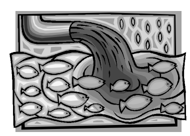
Data stream algoritmy

I.Kolingerová

Obsah:

1. Úvod
2. Typická úloha
3. Data stream modely
4. Příklady

1. Úvod



- Streaming algorithm – vstupem proud dat přicházející postupně po jedné položce
- Zaměřeny "na minulost" – spočítat nějakou funkci dat

  x
- Online algoritmy – jak naše rozhodnutí ovlivní budoucnost
- Vypadá snadné, ale: nemáme místo na všechna data, jen O(log n) nebo dokonce O(1) paměti

2

Úvod

- Data stream – data přicházejí rychle, takže
  - obtížné předat je programu všechna
  - obtížné spočítat složitější funkce na velkých částech vstupu
  - obtížné je dočasně nebo trvale ukládat
- Neformálně: streaming zahrnuje
  - malý počet průchodů daty (obvykle jeden)
  - sublineární paměť (sublineární ve stavovém prostoru nebo v počtu položek streamu?)
  - sublineární čas na výpočet (někdy)

3

Úvod

- Podobné dynamickým, online, aproximačním nebo randomizovaným alg., ale s více omezeními
- Nesměšovat s alg. pro vnější paměti – tam data v souborech, pomalé, ale přes všechny potíže se k nim lze dostat opakovaně, ukládat atd.

4

Úvod

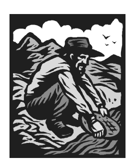
Jak zvládat taková data?

- Paralelizace (Často vysoce paralelizovatelné úlohy, kromě ukládání)
- Řízení dat. rychlosti vzorkováním (Příklad: experimenty s částicemi s vysokou energií v CERN – TB/s dat, redukováno hw v reálném čase na GB/s)
- Zaokrouhlení dat. struktur na bloky (Př.: hledání podvodů v telef. síti – užití grafu až do velikosti 1 dne a srovnáváni s předchozím dnem)
- Hierarchická detailní analýza
- Kladení si imaginativních otázek (může přinést nová řešení)

5

Úvod

- Obvykle jen aproximace, ale ne vždy
- Aplikace:
  - Sítě – např. routery – sleduje pakety, cca milióny za s, moc velké na uložení, ale chceme spočítat např. kam jdou, kde odmítány služby atd.
  - Databáze – sledování updatů a dotazů, chceme statistiku, např. které položky nejčastěji požadovány atd.



6

2. Typická úloha

Paul a Carole, Paul zadává stream permutovaných čísel z {1..n}, 1 vyřadil, Carole má uhodnout, které

Nápad?

7

Typická úloha

Paul a Carole, Paul zadává stream permutovaných čísel z {1..n}, 1 vyřadil, Carole má uhodnout, které

Řešení: snadné :-)

- C udržuje sumu čísel s a počet c, když je c n-1, zbylé číslo je n(n+1)/2 – s

8

Typická úloha

Co 2 chybějící čísla?

Nápad?

9

Typická úloha

Co 2 chybějící čísla?

Řešení: opět snadné, C kromě s uchovává ještě s´, sumu čtverců čísel ve streamu, nakonec spočítá

$i+j \;\; = n(n+1)/2 - s$

$i^2+j^2 = n(n+1)(2n+1)/6 - s'$

Jde zobecnit pro k chybějících prvků

10

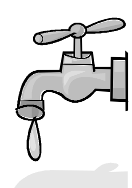
3. Data stream modely

- Vstupní data a_1, a_2,... přicházejí sekvenčně, prvek po prvku, a popisují signál A, 1D funkci A:[1..N]-> R.
- Modely se liší podle způsobu popisu A:
  - Time Series Model
  - Cash Register Model
  - Turnstile Model

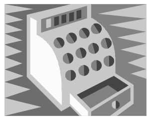
11

Data stream modely

- Time Series Model – a_i =A[i], objevují se v rostoucím pořadí i
- Vhodný model pro časové posloupnosti, kde např. sledujeme provoz na IP adrese každých 5 min, objem burzov. obchodů každou min. apod.



12



Data stream modely

- Cash Register Model – a_i jsou inkrementy k A[j], $a_i = (j,I_i)$, $I_i \geq 0$. Tj. $A_i[j]= A_{i-1}[j]+I_i$, kde A_i je stav signálu po shlédnutí i-té položky ve streamu. Více a_i může postupně inkrementovat jeden A[j].
- Patrně nejpopulárnější data stream model, pro aplikace typu monitorování IP adres, kt. přistupují k Web serveru – mohou přistupovat vícekrát

Data stream modely

- Turnstile Model – a_i jsou updaty A[j], $a_i = (j,U_i)$, Tj. $A_i[j]= A_{i-1}[j]+U_i$, kde U_i může být kladné i záporné.
- Nejobecnější data stream model, pro aplikace typu sledování cestujících v podzemce – turniket sleduje přicházející i odcházející
- Vhodné pro plně dynamické úlohy
- Těžké získat nějaké řešení v tomto modelu

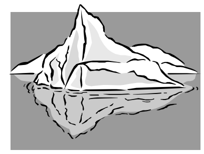
- Někdy $A_i[j] \geq 0$ – strict Turnstile model – tj. lidé odcházejí jen tím turniketem, kterým přišli
- Aplikace: např. databáze – lze zrušit jen záznam, který jsme předtím vložili
- Naopak non-strict model, $A_i[j] < 0$ pro nějaké i: např. signál nad rozdílem dvou cash registerových streamů

4. Příklady

Náhodné vzorkování s rezervoárem

- Počet záznamů N není znám předem - vzorkování nutno provádět dynamicky během čtení ze streamu
- Chceme: nezkreslený vzorek velikosti n
- Udržujeme rezervoár z data streamu velikosti n, naplníme prvními n body ze streamu
- Dále zařadíme do rezervoáru (t+1). bod ze streamu s p=n/(t+1) místo bodu vybraného náhodně z rezervoáru
- Celkově má na konci vzorkování každý bod streamu stejnou p zařazení (n /N)

Iceberg queries



- Chceme frekvence f (nebo jiné funkce) pro prvky nad určitým prahem zadaným uživatelem.
- Běžné řešení: na 2 průchody
  - V 1. průchodu udržujeme haš. tabulku čítačů, inkrementuje při příchodu prvku přísluš. čítač.
  - Pak komprese do bitmapy, kde 1 pro velké hodnoty čítače.
  - 2. průchod – udržovat přesné frekvence jen pro ty prvky, které se hashují do hodnoty odpovídající 1 v bitmapě.
- Modifikace pro stream obtížná – po 1. průchodu nemáme frekvence

17

Iceberg queries

- Vstup: práh s, chyba ε, pravd. selhání δ
- Záruky řešení: žádné chybné negativní odpovědi, žádné chybné pozitivní odpovědi pro f pod (s- ε)N , odhad frekvence f menší o max. εN
- Př.: s=0.1%, ε=0.01%,
- tj. na výstup všechny prvky s f > 0.1%, žádné s f < 0.09%, prvky mezi mohou a nemusí být výstupem, všechny f se liší od skut. max. o 0.01%

18

Data stream řešení 1

Sticky sampling

- Pravděpodobnostní alg.
- S <- Ø, sampling rate r <- 1
- Vzorkujeme prvky s pravděp. 1/r
- Pokud e už v S, inkrementujeme mu f, jinak přidáme do S (e,1) s pravděpodobností 1/r
- 1/r se během života streamu snižuje

(t= 1/ε log ($s^{-1}\delta^{-1}$), 2t prvků s r=1, pak 2t s r=2, pak 4t s r=4 atd.)

- Když se mění r, prohledají se položky S a následující úprava:

19

Sticky sampling

- Pro každý prvek házíme mincí, až je hod úspěšný, při neúspěšném dekrement f. Pokud f klesne na 0, prvek vyřadíme z S.
- Výstup: seznam položek s prahem s – všechny, kde f ≥ (s-ε)N

20

Data stream řešení 2

Lossy Counting

- Deterministický alg., stream koncepčně rozdělen na buckety šířky w=1/ε (zaokrouhl. nahoru)
- buckety číslovány od 1 ($b_{current}$)
- D <- Ø, ukládá se (e,f,Δ), Δ – max. možná chyba f (kolikrát se mohlo vyskytnout v předch. košících)
- nový prvek – pokud je v D, zvýšit jeho f; pokud není v D, vytvořit novou položku (e,1,$b_{current}$-1)
- pokud jsme na hranici bucketu, tj. N mod w = 0, zrušíme položky, pro které f+ Δ ≤ $b_{current}$
- Výstup: seznam položek s prahem s – všechny, kde f ≥ (s-ε)N

21

Př. pro cash register

Problém: dáno n čísel, určete majoritní položku, pokud existuje (tj. objevuje se alespoň N/2+1 x)

Alg.: Udržovat množinu K položek a počítat pro ně f. Pokud nová položka je v K, inkrementovat její f, pokud není, přidat ji s čítačem 1. Pokud K plné, dekrementovat čítače o 1 a eliminovat položky s čítači = 0.

22

Majoritní položka

Př.:K=10

Čísla: 1, 0, 5, 10, 13, 20, 21, 4, 2, 7, 1, 0, 5, 13, 20, 1, 0, 5, 5, 5, 41, ...

1	0	5	10	13	20	21	4	2	7

buffer

3	3	5	1	2	2	1	1	1	1

čítač f

Došla paměť v bufferu =>
dekrement f a eliminace položek s 0

23

Majoritní položka

1	0	5	10	13	20	41			

buffer

2	2	4	0	1	1	0	0	0	0

čítač f

atd.

- Možná přeceníme méně častou položku
- Bez dekrementu by k tomu nedošlo, ale kam s těmi „hausnumery“?

24

Majoritní položka

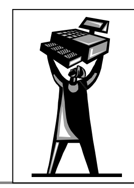
Určitě nevyhodíme častou položku ?

- Nechť se objeví 1x v K položkách, nechť ji vždy odstřelíme; přežije to majoritní položka?
- Celkem se objeví N/K x, takže pokud K aspoň 2, přežije.

Závěr: nedá chybné negativní odpovědi, může dát chybné pozitivní odpovědi, užitá paměť O(|K|), čas amortiz. O(1) na 1 položku.

25

Zlepšení



Alg. Sample and Count: Udržujeme vzorek K položek s čítači. Pokud je nová položka v K, inkrement čítače, jinak ji přidáme do K s pravděp. r/N

Výstup: všechny položky s čítačem nejméně (1/|K|-ε)N

Alg. Lossy Counting: Rozdělíme stream do oken velikosti 1/ε, na hranici oken dekrement čítačů o 1

Výstup: všechny položky s čítačem nejméně (1/|K|-ε)N

Analýza: LC nedává falešné negativní odpovědi, nejlepší chování z 3 uvedených

26

Př. Počet vzájemně různých položek

Problém: Vstup je stream $a_1, a_2,...,a_n$, a_i z {1,2,...,m}, odhadněte $F0=|\{a_1, a_2,...,a_n, \}|$

Aplikace: kolik transakcí odlišnou kartou za den? Kolik odlišných web. stránek za den?

Data z db transakcí, CD nebo cash registeru plateb

27

Počet vzájemně různých položek

Přesný alg.: Udržovat pole a[1..U], napřed všechny prvky rovny 0, a čítač C.

0	0	0	0	0	0	0	0	0	0

C $\boxed{0}$

Když přijde položka i, podívat se na a[i], pokud 0, inkrement C a a[i] <- 1. Vrátit C jako počet různých položek.

C $\boxed{4}$

a1, a2, a5, a5, a2, a4,...

1	1	0	1	1	0	0	0	0	0

28

Čas: O(1) na update a dotaz, paměť: O(U)

Př. Počet vzájemně různých položek

Přibližný alg.:

- Udržovat pole a[0..log m], napřed všechny prvky rovny 0, užít hash funkci f:{1..m} -> {0..log m}
- Spočítat f(i) pro každou položku ze streamu a nastavit a[f(i)] na 1
- Extrahovat z toho přibližný počet různých položek

29

Př. na sublinear time

Problém: Dána vzájemně různá čísla A[1..n], určete číslo v horní polovině hodnot.

Alg.: Vyberte k čísel rovnoměrně náhodně. Určete z nich MAX a vraťte jako řešení.

Pravděp., že řešení nesprávné: pravd., že všech k vybraných čísel leží v dolní polovině, tj. téměř $(1/2)^k$

Pro chybu δ vzít log (1/δ) vzorků.

Najít takto MIN a MAX je obtížné.

30

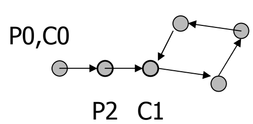
Př. na sublinear space

Problém: Je dán jednosměr. zřetězený seznam a pointer na jeho začátek, určete, zda je v seznamu smyčka, s užitím nejvýše O(1) přídavné paměti

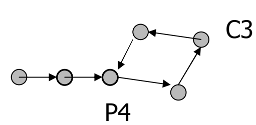
Alg.: sudá kola: Paul se pohne o 1 krok
lichá kola: Carol se pohne o 2 kroky

Smyčka

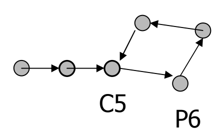
Po 2. kole:



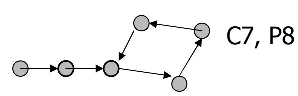
Po 4. kole:



Po 6. kole:

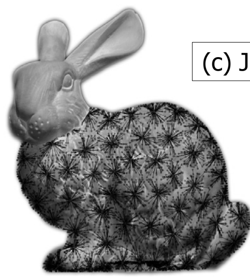


Po 8. kole:



Př. Konstrukce shluků (clusterů) bodů v E^2/E^3 pro data streamy

Problém: Je dán stream bodů, sestavte z nich víceúrovňovou shlukovou reprezentaci tak, aby se nejvyšší úroveň vešla do vnitřní paměti



(c) J.Skála

33

Konstrukce shluků bodů

Motivace:

- Geometrické modely začínají být příliš velké (stovky mil. vrcholů)
- Používány externí paměti – náhodný přístup velmi drahý
- Clusterování děláno hierarchicky → modely s více úrovněmi detailů

34

Řešení bez data streamu

Konstrukce shluků bodů

- Na začátku náhodný výběr několika středů shluků s pravděp.= vzdálenosti od nejbl.shluku/cena otevření shluku
- Pro každý bod zvažováno připojení k existujícímu shluku/otevření nového clusteru podle ceny
- Vylepšení počátečního řešení:
  - Vyplatí se udělat nový shluk v náhodně vybraném bodě?
  - Které body se k němu přesunou?
  - Vyplatí se zavřít nějaký shluk, kde zbylo po předchozím kroku málo bodů?
- Počát. řešení v $O(N^2)$, cena změny v $O(N)$, vyhodnoceno $O(N \log N)$ krát, celková složitost cca $O(N^2 \log N)$

35

Řešení pro data stream

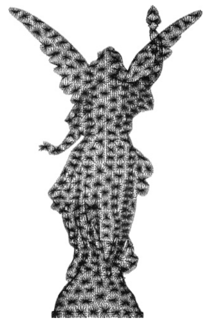
Konstrukce shluků bodů

- Zpracování po blocích podle velikosti OP, není nutné seřazení bodů
- Pak další blok dat
- Když plná paměť, jde se o úroveň výše
- 1 průchod daty (ale vícekrát těmi v OP)

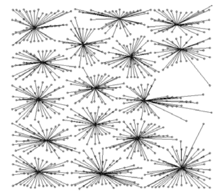
36

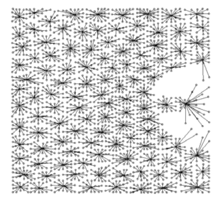
Konstrukce shluků bodů

Ukázky



(c) J.Skála





Výzva na semestrálku:

implementovat jinou metodu shlukování a srovnat výsledky 37