ПОИСК ПОВТОРЯЮЩИХСЯ ШАБЛОНОВ И АССОЦИАТИВНЫХ ПРАВИЛ

Все в мире повторяется.
Ф. Бэкон

Технологии анализа данных

Содержание

2

- Основные понятия
- Основные алгоритмы

Повторяющиеся шаблоны

3

- *Повторяющийся шаблон (frequent pattern)* – шаблон (множество предметов, последовательностей, структур и др.), который часто встречается в исходном множестве данных.
- Мотивация: найти скрытые закономерности в данных
  - Какие товары часто продаются совместно? (пиво и подгузники :-)
  - Какие цепочки ДНК наиболее вероятно составляют новое лекарство?
  - Как автоматически классифицировать web-документы?

Как демографическая ситуация влияет на покупки?

Имеет ли значение при покупке определенная марка молока?

Где в магазине нужно разместить помидоры для увеличения их продаж?

Хлеб обычно покупается вместе с молоком?

Хлеб покупается тогда, когда покупаются одновременно молоко и яйца?

Основные понятия

4

- *Набор товаров (itemset)*
  - {молоко, хлеб, сахар}
  - *k-элементный набор (k-itemset)*
- *Поддержка (support)*
  - *Support count P* – частота встречаемости заданного набора
    - *P*({молоко, хлеб, сахар})=2
  - *Поддержка s* – доля транзакций, содержащих заданный набор
    - *sup*({молоко, хлеб, сахар})=2/5
- *Часто встречающийся набор товаров (frequent itemset)* – набор, имеющий поддержку не ниже заданного порога *minsup*.
- *Задача анализа рыночной корзины (market basket analysis problem)* – для заданного порога *minsup* найти все часто встречающиеся наборы товаров.

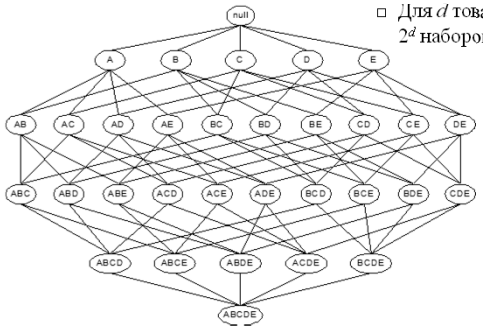
TID	Транзакция
1	хлеб, молоко
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар
4	хлеб, кофе, молоко, сахар
5	хлеб, кола, молоко, сахар

Технологии анализа данных © М.Л. Цымблер

Нахождение частых наборов

5

- Для d товаров имеем 2^d наборов-кандидатов



Технологии анализа данных © М.Л. Цымблер

Нахождение частых наборов

6

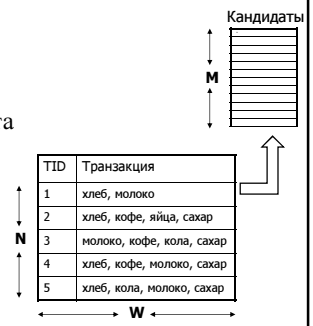
- Brute force
- Apriori
- Вертикальный формат данных

Технологии анализа данных © М.Л. Цымблер

Brute force

7

- Каждый набор является кандидатом
- Подсчитать поддержку каждого набора-кандидата
- Проверить каждую транзакцию для каждого набора-кандидата
- Сложность: $O(N \cdot M \cdot W) = O(N \cdot 2^d \cdot W)$



TID	Транзакция
1	хлеб, молоко
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар
4	хлеб, кофе, молоко, сахар
5	хлеб, кола, молоко, сахар

Стратегии

8

- Уменьшение количества наборов-кандидатов M
  - Полный перебор: $M=2^d$
  - Использование различных техник отсечения
- Уменьшение количества транзакций N
  - Уменьшение N при увеличении размера набора
  - Использование вертикального формата данных
- Уменьшение количества сравнений $N \cdot M$
  - Использование эффективных структур данных для хранения наборов-кандидатов или транзакций
  - Отсутствие необходимости проверки каждого кандидата для каждой транзакции

Уменьшение количества кандидатов

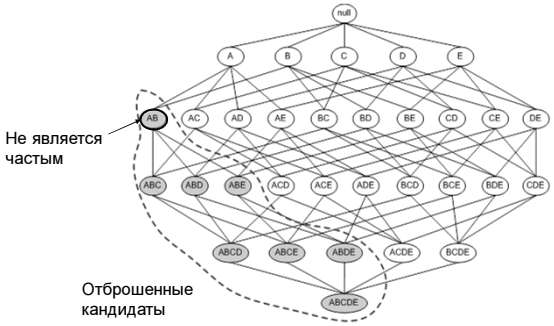
9

- *Принцип Apriori*
  - Если набор является частым, то все его подмножества также являются частыми
- *Антимонотонность поддержки*
  - Поддержка набора не превышает поддержки его подмножеств

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

Принцип Apriori

10



Принцип Apriori

11

minsup=3

1-наборы

Набор	К-во
{хлеб}	4
{кола}	2
{молоко}	4
{кофе}	3
{сахар}	4
{яйца}	1

2-наборы

Набор	К-во
{хлеб, молоко}	3
{хлеб, кофе}	2
{хлеб, сахар}	3
{молоко, кофе}	2
{молоко, сахар}	3
{кофе, сахар}	3

3-наборы

Набор	К-во
{хлеб, молоко, сахар}	3

(нет необходимости генерировать наборы-кандидаты, содержащие колу или яйца)

Полный перебор подмножеств:
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
Отбрасывание на основе поддержки:
6 + 6 + 1 = 13

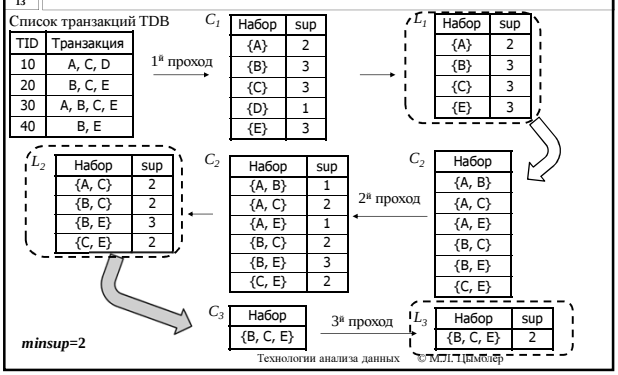
Алгоритм Apriori: идеология

12

- Принцип:
  - для любого набора, который не является частым, его надмножество не должно попасть в список наборов-кандидатов
- Метод:
  - Сканировать все транзакции для получения частых 1-наборов
  - Генерировать (k+1)-наборы-кандидаты из частых k-наборов
  - Проверить кандидаты по всем транзакциям
  - Если отсутствуют частые наборы или наборы-кандидаты, закончить

Алгоритм Apriori: пример

13



Алгоритм Apriori

14

C_k: k-набор-кандидат
L_k : частый k-набор

L_1 = {частые товары};
for (k=1; L_k!=∅; k++) **do begin**
  C_{k+1}=наборы-кандидаты, сгенерированные из L_k;
  for each t in TDB **do**
    увеличить счетчики кандидатов в C_{k+1}, содержащихся в t
    L_{k+1}=кандидаты из C_{k+1} с поддержкой ≥ *minsup*
  end
return $\cup_k L_k$;

Важные детали Apriori

15

- Как генерировать наборы-кандидаты?
  - Шаг 1: соединение (self-joining) L_k*L_k
  - Шаг 2: отсечение (pruning)
- Как вычислять поддержку наборов-кандидатов?
- Пример генерации кандидатов
  - L_3={*abc*, *abd*, *acd*, *ace*, *bcd*}
  - Соединение L_3*L_3
    - *abcd* из *abc* и *abd*
    - *acde* из *acd* и *ace*
  - Отбрасывание
    - *acde* удалено, поскольку *ade* ∉ L_3
  - C_4={*abcd*}

Как генерировать кандидатов?

16

- Пусть элементы L_{k-1} упорядочены
- Шаг 1: соединение L_{k-1}

  insert into C_k

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

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

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Шаг 2: отсечение

  forall ***itemsets c in*** C_k do

  forall ***(k-1)-подмножеств s из c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Факторы, влияющие на сложность

17

- Выбор *minsup*
  - Чем меньше *minsup*, тем больше частых наборов. Это увеличивает количество кандидатов и макс. длину частых наборов.
- Количество товаров
  - Необходимость хранения счетчика поддержки для каждого товара.
  - Если количество частых товаров возрастает, то возрастают накладные расходы на вычисления и ввод-вывод
- Количество транзакций
  - Поскольку Ariori предполагает множество проходов, время работы возрастает с увеличением количества транзакций
- Средняя длина транзакции
  - Количество подмножеств увеличивается при увеличении мощности множества

Улучшение Apriori

18

- Уменьшение количества сравнений
- Уменьшение количества транзакций
- Фрагментация данных при поиске кандидатов
- Сэмплинг: анализ подмножества исходных данных

Уменьшение количества сравнений

19

- При сканировании транзакций для подсчета поддержки каждого набора-кандидата количество сравнений можно уменьшить, если хранить кандидатов в хеш-структуре
  - Вместо проверки каждого кандидата – проверка хеш-адреса кандидата

h(i1,i2) = (Ord(i1) * 10 + Ord(i2) mod 7

H2

minsup = 3

h(i1,i2)	0	1	2	3	4	5	6
Поддержка наборов	2	2	4	2	2	4	4
Наборы	{A,D} {C,E}	{A,E} {A,E}	{B,C} {B,C} {B,C} {B,C}	{B,D} {B,D}	{B,E} {B,E}	{A,B} {A,B} {A,B} {A,B}	{A,C} {A,C} {A,C} {A,C}

Уменьшение количества транзакций

20

- Транзакция, которая не содержит частый *k*-набор, не может содержать любой частый *j*-набор для *j>k*, и может не просматриваться при обработке *j*-наборов.

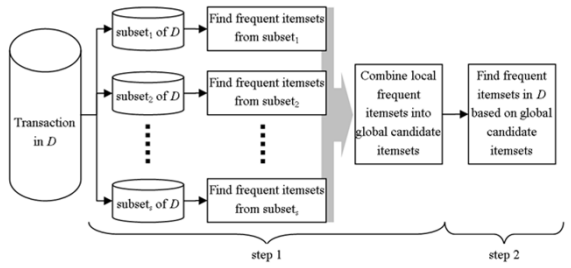
Фрагментация данных при поиске кандидатов

21

- Требует лишь двух просмотров списка транзакций, если каждый фрагмент может разместиться в оперативной памяти.
- Фаза 1
  - Разделить TDB на *n* фрагментов
  - Для каждого фрагмента найти локальные частые наборы. Частый набор фрагмента *i* имеет поддержку *sup_count* > *minsup**d_i, где d_i – размер фрагмента.
  - Для каждого набора сохранить *tid* транзакций, включающих в себя набор.
- Фаза 2
  - Второй просмотр TDB, чтобы найти действительное значение каждого локального частого набора.

Фрагментация данных при поиске кандидатов

22



Сэмплинг: анализ подмножества исходных данных

23

- Выбрать случайным образом часть *S* исходных данных *TDB* так, чтобы *S* можно было разместить в оперативной памяти
- Найти частые наборы L_S в *S*. Можно при этом уменьшить *minsup*, чтобы уменьшить количество пропущенных частых наборов.
- При необходимости
  - Найти реальную поддержку наборов из L_S, используя *TDB-S*.
  - Если L_S, не содержит все частые наборы из *TDB*, выполнить сэмплинг повторно.

Узкое место Apriori

24

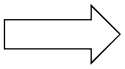
- Многочисленные просмотры TDB
- Анализ длинных шаблонов требует большого количества просмотров и генерации большого количества кандидатов
  - Пример: найти частые наборы для $i_1 i_2 \ldots i_{100}$
    - Количество просмотров: 100
    - Количество кандидатов: 2^{100}-1
- Узкое место: генерация и проверка кандидатов

Вертикальный формат данных

25

- Для каждого товара хранится список транзакций, в которые он входит.

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B



A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

tid-списки

Вертикальный формат данных

26

- Подсчет поддержки любого *k*-набора – мощность пересечения *tid*-списков двух его (*k*-1)-наборов.

A
1
4
5
6
7
8
9

∩

B
1
2
5
7
8
10

→

AB
1
5
7
8

- Плюсы: быстрый подсчет поддержки.
- Минусы: промежуточные *tid*-списки могут не помещаться в оперативной памяти.

Использование РСУБД

27

- Алгоритмы интеллектуального анализа данных могут быть реализованы в реляционных СУБД.
- Плюсы:
  - Данные, подлежащие интеллектуальному анализу, не нужно экспортировать для последующего использования внешними утилитами. Результаты анализа не нужно импортировать обратно в реляционную базу данных.
  - Нет ограничения на использование оперативной памяти.
- Минусы:
  - SQL менее гибок, чем ЯВУ.
  - РСУБД обычно менее эффективны, когда объем данных, подлежащих анализу, позволяет разместить их в оперативной памяти.

SQL: нахождение частых наборов

28

- Реляционная таблица для хранения транзакций
  - TDB (tid, item)

tid	Транзакция
1	хлеб, молоко
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар

tid	item
1	хлеб
1	молоко
2	хлеб
2	кофе
2	яйца
2	сахар
3	молоко
3	кофе
3	кола
3	сахар

SQL: нахождение частых наборов

29

```
-- Генерация кандидатов
insert into Cand
 select *
 from TDB
 where item in (
   select item
   from TDB
   group by item
   having count(*) >= minsup);
```


SQL: нахождение частых наборов

30

```
-- Частые 2-наборы
select A.item, B.item, count(A.tid) as sup_count
from Cand A, Cand B
where A.tid=B.tid and A.item<B.item
group by A.item, B.item
having count(A.tid) >= minsup;

-- Частые k-наборы (k>2)
-- Как автоматизировать k:=k+1 ?
```

Компактное представление частых наборов

31

- Некоторые наборы являются избыточными, поскольку имеют ту же поддержку, что и их надмножества.

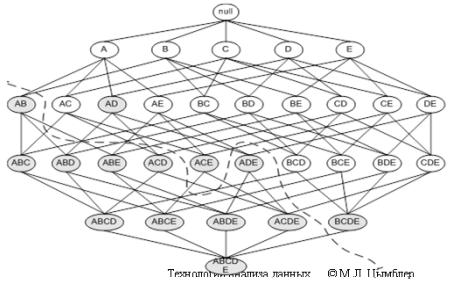
TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

- Кол-во частых наборов $= 3 \times \sum_{k=1}^{10} \binom{10}{k}$
- Нужно компактное представление

Максимально частые наборы

32

- Набор является *максимально частым*, если ни одно из его надмножеств не является частым.



Замкнутый набор

33

- Набор является *замкнутым*, если ни одно из его надмножеств не имеет ту же поддержку.

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

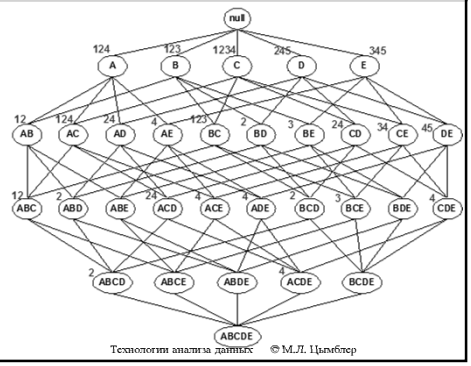
Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Максимальные и замкнутые наборы

34

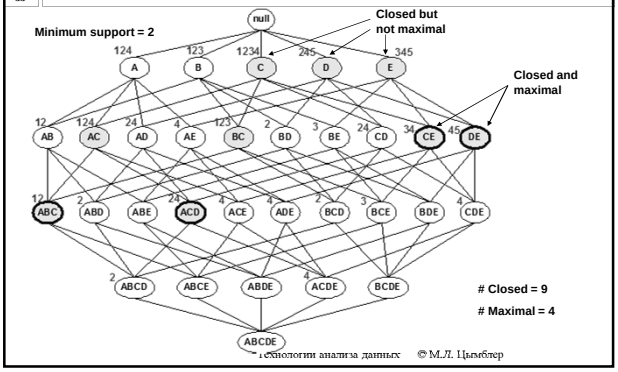
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

Номера показывают транзакции, в которых встречается набор



Максимальные и замкнутые частые наборы

35



Максимальные и замкнутые наборы

36

Ассоциативные правила: зачем?

38

- Используя данное множество транзакций, мы хотим найти правила, которые предскажут факт покупки товара на основе фактов покупки других товаров в транзакциях.

{сахар} → {кофе},
{молоко, хлеб} → {яйца, кола},
{кофе, хлеб} → {молоко}

TID	Транзакция
1	хлеб, молоко
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар
4	хлеб, кофе, молоко, сахар
5	хлеб, кола, молоко, сахар

Поддержка и доверие

39

- *Ассоциативное правило* на наборах X и Y – это выражение вида X→Y ("если X, то Y").
- *Поддержка правила (rule support)*
  - показывает долю транзакций, содержащих X и Y.
  - *sup*(X→Y)=P(X,Y)/|TID|
- *Доверие к правилу (rule confidence)*
  - показывает, как часто товары из Y возникают в транзакции, которая содержит X
  - *conf*(X→Y)=P(X,Y)/P(X)

Поддержка и доверие

40

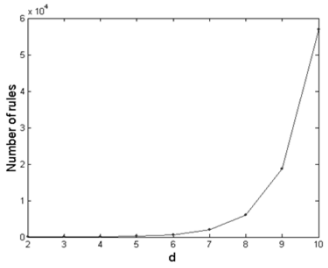
- {молоко,сахар}→{кофе}
  - sup=P(молоко,сахар,кофе)/|TID|=2/5
  - conf=P(молоко,сахар,кофе)/P(молоко,сахар)=2/3
- {кофе}→{молоко,сахар}
  - sup=2/5
  - conf=P(кофе,молоко,сахар)/P(кофе)=2/4

TID	Транзакция
1	хлеб, молоко, кофе
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар
4	хлеб, кофе, молоко, сахар
5	хлеб, кола, молоко, сахар

Вычислительная сложность

41

- *d* товаров
  - наборов: 2^d
  - правил: $3^d-2^{d+1}+1$



Поиск ассоциативных правил

42

- Для заданного множества транзакций *TDB* и пороговых значений *minsup* и *minconf* найти все правила, имеющие
  - поддержку ≥ *minsup*
  - доверие ≥ *minconf*
- Полный перебор
  - получить список всех возможных правил
  - вычислить поддержку и доверие каждого правила
  - отбросить правила, не удовлетворяющие пороговым значениям

Поиск ассоциативных правил

43

TID	Транзакция
1	хлеб, молоко, кофе
2	хлеб, кофе, яйца, сахар
3	молоко, кофе, кола, сахар
4	хлеб, кофе, молоко, сахар
5	хлеб, кола, молоко, сахар

Правило	**sup**	**conf**
{молоко,сахар}→кофе	0,4	0,67
{молоко,кофе}→сахар	0,4	1,0
{сахар,кофе} →молоко	0,4	0,67
кофе→{молоко,сахар}	0,4	0,67
сахар→{молоко,кофе}	0,4	0,5
молоко→{сахар,кофе}	0,4	0,5

- Наблюдения
  - правила – разбиение одного и того же набора
  - правила, полученные из одного и того же набора, имеют одну и ту же поддержку, но могут иметь разное доверие
  - при поиске правил требования поддержки и доверия можно отделить друг от друга.

Поиск ассоциативных правил

44

- Двухшаговый подход
  - Найти частые наборы
    - найти наборы с поддержкой больше *minsup*
  - Генерировать правила
    - генерировать правила с доверием больше *minconf* из каждого частого набора, где каждое правило является бинарным разбиением частого набора
- Нахождение частых наборов по-прежнему остается вычислительно сложной задачей

Заключение

45

- Задача анализа рыночной корзины – поиск часто встречающихся наборов товаров.
- Алгоритм Apriori поиска частых наборов.
- Поиск ассоциативных правил.