Анализ потоков данных (Часть 1)

Mining Data Streams (Part 1)

http://mmds.org

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman

Потоки данных

- Во многих ситуациях анализа данных весь набор данных заранее неизвестен

- Управление потоками становится важным, когда the input rate контролируется извне:
  - Запросы Google
  - Обновление статусов в Twitter или Facebook

- Можно считать, что поток данных бесконечен и нестационарен (распределение изменяется с течением времени)

Модель потока данных

- Входные элементы поступают с высокой интенсивностью по одному или более входному порту (т.е., потоки)
  - Будем называть элементы потока кортежами (tuples)
- Система не может хранить целый поток так, чтобы к нему был доступ
- Q: Как выполнять вычисления о потоке используя только ограниченные объем (вторичной) памяти?

SGD – поточный алгоритм

- Стохастический градиентный спуск (SGD) – пример поточного алгоритма

- В машинном обучении: Online Learning
  - Позволяет моделировать задачи с непрерывным потоком данных
  - Нужен алгоритм для обучения из потока и медленной адаптации к изменениям в данных

- Идея: Медленные обновления модели

- SGD (SVM, Perceptron) производят малые обновления
- Сначала: Обучить классификатор на обучающих данных.
- Затем: По каждому элементу потока немного обновляем модель (используя малую интенсивность обучения)

Задачи для потоков

- Типы запросов по потоку:
  - Выборка данных из потока
    - Построение случайной выборки
  - Запросы по скользящим окнам
    - Количество элементов типа x среди последних k элементов потока

Задачи для потоков

- Типы запросов по потоку:
  - Фильтрация потока данных
    - Выбор элементов со свойством x из потока
  - Подсчет различных элементов
    - Количество различных элементов среди последних k элементов потока
  - Оценка моментов
    - Оценка среднего/стандартного отклонения последних k элементов потока

– Нахождение частых элементов

Приложения (1)

- Анализ потоков запросов
  - Google хочет знать, какие запросы более частые сегодня по сравнению со вчера
- Анализ потоков кликов
  - Yahoo хочет знать, какие из ее страниц получают необучный набор хитов за последний час
- Анализ новостных лент социальных сетей

— Например, поиск trending-тем в Twitter, Facebook

Приложения (2)

- Сенсорные сети
  - Множество сенсоров (датчиков), посылающих данные центральному контроллеру
- Записи телефонных звонков
  - Data feeds into customer bills as well as settlements between telephone companies
- IP-пакеты в маршрутизаторе

– Сбор информации для оптимальной маршрутизации

– Определение DDoS-атак

Выборка из потока данных:

Выборка фиксированной пропорции

Выборка растет вместе с ростом потока

- Будем сохранять выборку

- Две различные задачи:
  - (1) Произвести выборку фиксированной пропорции элементов в потоке (например, 1 из 10)
  - (2) Хранить случайную выборку фиксированного объема из потенциально бесконечного потока
    - В любой «момент времени» k требуется иметь случайную выборку из s элементов
    - Какое свойство выборки следует сохранять?

- Для всех шагов времени k каждый из k встреченных элементов имеет одну и ту же вероятность попадания в выборку

Выборка фиксированной пропорции

- Задача 1: Выборка фиксированной пропорции

- Сценарий: Поток запросов поисковой машины
  - Поток кортежей: (пользователь, запрос, время)
  - Ответить на вопрос типа: Как часто пользователь отправлял один и тот же запрос в течение одного дня
  - Имеется место для хранения 1/10-й потока запросов

- Наивное решение:
  - Сгенерировать случайное целое число в диапазоне [0..9] для каждого запроса
  - Сохранить запрос, если это число равно 0, в противном случае – не сохранять

Что не так с наивным подходом

- Простой вопрос: Какую долю запросов среднего пользователя поисковой машины составляют дублирующиеся запросы?
  - Пусть каждый пользователь отправляет x запросов единожды и d запросов дважды (всего $x + 2d$ запросов)
    - Правильный ответ: $d/(x + d)$
  - Предложенное решение: Сохраняем 10% запросов

  * В выборке будет содержаться $x/10$ недублирующихся запросов и $2d/10$ дублирующихся не менее одного раза

  * Но только $d/100$ пар дублирующихся

$$d/100 = 1/10 \cdot 1/10 \cdot d$$

  * Из d «дублирующихся» $18d/100$ встречаются ровно один раз

$$18d/100 = ((1/10 \cdot 9/10) + (9/10 \cdot 1/10)) \cdot d$$

- Таким образом, ответ, основанный на выборке, – это

$$\frac{\frac{d}{100}}{\frac{x}{100} + \frac{d}{100} + \frac{18d}{100}} = \frac{d}{10x + 19d}$$

Решение: Формировать выборку из пользователей

Решение:

Выберем 1/10-ю пользователей и включим все их запросы в выборку

Используем хеш-функцию, хеширующую идентификатор пользователя равномерно по 10 урнам

Обобщенное решение

- Поток кортежей с ключами:
  - Ключ – некоторое подмножество каждой компоненты кортежа; например, кортеж – (пользователь, поисковый запрос, время); ключ – пользователь
  - Выбор ключа зависит от решаемой задачи
- Для получения выборки, составляющей a/b долю потока:
  - Хешировать ключ каждого кортежа равномерно в b урн

– Включить кортеж в выборку, если его хэш-значение не превосходит a

Выборка из потока данных:

Формирование выборки фиксированного объема

Объем выборки не изменяется с ростом потока

Сохранение фиксированного объема выборки

- Задача 2: Выборка фиксированного объема

- Пусть требуется сохранять случайную выборку S размера в точности s
  - Т.е., ограничение на объем оперативной памяти

- Почему? Длина потока неизвестна наперед

- Пусть в момент времени n было просмотрено n элементов

– Каждый элемент содержится в выборке S с одинаковой вероятностью s/n

Решение: выборка фиксированного объема

- Алгоритм (Reservoir Sampling)
  - Все первые s элементов потока записать в S
  - Пусть пройдено $n-1$ элементов и теперь пришел n-й элемент ($n > s$)
    - С вероятностью s/n сохранить n-й элемент в выборку, иначе не сохранять
    - Если n-й элемент выбран для сохранения в выборку, то он заменяет один из s элементов в выборке S, равномерно выбранный случайным образом

- Утверждение: Этот алгоритм сохранятет выборку S с требуемым свойством:
  - После n элементов выборка содержит каждый пройденный элемент с вероятностью s/n

Доказательство: по индукции

- Докажем это утверждение по индукции:
  - Пусть после просмотра n элементов выборка содержит каждый пройденный элемент с вероятностью s/n
  - Требуется показать, что после прохождения элемента $n+1$ выборка сохранит это свойсто
    * Выборка содержит каждый пройденный элемент с вероятностью $s/(n+1)$

- Основание индукции:

- После того, как пройдено $n = s$, элементов выборка S обладает требуемым свойством
  - Каждый из $n = s$ элементов содержится в выборке с вероятностью $s/s = 1$

Доказательство: по индукции

Предположение индукции: После n элементов выборка S содержит каждый пройденный элемент с вероятностью s/n

Теперь поступает элемент $n+1$

Шаг индукции: Для элементов, которые уже включены в S, вероятность их сохранения в S:

$$\left(1-\frac{s}{n+1}\right)+\left(\frac{s}{n+1}\right)\left(\frac{s-1}{s}\right)=\frac{n}{n+1}$$

Таким образом, в момент времени n, кортежи в S были там с вероятностью s/n В момент времени $n \to n+1$ кортеж был в S с вероятностью $n/(n+1)$ Вероятность того, что кортеж будет включен в S в момент времени $n+1$

$$= \frac{s}{n} \cdot \frac{n}{n+1} = \frac{s}{n+1}$$

Запросы по

(длинному) скользящему окну

(sliding window)

Скользящие окна

- Полезная модель обрабоки потоков – вопросы касаются окна длины N – N элементов, поступивших последними.

- Интересный слуай: N настолько велико, что данные не могут храниться в памяти или даже на диске

  – Или у нас так много потоков, что окна для всех хранить невозможно

- Пример Amazon:

– Для каждого продукта X сохраняем поток из нулей и единиц, показывающий, был ли продукт продан в момент n-й транзакции

– Требуется отвечать на запрос, сколько раз продукт X был продан среди последних k продаж

Скользящие окна: 1 поток

Скользящие окна по одному потоку: $N = 6$

```
q w e r t y u i o p|a s d f g h|j k l z x c v b n m
q w e r t y u i o p a|s d f g h j|k l z x c v b n m
q w e r t y u i o p a s|d f g h j k|l z x c v b n m
q w e r t y u i o p a s d|f g h j k l|z x c v b n m

<-- Прошлое                                Будущее -->
```

Подсчет битов (1)

- Задача:
  - Дан поток нулей и единиц
  - Быть готовым ответить на запросы типа: Сколько единиц содержится в последних k битах? где $k \leq N$
- Очевидное решение:
  - Хранить последние N бит
  - При поступлении нового бита удалять $N+1$-й бит

Подсчет битов (2)

- Невозможно получить точный ответ, не храня целое окно
- Проблема: Что, если у нас нет возможности хранить N бит?
  - Например, обрабатывается 1 миллиард потоков и $N = 1$ миллиарду
- Но нас удовлетворит и приближенный ответ

Попытка: простое решение

- Q: Сколько единиц содержится в последних N битах?

- Простое решение, которое на самом деле не решение: гипотеза равномерности

- Организуем 2 счетчика
  - S: число единиц с начала потока
  - Z: число нулей с начала потока

- Сколько единиц содержится в последних N битах? $N \cdot \frac{S}{S+Z}$

- Но что, если поток неравномерный?
  - Что, если распределение изменяется с течением времени?

Метод DGIM

- Метод DGIM *не* предполагает равномерности
- Хранится $O(\log^2 N)$ бит на поток
- Метод дает приближенный ответ с погрешностью не более, чем 50%
  - Погрешность может быть уменьшена до любой доли > 0, если использовать более сложный алгоритм с пропорционально большим количеством бит

Идея: экспоненциальные окна

- Решение, которое не работает (как надо):
  - Подытоживать экспоненциально увеличивающиеся регионы потока, если смотреть назад
  - Удалять малые регионы, если они начинаются в той же точке, что и бóльший регион

Окно длины 16 содержит 6 единиц.

Можем восстановить счетчик последних N бит, за исключением того, что мы не знаем, какое количество последних 6 бит содержится в N

Что хорошо

- Хранится только $O(\log^2 N)$ бит
  - $O(\log N)$ счетчиков по $\log_2 N$ бит на каждый
- Простое обновление при поступлении новых битов
- Погрешность в подсчете не превосходит число единиц в «неизвестной» области

Что не так

- Если единицы распределены относительно равномерно, то погрешность из-за неизвестного региона остается малой – не более 50%

- Но может случиться, что все единицы находятся в неизвестной области в конце

- В таком случае погрешность неограничена!

Поправка: метод DGIM

- Идея: Вместо подытоживания блоков фиксированной длины подытоживать блоки с определенным числом единиц:
  - Пусть размеры блоков (число единиц) увеличиваются экспоненциально
- Когда количество единиц в окне невелико, размер блоков остается малым, поэтому погрешности малы

DGIM: Временные отметки (Timestamps)

- Каждый бит в потоке имеет временную отметку, начиная с $1, 2, \ldots$

- Записываем временные отметки по модулю N (размер окна), поэтому можем представить любую нужную временную отметку с помощью $O(\log_2 N)$ бит

DGIM: Урны

- Урна в методе DGIM – это запись, состоящая из:
  - (A) Временной отметки ее конца [$O(\log N)$ бит]
  - (B) Числа единиц между ее началом и концом [$O(\log \log N)$ бит]
- Ограничение на урны: Число единиц должно быть степенью числа 2
  - Это объясняет $O(\log \log N)$ в (B) выше

Представление потока в виде урн

- Либо одна, либо две урны с одинаковым количеством единиц, являющимся степенью числа 2
- Урны не накладываются в смысле временных отметок
- Урны отсортированы по размеру
  - Более ранние урны не меньше, чем более поздние

- Урны исчезают, когда их конечное время $> N$ временных единиц в прошлое

Обновление урн (1)

- При поступлении нового бита удалить последнюю (самую давнюю) урну, если ее конечное время на N временных единиц раньше, чем настоящий момент

- 2 случая: текущий бит равен 0 или 1

- Если текущий бит равен 0:
  не производить никаких других изменений

Обновление урн (2)

- Если текущий бит равен 1:
  - (1) Создать новую урну размера 1, в которой содержится только этот бит;
    Конечная временная отметка = текущее время
  - (2) Если теперь имеется три урны размера 1, скомпоновать две самых давних урны в одну урну размера 2
  - (3) Если теперь имеется три урны размера 2, скомпоновать две самых давних урны в урну размера 4

– (4) И так далее...

Как формировать запрос?

- Для оценки числа единиц в последних N битах:

  1. Суммировать размеры всех урн, кроме последней

  2. Добавить половину размера последней урны

- Замечание: Нам неизвестно, сколько единиц из последней урны содержится в нашем окне

Граница погрешности: доказательство

- Почему погрешность составляет 50%? Давайте докажем!
- Пусть размер последней урны равен 2^r
- Тогда, положив 2^{r-1} (т.е., половина) ее единиц все еще содержится в окне, получаем, что погрешность не превосходит 2^{r-1}
- Так как имеется по меньшей мере по одной урне каждого размера, меньшего 2^r, истинная сумма составляет не менее

$$1 + 2 + 4 + \ldots + 2^{r-1} = 2^r - 1$$

- Таким образом, погрешность не превосходит 50%

Дальнейшее уменьшение погрешности

- Вместо хранения 1 или 2 урн каждого размера, допускаем $r-1$ или r урн ($r > 2$)
  - За исключением урны наибольшего размера, таковых у нас может быть любое количество между 1 и r
- Погрешность не превосходит $O(1/r)$
- Правильным подбором r можно балансировать между количеством хранимых бит и погрешностью

Обобщения

- Можно ли использовать тот же прием для ответа на запросы: Сколько единиц содержится в последних k, где $k < N$?
  - А: Найти самую раннюю урну B, которая пересекается с k. Число единиц равно сумме размеров более поздних урн $+\frac{1}{2}$ размера B
- Можно ли обработать случай, когда поток состоит не из битов, а из целых чисел и требуется получить сумму последних k элементов?

Обобщения

- Поток положительных целых чисел
- Требуется вычислить сумму последних k элементов
  - Amazon: средняя цена последних k продаж
- Решение
  - (1) Если известно, что во всех не более m бит
    - Рассматриваем m бит каждого целого числа как отдельный поток

  * С помощью DGIM подсчитываем количество единиц в каждом целом числе

  * Сумма равна $= \sum_{i=0}^{m-1} c_i 2^i$
    c_i – оценка количества для i-го бита

- (2) Храним частичные суммы в урнах

  * Сумма элементов в урне размера b не превосходит 2^b

    Идея: Сумма в каждой урне не превосходит 2^b (кроме случая, когда в урне только одно целое число)

Резюме

- Выборка фиксированной пропорции из потока
  - Объем выборки растет вместе с ростом потока
- Выборка фиксированного объема
  - Reservoir sampling
  - Подсчет количества единиц в последних N элементах
    * Экспоненциально уменьшающиеся окна

* Обобщения:
  - Число единиц в любых последних k ($k < N$) элементах
  - Суммы целых чисел в последних N элементах