Лекция 7.1

ПОИСК АССОЦИАТИВНЫХ ПРАВИЛ

Постановка задачи

Опишем эту задачу в обобщенном виде. Для этого обозначим объекты, составляющие исследуемые наборы (itemsets), следующим множеством:

$I=\{i_1, i_2,\ldots i_j,\ldots,i_n\}$,

где i_j — объекты, входящие в анализируемые наборы; n — общее количество объектов.

В сфере торговли, например, такими объектами являются товары, представленные в прайс-листе (табл. 1).

Таблиuа 1

идентификатор	Наименование товара	цена
0	Шоколад	30.00
1	Чипсы	12.00
2	Кокосы	10.00
3	Вода	4.00
4	Пиво	14.00
5	Орехи	15.00

Они соответствуют следующему множеству объектов:

I={шоколад,чипсы,кокосы,вода,пиво,орехи}.

Описание транзакции:

$T=\{i_j|i_i \in I\}$

T_1= {чипсы, вода, пиво};
T_2 ={кокосы, вода, орехи}
Набор транзакпий, информация о которых доступна для анализа, обозначим следующим множеством:

$$D=\{T_1, T_2,\ldots T_r,\ldots,T_m\}$$

Где ***m-количество*** доступных для анализа транзакций.
Например, в магазине таким множеством будет:
D={{чипсы,вода,пиво},
{кокосы,вода,орехи},
{орехи,кокосы,чипсы,кокосы,вода},
{кокосы,орехи,кокосы}}

Для использования методов Data Mining множество D может быть представлено в виде табл. 2.

Таблица 2

Номер транзакции	Номер товара	Наименование товара	цена
0	1	Чипсы	12,00
0	3	Вода	4,00
0	4	Пиво	14,00
1	2	Кокосы	10,00
1	3	Вода	4,00
1	5	Орехи	15,00
2	5	Орехи	15,00
2	2	Кокосы	10,00
2	1	Чипсы	12,00
2	2	Кокосы	10,00
2	3	Вода	4,00
3	2	Кокосы	10,00
3	5	Орехи	15,00
3	2	Кокосы	10,00

Множество транзакций, в которые входит объект ij, обозначим следующим образом:

$$D_{i_j}=\{Tr \mid i_j \in Tr ; j=1..n; r=1..m\} \subseteq D$$

В данном примере множеством транзакций, содержащих объект вода, является следующее множество:

$D_{вода}$ = {{чипсы, вода, пиво},
{кокосы, вода, орехи},
{орехи, кокосы, чипсы, кокосы, вода}}.

Некоторый произвольный набор объектов (itemset) обозначим следующим образом:

$$F=\{i_j \mid i_j \in I; j=1..n\}.$$

Например ,

F={кокосы,вода}.

$$D_F=\{Tr \mid F \subseteq Tr ; r=1..n; r=1..m\} \subseteq D$$

D{кокосы,вода} == {{кокосы, вода, орехи},
{орехи, кокосы, чипсы, кокосы, вода} }

Отношение количества транзакций, в которое входит набор F, к общему количеству транзакций называется ***поддержкой*** (Support) набора F и обозначается Supp(F) :

$$\mathbf{Supp(F)}=\frac{|D_F|}{|D|}$$

Для набора {кокосы, вода} поддержка будет равна 0,5, т. к. данный набор входит в две транзакции (с номерами 1 и 2), а всего транзакций 4. При поиске аналитик может указать минималыюе значение поддержки интресующих его наборов Suppmin. Набор называется ***частым*** (large itemset), если значение его поддержки больше минимального значения поддержки, заданного пользователем:

Supp(F) > Suppmin.

Таким образом, при поиске ассоциативных правил требуется найти множест во всех частых наборов:

L={F|Supp(F)>Suppmin}

В данном примере частными наборами при Suppmin =0,5 являются следующие:

{чипсы} = Suppmin 0,5;

{чипсы, вода} Suppmin = 0,5;

{кокосы} Suppmin = 0,75;

{кокосы, вода} Suppmin = 0,5;

{кокосы, вода, орехи} Suppmin = 0,5;

{кокосы, орехи} Suppmin = 0,75;

{вода} Suppmin = 0,75;

{вода, орехи} Suppmin = 0,5;

{орехи} Suppmin = 0,75

Секвенциальный анализ

Последовательностью называется упорядоченное множество объектов. Для этого на множестве должно быть задано отношение порядка. Тогда последовательность объектов можно описать в следующем виде:

S={…,i_p,…,i_q}, где p<q.

Например, в случае с покупками в магазинах таким отношением порядка может выступать время покупок. Тогда последовательность

S={(вода,02.03.2003),(чипсы 05.03.2003),(пиво,10.03.2003)}

можно интерпретировать как покупки, совершаемые одним человеком в разное время (вначале была куплена вода, затем чипсы, а потом пиво).

Различают ***два вида последовательностей***: с циклами и без циклов. В первом случае допускается вхождение в последовательность одного и того же объекта на разных позициях: S={…,i_p,…,i_q}, где p<q, i_q= i_p.

Поддержкой последовательности S называется отношение количества транзакций, в которое входит последовательность S, к общему количеству транзакций. Последовательность является ***частой***, если ее поддержка превышает минимальную поддержку, заданную пользователем:

$$Supp(S) > Supp_{min}$$

Задачей секвенциальноrо анализа является поиск всех частых последовательностей:

$$L = \{S | Supp(S) > Supp_{min}\}$$

Например, при анализе последовательности покупок в супермаркете наборами являются покупки, совершаемые в разное время одними и теми же покупателями, а отношением порядка в них является хронология покупок:

D={{(вода),(пиво)),

{(кокосы, вода), (пиво), (вода, шоколад, кокосы)},

{(пиво, чипсы, вода), (пиво)}}.

Таблица 6.3

ID покупателя	Последовательность покупок
0	(пиво),(вода)
1	(кокосы,вода),(пиво),(вода,шоколад,кокосы)
2	(пиво,чипсы,вода)(пиво)

Интерпретировать такую последовательность можно следующим образом:

покупатель с идентификатором1 вначале купил кокосы и воду, затем купил пиво, в последний раз он купил воду, шоколад и кокосы.

Поддержка, например, последовательности {(пиво), (вода)} составит 2/3, т. к. она встречается у покупателей с идентификаторами 0 и 1.

У последнего покупателя также встречается набор {(пиво), (вода)}, но не сохраняется последовательность (он купил вначале воду, а затем пиво).

Таблица 6.4

Дата	Время	Источник ошибки	Код источника	Код ошибки	…
01.01.03	15:04:23	Станция 1	1001	a	…
01.01.03	16:45:46	Станция 1	1001	f	…
01.01.03	18:32:26	Станция 4	1004	z	…
01.01.03	20:07:11	Станция 5	1005	h	…
01.01.03	20:54:43	Станция 1	1001	q	…
…	…	…	…	…	…

В данной задаче объектами множества I являются коды ошибок, возникающих в процессе работы телекоммуникационной сети. Последовательность Ssid содержит сбои, происходящие на станции с идентификатором sid. Их можно представить в виде пар *(eid, t)*, где *eid* коц ошибки, а t время, когда она произошла. Таким образом, последовательность сбоев на станции с идентификатором sid будет иметь следующий вид:

$$S_{sid}=\{\{eid_1,t_1),(eid_2,t_2),\ldots,(eid_n,t_n)\}.$$

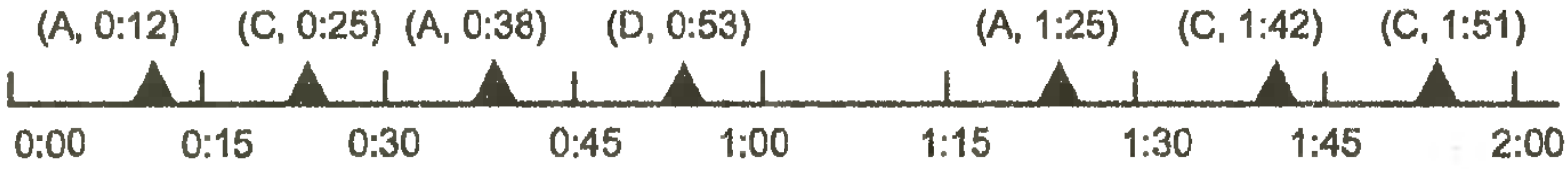
Для данных, приведенных в табл. 6.4, транзакции будут следующие:

T_{1001}={(a,15:04:23 01.01.03),(f,16:45:46 01.01.03),(q,20:54:43 01.01.03),…};

T_{1004} ={(z,18:32:26 01.01.03),…};

T_{1005}={(h,20:07:11 01.01.03),…}.

Отношение порядка в данном случае задается относительно времени появления ошибки (значения t).



При анализе такой последовательности ***важным является определение интервала времени между сбоями***. Оно позволяет предсказать момент и характер новых сбоев, а следовательно, предпринять профилактические меры. По этой причине при анализе данных интерес вызывает не просто последовательность событий, а сбои, которые происходят друг за другом.

Например, на рис. 1 изображена временная шкала последовательности сбоев, происходящих на одной станции.
При определении поддержки, например, для последовательности {A, C} учитываются только следующие наборы: {(A, 0: 12), (C, 0:25)}, {(A, 0:38), (C, 1 :42)}, {(A, 1 :25), (C, 1 :42)}. При этом не учитываются следующие последовательности: {(A, 0:12), (C, 1:42)}, {(A, 0:12), (C, 1:51)}, {(A, 0:38), (C,]:51)} и {(A, 1:25), (C, 1:51)}, т. к. они не следуют непосредственно друг за другом.

Разновидности задачи поиска ассоциативных правил

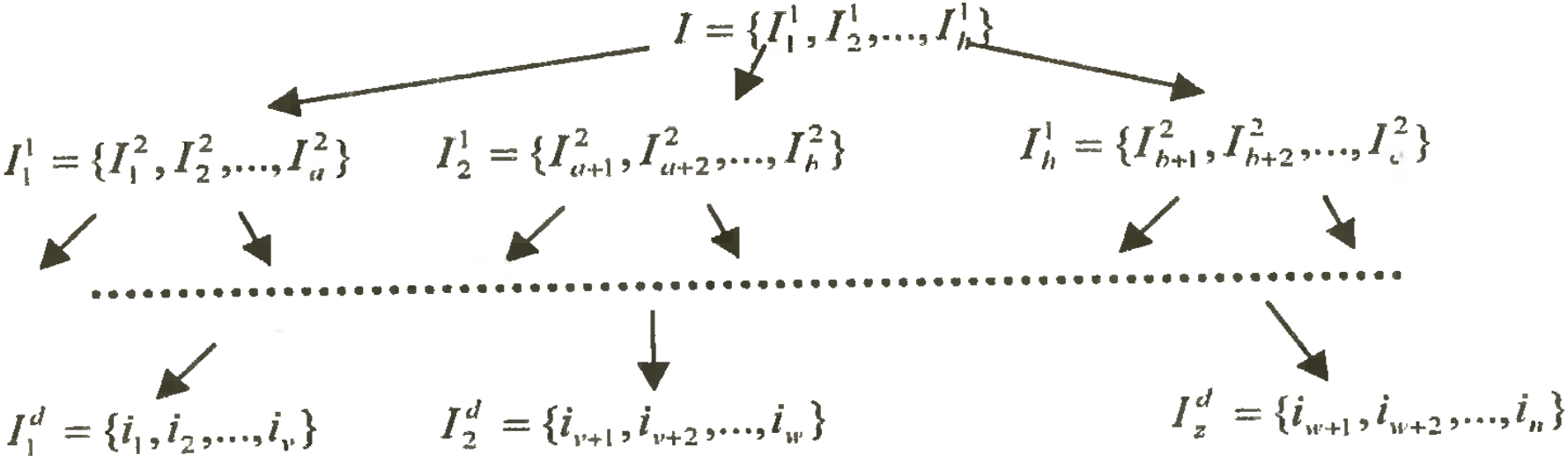


Рис.2. Иерархическое представление объектов множества I

Для примера, приведенного в табл. 6.1, такой иерархии может быть следующая категоризация товаров:

- □ напитки;
  - ● алкогольные;
  - □ пиво;
  - ● безалкогольные;
  - □ вода;
- □еда;
  - ●шоколад;
  - ● чипсы
  - ● кокосы;
  - ● орехи.

Наличие иерархии изменяет представление о том, когда объект *i* присутствует в транзакции T.

Очевидно, что поддержка не отдельного объекта, а группы, в которую он входит, больше:

$$Supp(Iq^g) \geq Supp(i_j),$$

где $i_j \in Iq^g$.

Это связано с тем, что при анализе групп подсчитываются не только транзакции, в которые входит отдельный объект, но и транзакции, содержащие все объекты анализируемой группы.

Например, если поддержка ***Supp{кокосы, вода} = 2/4***, то поддержка ***Supp{еда,напитки}***= 2/4, т. к. объекты групп ***еда И напитки*** входят в транзакции с идентификаторами 0, 1 и 2.

Представление результатов

Результаты, получаемые при решении этой задачи, принято представлять в виде ассоциативных правил. В связи с этим при их поиске выделяют два

основных этапа:

□ ***нахождение всех частых наборов объектов;***

□ ***генерация ассоциативных правил из найденных частых наборов объектов.***

Ассоциативные правила имеют следующий вид:

если (условие) то (результат)

где ***условие*** — обычно не логическое выражение (как в классификационных правилах), а набор объектов из множества I , с которыми связаны (ассоциированы) объекты, включенные в ***результат*** данного правила.

Например, ассоциативное правило:
если (кокосы, вода) то (орехи)
означает, что если потребитель покупает кокосы и воду, то он покупает и орехи.
Как уже отмечалось, в ассоциативных правилах условие и результат являются объектами множества I :
если X то Y
где $X \in I$,$Y \in I$, $X \cup Y = \varphi$.
Ассоциативное правило можно представить как импликацию над множеством $X \Rightarrow Y$, *где* $X \in I$,$Y \in I$, $X \cup Y = \varphi$.
Основным достоинством ассоциативных правил является их легкое восприятие человеком и простая интерпретация языками программирования. Однако они не всегда полезны.

Выделяют три вида правил:

- ***полезные правила*** — содержат действительную информацию, которая ранее была неизвестна, но имеет логичное объяснение. Такие правила могут быть использованы для принятия решений, приносящих выгоду;
- ***тривиальные правила*** — содержат действительную и легко объяснимую информацию, которая уже известна. Такие правила, хотя и объяснимы, но не могут принести какой-либо пользы, т. к. отражают или известные законы в исследуемой области, или результаты прошлой деятельности. Иногда такие правила могут использоваться для проверки выполнения решений, принятых на основании предыдущего анализа;
- ***непонятные правила*** — содержат информацию, которая не может быть объяснена.

Такие правила могут быть получены или на основе аномальных значений, или глубоко скрытых знаний. Напрямую такие правила нельзя использовать для принятия решений, т. к. их необъяснимость может привести к непредсказуемым результатам. Для лучшего понимания требуется дополнительный анализ.

Ассоциативные правила строятся на основе частых наборов. Так, правила,построенные на основании набора F (т. е. $X \cup Y = F$), являются всеми возможными комбинациями объектов, входящих в него.

Например, для набора {кокосы, вода, орехи} могут быть построены следующие правила:

если (кокосы) то (вода);
если (кокосы) то (орехи);
если (кокосы) то (вода, орехи);
если (вода, орехи) то (кокосы);
если (вода) то (кокосы);
если (вода) то (орехи);
если (вода) то (кокосы, орехи);
если (кокосы, орехи) то (вода);
если (орехи) то (вода);
если (орехи) то (кокосы);
если (орехи) то (вода, кокосы);
если (вода, кокосы) то (орехи).

Для полезности вводятся следующие величины:

Поддержка (support) — показывает, какой процент транзакций поддерживает данное правило. Так как правило строится на основании набора, то, значит, правило X ⇒Y имеет поддержку, равную поддержке набора F , который составляют X и Y :

$$Supp_{X=>Y} = Supp_F = \frac{|D_{F=X\cup Y}|}{|D|}$$

Например,

***Supp** если (вода, кокосы) то (орехи) = Supp {вода, кокосы, орехи} = 2/4.*

Достоверность (confidence) — показывает вероятность того, что из наличия в транзакции набора X следует наличие в ней набора Y Достоверностью правила X ⇒Y является отношение числа транзакций, содержащих наборы X и Y , к числу транзакций, содержащих набор X :

$$Conf_{X=>Y} = \frac{|D_{F=X\cup Y}|}{|D|x} = \frac{Supp_{X\cup Y}}{Supp_x}$$

Очевидно, что чем больше достоверность, тем правило лучше, причем у правил, построенных на основании одного и того же набора, достоверность будет разная, ***например***:

Conf если(вода) то (орехи)=2/3;

Conf если(орехи) то (вода)=2/3;

Conf если(вода,кокосы) то (орехи)=1;

Conf если(вода) то (орехи,кокосы)=2/3;

К сожалению, достоверность не позволяет оценить полезность правила,поэтому ввели следующую величину:

Улучшение (improvement) — показывает, полезнее ли правило случайного угадывания. Улучшение правила является отношением числа транзакций, содержащих наборы X и Y , к произведению количества транзакций, содержащих набор X , и количества транзакций, содержащих набор Y :

$$\textbf{Impr } X => Y = \frac{|D_{F=X \cup Y}|}{|Dx||Dy} = \frac{Supp_{X \cup Y}}{Supp_X * Supp_Y}$$

Например,

Impr если (вода, кокосы) то (орехи) = 0,5/(0,5·0,5) = 2.

Если улучшение больше единицы, то это значит, что с помощью правила предсказать наличие набора Y вероятнее, чем случайное угадывание, если меньше единицы, то наоборот.

Алгортим Apriori

Он использует одно из свойств поддержки, гласящее: поддержка любого набора объектов не может превышать минимальной поддержки любого из его подмножеств:

$Supp_F \leq Supp_E$ при $E \subset F$

Например, поддержка 3-объектного набора *{пиво, вода, чипсы}* будет всегда меньше или равна поддержке 2-объектных наборов *{пиво, вода}, {вода,чипсы}, {пиво, чипсы}*. Это объясняется тем, что любая транзакция, содержащая *{пиво, вода, чипсы}*, содержит также и наборы *{пиво, вода}, {вода, чипсы},{пиво, чипсы}*, причем обратное неверно.

L_1 = {часто встречающиеся 1-элементные наборы}

для ($k=2$; $L_{k-1} <> \varphi$; $k++$)

C_k = Apriorigen(F_{k-1}) // генерация кандидатов для всех транзакций t ∈ D выполнить
C_t = subset(C_k, t) // удаление избыточных правил для всех кандидатов c ∈ Ct выполнить
c.count ++
конец для всех
конец для всех
L_k = { c ∈ C_k | c.count >= $Supp_{min}$} // отбор кандидатов
конец для
Результат = $\cup_k L_k$

L_k-множество k-элементных частых наборов, чья поддержка не меньше заданной пользователем. Каждый член множества имеет набор упорядоченных (Ij < Ip,если j < p) элементов F и значение поддержки набора
$Supp_F > Supp_{min}$:

C_k -множество кандидатов k -элементных наборов потенциально частых. Каждый член множества имеет набор упорядоченных (ij< ip , если j < p) элементов F и значение поддержки набора Supp .

Опишем данный алгоритм по шагам.

Шаг1. Присвоить k = 1 и выполнить отбор всех 1-элементных наборов,у которых поддержка больше минимально заданной пользователем $Supp_{min}$.

Шаг2. k = k +1.

Шаг3. Если не удается создавать k-элементные наборы, то завершить алгоритм, иначе выполнить следующий шаг.

Шаг4. Создать множество k-элементных наборов кандидатов в частые наборы. Для этого необходимо объединить в k-элементные кандидаты (k – 1)-элементные частые наборы. Каждый кандидат $c \in Ck$ будет формироваться путем добавления к (k – 1)-элементному частому набору — p элемента из другого (k – 1)-элементного частого набора — q.

Причем добавляется последний элемент набора q, который по порядку выше, чем последний элемент набора *p* ($p.item_{k-1} < q.item_{k-1}$). При этом первые все (k – 2) элемента обоих наборов одинаковы ($p.item_1 = q.item_1, p.item_2 = q.item_2, ...,$

$p.item_{k-2} = q.item_{k-2}$).

Это может быть записано в виде следующего SQL-подобного запроса.

insert into C_k

select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$

from L_{k-1} p, L_{k-1} q

where $p.item_1 = q.item_1, p.item_2 = q.item_2, ..., p.item_{k-2} = q.item_{k-2},$

$p.item_{k-1} < q.item_{k-1}$

Шаг5 Для каждой транзакции T из множества D выбрать кандидатов Ct из множества C_k, присутствующих в транзакции T. Для каждого набора из построенного множества C_k удалить набор, если хотя бы одно из его (k – 1)подмножеств не является часто встречающимся, т. е. отсутствует во множестве L_{k-1}.Это можно записать в виде следующего псевдокода:

для всех наборов $c \in C_k$ *выполнить*

для всех (k-1) – поднаборов s из c выполнить

если $(s \notin L_{k-1})$ *то удалить c из* C_k

Шаг6 Для каждого кандидата из множества C_k увеличить значение поддержки на единицу.

Шаг7 Выбрать только кандидатов L_k из множества C_k, у которых значение поддержки больше заданной пользователем $Supp_{min}$. Вернуться к шагу 2.

Результатом работы алгоритма является объединение всех множеств L_k всех k.

Рассмотрим работу алгоритма на примере, приведенном в табл. 6.1, при

$Supp_{min}$= 0,5 . На первом шаге имеем следующее множество кандидатов C_1

(указываются идентификаторы товаров) (табл. 5).

Таблица 5

№	Набор	Supp
1	{0}	0
2	{1}	0,5
3	{2}	0,75
4	{4}	0,25
5	{3}	0,75
6	{5}	0,75

Заданной минимальной поддержке удовлетворяют только кандидаты 2, 3, 5 и 6, следовательно: L1={{1},{2},{3},{5}}.
На втором шаге увеличиваем значение k до двух. Так как можно построить 2-элементные наборы, то получаем множество C_2 (табл. 6).

Таблица 6

№	Набор	Supp
1	{1,2}	0,25
2	{1,3}	0,5
3	{1,5}	0,25
4	{2,3}	0,5
5	{2,5}	0,75
6	{3,5}	0,5

Из построенных кандидатов заданной минимальной поддержке удовлетворяют только кандидаты 2, 4, 5 и 6, следовательно:
L_2={{1,3},{2,3},{2,5},{2,5}}.
На третьем шаге перейдем к созданию 3-элементных кандидатов и подсчету их поддержки. В результате получим следующее множество C_3 (табл. 6.7).

№	Набор	Supp
1	{2,3,5}	0,5

Данный набор удовлетворяет минимальной поддержке, следовательно:
L_3= {{2,3,5}}
Так как 4-элементные наборы создать не удастся, то результатом работы алгоритма является множество:
$L=L_1 \cup L_2 \cup L_3$= {{1}, {2}, {3}, {5}, {1, 3}, {2, 3}, {2, 5}, {3, 5}, {2, 3, 5}}.

После построения хэш-дерево с кандидатами-наборами ,легко подсчитать поддержку для каждого кандидата. Для этого нужно "пропустить" каждую транзакцию через дерево и увеличить счетчики для тех кандидатов, чьи элементы также содержатся и в транзакции, $C_k \cap T_i = C_k$

Выводы

- **Задачей поиска ассоциативных правил** является определение часто встречающихся наборов объектов в большом множестве наборов.

- **Секвенциальный анализ заключается** в поиске частых последовательностей. Основным отличием задачи секвенциального анализа от поиска ассоциативных правил является установление отношения порядка между объектами.

- **Наличие иерархии в объектах** и ее использование в задаче поиска ассоциативных правил позволяет выполнять более гибкий анализ и получать дополнительные знания.

Результаты решения задачи представляются в виде ассоциативных правил, условная и заключительная часть которых содержит наборы объектов.

Основными характеристиками ассоциативных правил являются поддержка, достоверность и улучшение.

Поддержка (support) показывает, какой процент транзакций поддерживает данное правило.

Достоверность (confidence) показывает, какова вероятность того, что из наличия в транзакции набора условной части правила следует наличие в ней набора заключительной части.

Улучшение (improvement) показывает, полезнее ли правило случайного угадывания.

Задача поиска ассоциативных правил решается в два этапа. На первом выполняется поиск всех частых наборов объектов. На втором из найденных частых наборов объектов генерируются ассоциативные правила.

Алгоритм Apriori использует одно из свойств поддержки, гласящее: поддержка любого набора объектов не может превышать минимальной поддержки любого из его подмножеств.