Введение в анализ данных: Поиск ассоциативных правил

Юля Киселёва
juliakiseleva@yandex-team.ru
Школа анализа данных

По мотивам прошлой лекции

Входная матрица

1	4	3
3	2	4
7	1	7
6	3	6
2	6	1
5	7	2
4	5	5

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

h

Матрица сигнатур

1	2	3	4
2	1	2	1
2	1	4	1
1	2	1	2

Похожести

	1-3	2-4	1-2	3-4
Колонки	0.75	0.75	0	0
Сигнатуры	0.67	1.00	0	0

План на сегодня

- Поиск частотных объектов
  - Мотивация
  - Ассоциативные правила
- Алгоритмы для поиска частотных объектов

Поиск ассоциативных правил

Организация товарных полок в супермаркете - Market-basket модель:

- Цель: Определить товары, которые покупаются вместе достаточно большим числом покупателей
- Способ: Проанализировать историю покупок пользователей, чтобы найти зависимости между товарами
- Классическое правило:
  - Если кто-нибудь покупает подгузник и молоко, то он с большой вероятностью приобретет пиво
  - Не удивляетесь если в супермаркете возле подгузников увидите пиво

Market-Basket Модель

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- Большое количество объектов
  - Например, товары проданные в супермаркете
- Большое количество корзин, каждая из которых набор объектов
  - Например, товары, которые один покупатель купил за одну покупку
- Найти «интересные» взаимосвязи в данных

Ассоциативные правила: Способ построения

Дано: набор корзин с товарами
Задача: найти ассоциативные правила:

- Пользователи, которые покупаю {x,y,z} также покупают {u,v}
- Пример: Amazon

2-х шаговый способ:

1. Найти частотные наборы объектов
2. Сформировать ассоциативные правила

Входные данные

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Выходные данные:
Построенные правила

{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Частотные объекты

- Дано: набор = ***I***, количество корзин = ***N***
- Простое предположение: Найти наборы объектов, которые встречаются «часто» в корзине
- Порог для наборе ***I***: общее количество корзин, которые содержат весь набор(=***K***)
  - Часто рассматривается как относительное число (***K/N***)
- Имеем порог ***s***, тогда все наборы, которые встречаются более чем в ***s*** корзинах будем называть частотными

Пример: частотные объекты

- Набор = {молоко, кока-кола, пепси, пиво, сок}
- Порог = 3

B1={м, к, пив} B2={м, пеп, с}

B3={м, пив} B4={к, с}

B5={м, пеп, пив} B6={м, к, пив, с}

B7={к, пив, с} B8={пив, к}

- Частотные наборы: {м}, {к}, {пив}, {с}
- {м, пив} {пив, к} {к, с}

Приложения (1)

- Наборы = товары; Корзины = набор товаров, который покупают за один поход в магазин
- Реальная рыночная корзина: магазины хранят терабайты данных о том, что покупают
  - Позволяет определять, как типичный покупатель ориентируется в магазине, что позволяет расставлять товары более заманчиво
  - "трюки", например, сделать скидки на подгузники и повысить цены на пиво
- Amazon: люди, которые купили *X* также купили *Y*

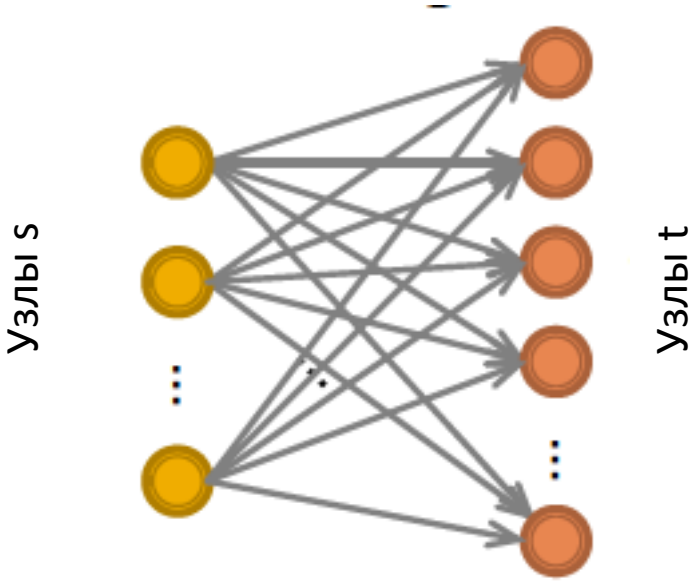
Приложение 2

- Корзины = предложения ; объекты = документы, которые содержат это предложение

  – Объекты, которые часто встречаются вместе (в одной корзине) возможно являются устойчивым выражением

Приложение 3

- Поиск групп в большом графе (например, в веб-графе)
- Корзины = узлы, объекты = соседние пользователи



Поиск двухстороннего подграфа Ks,t от большого графа

Данный граф помогает определить: О каких темах говорят одинаковые пользователи слева (темы справа)

План

- Определить:
  - Частотные объекты
  - Ассоциативные правила
    - Порог, уровень доверия, интересность
- 2 алгоритма для поиска частотных объектов
  - A-Priori алгоритм
  - PCY
  - Случайный отбор

Ассоциативные правила

- Ассоциативные правила:

  If-then правила, относящиеся к набору в корзине

- $\{i_1, i_2, \ldots, i_k\} \rightarrow j$ означает, что: «Если корзина содержит все объекты $i_1, \ldots, i_k$, тогда скорее всего она содержит и j»

- Уровень доверия представленного ассоциативного правила - это вероятность события j

$$I = \{i_1, \ldots, i_k\}$$

Интересные ассоциативные правила

- Не все ассоциативные правила с высоким уровнем доверия интересны
  - Правило X -> молоко имеет высокий уровень доверия для многих объектов X, так как молоко является часто покупаемым продуктом (независимо от X)
- Интересность ассоциативного правила *I -> j*: разность между его уровнем доверия и доля корзин, которые содержат *j*
- Интересными называются правила, которые имеют самые нижние и самые верхние значения для интереса

Пример: уровень доверия и интерес

B1={м, к, пив} B2={м, пеп, с}
B3={м, пив} B4={к, с}
B5={м, пеп, пив} B6={м, к, пив, с}
B7={к, пив, с} B8={пив, к}

- Ассоциативное правило: { м, пив} -> к

- Уровень доверия = 2/4 = 0.5
- Интересность = |0.5 – 5/8|=1/8
  - Кока-кола содержится в 5 корзинах из 8
  - Правило не представляет большого интереса

Поиск ассоциативных правил

- *Задача:* найти ассоциативные правила с порогом >= ***s*** и уровнем доверия >= ***c***
- ***Сложно:*** поиск частотных наборов
- если правило $\{i_1, i_2, \ldots, i_k\} \rightarrow j$ определяется высоким порогом и уровнем доверия, то оба набора $\{i_1, i_2, \ldots, i_k\}$ и $\{i_1, i_2, \ldots, i_k, j\}$ будут «частотными»

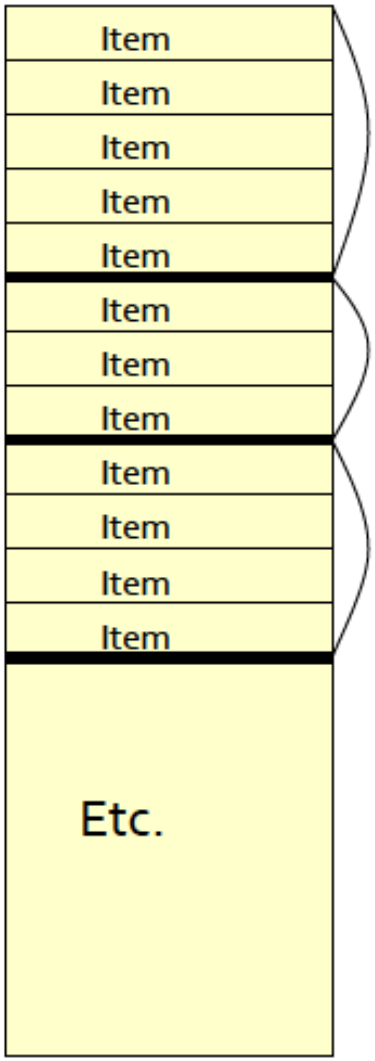
Анализ ассоциативных правил

- Шаг 1: поиск всех частотных наборов
  - Рассмотрим далее
- Пусть набор $I = \{i_1, i_2, \ldots, i_k\}$ частотный
- Шаг 2: Создание правила
  - Для каждого под поднабора ***A*** набора ***I*** **,** создадим правило $A \rightarrow I \setminus A$
    - Так как ***I*** частотный, тогда ***A*** тоже частотный
    - Вариант 1: за один проход вычислить уровень доверия для правила
    - Вариант 2: conf(AB->DC) = supp (ABCD)/supp(AB)
      - Наблюдения: если ABC-> D ниже уровня доверия, тогда и AB-> CD
  - Получаемые правила должны быть выше определенного уровня доверия

Вычислительная модель

- На практике данные чаще хранятся в файлах, а не в базе данных:
  - Хранение на диске
  - Данный сохраняются корзина за корзиной
  - Представление корзины в виде пары, троек и т.д., по мере того, как читаются данные из корзины
    - Необходимо k вложенных циклов, чтобы создать все наборы, размера k

Организация файла



Корзина 1

Корзина 2

Корзина 3

Вычислительная модель (2)

- На практике, алгоритмы для построения читает данные за проходы – каждая корзина читается в свою очередь
- Будем мерить производительность числом проходов, за которые алгоритм обходит данные
- Для многих алгоритмов для поиска частотных объектов, оперативная память – это узкое место
  - При чтении корзин, нужно что-то считать и сохранять
  - Число различных величин, которые мы можем посчитать ограничено оперативной памятью
  - Считать все в память – это дорого

Naïve Алгоритм: Посчет пар в памяти

- Подход 1:

  Сохраним тройки [i, j, c], где c = count(i,j)

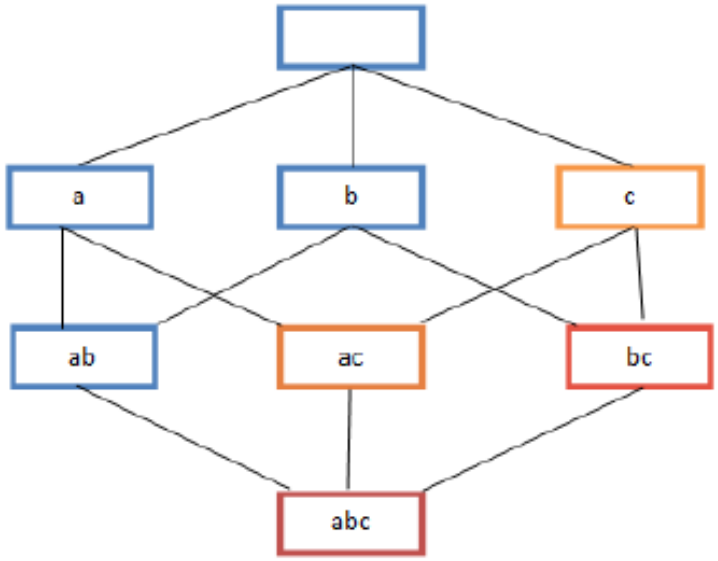
  Общее количество пар = n(n-1)/2

  Что если большинство пар не частотные?

Алгоритм на треугольной матрице

- Подход 2: Подсчитаем все пары
  - Число элементов 1,2,3, …, n
  - Count{i,j} только если i<j
- Сохраняем элементы в лексикографическом порядке: {1,2},{1,3}, …, {1,n}, {2,3},{2,4}, ….,{2,n},{3,4}, …
- Получаем менее 1/3 всех возможных пар

A-Priori Алгоритм



- Двухпроходный алгоритм

A-priori лимитирует потребности памяти

- Основная идея: монотонность

Если набор элементов ***I*** появился хотя бы s раз, тогда это справедливо и для поднабора ***J*** содержащегося в ***I***

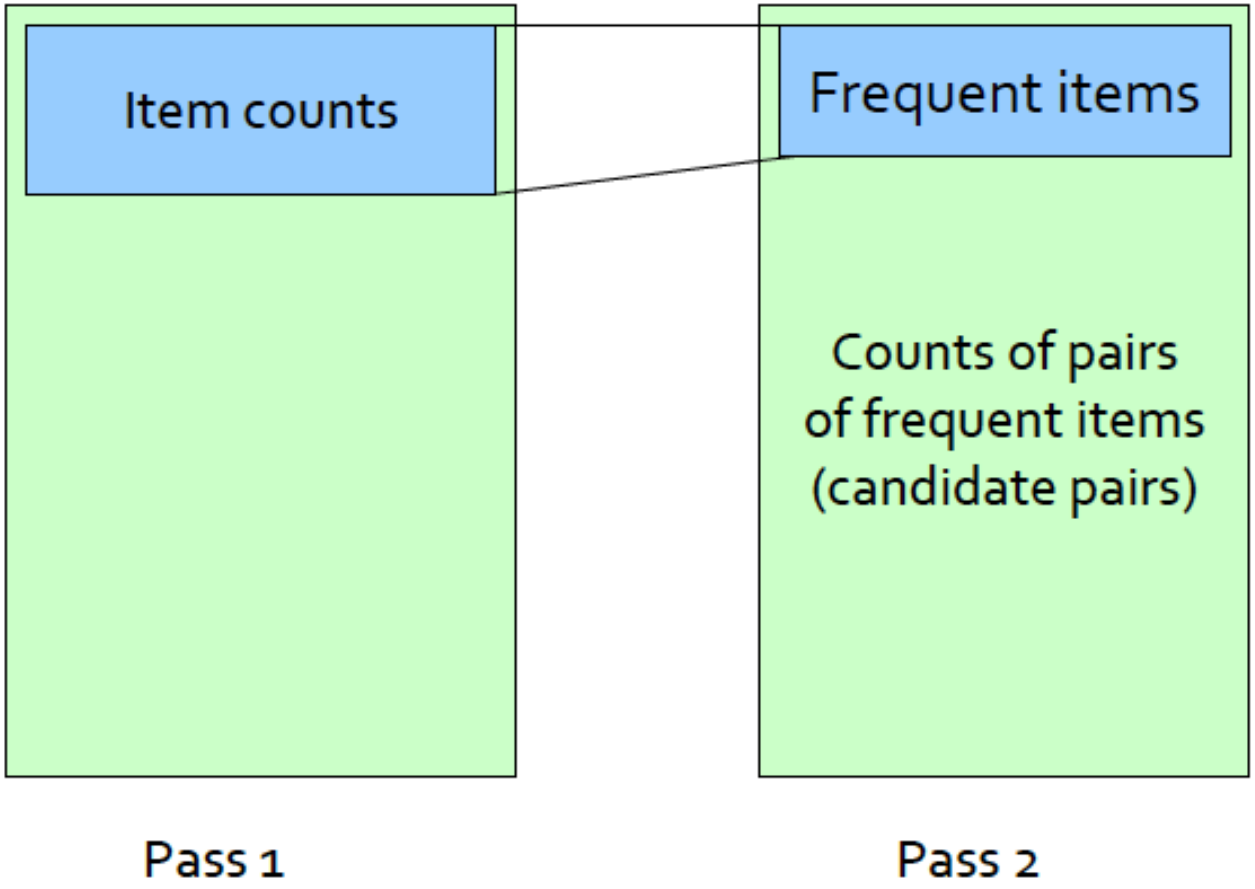
- Противоположное утверждение для пар:

Если элемент ***i*** не появлялся в ***s*** корзинах, тогда ни одна пара, содержащая ***i*** не может появится из этих ***s*** корзин

A-Priori Алгоритм (2)

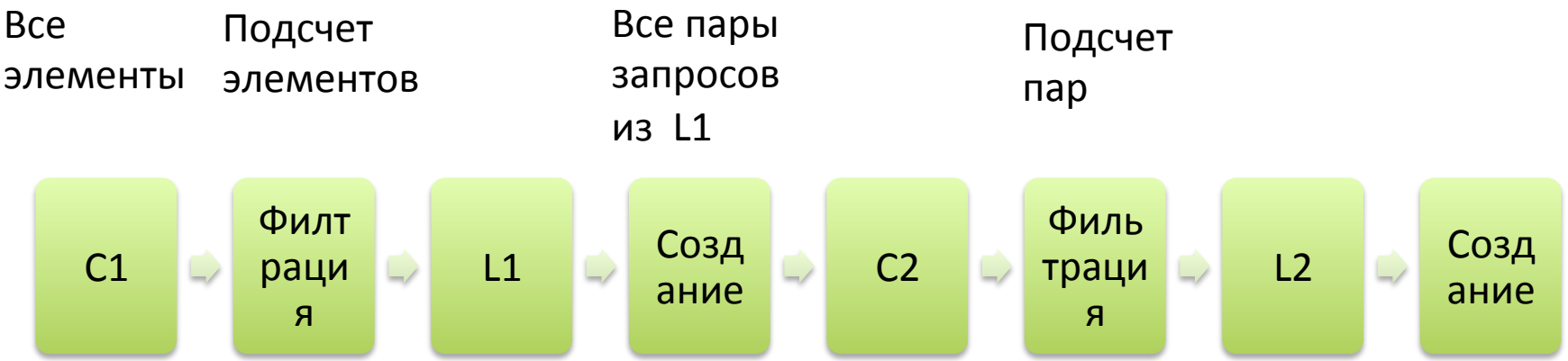
- Проход 1: Читаем содержимое корзин и считаем частоту каждого отдельного элемента
  - Необходимое количество памяти пропорционально # элементов
- Элемент, который встретился хотя бы s раз называется частотным
- Проход 2: Читаем содержимое корзины еще раз и складываем в память, только те пары, элементы которых являются частотными (из прохода 1)
  - Необходима память, пропорциональная квадрату частотных элементов
  - Плюс список частотных элментов

Основная память: картина для A-priori



Частотные элементы

- Для каждого k, мы строим два набора k-кортежей
  - Ск = кандидаты среди k-кортежей = те которые могут быть частотными (порог > s), базируясь на информации о k-1 кортежах
  - Lk – набор частотных k-кортежей



Частотные элементы (2)

- C1 = все элементы
- L1 = частотные элементы
- C2 = пары, в которых оба элемента частотные (из L1)
- L2 = частотные пары из L2 (порог >= s)

В общем случае:

- Ck = k-кортеж, каждый k-1-кортеж из Lk-1
- Lk = члены Ck с порогом >=s

PCY (Park-Chen-Yu) Алгоритм

- Наблюдения:

На первом этапе A-priori часть памяти не занято

- Мы сохраняем только отдельные частоты элементов
- Мы можем использовать незанятую память, чтобы уменьшить количество нужной памяти на шаге2?
- Шаг 2 для PCY: в дополнение к подсчёты частотности элементов, создаем хеш-таблицу с таким количеством бакетов, которое может поместиться в память.
  - Продолжаем подсчет для каждого бакета, в каждый из которых пары элементов были захешированы
  - Сохраняем только итоговую сумму, а не сами пары

PCY Алгоритм – первый проход

```
FOR (each basket) {
    FOR (each item in the basket)
      add 1 to item's count;
    FOR (each pair of items) {
      hash the pair to a bucket;
      add 1 to the count for that
        bucket
    }
}
```

Наблюдения о бакетах

- Если бакет содержит частотные пары, тогда бакет тоже частотный
- Хотя бакет, который не содержит частотных пар, может быть частотный
- Но для бакет с итогом меньшим s справедливо утверждение, что ни один элемент не частотный
  - Соответственно все пары, которые содержаться в не частотном бакете могут быть удалены из рассмотрения

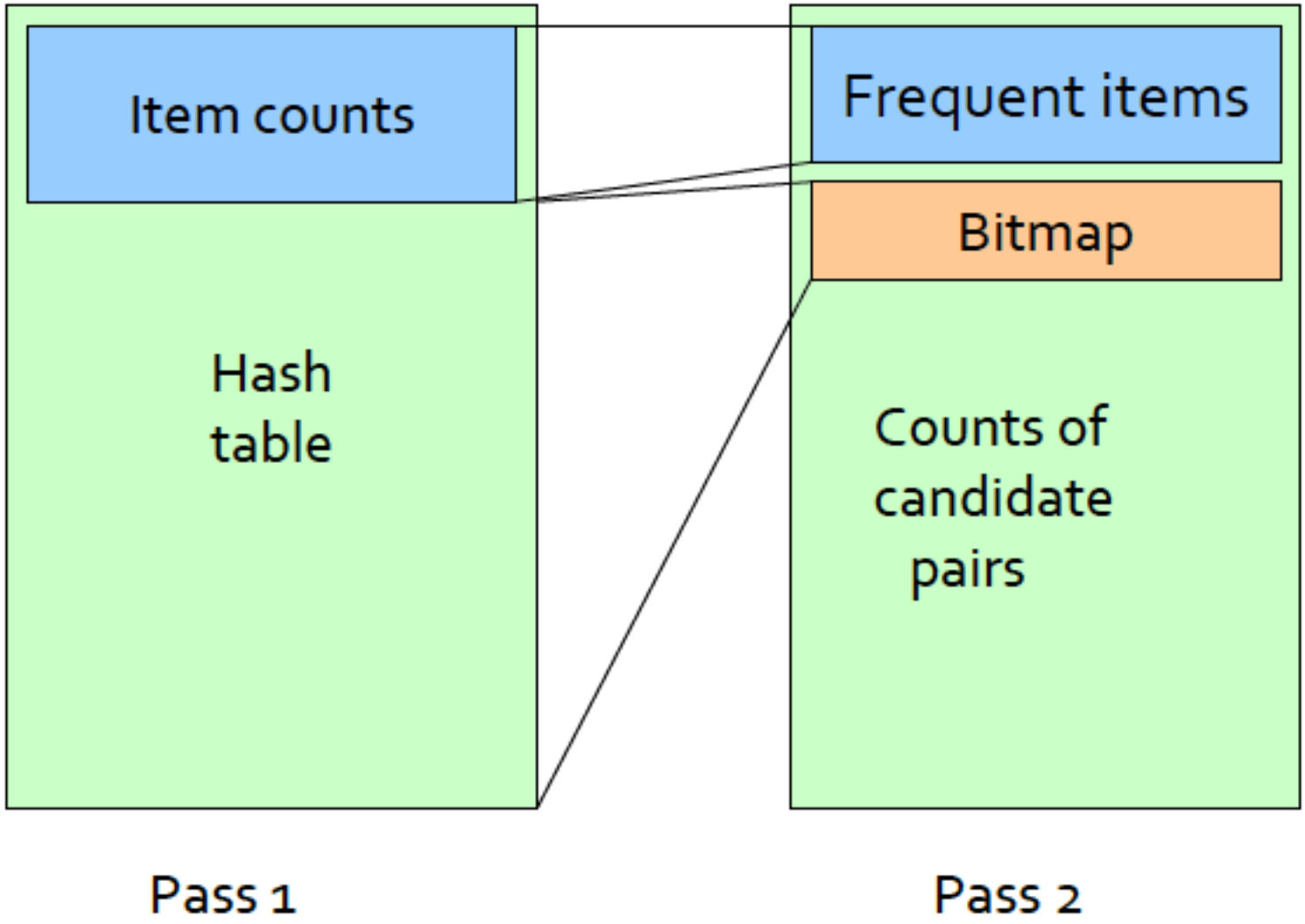
PYC Алгоритм – между шагами

- Заменяем бакет на битовый вектор:
  - 1 означает означает, что итоговая сумма больше порога s, 0 – обратное
- 4 байта необходимые для интеджер, будут заменены на 1 бит, битовый вектор требует на в 1/32 памяти

PYC Алгоритм: второй шаг

- Подсчитаем все пары ***{i, j}***, которые могут быть рассмотрены как пара кандидатов:
  1. Оба ***i*** и ***j*** – это частотные элементы
  2. Пара ***{i, j}***, которая захеширована в бакет, битовый вектор которого = 1(частотный бакет)
- Оба свойства необходимы быть выполнены, чтобы пара рассматривалась как частотная
- Замечание: данный подход тоже может быть улушен (MultiStage алгоритм + Multihash)

Map Reduce: Enviroment



Случайный отбор

- Возьмем случайную выборку корзин

- Запустим a-priori (или одно из его улучшений) в оперативной памяти
  - Мы не «платим» за дисковые операции I/O каждый раз когда увеличиваем размер набора
  - Следует уменьшить порог, пропорционально отобранному случайным образом наборы

Еще алгоритмы

- SON (Savasere, Omiecinski, and Navathe)
- Toivonen

Резюме

- Познакомились:
  - с ассоциативными правилами
  - с A-priori алгоритмом
  - С улучшением A-priori - PCY