Методы машинного обучения. Обучение без учителя: поиск ассоциативных правил

Воронцов Константин Вячеславович
www.MachineLearning.ru/wiki?title=User:Vokov
вопросы к лектору: voron@forecsys.ru

материалы курса:
github.com/MSU-ML-COURSE/ML-COURSE-21-22
орг.вопросы по курсу: ml.cmc@mail.ru

МФТИ • 29 февраля 2022

Выявление структуры данных на основе сходства:

- кластеризация (clustering)
- оценивание плотности распределения (density estimation)
- одноклассовая классификация (anomaly detection)

Преобразование признакового пространства:

- Метод главных компонент (principal components analysis)
- Матричные разложения (matrix factorization)
- Автокодировщики (autoencoders)
- Многомерное шкалирование (multidimensional scaling)

Поиск взаимосвязей в данных путём синтеза учителя:

- Поиск ассоциативных правил (association rule learning)
- Самостоятельное обучение (self-supervised learning)
- Состязательные сети (generative adversarial net)

Содержание

Определения и обозначения

X — пространство объектов
$X^\ell = \{x_1, \dots, x_\ell\} \subset X$ — обучающая выборка
$\mathscr{F} = \{f_1, \dots, f_n\}$, $f_j\colon X \to \{0,1\}$ — бинарные признаки (items)

Каждому подмножеству $\varphi \subseteq \mathscr{F}$ соответствует конъюнкция

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

Если $\varphi(x) = 1$, то «признаки из φ *совместно встречаются* у x»

Частота встречаемости (*поддержка*, support) φ в выборке X^ℓ

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

Если $\nu(\varphi) \geqslant \delta$, то «набор φ *частый*» (frequent itemset)
Параметр δ — *минимальная поддержка*, `MinSupp`

Определения и обозначения

Определение

Ассоциативное правило (association rule) $\varphi \to y$ — *это пара непересекающихся наборов* $\varphi, y \subseteq \mathscr{F}$ *таких, что:*

1) наборы φ *и* y *совместно часто встречаются,*

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

2) если встречается φ*, то часто встречается также и* y*,*

$$\nu(y|\varphi) \equiv \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \geqslant \varkappa.$$

$\nu(y|\varphi)$ — *значимость* (confidence) правила.

Параметр δ — *минимальная поддержка*, `MinSupp`.

Параметр $\varkappa$ — *минимальная значимость*, `MinConf`.

Классический пример

Анализ рыночных корзин (market basket analysis) [1993]
признаки — товары (предметы, items)
объекты — чеки (транзакции)

$f_j(x_i) = 1$ — в i-м чеке зафиксирована покупка j-го товара.

Пример: «если куплен хлеб φ, то будет куплено и молоко y с вероятностью $\nu(y|\varphi) = 60\%$; причём оба товара покупаются совместно с вероятностью $\nu(\varphi \cup y) = 2\%$».

Возможные применения:
- оптимизировать размещение товаров на полках,
- формировать персональные рекомендации,
- планировать рекламные кампании (промо-акции),
- более эффективно управлять ценами и ассортиментом.

Ассоциативные правила — это логические закономерности

Определение

Предикат $\varphi(x)$ — *логическая закономерность класса* $c \in Y$

$$\mathrm{Supp}(\varphi) = \frac{p(\varphi)}{\ell} \geqslant \delta; \qquad \mathrm{Conf}(\varphi) = \frac{p(\varphi)}{p(\varphi)+n(\varphi)} \geqslant \varkappa$$

$$p(\varphi) = \#\bigl\{x_i \in X^\ell \colon \varphi(x_i) = 1 \text{ и } y(x_i) = c\bigr\} \quad +\text{примеры класса } c$$
$$n(\varphi) = \#\bigl\{x_i \in X^\ell \colon \varphi(x_i) = 1 \text{ и } y(x_i) \neq c\bigr\} \quad -\text{примеры класса } c$$

Для «$\varphi \to y$» возьмём целевой признак $y(x) = \bigwedge\limits_{f \in y} f(x)$. Тогда

$$\nu(\varphi \cup y) \equiv \mathrm{Supp}_1(\varphi) \geqslant \delta; \quad \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \equiv \mathrm{Conf}_1(\varphi) \geqslant \varkappa$$

Вывод: различия двух определений — чисто терминологические

Два этапа построения правил. Свойство антимонотонности

Поскольку $\varphi(x) = \bigwedge\limits_{f\in\varphi} f(x)$ — 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).

Алгоритм APriory (основная идея — поиск в ширину)

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

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

Выделение ассоциативных правил

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

функция AssocRules (R, φ, y)
- **вход:** (φ, y) — ассоциативное правило;
- **выход:** R — список ассоциативных правил;
- **для всех** $f \in \varphi$: $\mathrm{id}_f > \max\limits_{g \in y} \mathrm{id}_g$ (чтобы избежать повторов y)
  - $\varphi' := \varphi \setminus \{f\}$; $\quad y' := y \cup \{f\}$;
  - **если** $\nu(y'|\varphi') \geqslant \varkappa$ **то**
    - добавить ассоциативное правило (φ', y') в список R;
    - **если** $|\varphi'| > 1$ **то**
      - AssocRules (R, φ', y');

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

Модификации алгоритмов индукции ассоциативных правил

- Более эффективные структуры данных для быстрого поиска частых наборов.
- Поиск правил по случайной подвыборке объектов при пониженных $\delta, \varkappa$, проверка правил на полной выборке.
- Иерархические алгоритмы, учитывающие иерархию признаков (например, товарное дерево).
- Учёт времени: инкрементные и декрементные алгоритмы.
- Учёт времени: поиск последовательных шаблонов (sequential pattern).
- Учёт информации о клиентах.

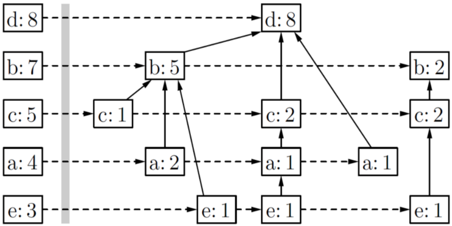
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

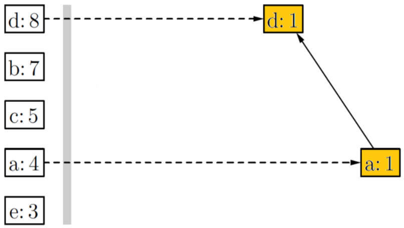
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

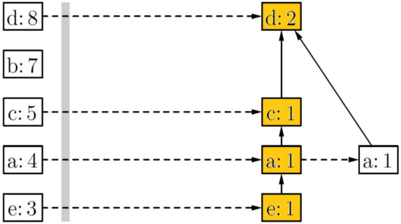
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

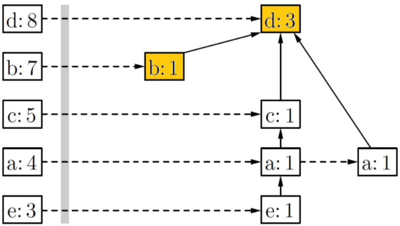
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

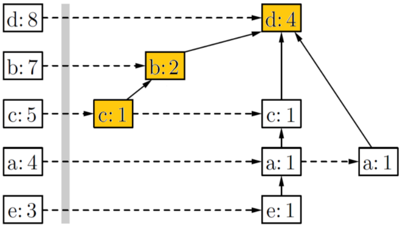
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

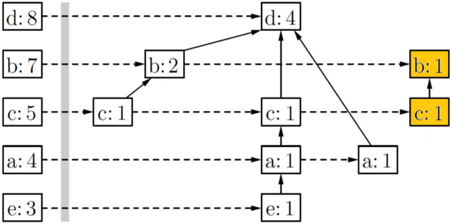
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

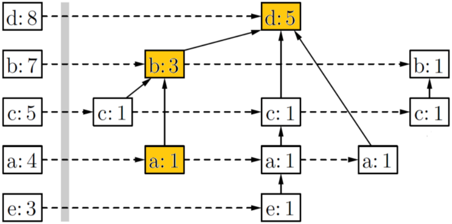
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

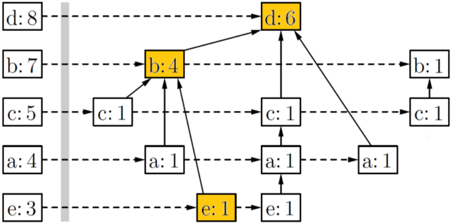
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

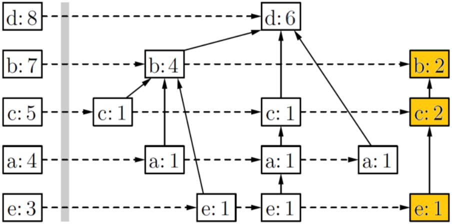
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

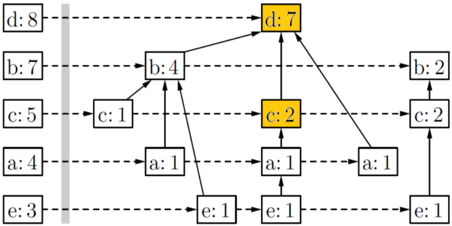
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

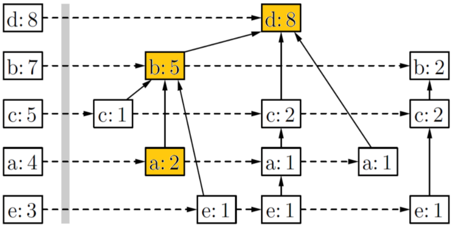
Префиксное FP-дерево (FP — frequent pattern). Пример.

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

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

матрица, $\ell = 10$	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 $\delta = \frac{3}{\ell}$ признаки f, g не частые

Префиксное FP-дерево (FP — frequent pattern)

В каждой вершине v дерева T задаётся тройка $\langle f_v, c_v, S_v \rangle$:

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

Обозначения:

$V(T, f) = \{v \in T \colon f_v = f\}$ — все вершины признака (уровня) f.
$C(T, f) = \sum\limits_{v \in V(T,f)} c_v$ — сумма счётчиков поддержки признака f.

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

1. $\frac{1}{\ell} C(T, f) = \nu(f)$ — поддержка признака f.
2. T содержит информацию о $\nu(\varphi)$ всех частых наборов φ.

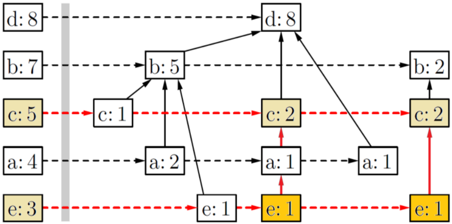
FP-дерево содержит информацию о всех частых наборах

Как по FP-дереву найти $\nu(\varphi)$ для произвольного набора φ:

1. выделить пути $[v_0, v]$, содержащие все признаки из φ
2. суммировать c_v нижних вершин всех таких путей

Пример: $\varphi = \{$"c", "e"$\}$, две записи, два пути, $\nu(\varphi) = \frac{2}{\ell}$:

матрица, $\ell = 10$	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



Алгоритм FP-growth

вход: X^ℓ — обучающая выборка;
выход: FP-дерево T; $\langle f_v, c_v, S_v \rangle$ для всех вершин $v \in T$;

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

Этап 2: рекурсивный поиск частых наборов по FP-дереву

FP-find(T, φ, R) находит по FP-дереву T все частые наборы, содержащие *частый набор* φ, и добавляет их в список R.

Две идеи эффективной реализации FP-find:

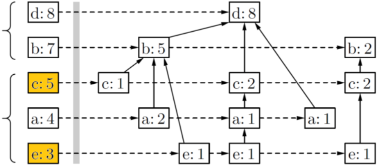
1. Вместо T достаточно передать *условное FP-дерево* $T|\varphi$, это FP-дерево, порождаемое подвыборкой $\{x_i \in X^\ell \colon \varphi(x_i) = 1\}$

2. Будем добавлять в φ только те признаки, которые находятся выше в FP-дереве. Так мы переберём все подмножества $\varphi \subseteq \mathscr{F}$.

Пример: $\varphi = \{\text{“c”}, \text{“e”}\}$

признаки для добавления в φ

признаки, из которых выбиралось φ



Этап 2: рекурсивный поиск частых наборов по FP-дереву

функция FP-find (T, φ, R)
- **вход**: FP-дерево T, частый набор φ, список наборов R;
- **выход**: добавить в R все частые наборы, содержащие φ;
- **для всех** $f \in \mathscr{F}$: $V(T, f) \neq \varnothing$ по уровням снизу вверх
  - **если** $C(T, f) \geqslant \ell\delta$ **то**
    - добавить частый набор $\varphi \cup \{f\}$ в список R;
    - $T' := T|f$ — условное FP-дерево;
    - найти по T' все частые наборы, включающие φ и f:
    - FP-find$\bigl(T', \varphi \cup \{f\}, R\bigr)$;

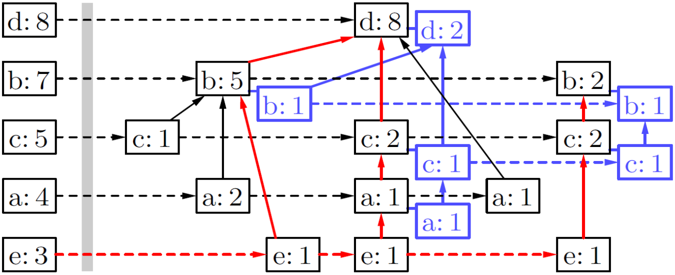
Условное FP-дерево $T' = T|f$ можно построить быстро, используя только FP-дерево T и не заглядывая в выборку.

Условное FP-дерево

Пусть FP-дерево T построено по подвыборке $U \subseteq X^\ell$.

Опр. *Условное FP-дерево* (conditional FP-tree) $T' = T|f$ — это FP-дерево, порождаемое подвыборкой $\{x_i \in U \colon f(x_i) = 1\}$, из которого удалены все вершины признака f и ниже.

Пример: CFP-дерево $T|$“e”



Быстрое построение условного FP-дерева $T' = T|f$

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

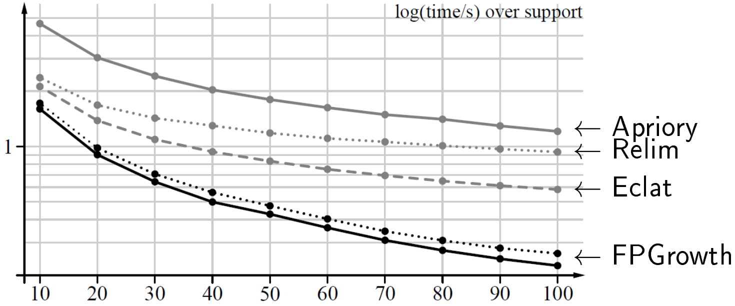
1 оставить в дереве только вершины на путях из вершин v признака f снизу вверх до корня v_0:
$$T' := \bigcup_{v \in V(T,f)} [v, v_0];$$
2 поднять значения счётчиков c_v от вершин $v \in V(T', f)$ снизу вверх по правилу
$$c_u := \sum_{w \in S_u} c_w \text{ для всех } u \in T';$$
3 удалить из T' все вершины признака f;

В дереве $T' = T|f$ остаются только признаки выше f, т.к. в момент вызова FP-find все наборы, содержащие признаки ниже f, уже просмотрены.

Эффективность алгоритма FPGrowth

Зависимость $\log_{10}$ времени работы алгоритма от MinSupp в сравнении с другими алгоритмами (на данных census).

Нижние кривые — две разные реализации FP-growth.



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

Резюме в конце лекции

- Поиск ассоциативных правил — обучение без учителя.
- Ассоциативное правило (по определению) — почти то же самое, что логическая закономерность.
- Простые алгоритмы типа APriory вычислительно неэффективны на больших данных.
- FP-growth — один из самых эффективных алгоритмов поиска ассоциативных правил.
- Для практических приложений используются его инкрементные и/или иерархические обобщения.