Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 8 – Bounded Differences and Bloom Filters

Randomisierte Algorithmik

Aufgabe 1 – Bälle, Behälter und beschränkte Differenzen

Sei $\lambda > 0$ eine Konstante, $m = \lambda n$ eine Anzahl von Bällen und n eine Anzahl von Behältern. Der j-te Ball werde in Behälter X_j platziert wobei $X_1, \dots, X_m \sim \mathcal{U}([n])$ unabhängige Zufallsvariablen sind. Die Kollisionszahl ist definiert als $C = |\{(i,j) \mid 1 \le i < j \le n, X_i = X_j\}|$. Ziel dieser Aufgabe ist es, eine Konzentrationsschranke an C zu bestimmen.

(i) Zeige $\mathbb{E}[C] = \Theta(n)$.

Für $i \in [n]$ sei $L_i = |\{j \in [m] \mid X_j = i\}|$ der Füllstand von Behälter i. Es ist weder schwer noch interessant zu zeigen, dass $\Pr[\max_{i\in[n]} L_i \le \log n] \ge 1 - O(1/n)$. Nimm das im Folgenden als gegeben hin (Beweis in der Musterlösung).

(ii) Definiere $C_{\text{cap}} := \sum_{i\in[n]} \binom{\min(\log n, L_i)}{2}$. Zeige $\Pr[C_{\text{cap}} = C] = 1 - O(1/n)$.

(iii) Zeige $\Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \ge t] \le \exp(-2t^2/(m \cdot \log^2 n))$.

(iv) Zeige $\Pr[C - \mathbb{E}[C] \ge n^{2/3}] = O(1/n)$.

Lösung 1

(i) Je zwei verschiedene Bälle kollidieren mit Wahrscheinlichkeit $1/n$. Also gilt $\mathbb{E}[C] = \binom{m}{2} \cdot \frac{1}{n} = \frac{m(m-1)}{2n} = \frac{\lambda(m-1)}{2} = \Theta(n)$.

Wir schauen uns nun die behauptete Schranke an die Füllstände an. Seien zunächst $i \in [n]$ und $k \in \mathbb{N}$ beliebig aber fest. Für eine Menge $S \subseteq [m]$ der Größe k sei $E_{S,i}$ das Ereignis, dass alle Bälle aus S in Behälter i platziert werden. Es gilt $\Pr[E_{S,i}] = n^{-k}$. Dann folgt:

$$\Pr[\max_{i\in[n]} L_i \ge k] = \Pr\left[\bigcup_{\substack{S\subseteq[m], i\in[n]\\ |S|=k}} E_{S,i}\right] \overset{\text{UB}}{\le} \sum_{\substack{S\subseteq[m], i\in[n]\\ |S|=k}} \Pr[E_{S,i}] = \binom{m}{k} \cdot n \cdot n^{-k} \le \frac{m^k}{k!} n^{-k+1} = \frac{\alpha^k n}{k!}.$$

Jetzt muss man nur noch $k = \log n$ einsetzen und verwenden, dass dann $\alpha^k = \text{poly}(n)$ sowie $m = \text{poly}(n)$ gilt, aber $(\log n)!$ schneller als jedes Polynom wächst.

(ii) Falls $\max_{i\in[n]} L_i \leq \log n$ gilt so folgt

$$C_{\text{cap}} = \sum_{i\in[n]} \binom{L_i}{2}.$$

Letzeres ist aber eine alternative Definitionsmöglichkeit für die Kollisionszahl C (summiere Kollisionen innerhalb jedes Behälters).

(iii) C_{cap} ergibt sich als Funktion aus den m unabhängigen Zufallsvariablen $X_1, \ldots, X_m$. Verändert sich eines der X_j, so kann sich C_{cap} um höchstens $\log n$ verändern. Das Ergebnis ergibt sich durch direkte Anwendung von McDiarmid's Ungleichung.
Bemerkung: Auf C selbst lässt sich die Strategie nicht gut anwenden. Der Extremfall ist, dass $X_1 = \ldots = X_{m-1} = 1$ und $X_m = 2$ gilt. Ändert man X_m auf 1, so entstehen $m - 1$ zusätzliche Kollsionen.

(iv) Wir setzen die vorherigen beiden Teilergebnisse zusammen:

$$\begin{aligned}
\Pr[C - \mathbb{E}[C] \geq n^{2/3}] &\leq \Pr[C \neq C_{\text{cap}} \vee C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{\text{UB}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{C \geq C_{\text{cap}}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \geq n^{2/3}] \\
&\leq O(1/n) + \exp(-2n^{4/3}/(m \log^2 n)) = O(1/n).
\end{aligned}$$

Folgende Aufgabe hat nicht wirklich mit randomisierten Algorithmen zu tun, außer dass wir die Abschätzung in der Vorlesung gebraucht haben. Daher "Bonus".

Aufgabe 2 – Bonus: Approximationen von *e*

Du weißt sicher, dass $e = \lim_{n\to\infty}(1 + \frac{1}{n})^n$ gilt (das ist sogar eine gängige *Definition* von e). Leite daraus ab, dass die folgenden beiden Gleichungen für alle $n \in \mathbb{N}$ gelten:

$$\begin{aligned}
(1 + \tfrac{1}{n})^n \leq e \leq (1 + \tfrac{1}{n})^{n+1} \\
(1 - \tfrac{1}{n})^n \leq e^{-1} \leq (1 - \tfrac{1}{n})^{n-1}.
\end{aligned}$$

Folgende Schritte bieten sich an.

(i) Zeige die linken beiden Ungleichungen.
Hinweis: Benutze mal wieder $1 + x \leq e^x$.

(ii) Zeige, dass die rechten Seiten monoton in n fallen.

(iii) Folgere aus (ii) und einer Grenzwertbetrachtung die rechten beiden Ungleichungen.

Lösung 2

(i) Es gilt jeweils:

$$(1+\tfrac{1}{n})^n \le (e^{1/n})^n = e$$
$$(1-\tfrac{1}{n})^n \le (e^{-1/n})^n = e^{-1}$$

(ii) Zunächst erhalten wir durch Logarithmieren des Hinweises für (i):

$$\forall x \in (-1,\infty) : \ln(1+x) \le x$$

Um zu zeigen, dass $f(n) = (1+\frac{1}{n})^{n+1}$ monoton in $n \in \mathbb{N}$ fällt genügt es zu zeigen, dass $\ln(f(x)) = (x+1)\ln(1+\frac{1}{x})$ monoton in $x \in (0,\infty)$ fällt. Dazu betrachten wir die Ableitung und zeigen, dass diese überall kleinergleich 0 ist:

$$(\ln(f(x))' = \ln(1+\tfrac{1}{x}) + \frac{x+1}{1+\frac{1}{x}} \cdot (-\tfrac{1}{x^2}) \le \tfrac{1}{x} - \frac{x+1}{(x+1)x} = 0.$$

Analog argumentieren wir für $g(x) = (1-\frac{1}{n})^{n-1}$ dass die Ableitung von $\ln(g(x))$ kleinergleich 0 ist:

$$(\ln(g(x))' = ((n-1)\ln(1-\tfrac{1}{n}))' = \ln(1-\tfrac{1}{x}) + \frac{x-1}{1-\frac{1}{x}} \cdot \frac{1}{x^2} \le -\tfrac{1}{x} + \frac{x-1}{(x-1)x} = 0.$$

(iii) Es gilt durch Separierung eines Faktors:

$$\lim_{n\to\infty}(1+\tfrac{1}{n})^{n+1} = \lim_{n\to\infty}(1+\tfrac{1}{n})^n \cdot \lim_{n\to\infty}(1+\tfrac{1}{n}) = e \cdot 1 = e.$$

Den Grenzwert von $(1-\frac{1}{n})^{n-1}$ können wir folgendermaßen bestimmen:

$$\lim_{n\to\infty}(1-\tfrac{1}{n})^{n-1} = \lim_{n\to\infty}(\tfrac{n-1}{n})^{n-1} = \lim_{n\to\infty}\left(\frac{1}{\frac{n}{n-1}}\right)^{n-1} = \frac{1}{\lim_{n\to\infty}(\frac{n}{n-1})^{n-1}}$$
$$= \frac{1}{\lim_{n\to\infty}(1+\frac{1}{n-1})^{n-1}} = \frac{1}{e} = e^{-1}.$$

Also konvergieren die rechten Seite gegen e bzw. e^{-1}. Das tun sie nach (iii) „von oben“. Also gelten die Ungleichungen wie behauptet.

Aufgabe 3 – Counting Bloomfilter (a.k.a.: Count-Min-Sketch)

Ein Counting Bloomfilter ist zunächst wie ein Bloomfilter: Er verwaltet n Schlüssel in einem Array der Größe m, der *load factor* ist $\alpha := \frac{n}{m}$ und es gibt k Hashfunktionen. Die Parameter seien wie in der Vorlesung gewählt, sodass ein (gewöhnlicher) Bloomfilter eine Falsch-Positiv-Wahrscheinlichkeit von ε hätte. Das Array enthält nun keine Bits mehr sondern natürliche Zahlen. Neben insert und query gibt es nun als weitere Operation delete.

Algorithm insert(x):
```
for i ∈ [k] do
    A[h_i(x)] + +
```

Algorithm delete(x):
```
for i ∈ [k] do
    A[h_i(x)] − −
```

Algorithm query(x):
```
for i ∈ [k] do
    if A[h_i(x)] = 0 then
        return false
return true
```

Wir lassen zu, dass ein Schlüssel *mehrfach* in den Counting Bloomfilter eingefügt wird. Insofern ist das verwaltete Objekt S nun eine *Multimenge*, deren n Elemente $\{x_1, \ldots, x_n\}$ jeweils mit einer Vielfachheit $a_1, \ldots, a_n$ vorliegen. Die Operation insert(x) soll der Multimenge S eine Kopie von x hinzufügen, das heißt die Vielfachheit von x erhöhen, falls x bereits in S enthalten war oder ein weiteres Element mit Vielfachheit 1 in S aufnehmen, falls x noch nicht in S enthalten war. Die Bedeutung von delete ist analog.

Wie zuvor auch darf query falsch-positive, aber keine falsch-negativen Antworten liefern.

(a) Wir fordern, dass delete nur für solche Elemente aufgerufen wird, die auch wirklich in S enthalten sind (mit Vielfachheit mindestens 1). Was geht kaputt wenn wir das nicht fordern?

Mit Counting Bloomfiltern lassen sich noch zusätzliche nützliche Operationen implementieren.

(b) Implementiere eine Operation count, die für $x \in D$ einen Schätzwert count(x) für die Vielfachheit a von x in S angibt. Zeige, dass $\Pr[a \neq \text{count}(x)] \leq \varepsilon$ gilt.

(c) Das wichtigste Argument dafür (Counting-) Bloomfilter zu verwenden, ist der geringe Speicherplatz im Vergleich zu einer exakten Datenstruktur. Wir sollten daher besser nicht große Integer Datentypen (etwa 64 Bit) für die Zähler verwenden. Stellen wir uns einen Anwendungsfall vor, in dem die „meisten“ Zähler niemals 8 Bit überschreiten. Wir nutzen daher ein Array A von 8-Bit Zählern. Diskutiere (kurz) die folgenden Vorschläge zum Umgang mit Zählerüberläufen. Was sind die Vorteile und welche Kompromisse werden eingegangen?

- Alice verhindert lediglich Zählerüberläufe, passt also die $A[h_i(x)] + +$ Operation in insert und die $A[h_i(x)] - -$ Operation in delete so an, dass der Zähler nur inkrementiert bzw. dekrementiert wird, wenn der maximale darstellbare Wert bzw. der minimale darstellbare Wert noch nicht erreicht ist.
- Bob schlägt vor, einen Zähler, der den Wert $(11111111)_2 = 255$ erreicht hat „einzufrieren“, das heißt zukünftige insert oder delete Operationen werden den Zähler nie mehr anpassen.
- Carol schlägt vor, die Bitfolge $(11111111)_2 = 255$ als Markierung dafür zu verwenden, dass der wahre Zählerwert Größer als 254 ist. Der wahre Zählerwert wird in diesem Sonderfall in einer Hashtabelle gespeichert.

Lösung 3

(a) Ein delete(y) für ein y, das niemals eingefügt reduziert unkontrolliert k Zähler im Counting Bloomfilter. So können Zähler 0 werden. Dadurch könnte es sein, dass für ein $x \in S$ anschließend query(x) = false gilt. Also wird für x eine falsch-negative Antwort gegeben, was wir nicht wollen.

(b) Wir geben das Minimum der mit x assoziierten Zähler zurück:

Algorithm count(x)**:**
- $r \leftarrow \infty$
- **for** $i \in [k]$ **do**
  - $r \leftarrow \min(r, A[h_i(x)])$
- **return** r

Zunächst fällt auf, dass $\text{count}(x) < a$ unmöglich ist, denn jedes Vorkommen von x wird von jedem mit x assoziierten Zähler gezählt. Es könnte allerdings sein, dass $\text{count}(x) > a$ gilt, wenn alle k Zähler zusätzlich von anderen Schlüsseln verwendet werden.

Um dies zu verstehen sei $A'[1..m] \in \{0,1\}^m$ der gewöhnliche Bloomfilter, in den die Elemente aus S mit Ausnahme von x eingefügt wurden (mit den selben Hashfunktionen, wie für den Counting Bloomfilter). Dann gilt:

$$
\begin{aligned}
\text{count}(x) \neq a &\Leftrightarrow \text{count}(x) > a \\
&\Leftrightarrow \min_{i \in [k]} A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : \text{es gibt einen Schlüssel in } S \text{ außer } x \text{ der } A[h_i(x)] \text{ verwendet} \\
&\Leftrightarrow \forall i \in [k] : A'[h_i(x)] = 1 \\
&\Leftrightarrow x \text{ ist falsch-positives Element für } A'
\end{aligned}
$$

Das letzte Ereignis hat nach Wahl der Konfigurationsparameter höchstens Wahrscheinlichkeit ε. Also gilt dies auch für das dazu äquivalente erste Ereignis.

(c) Zu den Vorschlägen ist folgendes zu sagen:

- Der Vorschlag von Alice ist einfach zu implementieren, allerdings werden falsch-negative Antworten möglich. Am einfachsten sieht man das an der Operationsfolge, die 256 mal den selben Schlüssel x einfügt und diesen anschließend 255 mal wieder löscht. Die letzte Einfügung geht verloren und die Löschungen machen alle relevanten Zähler wieder zu 0. Nun würde query(x) als Ergebnis falsch zurückgeben, obwohl der Schlüssel noch einmal in der Datenstruktur sein müsste. Doof.
- Bei Bobs Vorschlag kann ein Zähler niemals fälschlich zurück auf Null fallen, also kann es keinen falsch-negativen Antworten geben. Ein Nachteil ist, dass man eingefrorene Zähler niemals wieder los wird. Langfristig könnte daher die falsch-positiv Wahrscheinlichkeit steigen.

- Carols Vorschlag macht keine Kompromisse bei der Funktionalität. Die zusätzliche Hashtabelle kostet aber selbstverständlich Platz und Zugriffe darauf brauchen Zeit.

Aufgabe 4 – Schnitte Schätzen mit Bloomfiltern

Seien $n, m, k \in \mathbb{N}$. Alice und Bob wollen abschätzen, wie ähnlich ihr Musikgeschmack ist. Seien X die n Lieblingslieder von Alice und Y die n Lieblingslieder von Bob. Zu schätzen ist $\gamma := \frac{|X \cap Y|}{n} \in [0, 1]$. Beide gehen folgendermaßen vor.

- Alice konstruiert einen Bloomfilter $A[1..m] \in \{0,1\}^m$ für X unter Verwendung von k Hashfunktionen $h_1, \ldots, h_k$.
- Bob konstruiert einen Bloomfilter $B[1..m] \in \{0,1\}^m$ für Y unter Verwendung *derselben* k Hashfunktionen.
- Alice und Bob tauschen ihre Filter aus und berechnen $\delta := \frac{|\{i \in [m] | A[i] \neq B[i]\}|}{m}$.
- Alice und Bob berechnen basierend auf δ eine Schätzung $\bar{\gamma}$ für γ.

Löse folgende Teilaufgaben:

(a) Diskutiere: Welche Vor- und Nachteile könnte das Verfahren im Vergleich zum direkten Austausch von X und Y haben?

(b) Gewinne Intuition: Welche Werte von δ erwartest du (in etwa) für die Extremfälle, in denen $\gamma = 1$ bzw. $\gamma = 0$ gilt?
Hinweis: Du darfst hier und im Folgenden davon ausgehen, dass den Bloomfiltern eine „optimale“ Konfiguration mit $\alpha k = \ln(2)$ zugrundegelegt wurde.

(c) Berechne $\mathbb{E}[\delta]$ als Funktion von γ. Du darfst hierbei Terme niederer Ordnung unter den Tisch fallen lassen, also z.B. $(1 - \frac{1}{m})^m \approx e^{-1}$ schreiben, ohne ein $o(1)$ mitzuführen.
Hinweis: Zunächst scheint es, als könnten andere Parameter (z.B. $n, m, k, \alpha, \varepsilon$) auch eine Rolle spielen. Deren Einfluss verschwindet aber in Termen niederer Ordnung.

(d) Diskutiere: Welche Konzentrationsschranke eignet sich, um zu beweisen, dass δ mit hoher Wahrscheinlichkeit nahe an $\mathbb{E}[\delta]$ liegt?

(e) Stelle die Gleichung aus (c) um, sodass ersichtlich wird, wie eine Schätzung $\bar{\gamma}$ für γ aus δ berechnet werden kann.

(f) Spekuliere: Welche Rolle spielt die Wahl von k (bzw. von ε) im vorliegenden Kontext?

Lösung 4

(a)
- Vorteil: Der Platzverbrauch ist unter Umständen geringer.
- Nachteil: Wir können zwar erreichen, dass $\bar{\gamma}$ mit hoher Wahrscheinlichkeit in der Nähe von γ liegt, aber wir können γ nicht fehlerfrei berechnen.

- Vorteil: Die Elemente der Mengen X und Y werden nicht bekanntgegeben, d.h. Alice kann für jedes $x \in X$ abstreiten dass $x \in X$ gilt, ohne dass jemand sie der Lüge überführen kann.

(b) Für $\gamma = 1$ gilt offensichtlich $A[i] = B[i]$ für alle i und damit $\delta = 0$. Für $\gamma = 0$ haben die beiden Bloomfilter nichts miteinander zu tun und sind unabhängig. Laut Vorlesung sind in einem Bloomfilter mit $\alpha k = \ln(2)$ etwa die Hälfte der Einträge 1 und die andere Hälfte ist 0. Plausibel ist also, dass für alle $i \in [m]$ gilt: $\Pr[A[i] \neq B[i]] \approx \Pr_{C,D\sim\mathrm{Ber}(1/2)}[C \neq D] = 1/2$. Wir erwarten also $\delta \approx 1/2$.

(c) Wir schreiben kurz $h(z) := \{h_1(z), \ldots, h_k(z)\}$. Beachte: Ereignisse, die sich auf verschiedene Schlüssel oder verschiedene Hashfunktionen beziehen können wir wegen Unabhängigkeit „auseinanderziehen“. Im Folgenden nutzen wir x_0 für ein beliebiges Element in X und y_0 für ein beliebiges Element in $Y \setminus X$.

$$
\begin{aligned}
\mathbb{E}[\delta] &= \frac{\mathbb{E}[|\{i \in [m] \mid A[i] \neq B[i]\}|]}{m} = \frac{1}{m}\sum_{i=1}^{m} \Pr[A[i] \neq B[i]] = \Pr[A[i_0] \neq B[i_0]] \\
&= \Pr[(A[i_0] = 0 \wedge B[i_0] = 1) \vee (A[i_0] = 1 \wedge B[i_0] = 0)] = 2\Pr[A[i_0] = 0 \wedge B[i_0] = 1] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x) \wedge \exists y \in Y \setminus X : i_0 \in h(y)] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x)] \cdot \Pr[\exists y \in Y \setminus X : i_0 \in h(y)] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x)] \cdot (1 - \Pr[\forall \in Y \setminus X : i_0 \notin h(y)]) \\
&= 2\Pr[i_0 \notin h(x_0)]^{|X|} \cdot (1 - \Pr[i_0 \notin h(y_0)]^{|Y\setminus X|}) \\
&= 2\Pr[i_0 \neq h_1(x_0)]^{k|X|} \cdot (1 - \Pr[i_0 \neq h_1(y_0)]^{k|Y\setminus X|}) \\
&= 2(1 - \tfrac{1}{m})^{k|X|} \cdot (1 - (1 - \tfrac{1}{m})^{k|Y\setminus X|}) = 2(1 - \tfrac{1}{m})^{kn} \cdot (1 - (1 - \tfrac{1}{m})^{k(1-\gamma)n}) \\
&= 2(1 - \tfrac{1}{m})^{k\alpha m} \cdot (1 - (1 - \tfrac{1}{m})^{k(1-\gamma)\alpha m}) \\
&\approx 2e^{-k\alpha} \cdot (1 - e^{-k(1-\gamma)\alpha}) = 2e^{-\ln(2)} \cdot (1 - e^{-(1-\gamma)\ln(2)}) = 1 - \left(\tfrac{1}{2}\right)^{(1-\gamma)}.
\end{aligned}
$$

(d) Die Methode der beschränkten Differenzen bzw. McDiarmids Ungleichung ist hier geeignet. Die $k \cdot |X \cup Y|$ relevanten Hashwerte sind alle unabhängig. Verändert man einen davon dann ändert sich der Wert von δ um höchstens $\pm\frac{1}{m}$. Die Analyse der Vorlesung bzgl. der Konzentration von Z (Anzahl Nuller) überträgt sich problemlos.

(e) Aus (c) haben wir $\mathbb{E}[\delta] = 1-(\frac{1}{2})^{(1-\gamma)}$. Wir entfernen das „$\mathbb{E}$“ (weil wir nur δ zur Verfügung haben, nicht $\mathbb{E}[\delta]$) und ersetzen γ durch $\bar{\gamma}$ (weil wir also nicht γ ausrechnen können sondern nur eine Schätzung dafür). Umstellen von $\delta = 1-(\frac{1}{2})^{(1-\bar{\gamma})}$ ergibt: $\bar{\gamma} = 1-\log_2(1/(1-\delta))$.

(f) Je größer k ist, desto stärker wird die Konzentrationsschranke und desto besser wird entsprechend die Schätzung $\bar{\gamma}$. Es spricht aber wenig dagegen einfach $k = 1$ zu verwenden. Es ist sogar denkbar $k \in (0, 1)$ zu wählen mit der Bedeutung, dass ein Schlüssel nur mit Wahrscheinlichkeit k eine Position zugeteilt bekommt und mit Wahrscheinlichkeit $1 - k$ einfach verworfen wird. Diese Zufallsentscheidungen müssten widerum durch eine Hashfunktion getroffen werden, die Alice und Bob beide kennen. Mit $k = \Theta(\frac{1}{n})$ könnte man

einen Speicherbedarf erreichen der nicht mehr von n abhängt. Ähnlich wie bei Approximationsalgorithmen könnte man einen relativen Fehler und eine Fehlerwahrscheinlichkeit einführen und ausrechen wie groß k in Abhängigkeit dieser beiden Parameter gewählt werden müsste.