A PRIORI Y ECLAT, RESUMEN Y MANUAL DE USUARIO

# A PRIORI

## Resumen

El algoritmo a priori surge como respuesta al problema existente a la hora de analizar las reglas de asociación entre diferentes datos, puesto que con unas decenas de datos diferentes podemos encontrar miles de reglas diferentes. Este algoritmo encuentra reglas de asociación entre diferentes datos en base a dos parámetros que establecemos arbitrariamente que tienen que cumplirse en ellas, que son el soporte y la confianza. El soporte de un conjunto de datos X {\displaystyle X\,\!} D {\displaystyle D\,\!} se define como la proporción de apariciones que contiene dicho conjunto de datos en el espacio muestral. La confianza define la probabilidad de encontrar relacionadas una parte derecha de una regla de asociación con una parte izquierda determinadas. El principio en el que se basa apriori es que si un conjunto de datos es frecuente, cualquier subconjunto del mismo también lo será.

## Manual de Usuario

Invocamos la función escribiendo apriori() y los diferentes parámetros que nos interesan para su uso son los siguientes:

Apriori(data, parameter = NULL, appearance = NULL, control = NULL)

Data: estructura de datos (por ejemplo, una matriz binaria o un data.frame) que se puede convertir en transacciones.

Parameter: diferentes parámetros de la clase APparameter. El comportamiento por defecto de los parámetros importantes es: soporte (support) de 10%, confianza (confidence) del 80% un límite máximo (maxlen) de 10 y mínimo (minlen) de 1 elemento por conjunto de elementos, teniendo en cuenta el conjunto vacio y un tiempo de check (maxtime) máximo de 5 segundos.

Appearance: diferentes parámetros de la clase APappearance. Con esto podemos restringir la aparición de algunos datos.

Control: diferentes parámetros de la clase APcontrol. Controla el rendimiento del algoritmo (sort, heap, filter…) y aspectos como la ordenación o el reporte del rendimiento (verbose).

# ECLAT

## Resumen

Eclat es una alternativa a apriori que utiliza para cada elemento una lista en la que se registra en qué transacción está el elemento mencionado, reduciendo así enormemente el tiempo de cómputo, pero aumentando el consumo de memoria. La idea básica es que usa estas listas y realiza intersecciones con otras para calcular el soporte del elemento, evitando generar subconjuntos que no existan en el espacio muestral.

## Manual de Usuario

Invocamos la función escribiendo eclat() y los diferentes parámetros que nos interesan para su uso son los siguientes:

Eclat(data, parameter = NULL, control = NULL)

Data: estructura de datos (por ejemplo, una matriz binaria o un data.frame) que se puede convertir en transacciones.

Parameter: diferentes parámetros de la clase ECparameter. El comportamiento por defecto de los parámetros importantes es: soporte (support) de 10% y un límite (maxlen) de 5 elementos por conjunto de elementos.

Control: diferentes parámetros de la clase ECcontrol. Controla el rendimiento del algoritmo (sort, heap, filter…) y aspectos como la ordenación o el reporte del rendimiento (verbose).