Ассоциативные правила

Наумов Д.А., доц. каф. КТ

Экспертные системы и искусственный интеллект, 2020

Содержание лекции

Что такое поиск ассоциативных правил?

Поиск ассоциативных правил

поиск часто в страчающихся шаблонов, ассоциаций, корреляций или структур среди множества элементов в транзакционной базе данных.

Понять покупательские привычки клиента, находя ассоциации и корреляции между различными товарами, которые клиенты размещают в их «корзину для покупок».
Практическое применение:

- анализ покупок;
- кросс-маркетинг;
- каталогизация;
- web-анализ;
- обнаружение мошеннических схем.

Что такое поиск ассоциативных правил?

Ассоциативное правило

Antecedent → Consequent [support, confidence]

- Antecedent – антецедент, причина;
- Consequent – косеквент, следствие;
- Support – поддержка (мера интересности правила);
- Confedence – значимость (мера интересности правила).

Примеры:

$$buys(x, "computer") \rightarrow buys(x, "financialmanagementsoftware")$$

$$[0.5\%, 60\%]$$

$$age(x, "30..39") and income(x, "42..48K") \rightarrow buys(x, "car")$$

$$[1\%, 75\%]$$

Как можно использовать ассоциативные правила?

- пусть правило имеет вид:

$$\{Bagels, ...\} \rightarrow \{Potato\ Chips\}$$

- **Potato chips** (следствие) – продажи чего мы собираемся (можем) увеличивать;
- **Bagels** (антецедент) – какие продукты будут влиять на продажу, если объявить скидки;
- **Bagels -> Potato chips** – какие продукты следует размещать рядом, чтобы увеличить продажи Potato Chips.

Практическое применение:

- оптимизировать размещение товара на полках;
- формировать персональные рекомендации;
- планирование промо-акции;
- более эффективно управлять ценами и ассортиментом.

Ассоциативные правила: основные понятия

Исходные данные

1. база данных **транзакций**
2. транзакция содержит список **элементов**

Результаты поиска ассоциативных правил

1. все правила, которые связывают наличие одного **набора** (itemset) с другим набором элементов

Например, 98% людей, которые покупают шины и автоаксессуары, также заказывают услуги шионмонтажа.

Ассоциативные правила: поддержка и значимость

Пусть имеется ассоциативное правило:

$$A \Rightarrow B[s, c]$$

Поддержка (Support)

обозначает, как часто правило встречается в транзакциях.

$$support(A \Rightarrow B[s, c]) = p(A \cup B)$$

Значимость (confidence)

обозначает процент транакций, содержащая **A**, которые содержат также **B**.

$$confidence(A \Rightarrow B[s, c]) = p(B|A) = sup(A, B)/sup(A)$$

Значимость – это оценка условной вероятности.

Пример

Trans. Id	Purchased Items
1	A,D
2	A,C
3	A,B,C
4	B,E,F

Itemset:

A,B or B,E,F

Support of an itemset:

Sup(A,B)=1 Sup(A,C)=2

Frequent pattern:

Given min. sup=2, {A,C} is a frequent pattern

For minimum support = 50% and minimum confidence = 50%, we have the following rules

A => C with 50% support and 66% confidence

C => A with 50% support and 100% confidence

Математические обозначения

X – пространство объектов;
$F = f_1, \ldots, f_n, f_i : X \to 0, 1$ – бинарные признаки (items);
$X^l = \{x_1, \ldots, x_l\} \subset X$ – обучающая выборка.
Каждому подмножеству $\varphi \subseteq F$ соответствует конъюнкция

$$\varphi(x) = \bigwedge_{f \in \varphi} f(x), x \in X$$

Если $\varphi(x) = 1$, то «признаки из φ совместно встречаются в x»/
Поддержка (support) φ в выборке X^l

$$\nu(\varphi) = \frac{1}{l} \sum_{i=1}^{l} \varphi(x_i)$$

Если $\nu(\varphi) \geq \delta$, то «набор φ частый» (frequent itemset).
Параметр δ – минимальная поддержка, (min support).

Математические обозначения

Определение

Ассоциативное правило (association rule) $\varphi \to y$ – это пара непересекающихся наборов $\varphi, y \subseteq F$, таких, что:

1. наборы φ и y совместно часто встречаются,

$$\nu(\varphi \cup y) \geq \delta;$$

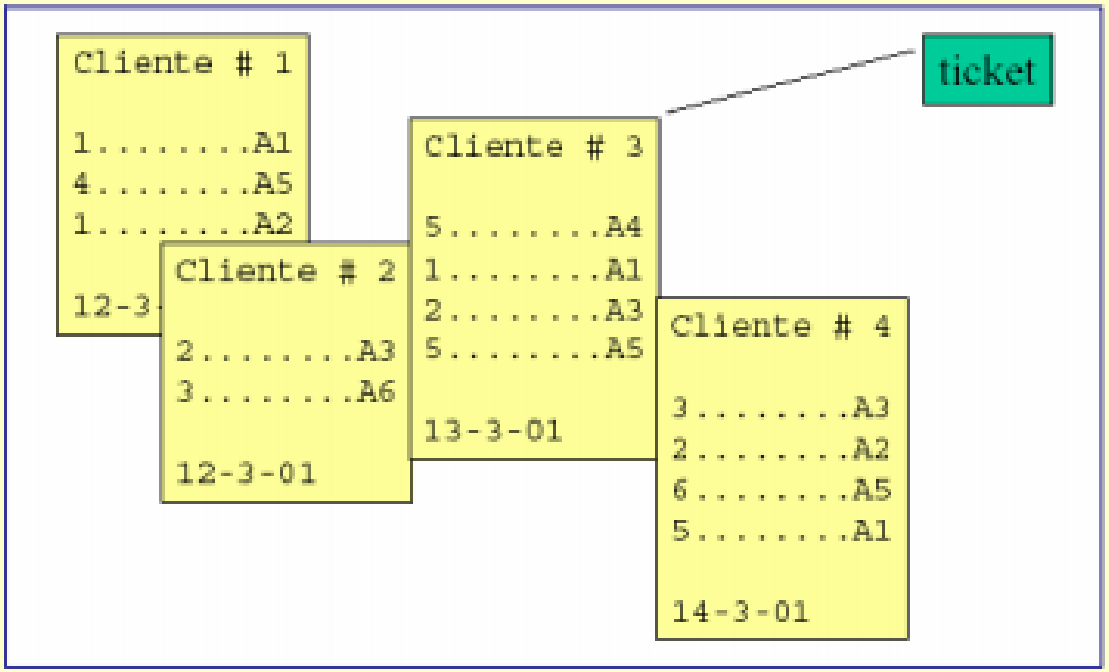
2. если встречаются φ, то часто встречается также и y:

$$\nu(y|\varphi) \equiv \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \geq \chi$$

$\nu(y|\varphi)$ – значимость (confidence) правила.
Параметр δ – минимальная поддержка (min support).
Параметр χ – минимальная значимость (min conf).

Содержание лекции

Логические (булевые) ассоциативные правила



Each transaction is represented by a Boolean vector

Cliente	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
1	1	1	0	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	1	0	0	0	0	0	0	0
3	1	0	1	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	1	0	0	0	0	0	0	0	0
5	0	0	1	0	0	1	0	1	1	1	0	0	0
6	0	1	0	0	0	0	0	1	0	1	0	0	0
7	1	0	0	0	0	0	1	1	0	1	0	1	1
8	0	1	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	1	0

Пример поиска ассоциативных правил

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A, C\}$) = 50%

confidence = support($\{A, C\}$) / support($\{A\}$) = 66.6%

Принцип Apriori

Поскольку $\varphi(x) = \bigwedge_{f \in \varphi} f(x)$ — конъюнкция, имеет место

свойство антимонотонности:

для любых $\psi, \varphi \subset \mathscr{F}$ из $\varphi \subset \psi$ следует $\nu(\varphi) \geqslant \nu(\psi)$.

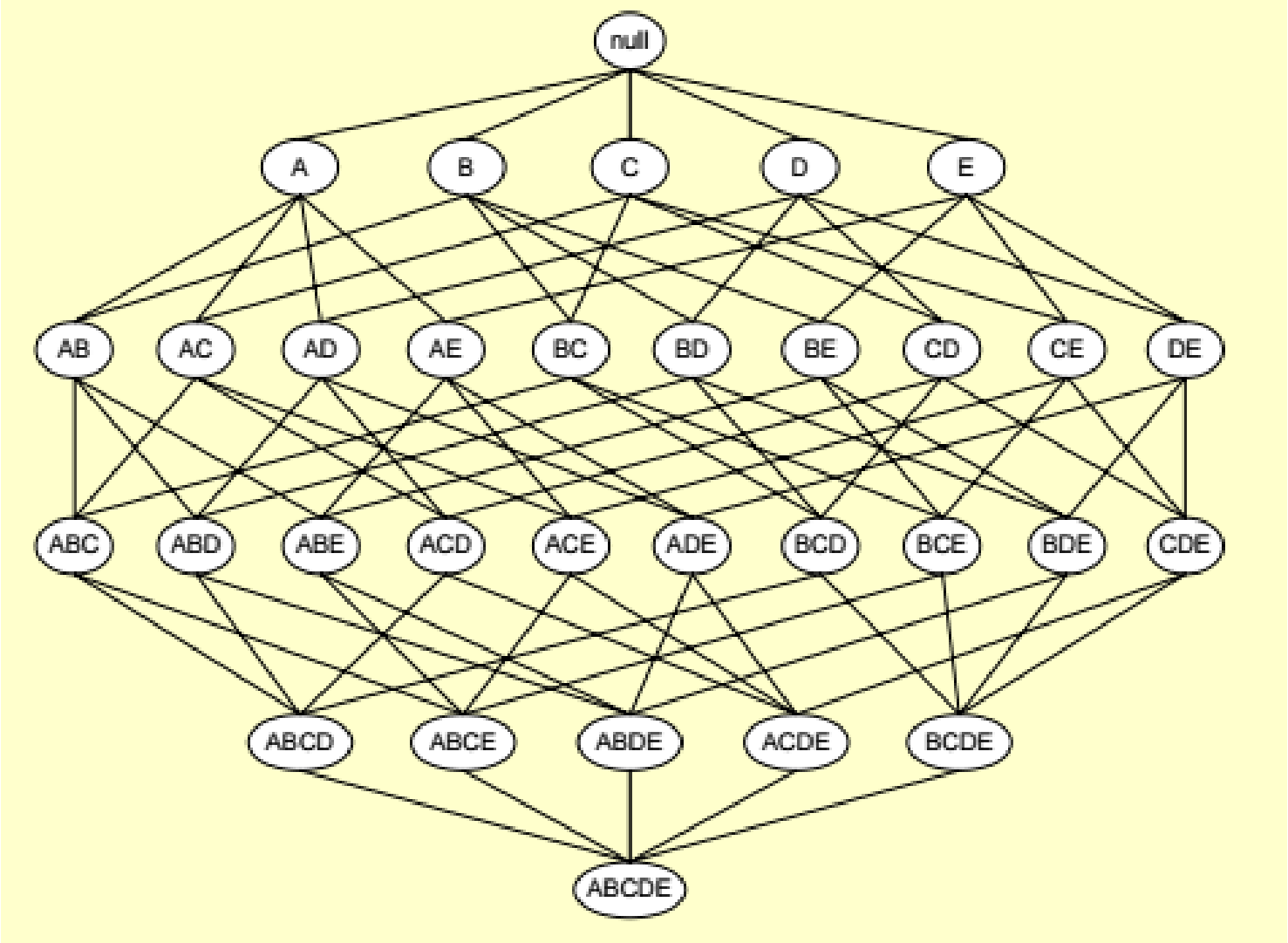
Следствия:

1. если ψ частый, то все его подмножества $\varphi \subset \psi$ частые.
2. если φ не частый, то все наборы $\psi \supset \varphi$ также не частые.
3. $\nu(\varphi \cup \psi) \leqslant \nu(\varphi)$ для любых φ, ψ.

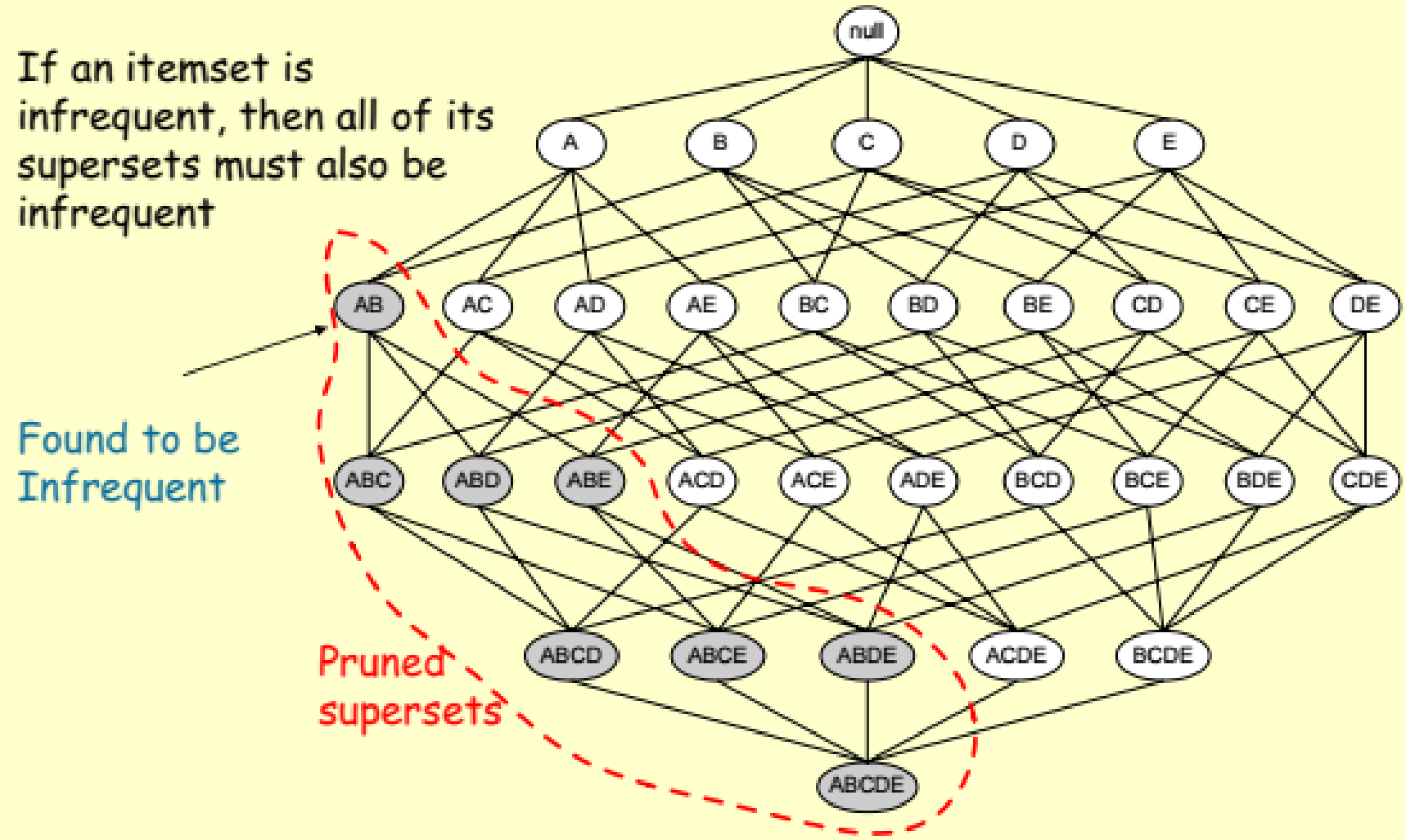
Два этапа поиска ассоциативных правил:

1. поиск частых наборов
   (многократный просмотр транзакционной базы данных).
2. выделение ассоциативных правил
   (простая эффективная процедура в оперативной памяти).

Принцип Apriori: множество наборов



Принцип Apriori: не рассматриваемые наборы

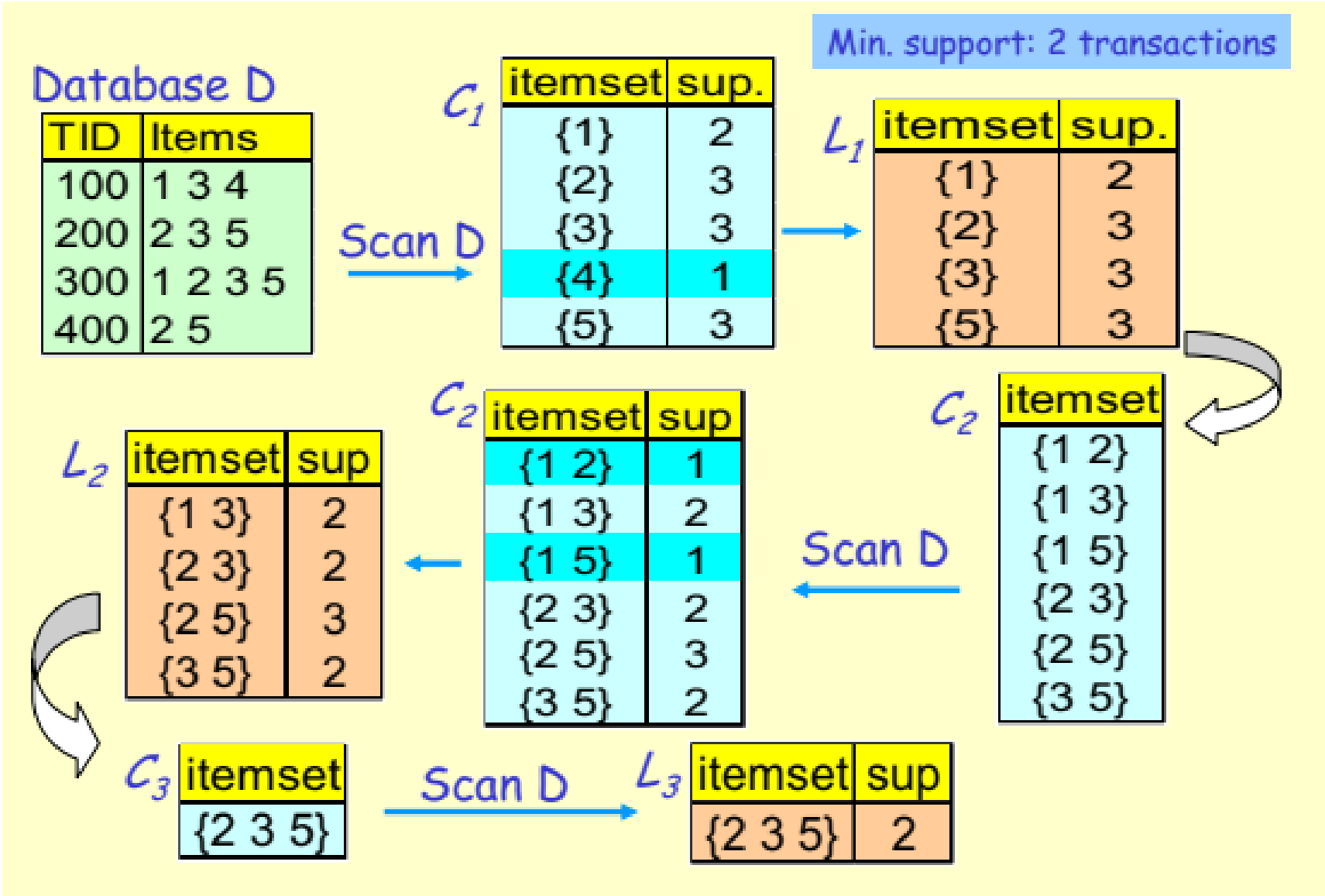


Принцип Apriori: основная идея – поиск в ширину

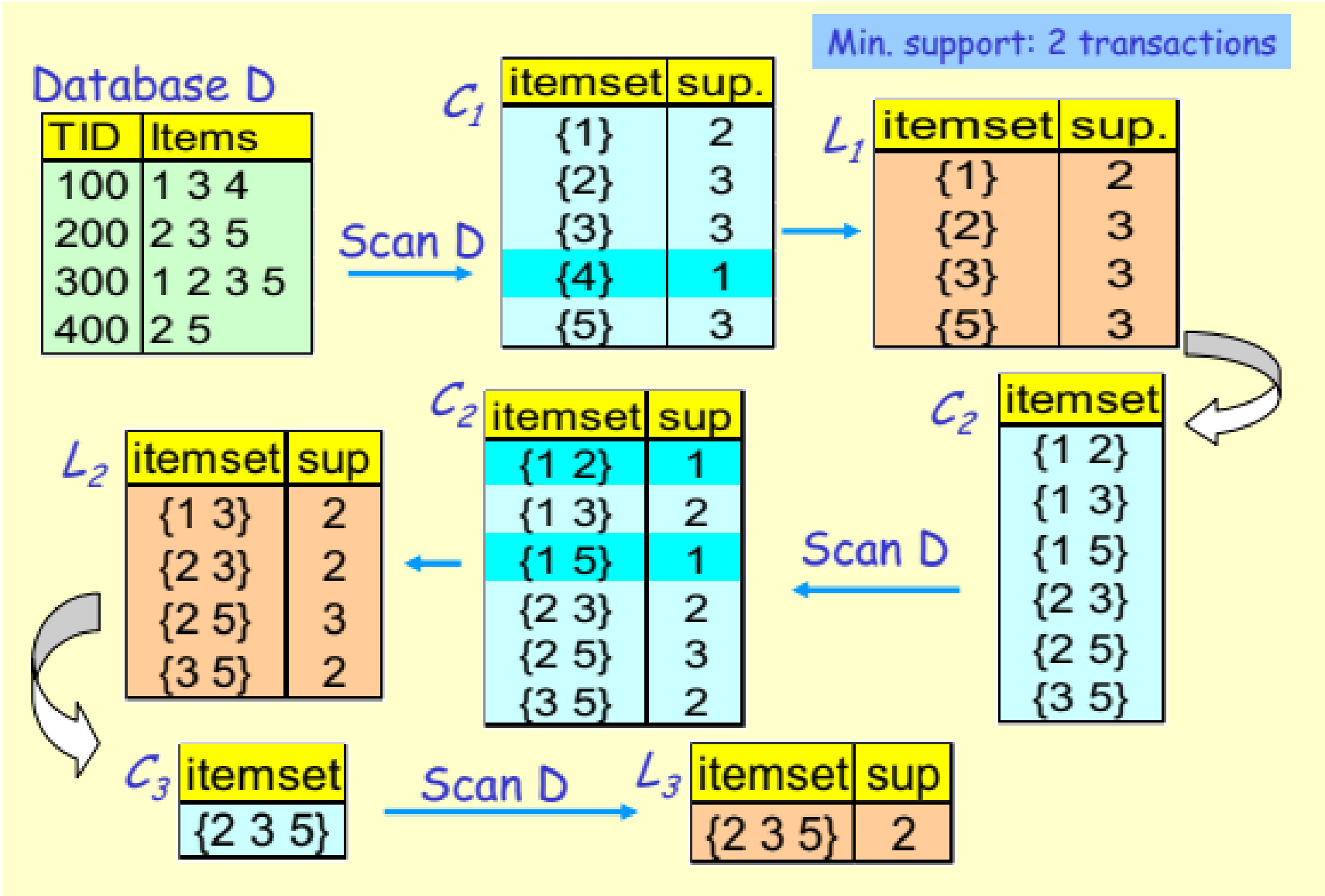
вход: X^ℓ — обучающая выборка; $\delta = \texttt{MinSupp}$; $\varkappa = \texttt{MinConf}$;
выход: $R = \{(\varphi, y)\}$ — список ассоциативных правил;

1 множество всех частых исходных признаков:
$G_1 := \{f \in \mathscr{F} \mid \nu(f) \geqslant \delta\}$;
2 **для всех** $j = 2, \ldots, n$
3 множество всех частых наборов мощности j:
$G_j := \{\varphi \cup \{f\} \mid \varphi \in G_{j-1},\ f \in G_1 \backslash \varphi,\ \nu(\varphi \cup \{f\}) \geqslant \delta\}$;
4 **если** $G_j = \varnothing$ **то**
5 **выход** из цикла по j;
6 $R := \varnothing$;
7 **для всех** $\psi \in G_j,\ j = 2, \ldots, n$
8 AssocRules $(R, \psi, \varnothing)$;

Принцип Apriori: пример



Принцип Apriori: пример



Принцип Apriori: получение ассоциативных правил

вход: X^ℓ — обучающая выборка; $\delta = \mathtt{MinSupp}$; $\varkappa = \mathtt{MinConf}$;
выход: $R = \{(\varphi, y)\}$ — список ассоциативных правил;

1 множество всех частых исходных признаков:
$G_1 := \{f \in \mathscr{F} \mid \nu(f) \geqslant \delta\}$;
2 **для всех** $j = 2, \ldots, n$
3 множество всех частых наборов мощности j:
$G_j := \{\varphi \cup \{f\} \mid \varphi \in G_{j-1},\ f \in G_1 \backslash \varphi,\ \nu(\varphi \cup \{f\}) \geqslant \delta\}$;
4 **если** $G_j = \varnothing$ **то**
5 **выход** из цикла по j;
6 $R := \varnothing$;
7 **для всех** $\psi \in G_j,\ j = 2, \ldots, n$
8 AssocRules $(R, \psi, \varnothing)$;

Выделение ассоциативных правил

$$confidence(A \Rightarrow B) = P(B|A) = support(A \cup B)/support(A)$$

- Для каждого частого набора элементов x сгенерировать все непустые подмножества x;
- Для каждого непустого подмножества x множества s получить правило:

$$s \Rightarrow (s \setminus x)$$

если

$$support(x)/support(s) > MinConf$$

Выделение ассоциативных правил

Этап 2. Простой рекурсивный алгоритм, выполняемый быстро, как правило, полностью в оперативной памяти.

```
1 функция AssocRules (R, φ, y)
  |  вход: (φ, y) — ассоциативное правило;
  |  выход: R — список ассоциативных правил;
2 |  для всех f ∈ φ: id_f > max_{g∈y} id_g (чтобы избежать повторов y)
3 |  |  φ' := φ \ {f};   y' := y ∪ {f};
4 |  |  если ν(y'|φ') ⩾ ϰ то
5 |  |  |  добавить ассоциативное правило (φ', y') в список R;
6 |  |  |  если |φ'| > 1 то
7 |  |  |  |  AssocRules (R, φ', y');
```

id_f — порядковый номер признака f в $\mathscr{F} = \{f_1, \ldots, f_n\}$

Пример

Задан часто встречающийся набор (A, B, E). Какие возможны ассоциативные правила?

- *Q: Given frequent set {A,B,E}, what are possible association rules?*
  - A => B, E
  - A, B => E
  - A, E => B
  - B => A, E
  - B, E => A
  - E => A, B
  - __ => A,B,E (empty rule), or true => A,B,E

Пример

Trans-ID	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE

Min_support: 60%
Min_confidence: 75%

Frequent Itemset	Support
{**BCE**},{AC}	60%
{BC},{CE},{A}	60%
{BE},{B},{C},{E}	80%

Rule	Conf.
{BC} =>{E}	100%
{BE} =>{C}	75%
{CE} =>{B}	100%
{B} =>{CE}	75%
{C} =>{BE}	75%
{E} =>{BC}	75%

Упражнение

TID	Items
1	Bread, Milk, Chips, Mustard
2	Beer, Diaper, Bread, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk, Chips
5	Coke, Bread, Diaper, Milk
6	Beer, Bread, Diaper, Milk, Mustard
7	Coke, Bread, Diaper, Milk

Упражнение

Bread	Milk	Chips	Mustard	Beer	Diaper	Eggs	Coke
1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	0
0	1	0	0	1	1	0	1
1	1	1	0	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	1	0	0
1	1	0	0	0	1	0	1

Упражнение

0.4*7= 2.8

C1	
Bread	6
Milk	6
Chips	2
Mustard	2
Beer	4
Diaper	6
Eggs	1
Coke	3

L1	
Bread	6
Milk	6
Beer	4
Diaper	6
Coke	3

C2	
Bread,Milk	5
Bread,Beer	3
Bread,Diaper	5
Bread,Coke	2
Milk,Beer	3
Milk,Diaper	5
Milk,Coke	3
Beer,Diaper	4
Beer,Coke	1
Diaper,Coke	3

L2	
Bread,Milk	5
Bread,Beer	3
Bread,Diaper	5
Milk,Beer	3
Milk,Diaper	5
Milk,Coke	3
Beer,Diaper	4
Diaper,Coke	3

C3	
Bread,Milk,Beer	2
Bread,Milk,Diaper	4
Bread,Beer,Diaper	3
Milk,Beer,Diaper	3
Milk,Beer,Coke	
Milk,Diaper,Coke	3

L3	
Bread,Milk,Diaper	4
Bread,Beer,Diaper	3
Milk,Beer,Diaper	3
Milk,Diaper,Coke	3

$$8 + C_2^8 + C_3^8 = 92 \quad >> \quad 24$$

Модификация алгоритма Apriori

Основные проблемы при генерации наборов:

- общее число транзакций может быть **очень большим**;
- одна транзакция может содержать **много элементов**.

Модификации алгоритма:

- более эффективные структуры данных для быстрого поиска;
- поиск по частичной случайной выборке при пониженных поддержке и значимости с последующей проверкой на полной базе;
- алгоритмы, учитывающие иерархию признаков;
- поиск последовательных шаблонов;
- учет информации о клиентах.

Модификации алгоритма Apriori

- **Проблема**: на каждом уровне осуществляется просмотр всей базы данных транзакций
- **AprioriTID**:
  - генерировать набора как в алгоритме Apriori, но БД используется для вычисления поддержки всех наборов за один проход;
  - требуется значительно больше памяти;
  - вычисляются и хранятся часто встречающиеся наборы $C\hat{}_k$ для каждой транзакции;
- **AprioriHybrid**
  - на начальном этапе используется алгоритм Apriori;
  - вычисляется размер $C\hat{}_k$;
  - как только $C\hat{}_k$ будет умещаться в памяти, переключиться на AprioriTid.

Какие правила интересны?

- все ли найденные правила будут полезны и интересны?
- как можно измерить «интересность» правила?

Субъективные критерии:

1. правило интересно, если оно неожиданно для пользователя;
2. правило полезно, если пользователь может его применить.

Объективные критерии:

1. поддержка (Support)
2. значимость (Confidence)
3. интересность (Lift, Interest, Correlation)
4. убедительность (Conviction)
5. влияние (Leverage, Piatetsky-Shapiro)
6. покрытие (Coverage)

Пример

Среди 5000 студентов:

- 3000 играют в баскетбол;
- 3750 едят хлопья;
- 2000 и играют в баскетбол, и едят хлопья.

$$\text{play basketball} \rightarrow \text{eat cereal [40\%, 66.7\%]}$$

$$\text{play basketball} \rightarrow \text{not eat cereal [20\%, 33.3\%]}$$

	basketball	not basketball	sum(row)	
cereal	2000	1750	3750	75%
not cereal	1000	250	1250	25%
sum(col.)	3000	2000	5000	
	60%	40%		

Пример

Lift (Correlation, Interest):

$$Lift(A \to B) = \frac{sup(A,B)}{sup(A) \cdot sup(B)} = \frac{P(B|A)}{P(B)}$$

A и B имеют отрицательную корреляцию, если значение $Lift < 1$, иначе корреляция положительная.

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

rule	Support	Lift
X ⇒ Y	25%	2.00
X ⇒ Z	37.50%	0.86
Y ⇒ Z	12.50%	0.57

Пример

Продолжение примера:

- play basketball => eat cereal [40%, 66.7%]

$$Lift = \frac{\frac{2000}{5000}}{\frac{3000}{5000} \times \frac{3750}{5000}} = 0.89$$

- play basketball => not eat cereal [20%, 33.3%]

$$Lift = \frac{\frac{1000}{5000}}{\frac{3000}{5000} \times \frac{1250}{5000}} = 1.33$$

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Убедительность

Убедительность – мера импликации, ее значение равно 1, если элементы не связаны.

$$Conv(A \to B) = \frac{sup(A \cdot sup(\neg B)}{sup(A, \neg B)} = \frac{P(A) \cdot P(\neg B)}{P(A, \neg B)} = \frac{P(A)(1 - P(B))}{P(A) - P(A, B)}$$

Замечание: $A \to B$ можно записать в виде $\neg(A, \neg B)$.

- play basketball => eat cereal [40%, 66.7%]
- eat cereal => play basketball conv: 0.85

$$Conv = \frac{\frac{3000}{5000}\left(1 - \frac{3750}{5000}\right)}{\frac{3000}{5000} - \frac{2000}{5000}} = 0.75$$

- play basketball => not eat cereal [20%, 33.3%]
- not eat cereal => play basketball conv: 1.43

$$Conv = \frac{\frac{3000}{5000}\left(1 - \frac{1250}{5000}\right)}{\frac{3000}{5000} - \frac{1000}{5000}} = 1.125$$

Влияние

Влияние (Leverage, Piatetsky-Shapiro, PS)

мера зависимости предпосылки и следствия.

$$PS(A \to B) = sup(A, B) - sup(A) \cdot sup(B)$$

Покрытие (coverage)

$$Coverage(A \to B) = sup(A)$$

Содержание лекции

Префиксное FP-дерево (FP-frequent pattern)

В каждой вершине v дерева T задаются:

- признак $f_v \in \mathscr{F}$;
- множество дочерних вершин $S_v \subset T$;
- поддержка $c_v = \nu(\varphi_v)$ набора признаков $\varphi_v = \{f_u : u \in [v_0, v]\}$, где $[v_0, v]$ — путь от корня дерева v_0 до вершины v.

Обозначения:

$V(T, f) = \{v \in T : f_v = f\}$ — все вершины признака f.

$C(T, f) = \sum_{v \in V(T,f)} c_v$ — суммарная поддержка признака f.

Свойства FP-дерева T, построенного по всей выборке X^ℓ:

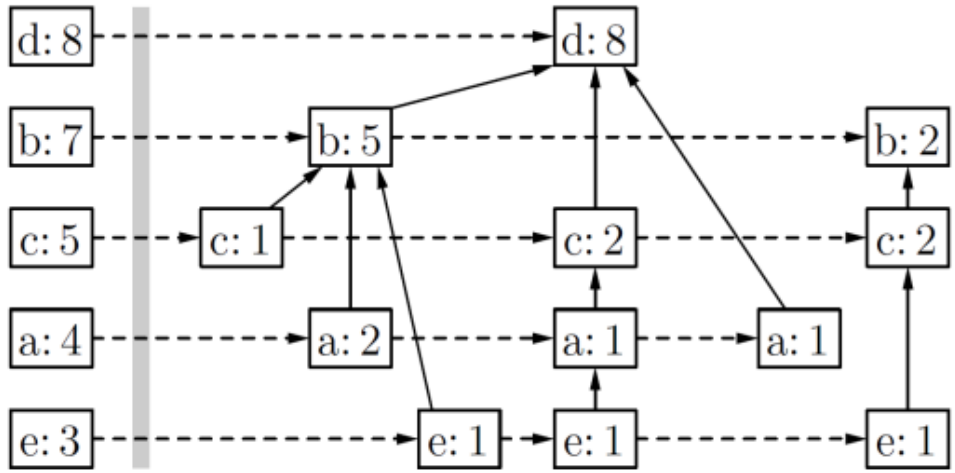
1. T содержит полную информацию о всех $\nu(\varphi)$, $\varphi \subseteq \mathscr{F}$.
2. $C(T, f) = \nu(f)$ для всех $f \in \mathscr{F}$.

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



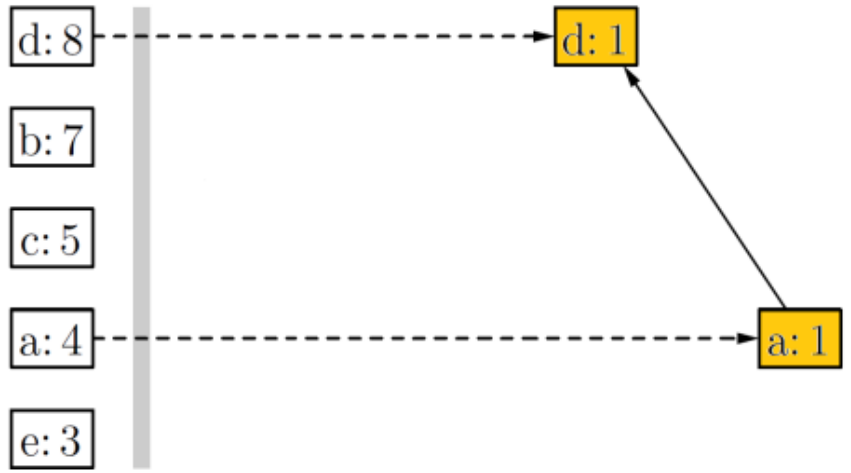
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица							слова
a	–	–	d	–	f	–	d a
a	–	c	d	e	–	–	d c a e
–	b	–	d	–	–	–	d b
–	b	c	d	–	–	–	d b c
–	b	c	–	–	–	–	b c
a	b	–	d	–	–	–	d b a
–	b	–	d	e	–	–	d b e
–	b	c	–	e	–	g	b c e
–	–	c	d	–	f	–	d c
a	b	–	d	–	–	–	d b a

(корень v_0 не показан)



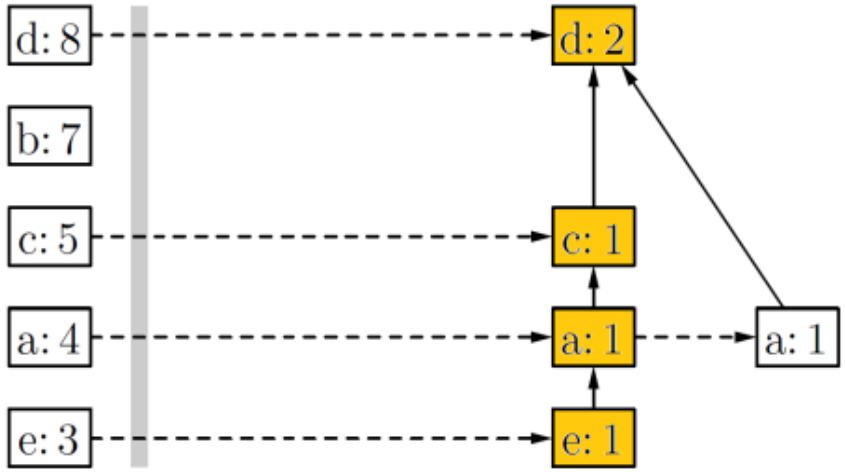
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



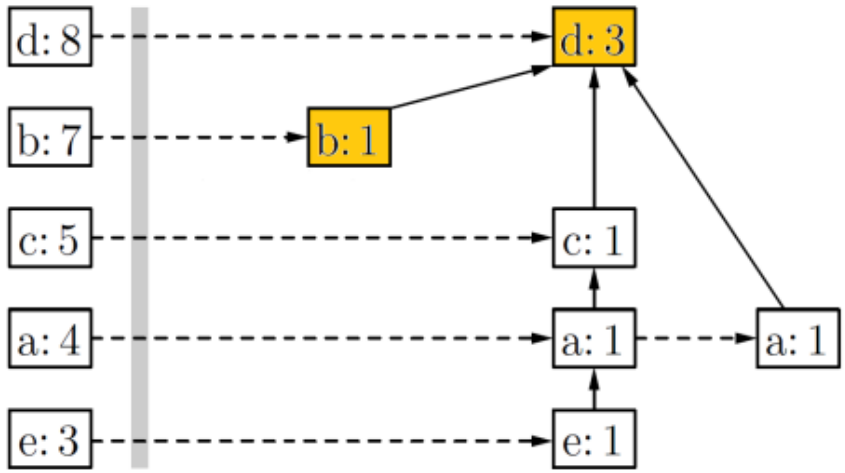
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



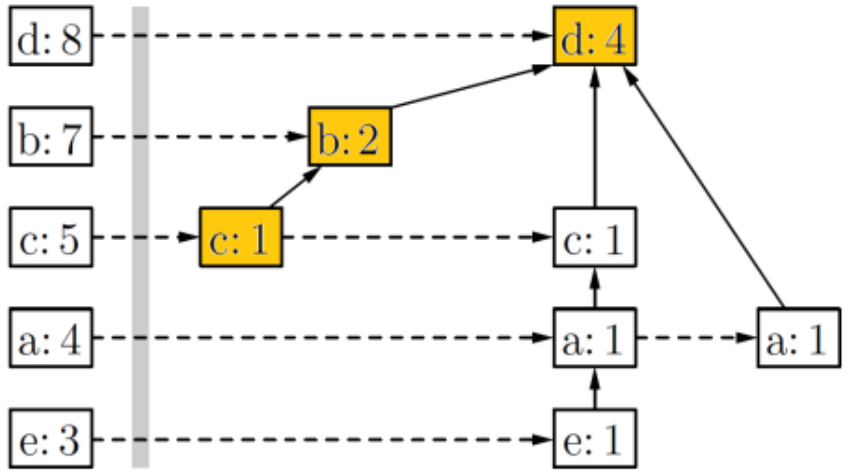
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



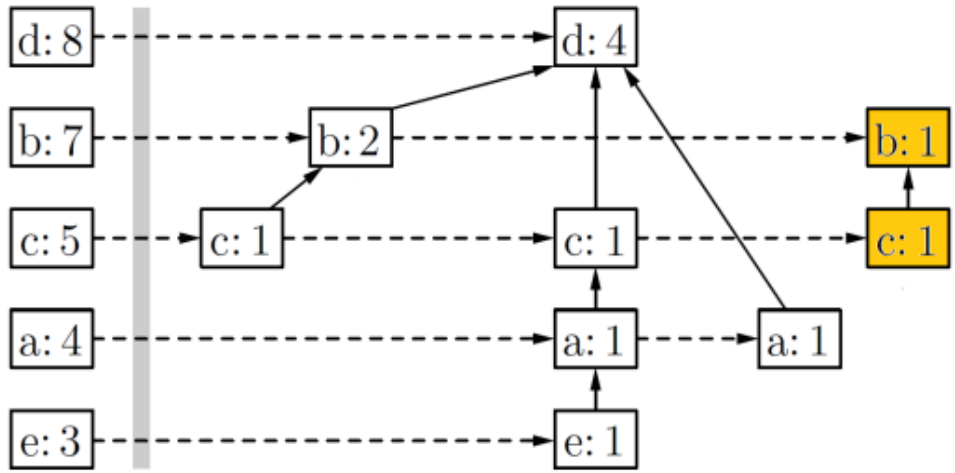
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



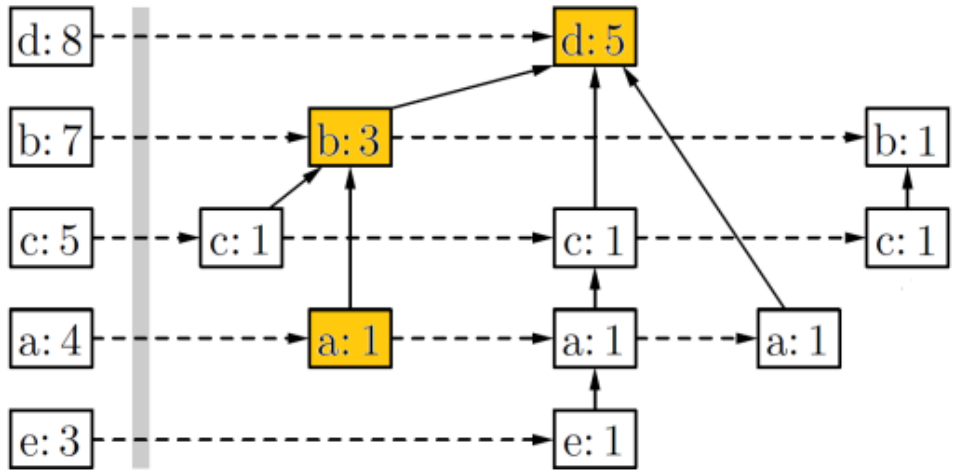
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



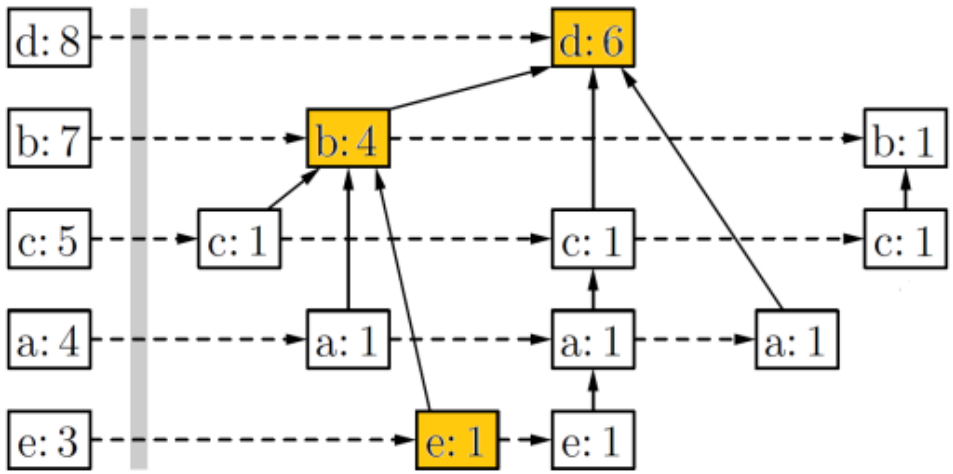
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



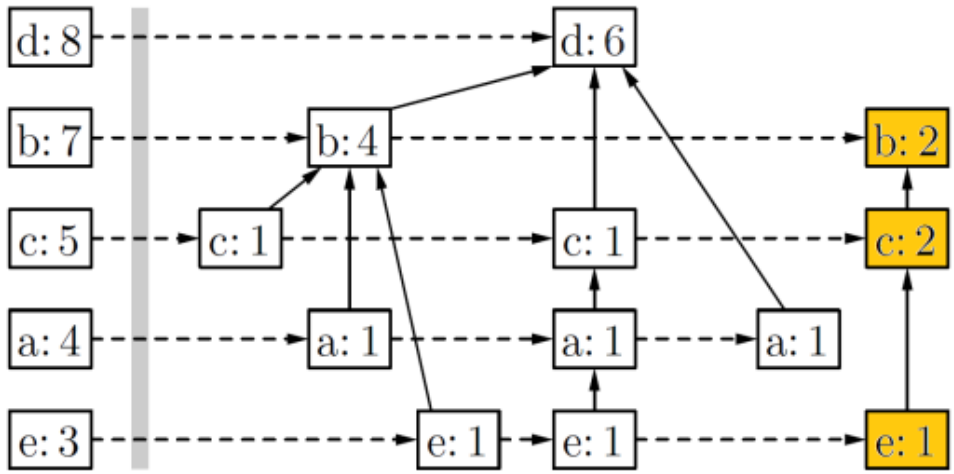
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



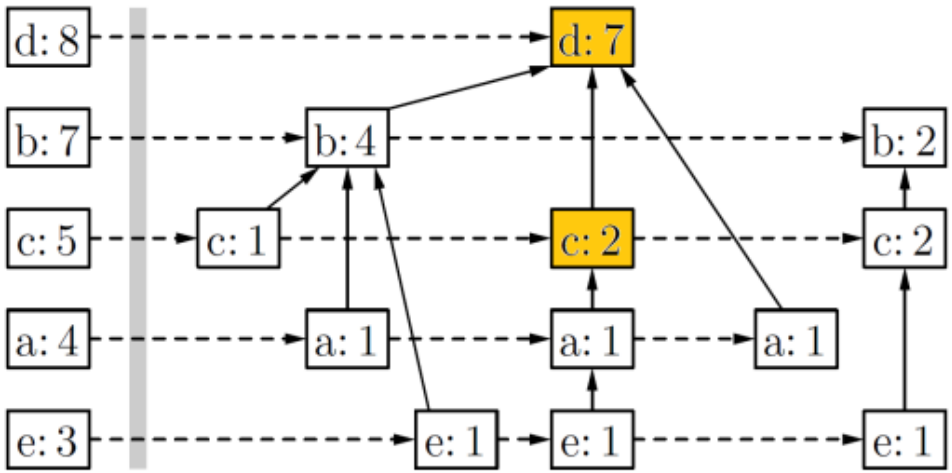
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



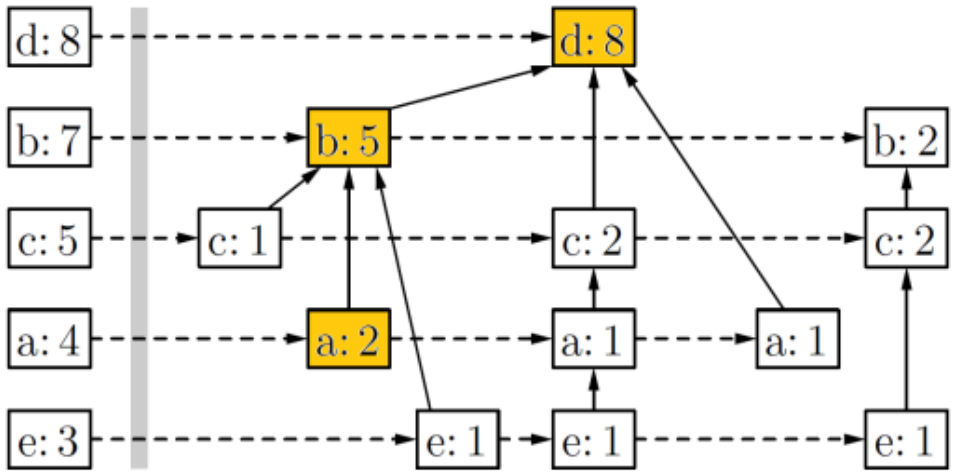
при $\delta = 3$ признаки f, g не частые

Упорядочим все признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$.

Тогда каждый объект описывается словом в алфавите $\mathscr{F}$;
FP-дерево — это эффективный способ хранения словаря;
уровни дерева соответствуют признакам, по убыванию $\nu(f)$.

матрица	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a

(корень v_0 не показан)



при $\delta = 3$ признаки f, g не частые

Вход: X^ℓ — обучающая выборка;
Выход: FP-дерево T, $\langle f_v, c_v, S_v \rangle_{v \in T}$;

1: упорядочить признаки $f \in \mathscr{F}$: $\nu(f) \geqslant \delta$ по убыванию $\nu(f)$;
ЭТАП 1: построение FP-дерева T по выборке X^ℓ
2: **для всех** $x_i \in X^\ell$
3: $v := v_0$;
4: **для всех** $f \in \mathscr{F}$ таких, что $f(x_i) \neq 0$
5: **если** нет дочерней вершины $u \in S_v$: $f_u = f$ **то**
6: создать новую вершину u; $S_v := S_v \cup \{u\}$;
$f_u := f$; $c_u := 0$; $S_u := \varnothing$;
7: $c_u := c_u + 1/\ell$; $v := u$;
8: *ЭТАП 2: рекурсивный поиск частых наборов по FP-дереву T*
FP-find $(T, \varnothing, \varnothing)$;

Вход: FP-дерево T, набор $\varphi \subset \mathscr{F}$, список правил R;
Выход: добавить в R все частые наборы, содержащие φ;

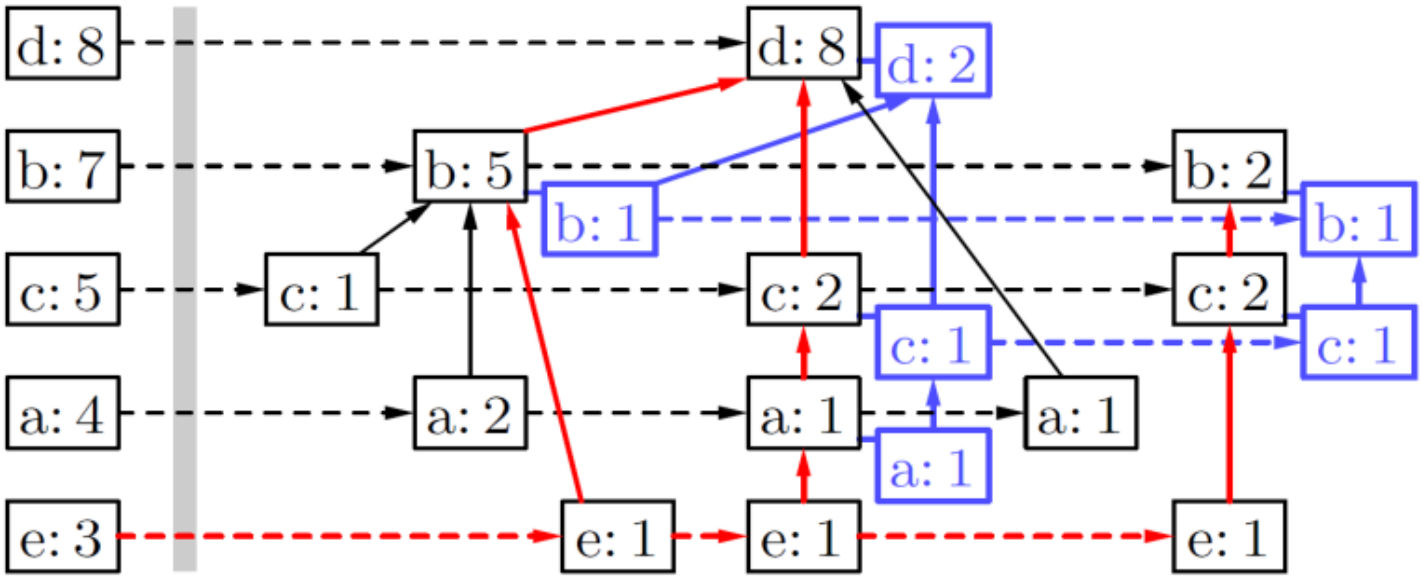
1: **ПРОЦЕДУРА** FP-find (T, φ, R);
2: **для всех** $f \in \mathscr{F}: V(T, f) \neq \varnothing$ по уровням снизу вверх
3: **если** $C(T, f) \geqslant \delta$ **то**
4: добавить частый набор $\varphi \cup \{f\}$ в список R:
$R := R \cup \{\varphi \wedge f\}$;
5: построить условное FP-дерево $T' := T|f$, а именно:
$T' :=$ FP-дерево по подвыборке $\left\{x_i \in X^\ell : f(x_i) = 1\right\}$;
6: найти по T' все частые наборы, включающие φ и f:
FP-find $\left(T', \varphi \cup \{f\}, R\right)$;

Условное FP-дерево $T' := T|f$ можно построить быстро, используя только FP-дерево T и не заглядывая в выборку.

Пусть FP-дерево T построено по выборке X^ℓ.

Опр. *Условное FP-дерево* (conditional FP-tree) — это FP-дерево $T' := T|f$, построенное по подвыборке $\{x_i \in X^\ell : f(x_i) = 1\}$, из которого удалены все вершины $v \in V(T', f)$ и все их потомки.

Продолжение примера: CFP-дерево $T|$"e"



Вход: FP-дерево T, признак $f \in \mathscr{F}$;
Выход: условное FP-дерево $T' = T|f$;

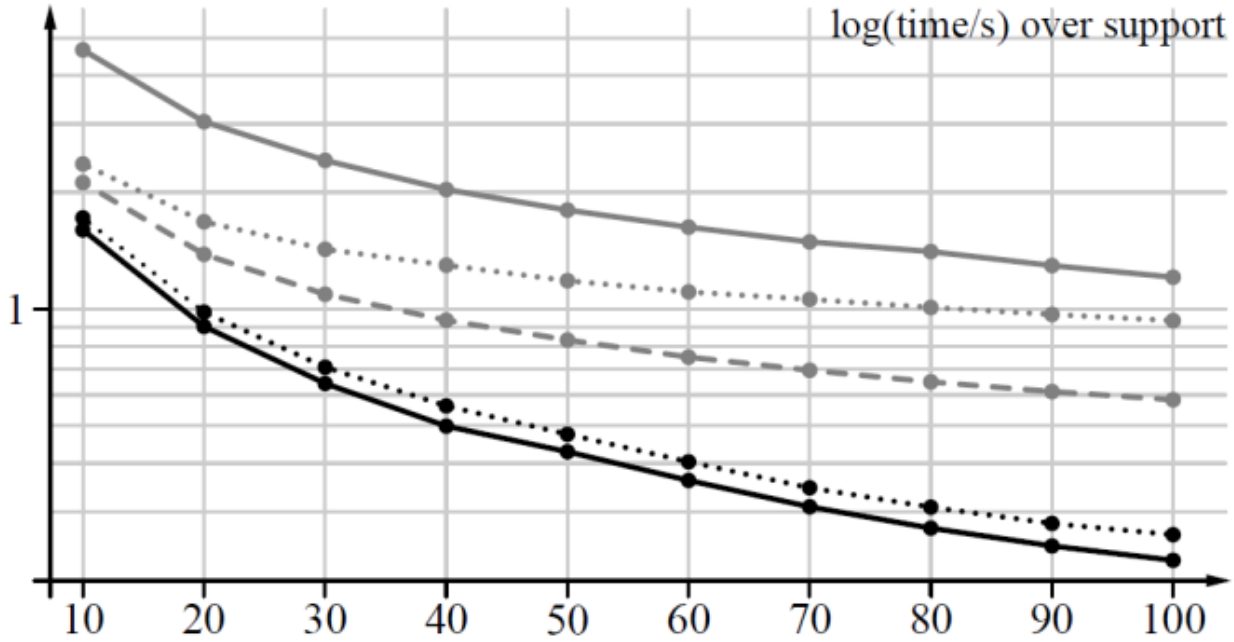
1: оставить в дереве только вершины на путях из вершин v признака f снизу вверх до корня v_0:
$T' := \bigcup\limits_{v \in V(T,f)} [v, v_0]$;

2: поднять значения счётчиков c_v от вершин $v \in V(T', f)$ снизу вверх по правилу
$c_u := \sum\limits_{w \in S_u} c_w$ для всех $u \in T'$;

3: удалить из T' все вершины признака f;
их поддеревья также не нужны и даже не создаются, т.к. в момент вызова FP-find все наборы, содержащие признаки ниже f, уже просмотрены.

Одна из типичных зависимостей log времени работы алгоритма от MinSupp (на выборке данных census).

Нижние кривые — две разные реализации FP-growth.

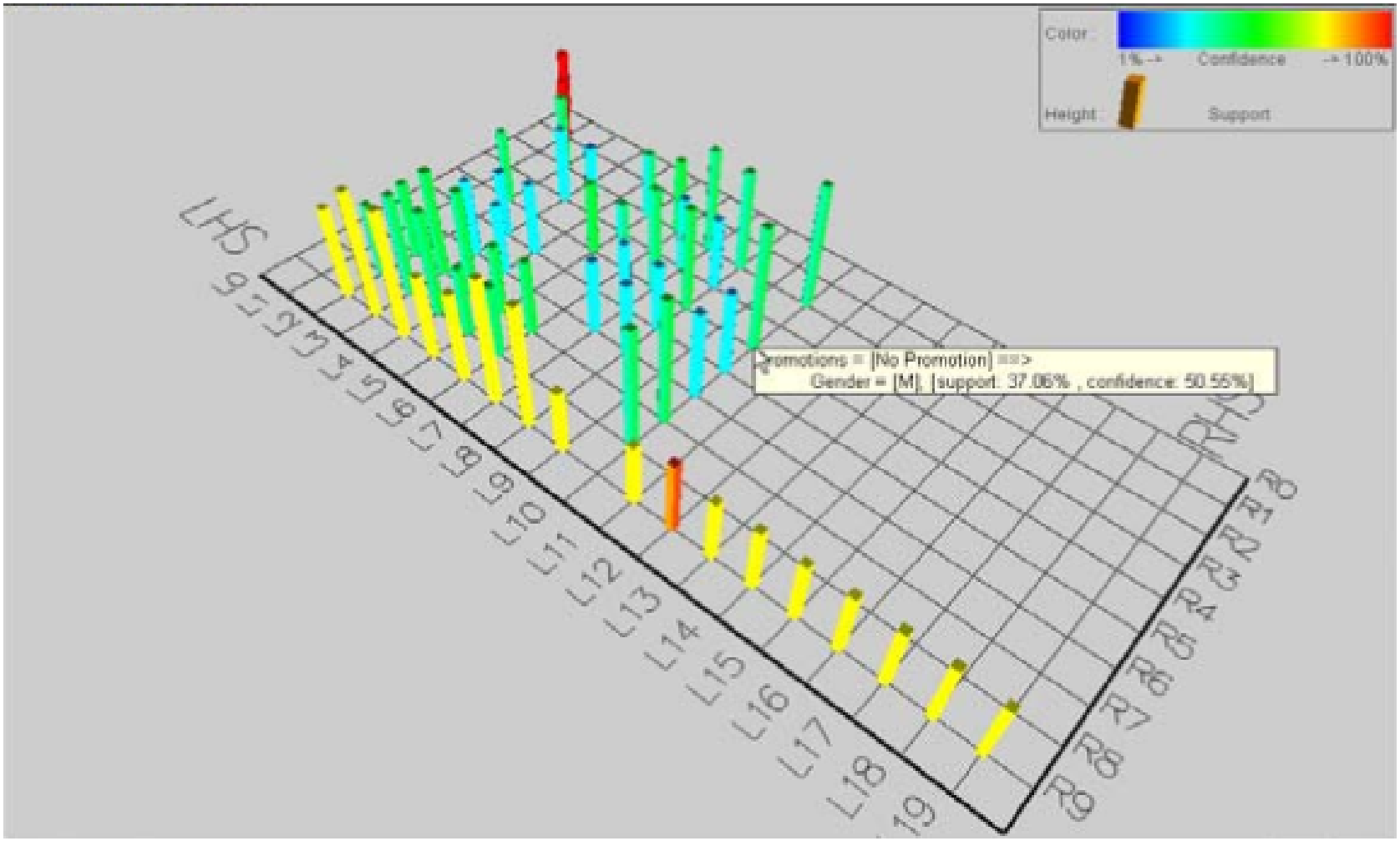


Christian Borgelt. An Implementation of the FPgrowth Algorithm. 2005.

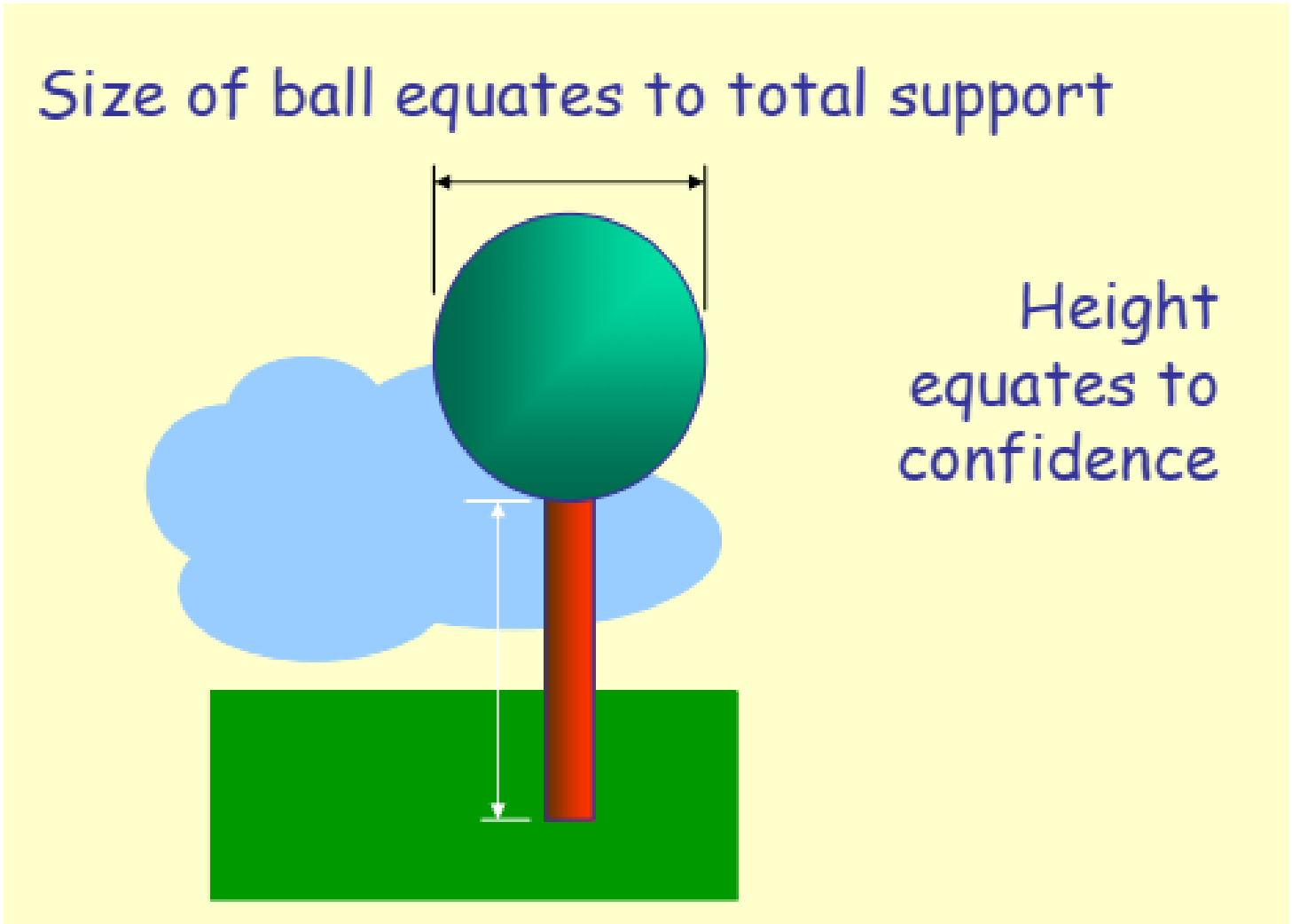
Содержание лекции

Визуализация ассоциативных правил

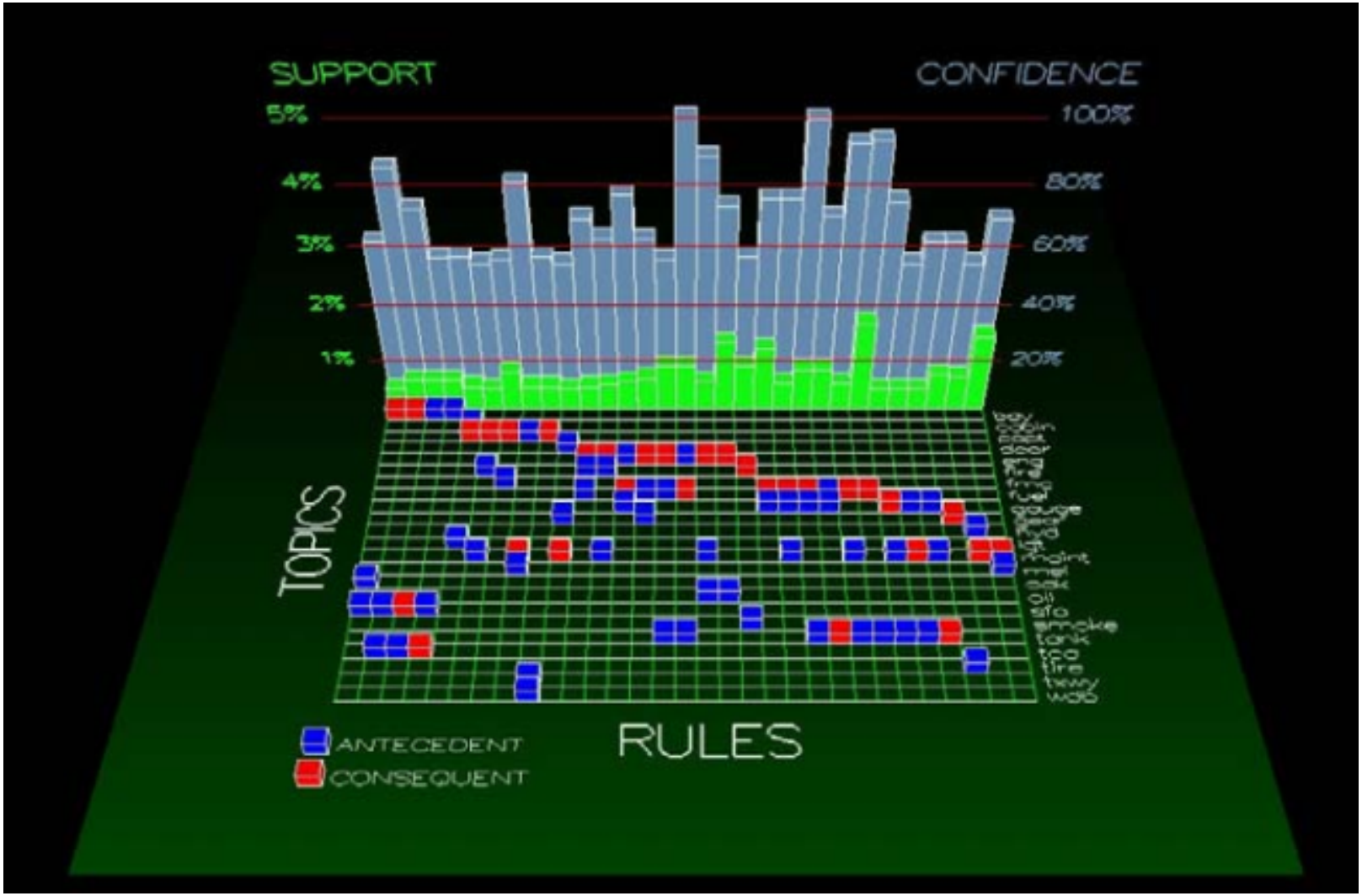
Визуализация ассоциативных правил



Визуализация ассоциативных правил



Визуализация ассоциативных правил



Выводы

- Поиск ассоциативных правил — обучение без учителя.
- Простые алгоритмы типа APriory вычислительно неэффективны на больших данных.
- FP-growth — один из самых эффективных алгоритмов поиска ассоциативных правил.
- Для практических приложений часто используются его инкрементные и/или иерархические обобщения.