 5
- {Pañales} ⟹ {Leche}: 4 de 5
- {Pañales} ⟹ {Pan}: 4 de 5
- {Pañales} ⟹ {Leche, Pan}: 0 de 5
- {Pan} ⟹ {Leche}: 4 de 6

- {Pan} ⟹ {Pañales}: 4 de 6
- {Pan} ⟹ {Leche, Pañales}: 0 de 6
- {Leche, Pañales} ⟹ {Pan}: 0 de 4
- {Leche, Pan} ⟹ {Pañales}: 0 de 4
- {Pañales, Pan} ⟹ {Leche}: 0 de 4

Aquí se ha de recordar que la intersección de los conjuntos de datos con los que se construyen las reglas de asociación ha de ser el conjunto vacío. Por esto no se han probado reglas de asociación como {Leche} ⟹ {Leche, Pan}.

Por ejemplo, en caso de que se fije una confianza del 75% solamente se tendrán cuatro reglas de asociación para este conjunto:

- Leche ⟹ Pañales: 4 de 5
- Leche ⟹ Pan: 4 de 5
- Pañales ⟹ Leche: 4 de 5
- Pañales ⟹ Pan: 4 de 5

Ahora es posible calcular los valores de las métricas que se han definido previamente. Por ejemplo, para la primera regla, Leche ⟹ Pañales, se observa que la leche aparece 5 veces, los pañales también aparecen 5 veces y la regla se verifica 4. Con esto se obtienen los valores:

- el soporte de la regla es 0,57 (4/7)
- la confianza es 0,8 ((4/7)/(5/7))
- la mejora de la confianza es 1,12 (4/7)/((5/7) * (5/7))
- la convicción es 1,42 ((1 - (5/7))/(1 – (4/5)))

Los cálculos se pueden repetir para el resto de las reglas obteniendo los resultados que se muestran en la tabla 6.2.

Regla	Soporte	Confianza	Mejora de la confianza	Convicción
Leche ⟹ Pañales	0,57	0,80	1,12	1,42
Leche ⟹ Pan	0,57	0,80	0,93	0,71
Pañales ⟹ Leche	0,57	0,80	1,12	1,41
Pañales ⟹ Pan	0,57	0,80	0,93	0,71

Tabla 6.2. Resumen de los valores de las reglas de asociación obtenidas.

Implementación del algoritmo Apriori

El algoritmo Apriori actualmente no se encuentra implementado en scikit-learn, por lo que es necesario utilizar algunas de las implementaciones disponibles en la red. Como ejemplo se utilizará una implementación que han publicado Everaldo Aguiar y Reid Johnson, de la Universidad de Notre Dame, basada en una versión previa de Marcel Caraciolo.

El código se puede encontrar en la página web Jupyter nbviewer,*The Apriori Algorithm*. (http://nbviewer.jupyter.org/github/cse40647/cse40647/blob/sp.14/10%20-%20Apriori.ipynb)

La implementación del algoritmo Apriori se encuentra en un archivo con el mismo nombre y extensión py, por lo que hay que invocarlo para importarlo al notebook. Para validar los cálculos realizados, previamente se puede utilizar la función apriori con el conjunto de datos. Asumiendo que los datos de la tabla 6.1. se encuentran en un array con nombre dataset se ha de escribir:

F, soporte = apriori.apriori(dataset, min_support = 0.55, verbose = True)

Con min_support se indica el soporte mínimo y con verbose se señala que se saquen los resultados por pantalla. En la variable F se guarda el listado de conjuntos de ítems y en la variable soporte se guarda además el soporte de cada conjunto. Al ejecutar esta línea se observan los resultados esperados:

{Leche}: sup = 0.714

{Pañales}: sup = 0.714

{Pan}: sup = 0.857

{Leche, Pañales}: sup = 0.571

{Pañales, Pan}: sup = 0.571

{Leche, Pan}: sup = 0.571

Para validar las reglas se ha de utilizar la función generate_rules, que necesita como entrada las dos salidas de la función anterior y la confianza mínima requerida. La forma de llamar a esta función es:

H = apriori.generate_rules(F, soporte, min_confidence=0.75, verbose = True)

Con min_confidence se indica la confianza mínima de las reglas y con verbose se señala que se saquen los resultados por pantalla, al igual que en el caso anterior. Al ejecutar la función, se vuelven a obtener los resultados esperados:

{Pañales} ---> {Leche}: conf = 0.8, sup = 0.571

{Leche} ---> {Pañales}: conf = 0.8, sup = 0.571

{Pañales} ---> {Pan}: conf = 0.8, sup = 0.571

{Leche} ---> {Pan}: conf = 0.8, sup = 0.571

V. Otros algoritmos asociativos

El algoritmo Apriori que se ha visto en la sección anterior ofrece buenos resultados, pero el esfuerzo necesario para la obtención de estos crece a medida que aumenta el tamaño de la base de datos. En cada una de las iteraciones para obtener los conjuntos de artículos candidatos es necesario recorrer la totalidad de la base de datos para contar la cantidad de apariciones de cada una de las combinaciones candidata. Cuando el tamaño de la base de datos crece y el soporte mínimo es bajo, esta búsqueda se convierte en un cuello de botella para el algoritmo.

Para solventar estos problemas en las bases de datos actuales, se han propuesto otros algoritmos con los que se puede reducir el número de búsquedas necesarias y reducir así el tiempo de cálculo. En esta sección se estudiarán algunas de estas alternativas, entre las que se encuentran:

- Eclat (*Equivalence Class Transformation*)
- FP-Growth

5.1. Eclat (*Equivalence Class Transformation*)

El algoritmo Eclat (ECLAT, *Equivalence Class Transformation*) ha sido propuesto en 1997 por Mohammed Javeed Zaki, Srinivasan Parthasarathy, M. Ogihara y Wei Li para intentar solucionar los problemas de rendimiento que se observan durante la búsqueda de conjuntos de artículos candidatos en Apriori cuando la base de datos aumenta de tamaño. En este algoritmo la búsqueda de candidatos se realiza creando una tabla en la que se almacenan los conjuntos de candidatos y el id del registro en el que se encuentran, al que se le suele denominar TID-list.

Los pasos que se deben seguir para la implementación de Eclat son los siguientes:

1. Obtener para todos los artículos disponibles en el conjunto de datos la lista de identificadores (TID-list) de las transacciones en las que aparecen durante la lectura de la base de datos.
2. En las TID-lists se seleccionan únicamente los conjuntos de artículos en los que el soporte se encuentra por encima del mínimo seleccionado.
3. Se calcula la intersección de todos los artículos para la crear las nuevas TID-lists de los conjuntos de pares de artículos y selecciona aquellos que superan el mínimo de soporte.
4. Se repite el paso 3 para crear las TID-lists para los conjuntos de n+1 artículos hasta que no se pueden obtener más.

La principal ventaja de este algoritmo es que solamente se ha de acceder una única vez a la base de datos para crear la tabla de TID-lists. En su contra se encuentra que el tamaño de las TID-lists puede llegar a ser demasiado grande como para poder ser gestionadas en memoria.

Otra ventaja es que es más fácil escalar este algoritmo, ya que ante la entrada de nuevos registros solamente hay que actualizar la tabla con los TID-list, sin necesidad de volver a recorrer la base de datos completa.

De forma análoga a la sección anterior, se puede utilizar el conjunto de datos de la tabla 6.1. para probar el algoritmo. El primer paso es crear una tabla con los TID-lists para todos los artículos, los resultados son los que se muestran en tabla 6.3.

Artículo	TID-lists
Pan	1, 2, 4, 5, 6, 7
Leche	1, 3, 4, 5, 6
Pañales	2, 3, 4, 5, 6
Cerveza	2, 3, 4
Huevos	2
Cola	3, 5, 7

Tabla 6.3. TID-lists para los conjuntos de un artículo.

En la sección anterior se había fijado como criterio de soporte mínimo un 55%, lo que equivale a 4 registros. Aplicando este mismo criterio, se han de eliminar de la tabla la cerveza los huevos y la cola, ya que las longitudes de sus TID-lists son de 3, 1, y 3 registros, respectivamente. Así, la tabla 6.3. se queda únicamente con las tres primeras filas.

Una vez que se ha aplicado el filtro, se han de crear todos los conjuntos de pares de artículos y calcular la intersección de los TID-lists de sus artículos. En este punto, para una implementación más eficaz, se ha de tener en cuenta que la intersección es conmutativa, por lo que solamente se ha de calcular una vez. Los resultados de los cálculos de la intersección se muestran en la tabla 6.4.

Artículos	TID-lists
{Pan, Leche}	1, 4, 5, 6
{Pan, Pañales}	2, 4, 5, 6
{Leche, Pañales}	3, 4, 5, 6

Tabla 6.4. TID-lists para los conjuntos de dos artículos.

En la tabla 6.4. se puede comprobar que las tres TID-lists de los conjuntos de pares de artículos posibles con los resultados filtrados de la tabla anterior tienen una longitud de cuatro artículos, por lo que superan el punto de corte. Ahora, simplemente, se ha de calcular la intersección de las TID-list de la tabla 6.4. con las de la tabla 6.3. El resultado es el que se muestra en la tabla 6.5.

Artículos	TID-lists
{Pan, Leche, Pañales}	4, 5, 6

Tabla 6.5. TID-lists para los conjuntos de tres artículos.

Finalmente, se ha llegado a la misma lista de candidatos a la que se había llegado utilizando el algoritmo Apriori. Pero en esta ocasión solamente ha sido necesario recorrer la base de datos una vez, a diferencia de las tres que han sido necesarias anteriormente. Esto se ha conseguido a costa de crear unas listas de identificadores (TID-lists) que pueden llegar a ser demasiado grandes y, por lo tanto, costosas de manipular en cuanto a memoria y tiempo de ejecución.

5.2. FP-Growth

El algoritmo FP-Growth ha sido propuesto por Jiawei Han, Jian Pei y Yiwen Yin en el año 2000. En su diseño se ha tenido en cuenta que recorrer la base de datos varias veces para contar los registros, como se hace en el Apriori, no es una forma eficiente de buscar la lista de candidatos para las reglas. Con la idea de evitar esta búsqueda se propuso la utilización de un árbol en el que se almacena la información necesaria para obtener el soporte de los conjuntos de artículos candidatos.

Los pasos que se deben seguir para la implementación de FP-Growth son los siguientes:

1. Al igual que en el Apriori, se cuenta el número de apariciones de todos los artículos, seleccionando únicamente los que superen un valor de soporte mínimo.
2. Los artículos seleccionados en el paso anterior se ordenan en función del número de apariciones que tienen en el conjunto de datos.
3. Se construye un árbol con el orden del paso anterior y se procede a añadir los registros de la base de datos, ordenándolos con los resultados del paso 2.
4. Para seleccionar una regla se ha de recorrer el árbol y parar cuando ya no se supere el umbral necesario.

Al igual que anteriormente, se pueden utilizar los datos de la tabla 6.1. para ejecutar este algoritmo. En caso de que el soporte se fije en 55 %, como en los ejemplos anteriores, los artículos que superan el corte son:

- {Leche}: 5
- {Pañales}: 5
- {Pan}: 6

Ahora se pueden ordenar en función del número de apariciones, de lo que resulta:

1. {Pan}: 6
2. {Leche}: 5
3. {Pañales}: 5

1

A partir de esto, se puede crear el árbol y volver a recorrer la base de datos para añadir los registros. Se empieza por añadir el primer registro del conjunto de datos [Pan, Leche], que es lo que se muestra en la figura 6.4.

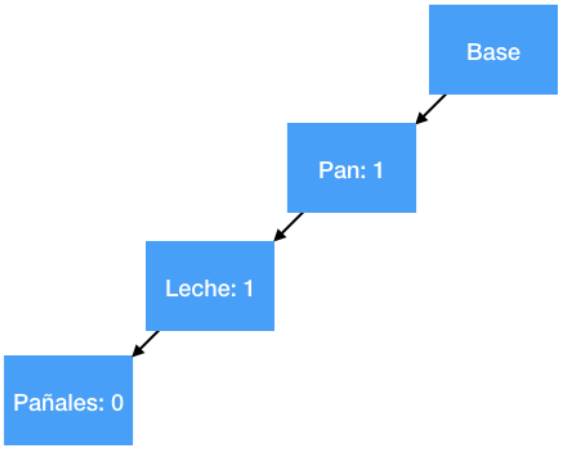

Figura 6.4. Creación del árbol del algoritmo FP- Growth.

2

El siguiente paso es añadir el segundo registro: [Pan, Pañales, Cerveza, Huevos], en el que solamente se han de tener en cuenta Pan y Pañales. Como el árbol está ordenado, se ha de crear una nueva rama que cuelga de Pan para apuntar los registros. Esto es lo que se ha creado en la figura 6.5.

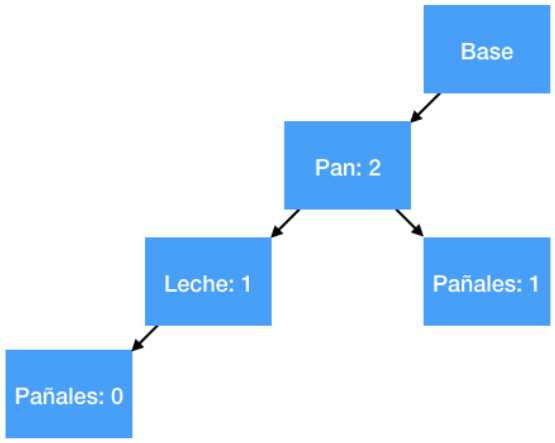

Figura 6.5. Árbol FP-Growth con los dos primeros registros.

3

El siguiente paso es añadir el tercer registro: [Leche, Pañales, Cerveza, Cola], del que solamente se ha de tener en cuenta los registros que superan el corte, es decir, Leche y Pañales.

Al igual que en el caso anterior, esta ruta no se encuentra en el árbol, por lo que se ha de construir, como se muestra en la figura 6.6.

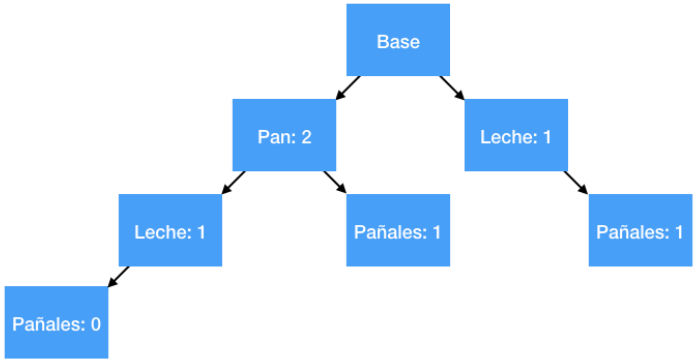

Figura 6.6. Árbol FP-Growth con los tres primeros registros.

4

Finalmente, se ha de añadir el resto, solamente teniendo en cuenta los registros que pasan el umbral y en el orden obtenido previamente, es decir:

- [Leche, Pan, Pañales, Cerveza] se añade a la ruta [Pan, Leche, Pañales]
- [Pañales, Pan, Leche, Cola] se añade a la ruta [Pan, Leche, Pañales]
- [Pan, Leche, Pañales] se añade a la ruta [Pan, Leche, Pañales]
- [Pan, Cola] se añade la ruta [Pan]

El resultado final de este proceso se muestra en el árbol de la figura 6.7., donde se puede ver un árbol con el número de veces que aparece cada uno de los registros.

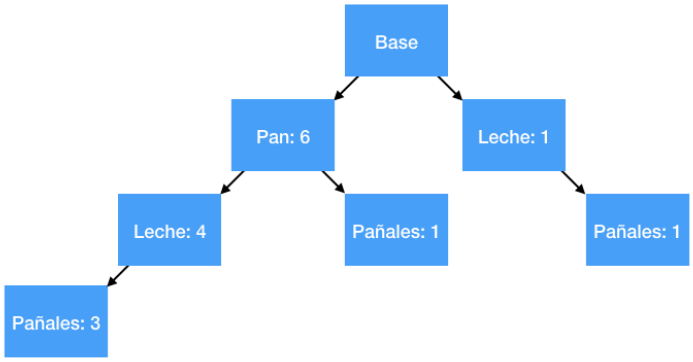

Figura 6.7. Árbol FP-Growth terminado.

5

El algoritmo no necesita volver a acceder a la base de datos, por lo que el número de accesos que se han necesitado han sido solamente dos, una ganancia considerable si se compara con lo que requiere el algoritmo Apriori.

Otra ventaja de este algoritmo sobre Apriori es que ya no es necesario calcular los conjuntos de artículos que se van a contar en cada paso, que es un proceso relativamente pesado, ya que esta información se guarda en el árbol.

Ahora, para terminar el árbol se ha de tener en cuenta que los productos de la rama principal se han de complementar con los de las otras ramas, esto es lo que se muestra en la figura 6.8.

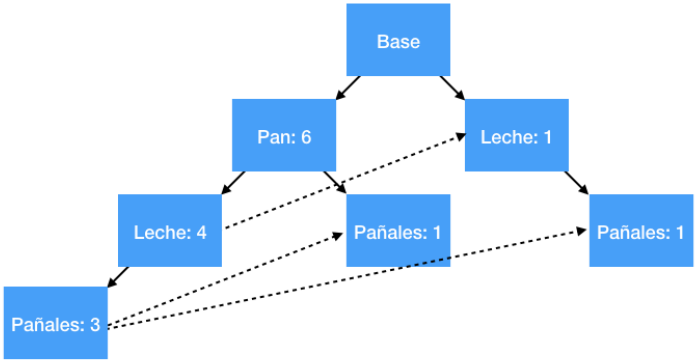

Figura 6.8. Fase final de la creación del árbol FP-Growth.

Para identificar los conjuntos de artículos candidatos para las reglas, hay que recorrer el árbol desde la base teniendo en cuenta el resto de ramas. Al recorrerlo, se comprueba si el artículo en el nodo alcanza en ese punto el valor mínimo, si es cierto, se añade la rama completa en la que se encuentra, en caso contrario, se deja de buscar.

Por ejemplo, comenzando por la base se selecciona [Pan] y se comprueba que este alcanza el valor mínimo, posteriormente se continua por la rama [Pan, Leche] y se verifica que también se llega al mínimo de soporte, que son 4 registros, pero al continuar por [Pan, Leche, Pañales] se comprueba que el mínimo es 3 y se para. Lo mismo sucede en la rama [Pan, Pañales], donde solamente se ha observado un registro. Luego se continua por la rama [Leche] y se obtiene [Leche, Pañales]. Al igual que en el caso de Apriori, las posibles rutas que se encuentran son:

- {Leche}
- {Pañales}
- {Pan}
- {Leche, Pañales}
- {Leche, Pan}
- {Pañales, Pan}

Algoritmo FP-Growth disponible en el paquete pyfpgrowth

El algoritmo FP-Growth se encuentra disponible en el paquete pyfpgrowth, el cual no forma parte de la instalación estándar de Anaconda, por lo que ha de instalarse antes de usarlo. Para esto se ha de ejecutar en la terminal el comando:

```
pip install pyfpgrowth
```

A partir de aquí hay que importar este paquete. Al igual que ocurría con el algoritmo Apriori, existen dos métodos importantes: uno para la obtención de los candidatos y un segundo para la obtención de las reglas. Los candidatos se pueden obtener con el código:

```
patterns = pyfpgrowth.find_frequent_patterns(dataset, 4)
```

donde se ha de pasar el conjunto de datos como primer parámetro y como segundo el número mínimo de registros. En esta implementación no se puede pasar el porcentaje y, por lo tanto, será necesario calcular este valor si está en porcentaje. Otra función que no tiene esta implementación es la posibilidad de sacar por pantalla los resultados, por lo que se han de imprimir para comprobar los mismos. Al ejecutar esta línea, se obtienen los resultados esperados:

```
{('Leche',): 5,

('Leche', 'Pan'): 4,

('Pan',): 6,

('Pan', 'Pañales'): 4,

('Pañales',): 5}
```

Posteriormente se ha de ejecutar la función para la obtención de las reglas de asociación. Esta es generate_association_rules y se le ha de pasar como primer parámetro la salida de la función anterior y, como segunda, la confianza mínima. Esto es, para obtener las reglas con una confianza mínima de 75% se ha de ejecutar:

```
rules = pyfpgrowth.generate_association_rules(patterns, 0.75)
```

Con lo que se obtienen los resultados esperados:

```
{('Leche',): (('Pan',), 0.8), ('Pañales,): (('Pan',), 0.8)}
```

VI. Resumen

En esta unidad se ha visto la forma en la que se pueden describir las relaciones de los eventos que suceden de forma conjunta en las bases de datos. Para esto se han utilizado las reglas de asociación.

En este punto se han estudiado los conceptos utilizados para identificar la capacidad de una regla de asociación:

- Soporte.
- Confianza.
- Mejora de la confianza.
- Convicción.

Posteriormente, se han estudiado diversos algoritmos que se pueden utilizar para la obtención de las reglas de asociación. Se ha comenzado por Apriori, uno de los algoritmos más populares debido a su simplicidad, pero cuyo rendimiento cae rápidamente a medida que se aumenta el tamaño de las bases de datos

Después, se han visto otros dos algoritmos propuestos después Apriori que reducen la cantidad de accesos necesarios a la base de datos. Estos son:

- Eclat (*Equivalence Class Transformation*).
- FP-Growth.

Ejercicios

Caso práctico

En el archivo "groceries.csv" se recogen los datos de las transacciones recogidas durante un mes en una tienda de comestibles (los datos han sido extraídos del paquete de R arules). En cada una de las filas de este archivo se encuentra una lista de artículos correspondiente a un tique de caja.

Hahsler, M., Buchta, C., Gruen, B. and Hornik, K. *arules: Mining Association Rules and Frequent Itemsets*. R package version 1.5-4. 2017. [En línea] URL disponible en: https://CRAN.R-project.org/package=arules.

También puedes descargar el siguiente archivo apriori.py

A partir de estos datos, obtén la lista de artículos que tienen un soporte mínimo de 0,15 utilizando tanto el algoritmo Apriori como FP-Growth. Posteriormente, obtén las reglas de asociación que se pueden deducir de este conjunto de datos con un soporte mínimo de 0,05 y confianza de 0,25.

Solución

El primer paso para solucionar este problema es importar el archivo, para esto se utiliza csv como se muestra en la figura 6.9.

```
import csv

groceries =  []
groceries_file = csv.reader(open("groceries.csv", "rb"))

for row in groceries_file:
    groceries.append(row)
```

Figura 6.9. Importación de los datos.

Posteriormente se puede obtener el listado con el conjunto de artículos que tienen un soporte mínimo de 0,15, para lo cual hay que importar Aprori y escribir:

F = apriori.apriori(groceries, min_support = 0.15, verbose = True)

Con lo que se obtienen los siguientes resultados:

{soda}: sup = 0.174

{whole milk}: sup = 0.256

{other vegetables}: sup = 0.193

{rolls/buns}: sup = 0.184

En el caso de FP-Growth, se ha de importar la librería pyfpgrowth. Como en la función find_frequent_patterns es necesario indicar el número de veces que debe aparecer el conjunto de datos, hay que calcular previamente cuál es este valor. Para ello, se necesita obtener el número de tiques en el conjunto de datos, esto es:

```
num_tickets = len(groceries)
```

Se obtiene un total de 9835. El 15% es 1475,25, por lo que ha de redondearse al entero inmediatamente superior, que es 1476. Esto se puede realizar utilizando la función ceil de la librería matemática de Phyton, esto es:

```
import math

num_tickets = len(groceries)

support_threshold = math.ceil(num_tickets * 0.15)
```

En este punto es importante destacar que se ha de utilizar el entero superior, en caso de que se utilice el entero inferior, es decir, 1475, el algoritmo podría obtener conjuntos de artículos que no superasen el corte, ya que es 14,99%. Como es de esperar, se obtienen los mismos resultados que en el caso anterior:

```
{('other vegetables',): 1903,

('rolls/buns',): 1809,

('soda',): 1715,

('whole milk',): 2513}
```

El código necesario para obtener las reglas de asociación utilizando arules es:

```
F, soporte = apriori.apriori(groceries, min_support = 0.05, verbose = False)

H = apriori.generate_rules(F, soporte, min_confidence = 0.25, verbose = True)
```

Se ha utilizado la opción verbose = False, dado que en caso contrario saldría la lista de candidatos que no se desea en esta ocasión, dada su longitud. Las reglas que se obtienen son:

```
{yogurt} ---> {whole milk}: conf = 0.402, sup = 0.056

{rolls/buns} ---> {whole milk}: conf = 0.308, sup = 0.057

{other vegetables} ---> {whole milk}: conf = 0.387, sup = 0.075

{whole milk} ---> {other vegetables}: conf = 0.293, sup = 0.075
```

El proceso se puede repetir con FP-Growth utilizando las líneas de código:

```
support_threshold = math.ceil(num_tickets * 0.05)

patterns = pyfpgrowth.find_frequent_patterns(groceries, support_threshold)

rules = pyfpgrowth.generate_association_rules(patterns, 0.05)

rules
```

Verificando que se obtienen los mismos resultados:

{('other vegetables',): (('whole milk',), 0.386757750919600),

('rolls/buns',): (('whole milk',), 0.30790491984521834),

('whole milk',): (('yogurt',), 0.2192598487863112),

('yogurt',): (('whole milk',), 0.40160349854227406)}

El código completo del caso se puede encontrar en elnotebook U7_caso.

Recursos

Bibliografía

- **“Mining Frequent Patterns Without Candidate Generation”.**: Han, J., Pei, J. y Yen, Y. En: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. SIGMOD 2000: 1–12
- **"Scalable algorithms for association mining".**: Zaki, M. J. En: IEEE Transactions on Knowledge and Data Engineering. 2000. 12 (3): 372–390.
- ***Building Machine Learning Systems with Python***. : Richert, W. y Coelho, L. P. Birmingham: Packt Publishing; 2013.
- ***Machine Learning in Action***. : Harrington, P. Nueva York: Manning Publication; 2012
- ***New Algorithms for Fast Discovery of Association Rules*** : Zaki, M. J., Parthasarathy, S., Ogihara, M. y Li, W. KDD; 1997.
- ***Parallel Algorithms for Discovery of Association Rules*** : Zaki, M. J., Parthasarathy, S., Ogihara, M. y Li, W. Data Min. Knowl. Discov. 1(4): 343-373 (1997)

Glosario.

- **Antecedente**: En una regla de asociación, es el conjunto de elementos que disparan la regla.
- **Confianza**: Porcentaje en el que se verifica una regla de asociación en relación al antecedente.
- **Convicción**: Es el ratio de la frecuencia esperada para el antecedente que sucede sin el precedente.
- **Mejora de la confianza**: Es la fracción del soporte observado en una regla de asociación respecto al teórico, suponiendo independencia entre el antecedente y el precedente.
- **Precedente**: En una regla de asociación, es el conjunto de elementos que son implicados por el antecedente.
- **Regla de asociación**: Mide la fuerza con la que dos eventos ocurren al mismo tiempo en un conjunto de datos
- **Soporte**: Es el porcentaje de transacciones en las que aparece un conjunto de elementos o una regla de asociación.