UNIVERSIDAD DE GRANADA

TRABAJO DE TEORÍA

Análisis de secuencias. Algoritmo AprioriAll

Tratamiento Inteligente de Datos

Máster Profesional en Ingeniería Informática

Curso académico 2022/2023

Autor

Ramón García Verjaga (rgarver@correo.ugr.es)

José Alberto Gómez García (modej@correo.ugr.es)

Introducción

Las reglas de asociación son un mecanismo que permite expresar la existencia de patrones entre diversos conjuntos de datos. Estos patrones nos permiten obtener información inicialmente oculta entre la gran cantidad de datos que podamos haber recopilado con el paso del tiempo.

Estas reglas nos pueden ayudar a la hora de tomar decisiones estratégicas en el marco de un negocio o producto, de manera que intentemos aumentar la rentabilidad o el beneficio que obtenemos del mismo.

Un par de ejemplos de uso de reglas de asociación son:

- el descubrimiento de los recorridos más frecuentes que se producen a la hora de navegar entre las diferentes páginas de un sitio web;
- o conocer qué productos suelen comprarse juntos en un supermercado o tienda online, de manera que se pueda optimizar la asignación de productos a estantes, o establecer técnicas de marketing para vender mejor los productos.

Este último ejemplo ya nos resulta familiar a los estudiantes de la asignatura, pues se encuentra entre el conjunto de prácticas que debemos resolver. Cabe destacar que las reglas de asociación surgieron precisamente con este objetivo, el análisis de las cesta de la compra.

A veces, únicamente tenemos en cuenta las transacciones sin tener en cuenta el posicionamiento temporal-espacial de las mismas. Cuando dispongamos de marcas de tiempo que nos indiquen cuándo se han realizado las transacciones sería interesante analizarlas e intentar extraer conocimiento de las mismas. Esto es lo que pretendemos con el análisis de secuencias.

¿Qué es una regla de asociación?

Una regla de asociación es una declaración del tipo:

«**SI** *x* **ENTONCES** *y*»

Es decir, es una proposición sobre la ocurrencia de cierta situación con una determinada probabilidad asociada, en una base de datos de tipo transaccional.

Al conjunto que representa la ***x*** se le denomina antecedente y al conjunto que representa la ***y*** se le llama consecuente.

Un ejemplo de regla de asociación podría ser:

«**SI** *condimentos* **ENTONCES** *carne***»**

Aunque se expresan de forma similar, no debemos confundir estas proposiciones con aquellas propias de la lógica proposicional (utilizadas en reglas de clasificación).

En este caso, y haciendo alusión al ejemplo de la cesta de la compra, lo que queremos representar es que dos artículos se compran juntos (**co-ocurrencia**), no que la compra del primero implique la compra del segundo.

Además, en la parte derecha de la proposición puede aparecer más de un único atributo. La restricción que se exige es que X e Y sean conjuntos de ítems (ítemsets).

Por lo general, solemos utilizar las reglas de asociación sobre atributos binarios. En el caso de la cesta de la compra, nos limitamos a indicar si un producto se ha comprado o no, ignorando la cantidad.

Existen familias de reglas de asociación que trabajan atributos multivalorados, o que utilizan rangos numéricos.

Otro tipo de reglas de asociación son aquellas que consideran relaciones entre distintos instantes temporales, o en una secuencia. Estas las abordaremos en las siguientes secciones con mayor detalle.

Conceptos básicos

Para entender el funcionamiento de algunos algoritmos que abordaremos posteriormente, necesitamos conocer los siguientes **conceptos básicos**:

La siguiente tabla de datos es nuestra **base de datos de transacciones**. Cada **transacción** contiene un **identificador** único y un **conjunto de ítems**.

Id	**Ítemsets**
1	{leche, pan}
2	{pan, mantequilla}
3	{cerveza}
4	{leche, pan, mantequilla}
5	{pan}

En primer lugar, existen varios elementos que conforman nuestro conjunto de datos:

- **Ítem**: es cada uno de los artículos que forman parte de una transacción. Por ejemplo: leche.
- **Transacción**: es cada una de las operaciones o acciones sobre las que obtenemos un identificador asociado a un conjunto de ítems (ítemset). Por ejemplo: <1, {leche, pan}>.
- **Ítemset**: es un conjunto de uno o más ítems. Por ejemplo: {leche, pan y mantequilla}.

- **K-ítemset**: es un conjunto de ítems que posee k elementos. Por ejemplo: los 2-ítemsets de nuestro conjunto de datos son: {leche, pan} y {pan, mantequilla}.

Con respecto a nuestro conjunto de datos, existen varios atributos:

Soporte (también conocido como cobertura): para un conjunto de ítems ***X*** se define como la proporción de transacciones que contiene dicho conjunto de ítems. Por ejemplo: la cobertura del ítem leche es de 2 (transacciones diferentes en las que aparece) entre 5 (transacciones totales que existe), que es igual a 0,4.

Itemset frecuente: es aquel cuyo soporte es igual o superior a un umbral establecido de antemano.

Un **ítemset frecuente** es **maximal** a un cierto nivel de soporte mínimo si es frecuente y ningún superconjunto del ítemset es frecuente.

Medidas de efectividad de las reglas de asociación

La efectividad de una regla de asociación determinada se mide por dos parámetros principales: el soporte y la confianza.

- El **soporte** se refiere a la frecuencia relativa con la que una determinada regla aparece en la base de datos que se está analizando.
- La **confianza** se refiere a la fiabilidad o soporte de una regla de asociación determinada. Se define como la proporción de casos en los que la regla de asociación es cierta o, en otras palabras, el porcentaje de veces que los elementos del antecedente (la parte «si» de la regla) aparecen en la misma transacción que los elementos del consecuente (la parte «entonces» de la regla).

Si tuviéramos la siguiente regla:

«**SI** *pan* **ENTONCES** *leche*»

Tendríamos los siguientes valores de efectividad:

- **Soporte** (Regla - **SI** *pan* **ENTONCES** *leche*): como hay 5 transacciones y el ítemset {pan, leche} aparece dos veces, tenemos un soporte para la regla de 2 entre 5 que es 0,4 o el 40%.
- **Confianza** (Regla - **SI** *pan* **ENTONCES** *leche*): Como {pan}, ítemset del antecedente, aparece en 4 transacciones y {leche}, ítemset del consecuente, solamente aparece en 2 de esas 4 transacciones, tenemos una confianza para la regla de 2 entre 4 que es 0,5 o el 50 %.

Generalmente, nos interesan reglas con un soporte mayor al 30 % y una confianza mayor al 70 %.

También, tenemos otra medida denominada **lift**, que mide la correlación entre antecedente y consecuente. Es decir, mide hasta qué punto ocurren el antecedente y el consecuente conjuntamente más o menos de lo esperado si fuesen independientes.

Casos de uso

Sistemas de recomendación: Los sistemas de recomendación utilizan, principalmente, técnicas como el filtrado colaborativo y las técnicas basadas en el contenido para recomendar nuevos usuarios. Los datos históricos recopilados sobre los usuarios anteriores pueden utilizarse para extraer reglas de asociación, las cuales tienen en cuenta la temporalidad en los eventos ocurridos. La minería de reglas de asociación puede utilizarse para aumentar significativamente la potencia de los sistemas de recomendación basados en contenidos y en filtrado colaborativo.

Mejora de los sitios web: Los weblogs son los archivos más importantes para conocer el comportamiento de los usuarios en un sitio web. El algoritmo Apriori puede utilizarse en los weblogs para conocer el comportamiento de los usuarios y tomar decisiones estratégicas sobre la interfaz de usuario. Esto tiene un gran impacto en la experiencia del usuario y podría aumentar el número de usuarios.

Medicina: Los médicos pueden utilizar reglas de asociación para servirse de apoyo en el diagnóstico de un paciente. Hay muchas variables a tener en cuenta a la hora de hacer un diagnóstico, ya que muchas enfermedades comparten síntomas. Mediante el uso de reglas de asociación los médicos pueden determinar la probabilidad condicional de una enfermedad determinada comparando las relaciones entre síntomas en los datos de casos anteriores.

Comercio. Los comerciantes pueden recopilar datos sobre patrones de compra, registrando los datos de compra a medida que los clientes van realizando compras. Los modelos pueden buscar coincidencias en estos datos para establecer reglas de asociación y determinar qué productos es más probable que se compren juntos. De esta forma, los comerciantes pueden ajustar las estrategias de marketing y ventas para aprovechar esta información.

Patrones de secuencias

Si nos paramos a pensar, los datos de la cesta de la compra suelen contener información temporal sobre el momento en que uno o varios artículos fueron comprados por los clientes. Esta información puede utilizarse para construir la secuencia de transacciones realizadas por un cliente en un determinado periodo de tiempo.

Del mismo modo, los datos basados en eventos recogidos en experimentos científicos o en la supervisión de sistemas físicos, como redes de telecomunicaciones, redes informáticas y redes de sensores inalámbricos, tienen una secuencia inherente.

Esto significa que existe una relación ordinal, normalmente basada en la precedencia temporal o espacial, entre los datos basados en sucesos.

Sin embargo, las reglas de asociación que hemos visto hasta el momento sólo enfatizan las relaciones de co-ocurrencia y no tienen en cuenta la información secuencial de los datos.

La información secuencial de los datos puede ser valiosa para identificar características recurrentes de un sistema dinámico o para predecir futuras ocurrencias de eventos de un sistema dinámico o predecir la ocurrencia futura de determinados acontecimientos.

Análisis de secuencias

Una secuencia es una lista ordenada de elementos:

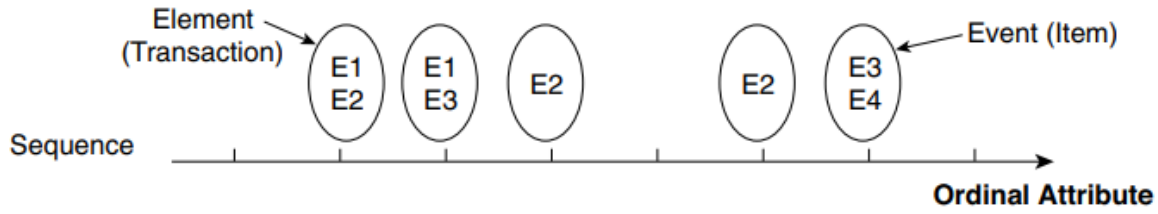
$$s = < e_1, e_2, \ldots, e_n >$$

Donde:

- Cada elemento contiene una colección de eventos: ei = $\{i_1, i_2, \ldots, i_k\}$
- Cada elemento tiene una localización temporal asociada.

La longitud de la secuencia es el número de elementos de la secuencia

Una k-secuencia es una secuencia de k eventos.



Algunos ejemplos de secuencias:

- ➔ **Visitas de una página web**: <{Página de inicio}, {Electrónica}, {Tablets}, {iPad}, {Shopping cart}, {Order confirmation}, {Volver al carrito}>
- ➔ **Préstamos de libros de una biblioteca**: <{Harry Potter y la Piedra Filosofal}, {Harry Potter y la Cámara Secreta}, {Harry Potter y el Prisionero de Azkaban}>

Como vemos en los ejemplos, la localización espacial de cada uno de los eventos tiene mucho sentido. Por una parte, en el ejemplo de la compra, … Por otra parte, en el ejemplo del préstamo, es interesante ver cómo se van prestando los libros en orden temporal en relación a la posición de los mismos en la saga, el primero, posteriormente, el segundo y, más tarde, el tercero.

Base de datos	Secuencia	Elemento (Transacción)	Evento (Ítem)
Clientes	Historial de compras de un cliente determinado	Conjunto de artículos comprados por un cliente en un instante concreto	Libros, productos…
Web	Navegación de un visitante del sitio web	Colección de ficheros vistos por el visitante tras un único clic de ratón	Página inicial, información de contacto, fotografía…
Eventos	Eventos generados por	Eventos generados por un	Tipos de alarmas

	un sensor	sensor en un instante t	generadas
Genoma	Secuencia de ADN	Elemento de la secuencia de ADN	Bases A, T, G, C

Decimos que una secuencia $<a_1, a_2, \ldots, a_n>$ está contenida en otra secuencia $<b_1, b_2, \ldots, b_m>$ si existe un conjunto de enteros $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \ldots, a_n \subseteq b_{in}$.

Secuencia	Subsecuencia	¿Incluida?
<{2, 4}, {3, 5, 6}, {8}>	<{2}, {3, 5}>	Sí
<{1, 2}, {3, 4}>	<{1}, {2}>	No
<{2,4}, {2,4}, {2,5}>	<{2}, {4}>	Sí

El **soporte** de una subsecuencia S se define como la fracción de secuencias de la base de datos que incluyen la subsecuencia S.

Un **patrón secuencial** es una **subsecuencia frecuente** (esto es, una subsecuencia con ***soporte ≥ MinSupp***)

La extracción de patrones patrones secuenciales nos plantea el problema a través del cual dados:

- una base de datos de secuencias y
- un umbral de soporte mínimo MinSupp,

encontrar todas las subsecuencias frecuentes (con soporte ≥ MinSupp).

Datos para obtención de reglas de asociación secuenciales

Las reglas de asociación secuenciales son aquellas que permiten expresar patrones de comportamiento que se dan en diferentes instantes temporales, pero que no suelen ser muy lejanos entre sí.

Estas reglas de asociación son interesantes para la recomendación de artículos que a un cliente le podrían interesar en función de su historial de compras o búsquedas, u organizar la logística de reposición y distribución de artículos. Este tipo de reglas de asociación son las responsables de que tras comprar, por ejemplo, un libro en Amazon, nos recomienden otros títulos que otros clientes compraron junto al nuestro o poco después.

Para la generación de reglas de asociación debemos trabajar con bases de datos que contengan algún tipo de identificador de usuario o cliente y algún identificador temporal que nos permita deducir en qué orden se realizaron las compras.

Una base de datos de este tipo podría tener una estructura similar a la mostrada en la siguiente tabla:

ID-Cliente	Inst-Temporal	Items
1	15/03/2003	{23, 56}
1	17/03/2003	{42, 13}
1	18/03/2003	{45, 33}
2	12/03/2003	{12, 13}
2	18/03/2003	{23, 34, 5, 8}

Los ítems son representados como enteros, los cuales corresponden a un identificador único que los describen, como podrían ser los códigos de barras.

Como hemos comentado anteriormente, una secuencia es una lista ordenada de elementos (transacciones). Denotaremos una secuencia como $<e_1, e_2, \ldots, e_n>$ donde e_i es una transacción, y cada colección de eventos (itemset) estará denotado como $\{i_1, i_2, \ldots, i_n\}$ donde i_i es un evento (ítem).

Así, si agrupamos las transiciones por el ID del cliente, dejando de tener la columna correspondiente al instante temporal, obtendremos esta otra base de datos:

ID-Cliente	Secuencia
1	< {23, 56}, {42, 13}, {45, 33} >
2	< {12, 13}, {23, 34, 5, 8} >

También, como ya hemos explicado, una secuencia $< a_1, a_2, \ldots, a_n >$ está contenida en otra secuencia $< b_1, b_2, \ldots, b_n>$ si existen valores enteros tales que $i_1 < i_2 < \ldots < i_n$ y $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \ldots, a_n \subseteq b_{in}$.

De esta manera, la subsecuencia < {23}, {42}, {33} > está contenida en la secuencia del cliente 1, en tanto que cada elemento de cada itemset está en los itemset de la secuencia del cliente y se respeta el orden.

De otra manera, la subsecuencia < {45}, {42} > no está contenida en la secuencia del cliente 1. A pesar de que cada itemset se encuentra contenido en los itemset de la secuencia, no se respeta el orden temporal. En cambio, la subsecuencia < {42}, {45} > sí sería válida.

Tampoco sería válida la subsecuencia < {42, 33} >, pues cada elemento de esta corresponden a itemsets distintos de la secuencia 1.

Principio del algoritmo Apriori

Selección de candidatos

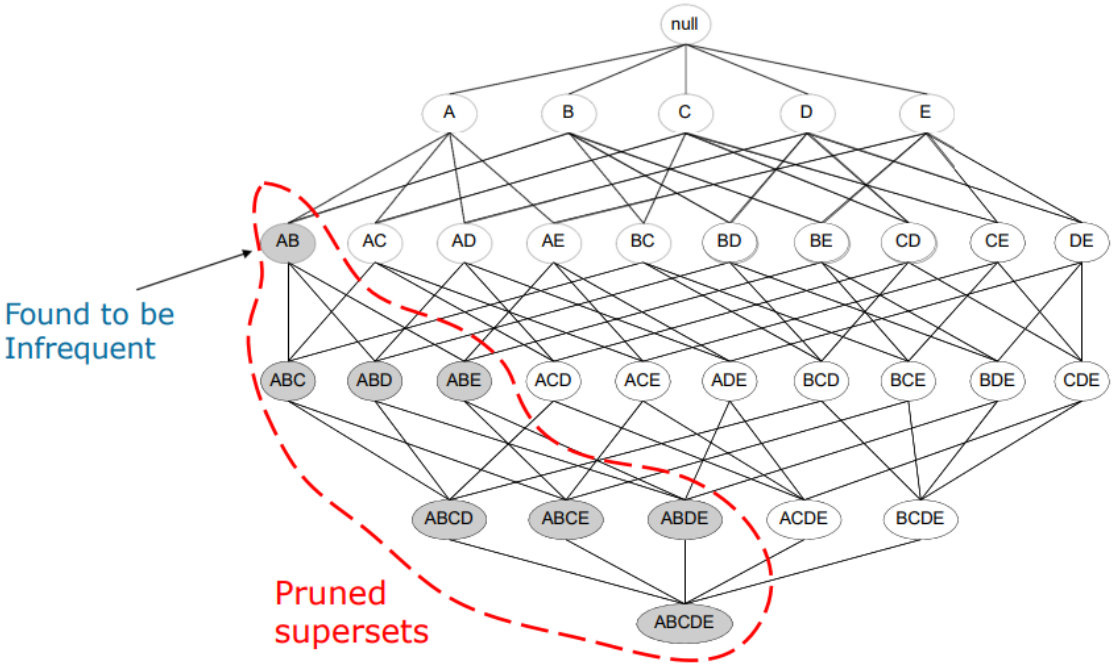
Si un conjunto de elementos es frecuente, todos sus subconjuntos también deben serlo.

El principio de Apriori se mantiene debido a la siguiente propiedad de la medida de soporte:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq S(Y)$$

El soporte de un itemset nunca supera el soporte de sus subconjuntos.

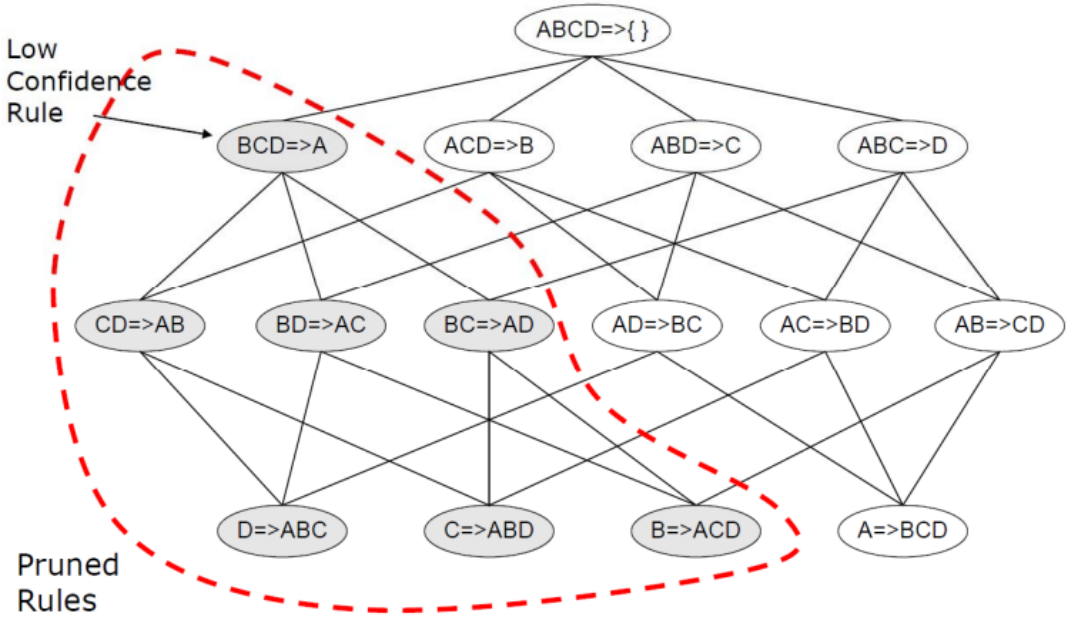
Esto se conoce como la **propiedad anti-monótona** del **soporte** (si un conjunto no pasa una prueba, ninguno de sus superconjuntos la pasan).



Generación de reglas

Es importante destacar que la **confianza** no tiene una **propiedad anti-monótona**. Pero la confianza de las **reglas generadas** a partir del **mismo conjunto** de elementos tiene una propiedad **anti-monótona**. Por ejemplo:

$$\text{Si } L = \{A, B, C, D\}, \text{ entonces } c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$



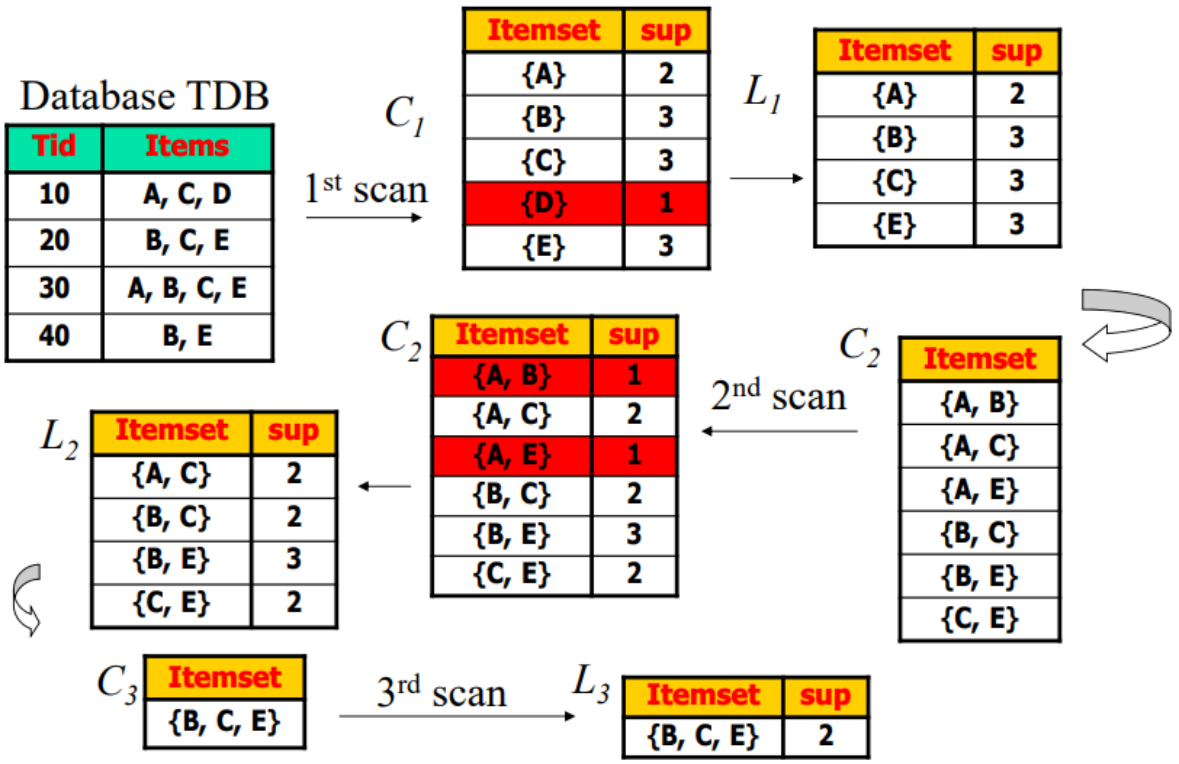
La regla candidata se genera combinando dos reglas que comparten el **mismo prefijo** en el **consecuente** de la **regla**.

- Combinar CD => AB y BD=>AC daría lugar a la regla candidata D => ABC.

Ejemplo del algoritmo Apriori para selección de candidatos

Antes de comenzar a aplicar de forma práctica lo visto teóricamente, nos podemos preguntar cuál sería la forma más adecuada de seleccionar el soporte mínimo. Tenemos que tener en cuenta que si es demasiado alto podríamos perder itemsets que involucren ítems interesantes y poco frecuentes; si es demasiado bajo, el problema va a ser muy costoso a nivel computacional, porque va a existir un gran número de ítemsets; por lo anterior, debemos ajustar el valor según el problema e, incluso, la fase del mismo en que nos encontremos. Un valor común sería el 30 %, combinado con un 70 % de confianza.

En el caso de nuestro ejemplo se elige un valor para el soporte de 2, que se corresponde con el 50 %.



En primer lugar, poseemos nuestra base de datos de transacciones. A partir de la misma, calculamos el soporte para los 1-ítemsets (en este caso el número de veces que aparece cada ítemset en las transacciones). Más tarde, nos quedamos con los 1-ítemsets cuyo soporte sea mayor o igual a 2, es decir, al 50 % (2 apariciones en 4 transacciones). A partir del conjunto resultante, formamos los 2-ítemsets y calculamos el soporte de cada uno de ellos. Nos quedamos con los 2-ítemsets cuyo soporte sea mayor o igual a 2 y repetimos el proceso. Formamos los conjuntos 3-ítemsets con el conjunto resultante. Como vemos, solo podemos formar uno ya que el resto no se encuentran en las transacciones. Calculamos el soporte para dicho conjunto y vemos que el soporte es 2.

Generalized Sequential Patterns (GSP)

Es un algoritmo utilizado para la minería de secuencias que se basa principalmente en el algoritmo Apriori. En primer lugar, descubrimos todos los elementos frecuentes por niveles. Simplemente significa contar las apariciones de todos los elementos simples en la base de datos. A continuación, filtramos las transacciones eliminando los elementos no frecuentes. Al final de este paso, cada transacción consta únicamente de los elementos frecuentes que contenía originalmente. Esta base de datos modificada se convierte en la entrada del algoritmo GSP.

Fase 1

Recorrer la base de datos para obtener todas las secuencias frecuentes de 1 elemento.

Fase 2

Mientras se encuentren nuevas secuencias frecuentes:

1. **Generación**: Generar k-secuencias candidatas a partir de las (k-1)-secuencias frecuentes.
2. **Poda**: Podar k-secuencias candidatas que contengan alguna (k-1)-secuencia no frecuente.
3. **Conteo**: Recorrer nuevamente el conjunto de datos para obtener el soporte de las candidatas.
4. **Eliminación**: Eliminar las k-secuencias candidatas cuyo soporte real esté por debajo de MinSupp.

```
F1 = the set of frequent 1-sequence
k=2,
do while Fk-1 != Null;
    Generate candidate sets Ck (set of candidate k-sequences);
    For all input sequences s in the database D
    do
        Increment count of all a in Ck if s supports a
    End do
    Fk = {a ∈ Ck such that its frequency exceeds the threshold}
    k = k+1;
End do
Result = Set of all frequent sequences is the union of all Fk's
```

El algoritmo anterior se parece al algoritmo Apriori. Sin embargo, una **diferencia** principal es la **generación de conjuntos candidatos**.

Supongamos que A → B y A → C son dos secuencias frecuentes de dos elementos. Los elementos implicados en estas **secuencias** son **(A, B)** y **(A, C)** respectivamente. La generación de candidatos en un estilo **Apriori** habitual daría **(A, B, C)** como un conjunto de 3 elementos, pero en el contexto actual obtenemos las siguientes **3 secuencias** como resultado de unir las 2 secuencias anteriores: **A → B → C**, **A → C → B** y **A → BC**.

La fase de generación de candidatos tiene esto en cuenta. El algoritmo **GSP** descubre **secuencias frecuentes**, teniendo en cuenta **restricciones temporales** como la separación máxima y la separación mínima entre los elementos de la secuencia.

Además, admite la noción de **ventana deslizante**, es decir, de intervalo de tiempo dentro del cual se observa que los elementos pertenecen al mismo suceso, aunque procedan de sucesos distintos.

Algoritmo AprioriAll

El algoritmo AprioriAll es una evolución del algoritmo Apriori, comentado en clase de teoría, cuya finalidad es la de generar reglas de asociación secuenciales y los conceptos mencionados en la sección anterior. Este algoritmo fue propuesto por Rakesh Agrawal y Ramakrishnan Skirant en 1995.

```
L1 = {large 1-sequences}
for (k = 2; Lk-1 ≠ {}; k++) do
   begin
   Ck = New candidates generated from Lk-1
   foreach customer-sequence c in the database do
      Increment the count of all candidates in Ck
         that are contained in c.
   Lk = Candidates in Ck with minimum support.
   end
Answer = Maximal Sequences in Lk

Notation:
Lk: Set of all large k-sequences
Ck: Set of candidate k-sequences
```

Este algoritmo se divide en cinco fases, las cuales pasamos a comentar a continuación:

1. **Ordenación**. La base de datos se debe ordenar haciendo uso de los identificadores únicos de clientes. Posteriormente, ordenamos en función del instante temporal de forma ascendente, de manera que transacciones más antiguas sean las primeras. Este paso implica convertir una base de datos de transacciones en una base de datos de secuencias de clientes.
2. **Selección de conjuntos de ítems.** En segundo lugar, y para cada cliente, seleccionamos los conjuntos de ítems que tienen una cobertura mínima que hayamos seleccionado de antemano. La cobertura es calculada con respecto a la presencia de un itemset en la secuencia de cada cliente.
3. **Transformación y renombramiento.** Cada secuencia se transforma de manera que sólo tenga ítems frecuentes. Hecho esto, cada conjunto de ítems frecuentes en cada secuencia es renombrado con un identificador único.
4. **Construcción de secuencias frecuentes.** A partir del conjunto de ítems frecuentes, se construye iterativamente el conjunto de secuencias que cumplan con el criterio de cobertura.
5. **Selección de secuencias maximales.** Se filtra el conjunto de secuencias frecuentes de manera que no haya subsecuencias partiendo desde las secuencias de mayor tamaño.

Para entender mejor el funcionamiento del algoritmo veamos un ejemplo de aplicación del mismo.

Supongamos que tenemos una base de datos como la mostrada en la siguiente tabla:

ID-Cliente	Inst-Temporal	Items
1	15/03/2003	{30}
1	17/03/2003	{90}

2	18/03/2003	{10, 20}
2	18/03/2003	{30}
2	18/03/2003	{40, 60, 70}
3	18/03/2003	{30, 50, 70}
4	13/03/2003	{30}
4	15/03/2003	{40, 70}
4	17/03/2003	{90}
5	14/03/2003	{90}

El **primer paso** del algoritmo nos indica que debemos agrupar las transacciones a partir del ID único de cliente, de manera que si lo hacemos obtendremos esta otra tabla:

ID-Cliente	**Secuencia**
1	< {30} ; {90} >
2	< {10, 20} ; {30} ; {40, 60, 70} >
3	< {30, 50, 70} >
4	< {30} ; {40, 70} ; {90} >
5	< {90} >

En el **segundo paso**, deberemos hallar los conjuntos de ítems con un mínimo de cobertura. El mínimo que vamos a usar es del 40%, de manera que para que un conjunto de ítems se considere frecuente debe encontrarse soportado por un mínimo de 2 clientes de los 5 que hemos supuesto. Una vez hayamos encontrado los conjuntos frecuentes, les haremos corresponder un identificador único. De esta manera, obtenemos los siguientes conjuntos de ítems frecuentes e identificadores:

- ➔ {30} con ID = 1.
- ➔ {40} con ID = 2.
- ➔ {70} con ID = 3.
- ➔ {40, 70} con ID = 4.
- ➔ {90} con ID = 5.

El lector puede que se pregunte por la inclusión del conjunto {40, 70}. Nótese que está soportado por el cliente 4 (aparece tal cual este conjunto) y por el cliente 2, dado que en la última transacción se compraron los ítems 40 y 70, en ese orden, y la no compra del ítem

60 no rompe la propiedad de orden especificada anteriormente cuando definimos las subsecuencias.

En el **tercer paso**, transformamos cada secuencia de manera que esta contenga sólo los conjuntos de ítems frecuentes. Posteriormente, los renombramos con los identificadores únicos vistos antes.

ID-Cliente	Secuencia	Secuencia transformada	Secuencia final renombrada
1	< {30} ; {90} >	< {30}; {90} >	< {1} ; {5} >
2	< {10, 20} ; {30} ; {40, 60, 70} >	< {30}; {{40} , {70} , {40, 70}} >	< {1} ; {2, 3, 4} >
3	< {30, 50, 70} >	< {{30} , {70}} >	< {1, 3} >
4	< {30} ; {40, 70} ; {90} >	< {30}; {{40}, {70}, {40,70}}, {90} >	< {1} ; {2, 3, 4} ; {5} >
5	< {90} >	< {90} >	< {5} >

Nótese que itemsets como {40, 60, 70} contienen tres subconjuntos frecuentes, el propio 40, el propio 70, y el conformado por ambos. De ahí que la secuencia transformada sea más larga y tenga conjuntos agrupados entre unas mismas llaves; que cuando son renombradas corresponden a “itemsets normales”, sin anidamiento.

Una vez obtenidas las secuencias finales renombradas pasamos a generar las secuencias que cumplan el requisito mínimo de cobertura especificado al principio del ejemplo, 2 clientes.

Este proceso se realiza de igual manera que se hace en el algoritmo Apriori “original”, teniendo en consideración las secuencias finales tras el renombramiento. Recordemos que este algoritmo hacía uso de la propiedad “apriori”, que nos decía que si un itemset no es frecuente entonces todo itemset que lo contenga tampoco es frecuente, para podar ramas y tener que realizar menos cálculos.

En la siguiente tabla resumen se muestran las secuencias generadas que superan la cobertura del 40% indicada.

Tamaño 1		Tamaño 2		Tamaño 3		Tamaño 4	
Seq.	**Soporte**	**Seq.**	**Soporte**	**Seq.**	**Soporte**	**Seq.**	**Soporte**
< 1 >	4	< 1, 2 >	2	< 1, 2, 3 >	2	< 1, 2, 3, 4 >	2
< 2 >	2	< 1, 3 >	3	< 1, 2, 4 >	2		
< 3 >	3	< 1, 4 >	2	< 1, 3, 4 >	2		

< 4 >	2	< 1, 5 >	2	< 2, 3, 4 >	2		
< 5 >	3	< 2, 3 >	2				
		< 2, 4 >	2				
		< 3, 4 >	2				

Finalmente, en el **quinto y último paso**, obtenemos las secuencias maximales, al igual que en el algoritmo Apriori "original". En el caso del algoritmo AprioriAll, también se podrían deshacer los mapeos que hicimos, de manera que sepamos qué productos son los que se encuentran en las secuencias generadas.

En este ejemplo, obtenemos dos secuencias maximales.

- ➔ < 1, 2, 3 , 4 > → < {30}, {40}, {70}, {40 70} >
- ➔ < 1, 5 > → < {30}, {90} >

A partir de las secuencias maximales, podríamos generar las diversas reglas de asociación, colocando unas y otras combinaciones de elementos en antecedente y consecuente. Eso sí, respetando siempre el orden de la secuencia. Algunas reglas generadas podrían ser:

- ✔ Si 30 entonces 40.
- ✔ Si 40 entonces 40, 70.
- ✔ Si 30 y 40 entonces 40 y 70.
- ✔ Si 30, 40 y 70; entonces 40 y 70.
- ✔ Si 30 entonces 90.

Cada una de estas reglas tendrá asociada una determinada confianza, será el umbral de confianza mínima que hayamos establecido el que nos dirá qué reglas sobreviven al último proceso de criba.

Conclusión

Hemos recordado lo que son las reglas de asociación, el uso que se les puede dar, la relación que tienen con el análisis de secuencias y cómo pueden ayudar a tomar determinadas decisiones.

Hemos aprendido lo que son los patrones de secuencias y la importancia que tienen para representar la espacio-temporalidad en relación a datos transaccionales.

Hemos expuesto el algoritmo AprioriAll para análisis de secuencias explicando las diferentes fases en las que está estructurado y ejemplificando cada una de ellas.

Hemos nombrado otras técnicas para análisis de secuencias.

En definitiva, tener en cuenta la temporalidad de los eventos a la hora de obtener información a través de la generación de reglas de asociación es muy importante. Para

multitud de acciones, como hemos podido ver, es importante saber en qué orden suceden los eventos y qué relación tienen los unos con los otros.

Bibliografía

[1] P-N. Tan, M. Steinbach and V. Kumar. Introduction to Data Mining. Pearson, 2005. Capítulo 7.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 1995, pp. 3-14.

[3] P. Fournier-Viger, J.C.W. Lin, R.U. Kiran, Y.S. Koh. A survey in sequential pattern mining, Data Science and Pattern Recognition. 2017.

[4] José Hernández Orallo, M.José Ramírez Quintana, Cèsar Ferri Ramírez. Introducción a la Minería de Datos. Pearson, 2004. Capítulo 9.

[5] Fernando Berzal. Patrones secuenciales. Obtenido de: https://elvex.ugr.es/idbis/dm/slides/22%20Pattern%20Mining%20-%20Sequences.pdf

[6] Wikipedia. GSP algorithm. Obtenido de: https://en.wikipedia.org/wiki/GSP_algorithm

[7] Gabriel Navarro. Material académico proporcionado en la asignatura Tratamiento Inteligente de Datos del Máster Profesional Universitario en Ingeniería Informática de la Universidad de Granada. 2023. Reglas de asociación.

[8] Association Rules Discovery. Obtenido de: https://www.cs.upc.edu/~mmartin/D7-%20Association%20rules.pdf