Анализ потоков данных (Часть 2)

Mining Data Streams (Part 2)

http://mmds.org

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman

Содержание лекции

- Новые поточные алгоритмы:
  - Фильтрация потока данных: фильтры Блума
    - Выбор элементов со свойством x из потока
  - Подсчет различных элементов: алгоритм Флажоле–Мартена
    - Количество различных элементов среди последних k элементов потока
  - Оценка моментов: метод AMS
    - Оценка стандартного отклонения последних k элементов

– Подсчет частых элементов

(1) Фильтрация потоков данных

Фильтрация потоков данных

- Каждый элемент потока данных – это кортеж (tuple)
- Для данного набора ключей S
- определить, какие кортежи из потока находятся в S
- Очевидное решение: хеш-таблица
  - Но предположим, что памяти для хранения всего S в хеш-таблице недостаточно

* Например, мы можем применять миллионы фильтров на одном и том же потоке

Приложения

- Пример: фильтрация спама в электронной почте
  - Мы знаем 1 миллиард «хороших» электронных адресов
  - Если сообщение пришло с одного из них, то это НЕ спам
- Системы publish-subscribe
  - Вы собираете много сообщений (новостных статей)

– Люди выражают заинтересованность в определенных наборах ключевых слов

– Определить, соответствует ли каждое сообщение интересам пользователя

Решение–«пробный дубль» (1)

- По заданному наборку ключей S, которые требуется отфильтровать
- создать побитовый массив B из n бит, изначально обнулив все элементы
- Выбрать хеш-функцию h с областью значений $[0, n)$
- Хешировать каждый элемент $s \in S$ в одну из n урн и положить этот бит равным 1, т.е. $B[h(s)] = 1$

- Хешировать каждый элемент a потока и выводить только те из них, которые хешируются в бит, равный 1

  - Выводить a, если $B[h(a)] == 1$

Решение–«пробный дубль» (2)

Вывести элемент, так как он может быть в S. Элемент хешируется в урну, в которую хешировался по меньшей мере один элемент из S.

Отбросить элемент. Он хешируется в урну с нулевым значением, поэтому он точно не содержится в S.

- Выдает ложноположительные результаты, но не ложноотрицательные
  - Если элемент находится в S, то он выводится обязательно, если нет, то он все равно может быть выведен

Решение–«пробный дубль» (3)

- $|S| = 1$ миллиард адресов электронной почты
  $|B| = 1\,GB = 8$ миллиардов бит

- Если электронный адрес находится в $|S|$, он обязательно хешируется в урну с битом 1, поэтому он всегда подается на вывод (нет ложноотрицательных)

- Около 1/8 битов положены равными 1, поэтому около 1/8 адресов, не принадлежащих S, подаются на вывод (ложноположительные)

– В действительности, менее 1/8, так как в один и тот же бит может хешироваться более одного адреса

Анализ: метание дротиков (1)

- Более точный анализ числа ложноположительных результатов

- Рассмотрим: Если метать m дротиков в n равновероятных мишений,какова вероятность, что в мишень попадет хотя бы один дротик?

- В нашем случае:
  - Мишени = биты/урны
  - Дротики = хеш-значения элементов

Анализ: метание дротиков (2)

- Дано m дротиков, n мишеней

- Какова вероятность того, что в мишень попадет хотя бы один дротик?

$$1 - (1 - 1/n)^m \to 1 - e^{-m/n}$$

Равно $1/e$ при $n \to \infty$

Вероятность непопадания дротика в некоторую мишень X

Вероятность попадания по меньшей мере одного дротика в мишень X

Анализ: метание дротиков (3)

- Доля единиц в массиве B = вероятности ложноположительного результата = $1 - e^{-m/n}$

- Пример: 10^9 дротиков, $8 \cdot 10^9$ мишеней
  - Доля единиц в $B = 1 - e^{-1/8} = 0.1175$
    - Сравните с оценкой выше: $1/8 = 0.125$

Фильтр Блума

- Рассмотрим: $|S| = m$, $|B| = n$
- Используем k независимых хеш-функций $h_1, \ldots, h_k$.
- Инициализация:
  - Положить все B равными 0
  - Хешировать каждый элемент $s \in S$ с помощью хеш-функции h_i, положить $B[h_i(s)] = 1$ (для каждого $i = 1, \ldots, k$)

- Во время выполнения
  - При поступлении элемента потока с ключом x
    - Если $B[h_i(x)] = 1$ *для всех* $i = 1, \ldots, k$, то считать, что x содержится в S
      - То есть, x хешируется в урну со значением 1 для каждой хеш-функции $h_i(x)$
    - В противном случае, отбросить элемент x

Фильтр Блума – Анализ

- Какую долю побитового вектора B составляют единицы?
  - Метание $k \cdot m$ дротиков в n мишеней
  - Поэтому доля единиц $(1 - e^{-km/n})$
- Но у нас k независимых хеш-функций, и элемент x пропускается, только если *все* k хешируют элемент x в урну со значением 1
- Следовательно, вероятность ложноположительных $= (1 - e^{-km/n})^k$.

Фильтр Блума – Анализ (2)

- $m = 1$ миллиард, $n = 8$ миллиардов
  - $k = 1$: $(1 - e^{-1/8}) = 0.1175$
  - $k = 2$: $(1 - e^{-1/4})^2 = 0.0493$
- Что происходит при увеличении k?
  - «Оптимальное» значение k: $n/m \ln(2)$
    - В нашем случае: Оптимальное $k = 8 \ln(2) = 5.54 \approx 6$

- Погрешность при $k = 6$: $(1 - e^{-1/6})^2 = 0.0235$

Фильтр Блума – Резюме

- Фильтры Блума гарантируют отсутствие ложно-отрицательных и требуют ограниченный объем памяти
  - Подходит для предварительной обработки до более дорогостоящих проверок
- Подходит для реализации на аппаратном обеспечении
  - Вычисление хеш-функций может быть распараллелено

- Что лучше: 1 большой B или k малых B?
  - Одинаково: $(1-e^{-km/n})^k$ против $(1-e^{-m/(n/k)})^k$
  - Но хранение одного большого B проще

Подсчет различных элементов

Подсчет различных элементов

- Задача:
  - Поток данных состоит из универсума элементов, выбранных из множества размера N
  - Сохранять счетчик количества различных элементов, встречавшихся до этого момента
- Очевидный подход: Хранить множество встречавшихся элементов
  - То есть, хранить хеш-таблицу всех различных встречавшихся элементов

Приложения

- Сколько разных слов найдено в веб-страницах на сайте?
  - Необычайно малое или большое количество слов может означать искусственность страниц (спам?)
- Сколько различных веб-страниц клиент запрашивает в течение недели?
- Сколько различных продуктов продано в последнюю неделю?

Использование малого хранилища

- Настоящая задача: Что, если у нас нет места для хранения набора элементов, которые нам встречались?

- Оценить количество несмещенным образом

- Принять то, что в подсчете может быть небольшая погрешность, но поставить ограничение на вероятность того, что ошибка слишком большая

Подход Флажоле–Мартена

- Выбрать хеш-функцию h, отображающую каждый из N элементов в не менее чем $\log_2 N$ бит

- Для каждого элемента потока a пусть $r(a)$ – количество хвостовых нулей в $h(a)$

  - $r(a) =$ позиция первой единицы, считая справа

    * Например, если $h(a) = 12$, то 12 – это 1100 в двоичной системе, поэтмоу $r(a) = 2$.

- Записать $R =$ максимальный встреченный $r(a)$

- $R = \max_a r(a)$ по всем пройденным элементам a

- Оценка количества различных элементов $= 2^R$

Почему это работает: интуитивно

- Очень-очень приблизительная прикидка, почему алгоритм Флажоле–Мартена работает:
  - $h(a)$ хэширует a с равной вероятностью в любую из N значений
  - Поэтому $h(a)$ – последовательность $\log_2 N$ бит, в которой 2^{-r} доля всех a имеют хвост длиной r нулей
    - Около 50% a хешируются в $***0$
    - Около 25% a хешируются в $**00$

  - Таким образом, если самый длинный встретившийся хвост равен $r = 2$ (т.е., окончания хеша элемента $*100$), то, вероятно, мы встретили около 4 различных элементов

- Следовательно, требуется хешировать 2^r элементов до того, как нам встретится таковой с нулевым суффиксом длины r

Почему это работает: более формально

- Теперь покажем, почему алгоритм Флажоле–Мартена работает

- С формальной точки зрения покажем, что вероятность нахождения хвоста из r нулей

  – Стремится к 1 при $m >> 2^r$

  – Стремится к 0 при $m << 2^r$

  где m – количество различных элементов, встретившихся в потоке

- Таким образом, 2^R почти всегда будет около m!

Почему это работает: более формально

- Какова вероятность того, что данная $h(a)$ заканчивается на не менее чем r нулей? – 2^{-r}
  - $h(a)$ хеширует элементы равномерно случайным огбразом
  - Вероятность того, что случайное число оканчивается на не менее чем r нулей – 2^{-r}
- Тогда вероятность НЕ встретить хвост длины r среди m элементов:

$$(1 - 2^{-r})^m$$

Почему это работает: более формально

- Заметим: $(1-2^{-r})^m = (1-2^{-r})^{2^r(m2^{-r})} \approx e^{-m2^{-r}}$

- Вероятность НЕ встретить хвост длины r:
  - Если $m << 2^r$, то вероятсность стремится к 1
    - $(1-2^{-r})^m \approx e^{-m2^{-r}} = 1$ при $m/2^r \to 0$
    - Следовательно, вероятность встретить хвост длины r стремится к 0
  - Если $m >> 2^r$, то вероятность стремится к 0
    - $(1-2^{-r})^m \approx e^{-m2^{-r}} = 0$ при $m/2^r \to \infty$

    - Следовательно, вероятность встретить хвост длины r стремится к 1

- Таким образом, 2^R будет почти всегда около m!

Почему это не работает

- $E[2^R]$ в действительности бесконечно
  - Вероятность уменьшается в два раза, когда $R \to R + 1$, но значение удваивается

- Обходные пути задействуют использование множества хеш-функций h_i и получения множества выборок R_i

- Как выборки R_i комбинируются?
  - Среднее? Что, если одно очень большое значение 2^{R_i}?

- Медиана? Все оценки – степени числа 2
- Решение
  * Разбить ваши выборки на малые группы
  * Взять медиану групп
  * Затем взять среднее медиан

(3) Вычисление моментов

Обобщение: моменты

- Пусть в потоке есть элементы, выбираемые из множества A из N значений

- Пусть m_i – число раз, когда значение i встречается в потоке

- k-й момент – это

$$\sum_{i \in A} (m_i)^k$$

Частные случаи

$$\sum_{i \in A} (m_i)^k$$

- 0-й момент = число различных элементов
  - Только что рассмотренная задача
- 1-й момент = count of the numbers of elements = длина потока
  - Легко вычислить
- 2-й момент = surprise number S = мера того, насколько распределение неравномерно

Пример: Surprise Number

- Поток длины 100
- 11 различных значений
- Item counts: 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9
  Surprise $S = 910$
- Item counts: 90, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
  Surprise $S = 8110$

Метод AMS

- Метод AMS годится для всех моментов
- Дает несмещенную оценку
- Мы сосредоточимся на втором моменте S
- Мы выбираем и следим за многими величинами X:
  - Для каждой величины X сохраняем $X.el$ и $X.val$
    - $X.el$ соответствует элементу i

  * $X.val$ соответсвует счетчику элемента i

  – Заметим, что это требует, чтобы счетчик был в оперативной памяти, поэтому количество X'ов ограничено

- Наша цель – вычислить $S = \sum_i m_i^2$

Одна случайная величина (X)

- Чему положить $X.val$ и $X.el$?
  - Пусть поток имеет длину n (ослабим это условие позже)
  - Выбрать случайный момент времени t ($t < n$) для начала, так чтобы любой момент времени был одинаково вероятен
  - Пусть в момент времени t поток имеет элемент i. Положить $X.el = i$.
  - Затем мы храним счетчик c ($X.val = c$) числа значений i в потоке начиная с выбранного момента времени t

- Тогда оценка второго момента ($\sum_i m_i^2$) равна:

$$S = f(X) = n(2 \cdot c - 1)$$

  - Отметим, что мы храним множество X'ов ($X_1, X_2, \ldots, X_k$) и наша окончательная оценка будет $S = 1/k \sum_j^k f(X_j)$

Анализ мат. ожидания

- Второй момент $S = \sum_i m_i^2$
- $c_t, \ldots$ – сколько раз элемент в момент времени t появляется начиная с момента времени t ($c_1 = m_a$, $c_2 = m_a - 1$, $c_3 = m_b$)
- $E[f(X)] = \frac{1}{n} \sum_{t=1}^{n} n(2c_t - 1) = \frac{1}{n} \sum_i n(1 + 3 + 5 + \ldots + 2m_i - 1)$

Анализ мат. ожидания

- $E[f(X)] = \frac{1}{n}\sum_i n(1+3+5+\ldots+2m_i-1)$
  - Небольшое стороннее вычисление: $(1+3+5+\ldots+2m_i-1) =$

$$\sum_{i=1}^{m_i}(2i-1) = 2\frac{m_i(m_i+1)}{2} - m_i = (m_i)^2$$

- Тогда $E[f(X)] = \frac{1}{n}\sum_i n(m_i)^2$
- Следовательно, $E[f(X)] = \sum_i (m_i)^2 = S$

- Получаем второй момент (в смысле мат. ожидания)!

Моменты высших порядков

- Для оценки k-го момента используем по существу тот же алгоритм, но заменяем оценку:
  - Для $k = 2$ было $n(2 \cdot c - 1)$
  - Для $k = 3$ используем: $n(3 \cdot c^2 - 3c + 1)$ (где $c = X.val$)
- Почему?
  - Для $k = 2$: Вспомним, что у нас было $(1 + 3 + 5 + \ldots + 2m_i - 1)$ и мы показали, что члены $c2 - 1$ (для $c = 1, \ldots, m$) в сумме дают m^2

* $\sum_{c=1}^{m} 2c - 1 = \sum_{c=1}^{m} c^2 - \sum_{c=1}^{m} (c-1)^2 = m^2$
* Следовательно: $2c - 1 = c^2 - (c-1)^2$

– Для $k = 3$: $c^3 - (c-1)^3 = 3c^2 - 3c + 1$

- В общем случае: Оценка $= n(c^k - (c-1)^k)$

Комбинация выборок

- На практике:
  - Вычислить $f(X) = n(2c - 1)$ для стольких переменных X, сколько поместится в памяти
  - Усреднить их по группам
  - Взять медиану средних
- Задача: Потоки бесконечны
  - Предполагалось, что существует число n, число позиций в потоке

– Но реальные потоки продолжаются бесконечно, так что n – это переменная – число полученных входных данных

Потоки бесконечны: корректировки

- (1) Величины X имеют n в качестве множителя – хранить n отдельно; просто храним счетчик в X

- (2) Пусть можно хранить только k счетчиков. Мы должны выбрасывать некоторые X с течением времени:

  – Цель: Каждое начальное время t выбирается с вероятностью k/n

  – Решение: (выборка фиксированного объема!)

* Выбрать первые k моментов времени для k величин

* При поступлении n-го элемента ($n > k$), включить его в выборку с вероятностью k/n

* Если элемент включен в выборку, выбросить одну из хранящихся величин X, с равной вероятностью

Подсчет наборов элементов

Подсчет наборов элементов

- Новая задача: Дан поток; какие элементы появляются чаще, чем s раз в окне?

- Возможное решение: Представлять себе поток урн как один бинарный поток на элемент
  - 1 = элемент есть; 0 = элемента нет
  - Использовать DGIM, чтобы оценить количество единиц во всех элементах

Обобщения

- В принципе, можно подсчитать частые пары или более большие наборы одинаковым образом
  - Один поток на набор элементов
- Недостатки:
  - Только приблизительно
  - Количество наборов элементов слишком велико

Экспоненциально затухающие окна

- Экспоненциально затухающие окна: Эвристика для выбора вероятных частых (наборов) элементов
  - Какие наиболее популярные фильмы «в текущий момент»
    * Вместо вычисления количества среди последних N элементов «в лоб»
    * Вычислить гладкое объединение по всему потоку

- Если поток – это $a_1, a_2, \ldots$, и мы берем сумму потока, взять за ответ в момент времени t:

$$\sum_{i=1}^{t} a_i (1-c)^{t-i}$$

  – c – константа, предположительно очень малая, порядка 10^{-6} или 10^{-9}

- При поступлении нового a_{t+1}:
  Умножить текущую сумму на $(1-c)$ и добавить a_{t+1}

Пример: подсчет элементов

- Если каждый a_i – «элемент», можно вычислить характеристическую функцию каждого возможного элемента x в виде экспоненциально затухающего окна
  - То есть: $\sum_{i=1}^{t} \delta_i \cdot (1-c)^{t-i}$
    где $\delta_i = 1$, если $a_i = x$, и 0 в противном случае
  - Предположим, что для каждого элемента x у нас есть бинарный поток (1, если x встречается, 0, если x не встречается)
  - Поступает новый элемент x:

    - Умножить все счетчики на $(1 - c)$
    - Добавить $+1$ к счетчику для элемента x

- Назовем эту сумму «весом» элемента x

Скользящие и затухающие окна

- Важное свойство: Сумма по всем весам $\sum_t (1-c)^t$ равна $1/[1-(1-c)] = 1/c$

Пример: подсчет элементов

- Какие фильмы наиболее популярны «в данный момент»?

- Пусть хотим найти фильмы с весом $> \frac{1}{2}$
  - Важное свойство: Сумма по всем весам $\sum_t (1-c)^t$ равна $1/[1-(1-c)] = 1/c$

- Таким образом:
  - Не может существовать более $2/c$ фильмов с весом $\frac{1}{2}$ или больше

- Следовательно, $2/c$ – граница числа фильмов, подсчитываемых в любой момент времени

Обобщение на набор элементов

- Посчитать (некоторые) наборы элементов в E.D.W.
  - Какие наборы элементы наиболее популярны в настоящий момент?
    * Проблема: Слишком много наборов элементов для того, чтобы хранить счетчики их всех в памяти
- Когда поступает урна B:
  - Умножить все счетчики на $(1 - c)$

– Для неподсчитанных элементов B создать новый счетчик

– Добавить 1 к счетчику любого элемента B и любого набора элементов, содержащихся в B, которые уже подсчитываются

– Удалить счетчики $< \frac{1}{2}$

– Инициализировать новые счетчики (следующий слайд)

Инициализация новых счетчиков

- Создать счетчик для набора множеств $S \subseteq B$, если каждое собственное подмножество S имело счетчик до поступления в урну B
  - Интуитивно: Если все подмножества S подсчитываются, это значит, что он «частые/популярные», и поэтому у S есть потенциал быть «популярным»
- Пример:
  - Начинаем подсчитывать $S = \{i, j\}$ ттогда i и j подсчитывались до того, как мы встретили B

– Начинаем подсчитывать $S = \{i, j, k\}$ ттогда $\{i, j\}$, $\{i, k\}$ и $\{j, k\}$ все подсчитывались того, как мы встретили B

Сколько счетчиков нужно?

- Счетчики для единственных элементов $< (2/c)$·среднее количество элементов в урне
- Счетчики для более больших наборов элементов = ??
- Но мы остаемся консервативными о начальных счетчиках больших наборов
  - Если бы мы подсчитывали каждый набор, который нам встретился, одна урна из 20 элементов инициализировала бы 1 M счетчиков