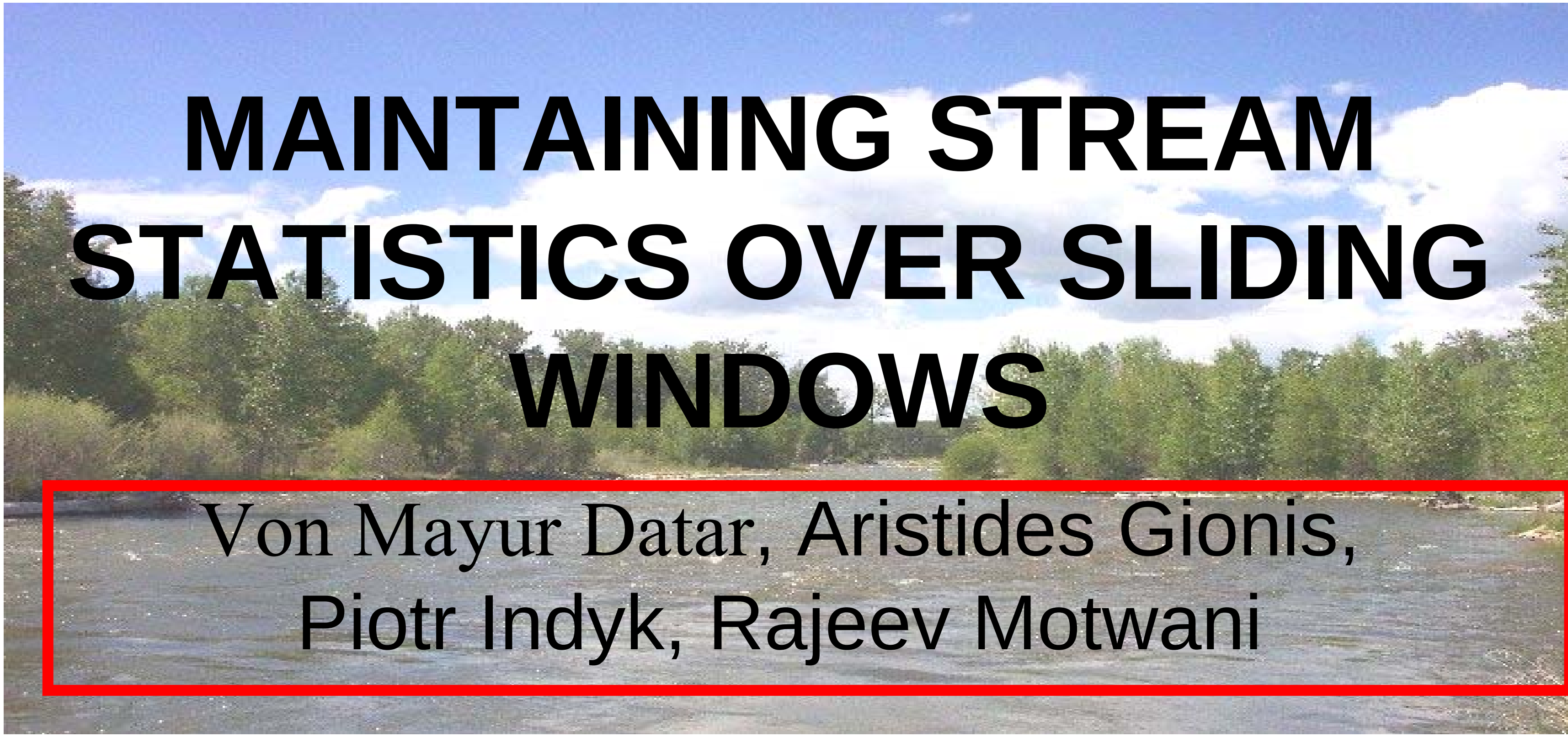


MAINTAINING STREAM STATISTICS OVER SLIDING WINDOWS

Von Mayur Datar, Aristides Gionis,
Piotr Indyk, Rajeev Motwani

Seminar aus maschinellem Lernen

Vorgetragen von Matthias Beckerle

Gliederung des Vortrags

- Einführung
  - Herausforderung
  - Ansatz der Autoren
    - „sliding window“
    - Ableitung von Problemen
- „simple counting problem“
  - Exponential Histogram
  - Algorithmus
  - Komplexitäten
- Erweiterungen und Ableitungen
  - „sum problem“
  - Timestamps
- Zusammenfassung
- Fragen / Diskussion

Einführung: Ein Vergleich

Datenbank:

- Große Teile der Daten werden immer wieder abgefragt
- Updates klein oder relativ unregelmäßig

Data stream:

- Wenn es unnötig ist, auf großen Datenmengen zu operieren
- Daten ändern sich oft

Herausforderung bei Data-Stream

- Ohne den Data-Stream komplett zu speichern wichtige Informationen beizubehalten, um sie später abfragen zu können

  → Techniken notwendig, die eine Art Zusammenfassung des Streams speichern
- Tradeoff: Größe der gespeicherten Daten ← → Genauigkeit der Daten

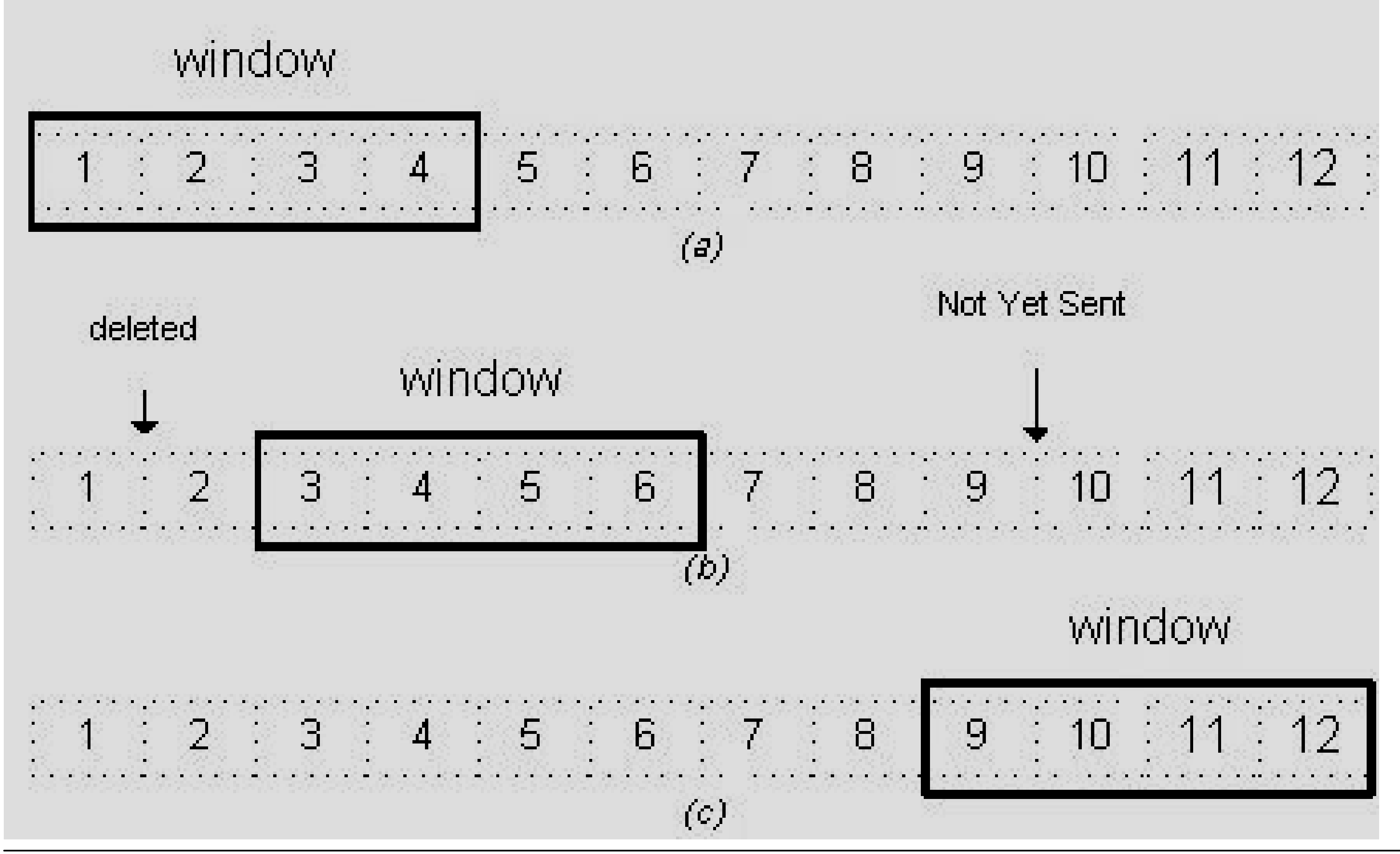
Ansatz der Autoren

- Es wird also nur ein Ausschnitt, das „sliding window“ betrachtet

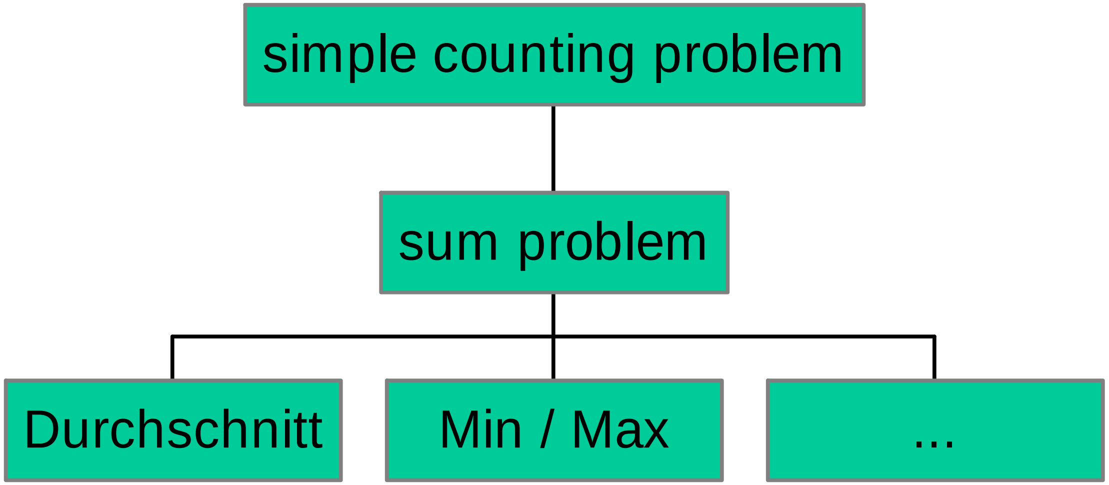
  → Es gibt nur N viele aktuelle Elemente (z.B. die letzen 100 Datensätze)

- Lösen von „einfachen“ Problemen, um Bausteine für komplexere Probleme zu bekommen

Sliding Window



Ableitungsbaum von Problemen



„simple counting problem“

Wieviele 1en waren in den letzten N Bits eines binären Datenstroms?

Bsp: N = 10

1010101001 $\underbrace{1001010101}$

Lösung: 5

„simple counting problem“

(Fortsetzung)

- Mit O(N) Speicherplatz kein Problem
- Erinnerung an letzen Vortrag: Ziel ist es, Lösungen mit O(poly(log N)) Speicherbedarf zu finden

Geht so was?

„simple counting problem“
(Fortsetzung)

- Es wird gezeigt, dass nur $O((1/e)\log^2 N)$ Speicher gebraucht wird
- Aber: Verlust der Accuracy
- Bei gleichem Fehler gilt:
  $\Omega > (k \log^2 (N/k))$

„simple counting problem“

(Fortsetzung)

2 * $\log^2 N$: (e hier 0,5 → k = 2)

N = 10: 32

N = 1000: 200

N = 10^6: 800

N = 10^21: 9800

„simple counting problem“

(Fortsetzung)

- Zuweisung von Werten an jedes Datenelement im „sliding window“:

Timestamps:	5	4	...	1		
Arrival time:	40	41	...	44	45	...
Elemente:	1	0	...	1	1	...

„sliding window“ der aktiven Elemente

Zukunft

Kernwerkzeug: “Exponential Histogram”

- „Alte“ Informationen werden weniger genau gespeichert als „neue“
  - Wenn Informationen „älter“ werden, werden sie zusammengefasst
- speicherschonend (optimal)

„simple counting problem“

(Fortsetzung)

Algorithmus:

1. Wenn neues Datenelement eintrifft
   → Errechne die neue „expire time“
   Wenn Timestamp des letzten Buckets abgelaufen ist
   → Lösche das Bucket und update den Counter „Last“ auf die Größe des letzten Buckets und den Counter „Total“ auf die totale Größe der Buckets
2. Wenn neues Datenlement 0 ist → Ignoriere es
   Sonst → Neues Bucket mit Größe 1 erstellen und „Total“ inkrementieren
3. Traversiere die Buckets in der Reihenfolge zunehmender Größe:
   Wenn k/2 + 2 (in unserem Fall = 3) Buckets gleicher Größe existieren
   → Verschmelze die 2 ältesten Buckets zu einem Bucket doppelter Größe

(siehe auch ausgeteiltes Blatt)

„simple counting problem“
(Fortsetzung)

1: 32, 32, 16, 8, 8, 4, 2 , 1	← 1	Benötigt keine Rechenzeit
2: 32, 32, 16, 8, 8, 4, 4, 2, 1, 1	← 1	
3: 32, 32, 16, 8, 8, 4, 4, 2, 1, 1, 1	← 1	
4: → 32, 32, 16, 8, 8, 4, 4, 2, 2, 1	(merged the older 1,s)	Benötigt O(1)
5: 32, 32, 16, 8, 8, 4, 4, 2, 2, 1, 1	← 1	
6: 32, 32, 16, 8, 8, 4, 4, 2, 2, 1, 1, 1	← 1	
7: → 32, 32, 16, 8, 8, 4, 4, 2, 2, 2, 1	(merged the older 1,s)	Benötigt Max. O(log N)
8: → 32, 32, 16, 8, 8, 4, 4, 4, 2, 1	(merged the older 2,s)	
9: → 32, 32, 16, 8, 8, 8, 4, 2, 1	(merged the older 4,s)	
10: → 32, 32, 16, 16, 8, 4, 2, 1	(merged the older 8,s)	

Rechenschritte:

→ Pro Datenelement im Schnitt O(1); O(log N) im worst case

„simple counting problem“

(Fortsetzung)

Anzahl der Buckets:

2*(log(N+1)+1)

N = 1 :	4 Buckets
N = 5 :	8 Buckets
N = 10 :	10 Buckets
N = 100 :	16 Buckets
N = 1000 :	22 Buckets
N = 10^6 :	42 Buckets
N = 10^21:	142 Buckets

Speicher pro Bucket:

log N + log log N

Größe des Buckets

Timestamp

Gesamt:

Anz. Buckets * Speicher pro Bucket:

$$\mathbf{O((1/e)\log^2 N)}$$

„sum problem“

Wie ist die Summe der letzten N Zahlen eines positiven Integer Stroms im Bereich [0...R]?

Bsp: N = 3

10, 45, 12, 15, $\underbrace{41, 3, 1002}$

Lösung: 1046

„sum problem“

(Fortsetzung)

- „Einfache“ Erweiterung des „simple counting problems“ :
  $O(k \log^2 N \log R)$ Speicher nötig
  (statt 1en werden Zahlen gespeichert)
- Optimiert nur:
  $O(k \log N (\log R + \log N))$

„sum problem“
(Fortsetzung)

Jede eintreffende Zahl v wird als v eintreffende 1en interpretiert
- Timestamps der 1en sind alle gleich
- Die 1en werden wieder normal im EH gespeichert

- (k/2 + 1)(log((2NR)/k + 1) + 1) Buckets
- log N + log(log N + log R) Bits pro Bucket
- zusammen: O((1/e)(log N + log R) (log N))
- Über l-canonical representation:

  Ankommende Objekte benötigen: O((log R / log N))
  Rechenzeit im worst case: O(log N + log R)
- Summenberechnungen brauchen nach wie vor nur O(1)

Beispiel für das Ableiten von Funktionen: Durchschnitt

- Summe berechnen
- Summe durch N teilen

$O(k \log N (\log R + \log N))$ Speicher

Erweiterung: Timestamps

- Umwandlung von letzten N Elementen in zeitliche Angaben wie die letzten 24 Stunden
  - z.B. durch sekundenweise Erhöhung der „arrival time“
  - Timestamp-Information wird größer

„Optimierung“ Timestamps

- Timestamps als Vielfaches von 2 speichern

  → log log N statt log N

  →Zeitangaben sind nicht mehr genau:

  Tradeoff

Weitere Beispiele

Beispiele für Ableitungen:

- Min / Max Berechnung
- Lp (p element aus [1,2]) Normen von Vektoren
- Hash tables
- ...

Zusammenfassung

- Streams brauchen viel Platz →
  Doppelte Komprimierung
  - Nur einen Ausschnitt betrachten
    - Fenstergröße N dynamisch änderbar
  - EH (Exponential Histograms) verwenden
    - Speicherausnutzung ist optimal mit EH
    - Neuere Daten werden genauer gespeichert als ältere
    - Informationsgewinnung mit O(1) realisierbar

Zusammenfassung
(Fortsetzung)

- EH ist als Konzept vielseitig anwendbar →

Altbekanntes nutzen

  - Lösung eines neues Problems von bereits effizient gelöstem Problem ableiten.
  - Anschließend optimieren

Aber:

Komprimierung funktioniert nicht verlustfrei!

→ Fehler Tradeoff einstellbar

→ Gerings möglicher Fehler bei gleicher Platzeinspaarung

Fragen?

Diskussion!

Danke für Ihre Aufmerksamkeit!

Quellen

- [1] N. Alon, Y. Matias, and M. Szegedy, *The space complexity of approximating the frequency*
- *moments*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing,
- ACM, New York, 1996, pp. 20{29.
- [2] C. Cortes, K. Fisher, D. Pregibon, and A. Rogers, *Hancock: A language for extracting signatures*
- *from data streams*, in Proceedings of the ACM SIGKDD International Conference
- on Knowledge Discovery and Data Mining, ACM, New York, 2000, pp. 9{17.
- [3] S. Chaudhuri, R. Motwani, and V. R. Narasayya, *On random sampling over joins*, in
- Proceedings of the ACM SIGMOD International Conference on Management of Data,
- ACM, New York, 1999, pp. 263{274.
- [4] M. Fang, H. Garcia-Molina, R. Motwani, N. Shivakumar, and J.D. Ullman, *Computing*
- *iceberg queries eciently*, in Proceedings of the 24th International Conference on Very
- Large Data Bases, New York, NY, 1998, pp. 299{310.
- [5] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, *An approximate L*1-*dierence*
- *algorithm for massive data streams*, in Proceedings of the 40th IEEE Symposium on
- Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1999, pp.
- 501{511.
- [6] P. Flajolet and G. Martin, *Probabilistic counting*, in Proceedings of the 24th IEEE Symposium

Quellen

(Fortsetzung)

- [7] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi, *Architecture of a Passive Monitoring*
- *System for Backbone IP Networks*, Technical report TR00-ATL-101-801, Sprint
- Advanced Technology Laboratories, Burlingame, CA, 2000.
- [8] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, *Surng wavelets on streams:*
- *One-pass summaries for approximate aggregate queries*, in Proceedings of the 27th International
- Conference on Very Large Data Bases, Rome, Italy, 2001, pp. 79{88.
- [9] S. Guha and N. Koudas, *Approximating a data stream for querying and estimation: Algorithms*
- *and performance evaluation*, in Proceedings of the Eighteenth International Conference
- on Data Engineering, San Jose, CA, 2002, pp. 567{576.
- [10] S. Guha and N. Koudas, *Data-streams and histograms*, in Proceedings of the Thirty-Third
- Annual ACM Symposium on Theory of Computing, ACM, New York, 2001, pp. 471{475.
- [11] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, *Clustering data streams*, in Proceedings
- of the 41st Annual IEEE Symposium on Foundations of Computer Science, IEEE
- Computer Society, Los Alamitos, CA, 2000, pp. 359{366.
- [12] M. R. Henzinger, P. Raghavan, and S. Rajagopalan, *Computing on Data Streams*, Technical
- report TR 1998-011, Compaq Systems Research Center, Palo Alto, CA, 1998

Quellen
(Fortsetzung)

- [13] P. Indyk, *Stable distributions, pseudorandom generators, embeddings and data stream computation*,
- in Proceedings of the 41st IEEE Symposium on Foundations of Computer Science,
- IEEE Computer Society, Los Alamitos, CA, 2000, pp. 189{197.
- [14] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel,
- *Optimal histograms with quality guarantees*, in Proceedings of the 24th International Conference
- on Very Large Data Bases, New York, NY, 1998, pp. 275{286.
- [15] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge,
- UK, 1995.
- [16] Cisco Systems, *Netflow Services and Applications*, White paper, Cisco Systems, San Jose,
- CA, 2000; also available online from http://www.cisco.com/warp/public/cc/pd/iosw/ioft/
- neflct/tech/napps wp.htm.