Università degli Studi di Udine

Dipartimento di Scienze Matematica, Informatiche e Fisiche

Tesi Magistrale in Informatica

Uno studio sugli algoritmi di sketching per la stima della cardinalità

Relatore
Prof. Gabriele Puppis
Università degli Studi di Udine

Laureando Magistrale
Daniele Ferroli

Matricola
137357

Anno Accademico
2024-2025

"Da scrivere"
— Rene Descartes

Contents

1 Introduzione

Random citation [**?**].

2 Background

In questa tesi, il modello che rappresenta i dati in input, a differenza di algoritmi più tradizionali, è chiamato *modello di stream di dati* (data stream model).

2.1 Modello di stream di dati

Nel modello di stream i dati [1] arrivano in modo continuo e potenzialmente infinita. Rispetto l'utilizzo di un database tradizionale, non è possibile accumulare tutto in memoria o su disco e interrogare i dati. Gli elementi devono essere processati al volo oppure vengono persi.

Inoltre, la velocità con cui i dati arrivano non è controllata dal sistema (più stream possono arrivare a velocità e con formati diversi) e lo spazio di memoria disponibile è limitato. Eventuali archivi storici possono esistere, ma non sono pensati per rispondere a query online in tempi ragionevoli.

Iniziamo a definire formalmente gli elementi di una stream e come vengono processati.

Def. 2.1 (Stream di dati)**.** Sia $\mathcal{U}$ un universo di chiavi. Senza perdita di generalità, assumiamo $\mathcal{U} \subseteq \mathbb{N}$. Uno *stream di dati* è una sequenza ordinata di elementi

$$S = \langle x_1, x_2, \ldots, x_s \rangle,$$

dove ogni $x_i \in \mathcal{U}$ e s può essere molto grande o non noto a priori.

Def. 2.2 (Frequenze)**.** Dato uno stream S, la *frequenza* di un elemento $a \in \mathcal{U}$ è

$$f(a) = |\{i \mid x_i = a\}|.$$

La collezione delle frequenze può essere vista come un vettore $f \in \mathbb{N}^{|\mathcal{U}|}$.

Def. 2.3 (Numero di distinti)**.** Il *numero di distinti* nello stream S è

$$F_0 = |\{a \in \mathcal{U} \mid f(a) > 0\}|.$$

Def. 2.4 (Frequency moments)**.** Per ogni $k \geq 0$, il *frequency moment* F_k è definito come

$$F_k = \sum_{a \in \mathcal{U}} f(a)^k.$$

In particolare, F_0 corrisponde al numero di distinti.

Def. 2.5 ((ε, δ)-approssimazione)**.** Un algoritmo A è detto (ε, δ)-*approssimante* per F_0 se, per ogni stream, produce una stima $\tilde{F}_0$ tale che

$$\Pr\big(|\tilde{F}_0 - F_0| \leq \varepsilon F_0\big) \geq 1 - \delta,$$

dove la probabilità è rispetto alla randomizzazione interna dell'algoritmo [2].

Esempio. Con $\varepsilon = 0{,}05$ e $\delta = 0{,}01$, l'algoritmo deve restituire una stima entro il 5% da F_0 con probabilità almeno 99%.

Supponiamo inoltre che l'universo abbia dimensione $n = |\mathcal{U}|$ e che ogni elemento x_i richieda b bit per essere rappresentato.

In questo contesto, un algoritmo di streaming deve:

- processare ciascun elemento con costo $O(1)$ o quasi costante;
- usare memoria molto più piccola di n e di $|\mathcal{U}|$;
- produrre una stima $\hat{F}_0$ con errore controllato, utilizzando m bits, dove $m \ll n$.

Nel modello dello stream dei dati esistono diverse tipologie di modelli [3]. In questa tesi adottiamo il seguente.

Def. 2.6 (Modello *insertion-only*)**.** Lo stream è una sequenza di aggiornamenti del tipo (a_t, Δ_t), con $a_t \in \mathcal{U}$ e $\Delta_t \geq 0$. Indichiamo con $A_t[j]$ la frequenza dell'elemento j dopo i primi t aggiornamenti; allora

$$A_t[j] = \begin{cases} A_{t-1}[j] + \Delta_t & \text{se } a_t = j, \\ A_{t-1}[j] & \text{altrimenti.} \end{cases}$$

Esempio. Se lo stream è una lista di valori, ogni elemento x_i può essere visto come un aggiornamento $(x_i, 1)$.

Def. 2.7 (Modello *turnstile*)**.** Nel modello *turnstile* sono ammessi aggiornamenti con Δ_t anche negativi, così che le frequenze possano aumentare o diminuire.

Esempio. Un aggiornamento $(a_t, -1)$ decrementa la frequenza dell'elemento a_t.

Def. 2.8 (Modello *sliding window*)**.** Nel modello a *sliding window* si considerano solo gli ultimi W aggiornamenti della stream, e gli elementi più vecchi vengono dimenticati.

Esempio. Con $W = 10^6$, le statistiche sono calcolate solo sugli ultimi un milione di elementi.

I modelli *turnstile* e *sliding window* esulano dallo scopo di questa tesi.

Per rispettare questi vincoli si ricorre a funzioni hash che approssimano una distribuzione uniforme sugli elementi e a strutture compatte, chiamate **sketch**, che riassumono le informazioni essenziali dello stream senza conservarlo esplicitamente.

Def. 2.9 (Spazio sublineare)**.** Un algoritmo usa *spazio sublineare* se la memoria m cresce asintoticamente meno di n, cioè $m = o(n)$, dove $n = |\mathcal{U}|$. In pratica si richiede che m dipenda in modo polilogaritmico da n e polinomiale da $1/\varepsilon$ e $\log(1/\delta)$.

Esempio. Per il problema dei distinti esistono algoritmi che ottengono una $(1 \pm \varepsilon)$-approssimazione usando $O(\varepsilon^{-2} + \log n)$ bit [4], che è molto meno dei $\Omega(n)$ bit necessari per memorizzare l'insieme dei distinti.

2.2 Algoritmi di streaming

In questa sezione si definisce cosa si intende per algoritmo di streaming e quali vincoli lo distinguono dagli algoritmi tradizionali. Da completare:

- Definizione formale di algoritmo di streaming.
- Stato compatto M e operazioni:

$$M \leftarrow \mathrm{Update}(M, x), \qquad \hat{F}_0 \leftarrow \mathrm{Query}(M).$$

- Modello di costo: un passaggio, update $O(1)$, query su stato compatto.
- Limiti di memoria e trade-off accuratezza/spazio.
- Differenza tra stime in tempo reale e analisi offline.
- Randomizzazione e garanzie probabilistiche.
- (Definizione) mergeabilità/composability degli sketch.

2.3 Funzioni hash

Qui si introduce il ruolo delle funzioni hash nella riduzione dello spazio e nella randomizzazione delle stime. Da completare:

- Definizione di hash e proprietà desiderate.
- Modello ideale: $h : \mathcal{U} \to [0, 1)$ o $h : \mathcal{U} \to \{0, 1\}^w$.
- Assunzioni tipiche: uniformità, indipendenza (o hash quasi-uniformi).
- Impatto di scelte non ideali sull'errore degli stimatori.
- Eventuale gestione del seed per riproducibilità.

2.4 Sketch

Si definisce lo sketch come struttura compatta che riassume lo stream. Da completare:

- Definizione generale di sketch e operazioni supportate (update, query).
- Dimensione dello sketch e relazione con i parametri (k, m, L).
- Esempi intuitivi (registri, bitmap, contatori).
- Due forme usate in tesi:
  - bitmap / pattern di bit (Probabilistic Counting).
  - registri e leading zeros (LogLog, HLL, HLL++).
- (Cenno) serializzabilità e footprint in byte.

2.5 Stimatori

Si descrive cosa si intende per stimatore e in che senso la stima è corretta. Da completare:

- Definizione di stimatore $\hat{F}_0$ e proprietà desiderate.
- Distinzione tra stimatore corretto (unbiased) e stimatore biased.
- Consistenza e varianza dello stimatore.
- Cenno a tecniche di bias correction.

2.6 Metriche di errore

Questa sezione introduce le metriche di qualità della stima. Da completare:

- Errore assoluto e relativo.
- Bias e unbiased estimator.
- Varianza e standard error.
- RMSE e MAE come metriche aggregate.
- RSE teorica vs osservata (quando applicabile).
- Definizioni allineate alle colonne CSV del framework.

2.7 Famiglia di algoritmi per count-distinct

Sezione ponte (senza dettagli implementativi) per motivare il Capitolo 3. Da completare:

- Baseline esatta vs sketch probabilistici.
- Linea FM/Probabilistic Counting $\rightarrow$ LogLog $\rightarrow$ HLL $\rightarrow$ HLL++.
- Differenze qualitative: riduzione varianza, correzioni di range, uso di registri.

2.8 Spazio–accuratezza: ordini di grandezza

Sezione opzionale ma utile per giustificare l'uso degli sketch. Da completare:

- Dipendenza dello spazio da (ε, δ).
- Interpretazione di "ottimalità" a livello di ordine di grandezza.

3

Un’analisi sullo stato dell’arte

TODO: scrivere l’analisi dello stato dell’arte.

4

Implementazione

4.1 Algoritmi

4.2 Framework di benchmark

Example of Algorithm 4.1 reference.

Algorithm 4.1 Pseudocode

$i \leftarrow 10$
if $i \geq 5$
  $i \leftarrow i - 1$
else
  if $i \leq 3$
    $i \leftarrow i + 2$
  end if
end if

5
Risultati sperimentali

Example of Table 5.1, made using `https://www.tablesgenerator.com/`.

A	B
1	2
3	4

Table 5.1: Interesting results.

6
Conclusioni

References

[1] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining Data Streams*. Cambridge University Press, 2014, p. 123–153.

[2] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Randomization and Approximation Techniques in Computer Science (RANDOM 2002)*, ser. Lecture Notes in Computer Science, vol. 2483. Springer, 2002, pp. 1–10.

[3] S. Muthukrishnan, "Data streams: Algorithms and applications," Rutgers University, Tech. Rep., 2005.

[4] D. M. Kane, J. Nelson, and D. P. Woodruff, "An optimal algorithm for the distinct elements problem," in *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2010)*. ACM, 2010.

Acknowledgments