دانشکده علوم ریاضی
دانشگاه صنعتی شریف

الگوریتم‌های خلاصه‌سازی برای مه‌داده

محمدهادی فروغمنداعرابی
پاییز ۱۳۹۹

خلاصه‌سازی خطی برای تقریبی از آرایه با افزایش و کاهش خانه‌ها

جلسه هفتم

نگارنده: امیر آذرمهر

۱ صورت مسئله

فرض کنید یک بردار $x \in \mathbb{R}^n$ داریم، که طی زمان تغییراتی می‌کند. می‌خواهیم با استفاده از حافظه‌ی $m \ll n$، تقریبی از مقدار x را ذخیره کنیم. به صورت دقیق‌تر برای نوع تغییرات x سه حالت در نظر گرفته می‌شود:

- فقط اضافه کردن؛ یعنی با هر تغییر به ازای یک i قرار می‌دهیم $x_i =\leftarrow x_i + 1$
- شمارنده‌ی مثبت[1]؛ یعنی هر بار قرار می‌دهیم $x_i \leftarrow x_i \pm 1$، به شکلی که تمام مؤلفه‌ها همواره نامنفی باشند.
- شمارنده‌ی آزاد[2]؛ یعنی هر بار قرار می‌دهیم $x_i \leftarrow x_i \pm 1$ و محدودیتی روی مقدار مؤلفه‌ها وجود ندارد.

همچنین برای نگهداری تقریبی x یکی از دو شیوه‌ی زیر را برای پاسخگویی الگوریتم در نظر می‌گیریم:

- پرسش نقطه‌ای (مؤلفه‌ای)[3] ℓ_1: در این شیوه پارامتر k یک عدد ثابت است. اندیس i به الگوریتم داده می‌شود و انتظار می‌رود که مقدار x_i را با حداکثر خطای مطلق $\|x\|_1/k$ گزارش دهد.
- پرسش وزین‌ها[4] ℓ_1: در این شیوه هم پارامتر k ثابت است و از الگوریتم انتظار می‌رود که هنگام پرسش، یک زیرمجموعه‌ی S از اندیس‌ها خروجی دهد. به شکلی که اندازه‌ی S از مرتبه $\mathcal{O}(k)$ باشد و تمام اندیس‌هایی مثل i که $|x_i| \geq \|x\|/k$ در S آمده باشند. **نکته:** دقت کنید

---

[1] strict turnstile
[2] general turnstile
[3] point query
[4] heavy hitters

مجموعه‌ی اندیس‌های دارای خاصیت گفته شده، همواره حداکثر k عضو دارد و الگوریتم از این جهت تقریبی است که اجازه دارد $\mathcal{O}(k)$ اندیس بازگرداند. این آزادی بیشتر اجازه می‌دهد که حافظه‌ی کمتری مصرف شود.

۲ کاهش پرسش وزین‌ها به پرسش نقطه‌ای

قضیه‌ای که در ادامه می‌آید، به مفهومی نشان می‌دهد که مسئله پرسش نقطه‌ای سخت‌تر از پرسش وزین است. توجه داشته باشید که الگوریتم‌هایی که مورد بررسی هستند همگی تصادفی هستند و به احتمال مشخصی پرسش را درست جواب می‌دهند.

قضیه ۱. *اگر الگوریتم $\mathcal{A}$ برای پاسخ دادن پرسش نقطه‌ای ℓ_1 با پارامتر $3k$ وجود داشته باشد که هر پرسش را به احتمال حداکثر کمتر از δ/n اشتباه پاسخ دهد. آنگاه الگوریتمی برای پاسخ دادن پرسش وزین‌ها ℓ_1 با پارامتر k وجود دارد که هر پرسش را به احتمال کمتر از δ اشتباه پاسخ دهد.*

اثبات. برای ساخت الگوریتم $\mathcal{A}'$ که پرسش وزین‌ها را پاسخ دهد به این شکل عمل می‌کنیم: از الگوریتم $\mathcal{A}$ برای اعمال تغییرها استفاده می‌کنیم. برای پاسخ دادن پرسش، مجموعه‌ی S را برابر با $3k$ بزرگترین مؤلفه، طبق تقریبی که $\mathcal{A}$ از آن‌ها دارد، قرار می‌دهیم. باید ثابت کنیم که S، به احتمال مورد نظر، دو خاصیت مورد نظر را دارد.

این پدیده که $\mathcal{A}$ پرسش iـاُم را اشتباه پاسخ دهد را E_i می‌نامیم. طبق فرض احتمال وقوع E_i حداکثر δ/n است. پس طبق کران اجتماع[5] احتمال این که حداقل یکی از پرسش‌ها اشتباه پاسخ‌داده شود کمتر از δ است. پس اینکه همه‌ی پرسش‌ها به درستی توسط $\mathcal{A}$ پاسخ داده شوند، حداقل $1 - \delta$ است. نشان می‌دهیم در این صورت S خواص مورد نظر را دارد.

اندازه‌ی S طبق تعریف از مرتبه‌ی $\mathcal{O}(k)$ است. برای اثبات خاصیت دوم، مقدار $t = \frac{2}{3k}\|x\|$ را در نظر بگیرید. اثبات می‌کنیم تمام مؤلفه‌هایی که مقدار گزارش‌شده‌ی آن‌ها (منظور تقریبی است که $\mathcal{A}$ خروجی می‌دهد) از t بزرگترمساوی باشد در S قرار خواهد گرفت. چون که اگر به ازای i داشته باشیم $|x_i| < \frac{1}{3k}\|x\|$ آنگاه مقدار گزارش‌شده‌ی آن، طبق خواص الگوریتم $\mathcal{A}$، مقدار کوچکتر از $t = \frac{2}{3k}\|x\|$ دارد. پس از آنجا که حداکثر $3k$ اندیس i موجود هستند به طوری که $|x_i| \geq \frac{1}{3k}\|x\|$، تعداد اندیس‌ها با مقدار گزارش‌شده‌ی بزرگترمساوی t هم حداکثر $3k$ است. پس تمامی این اعداد در S ظاهر می‌شوند. حالا در نظر داشته باشید که به ازای هر i که $|x_i| \geq \frac{1}{k}\|x\|$، باز هم طبق خواص الگوریتم $\mathcal{A}$، مقدار گزارش شده حداقل t است. پس تمام این اعداد در S قرار می‌گیرند. که خاصیت دوم مجموعه و در نتیجه درستی الگوریتم را به اثبات می‌رساند. □

۳ الگوریتم COUNTMIN: پرسش نقطه‌ای برای شمارنده‌ی مثبت

به صورت کلی یک الگوریتم ارائه می‌دهیم که با احتمال ثابت پاسخ درست می‌دهد. با ترکیب خروجی اجراهای موازی این الگوریتم، یک پاسخ بدست می‌آوریم که به احتمال دلخواه درست است. ماتریس $C \in \mathbb{R}^{L\times B}$ که در ابتدا تمام صفر است در نظر بگیرید (هر سطر مستقل از بقیه سطرها است و قرار است به احتمال ثابت پاسخ درست دهد). در اینجا $B = 2k$ و $L = \lceil \log(1/\delta) \rceil$ همچنین برای هر سطر a یک تابع درهم‌سازی ۲ـطرفه h_a از $[n]$ به $[B]$ در نظر بگیرید، به طوری که تابع‌های سطرهای متفاوت از هم کاملاً مستقل باشند. حالا به ازای تغییر $x_i \leftarrow x_i + \Delta$ برای هر سطر a مؤلفه‌ی $h_a(i)$ـاُم را با Δ جمع می‌کنیم. در نهایت وقتی مقدار x_i درخواست می‌شود مقدار $\min_a C_{ah_a(i)}$ را گزارش می‌دهیم.

۱.۳ اثبات درستی

برای گزارش مقدار x_i ابتدا مقدار $C_{ah_a(i)}$ را برای یک سطر بررسی می‌کنیم. این مقدار برابر است با جمع تمام مؤلفه‌هایی از x که توسط تابع h_a به خانه‌ی $C_{ah_a(i)}$ وصل می‌شوند. به این ترتیب اگر Z_j متغیر تصادفی مشخصه‌ی این پدیده در نظر بگیرید که $h_a(j) = h_a(i)$ آنگاه داریم:

$$C_{ah_a(i)} = x_i + \underbrace{\sum_{j\neq i} Z_j x_j}_{E}$$

در اینجا E مقدار خطای محاسبه‌ی x_i و همواره یک عدد نامنفی است. امیدریاضی آن را حساب می‌کنیم:

$$\begin{aligned}\mathbb{E}[E] &= \sum_{j\neq i} \mathbb{E}[Z_j] x_j \\ &\leq \frac{1}{B}\sum_{j\neq i} x_j \\ &\leq \frac{1}{B}\|x\|_1\end{aligned}$$

[5] union bound

که در اینجا خط دوم از ۲ــطرفه بودن تابع درهم‌سازی نتیجه می‌شود. حالا با استفاده از نامساوی مارکوف یک کران بالا برای احتمال اینکه خطا بیش از حد مجاز باشد ارائه می‌دهیم:

$$\mathbb{P}r[E > \|x_j\|/k] \leq \frac{\mathbb{E}[E]}{\|x_j\|/k} \leq \frac{k}{B} = \frac{۱}{۲}$$

به این ترتیب هر $C_a h_a(i)$ به ازای هر سطر a با احتمال $\frac{۱}{۲}$ یک تقریب قابل قبول است. از آنجا که خطاها همیشه مثبت هستند، مقدار $\min_a C_a h_a(i)$ در صورتی غیر قابل قبول است که همه‌ی سطرها غیر قابل قبول باشند. این اتفاق به احتمال کمتر از $۲^{-L} = \delta$ رخ می‌دهد. که درستی الگوریتم را اثبات می‌کند.

۲.۳ تحلیل زمان اجرا و حافظه

برای هر تغییر یا هر پرسش نقطه‌ای، باید عدد ورودی را بخوانیم و یک مؤلفه از هر سطر را تغییر دهیم یا بررسی کنیم. زمان مربوط به این کار $\mathcal{O}(\log(n)) + \mathcal{O}(\log(۱/\delta)) = \mathcal{O}(log(n/\delta))$ است. همچنین باید کل ماتریس را همواره نگهداری کنیم که حافظه‌ی مصرفی از مرتبه‌ی $\mathcal{O}(k\log(۱/\delta))$ دارد. موقع پاسخ دادن پرسش‌ها باید k اندیس را نگهداری کنیم که حافظه‌ی $\mathcal{O}(k\log(n))$ نیاز دارد. پس در کل حافظه‌ی مصرفی $\mathcal{O}(k\log(n/\delta))$ است.

نکته: توجه کنید حافظه‌ی لازم برای نگهداری توابع درهم‌سازی بررسی نشده است و در بخش‌های بعد هم بررسی نمی‌شود.

۳.۳ پرسش وزین‌ها

طبق قضیه ؟؟ با همین حافظه، زمان $\mathcal{O}(\log(n/\delta))$ برای تغییر و زمان $\mathcal{O}(n\log(n/\delta))$ برای پرسش‌ها، می‌توان پرسش وزین‌ها را حل کرد. زمان اجرای مربوط به پرسش‌ها خیلی زیاد است که در بخش بعد به آن می‌پردازیم.

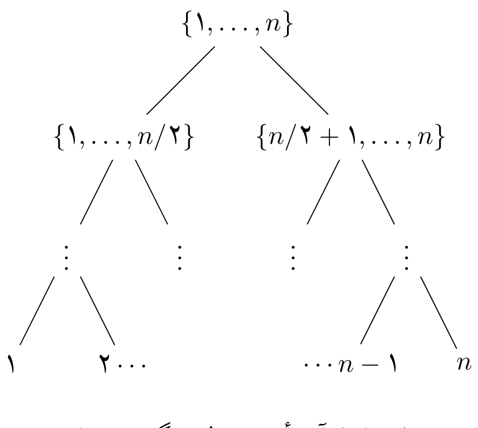
۴.۳ دلیل نام‌گذاری فشرده‌سازی خطی

دلیل این نام‌گذاری به این خاطر است که هر عدد که در ماتریس نگه می‌داریم، یک ترکیب خطی از اعداد اصلی است. به صورت کلی در این روش‌ها یک ماتریس تصادفی A داریم و به جای ذخیره کردن x مقدار $y = Ax$ را نگه‌داری می‌کنیم. برای تغییر $x \leftarrow x + e_i$ کافی است قرار درهم‌سازی $y \leftarrow y + Ae_i$. پاسخ پرسش‌ها به شیوه‌ای از روی y بدست می‌آید.

برای مثال، اگر یک سطر از ماتریس C در الگوریتم COUNTMIN را در نظر بگیرید، برابر است با Ax که در ستون iــاُم مؤلفه $h(i)$ برابر با ۱ و بقیه مؤلفه‌ها برابر با ۰ است.

۴ پرسش‌وزین‌های سریع‌تر

یک درخت دودویی کامل با $\log(n) + ۱$ طبقه در نظر بگیرید، که هر رأس آن متناظر با یک زیرمجموعه از اندیس‌ها است. به این شکل که هر برگ متناظر با یک اندیس است و هر رأس غیر برگ متناظر با اجتماع دو فرزندش. مثلاً ریشه متناظر با تمام اندیس‌ها است.



عدد متناظر با یک رأس را مجموع x روی اندیس‌های داخل آن رأس در نظر می‌گیریم. به این ترتیب در طبقه jــاُم تعداد ۲^j عدد در نظر گرفته‌ایم. برای هر طبقه به صورت موازی و مستقل یک الگوریتم COUNTMIN در نظر می‌گیریم، که پارامتر آن $۴k$ و احتمال خطای آن $\eta = \delta/۴k\log n$ است. به این ترتیب طبق تحلیل‌های بخش قبل حافظه‌ی استفاده شده از مرتبه‌ی $\mathcal{O}(k\log(۱/\eta)\log(n)) = \mathcal{O}(k\log((k\log n)/\delta)\log n)$ است.

برای تغییرات، به ازای هر طبقه، تغییر را به رأسی که اندیس مورد نظر را در بر می‌گیرد، اضافه می‌کنیم. همچنین یک شمارنده معمولی نگه می‌داریم که مقدار دقیق $\|x\|_۱$ را نگهداری کند.

برای پاسخ دادن پرسش وزین‌ها، دقت کنید که اگر یک اندیس در بردار اصلی وزین باشد (یعنی مقدار بزرگتر از $\|x\|_۱/k$ داشته باشد)، تمام رأس‌های روی مسیر از ریشه تا برگ متناظر با این اندیس وزین خواهند بود. به این ترتیب از طبقه‌ی اول شروع می‌کنیم و برای هر طبقه‌ی j یک مجموعه L_j نگه می‌داریم که وزین‌های آن طبقه را در بر داشته باشد. برای شروع $L_۰$ را برابر با ریشه قرار می‌دهیم. برای L_j، به ازای هر رأس داخل $L_{j-۱}$ هر فرزند آن را بررسی می‌کنیم اگر مقدار گزارش شده از $\frac{۳}{۴k}\|x\|_۱$ بیشتر بود، فرزند را در L_j قرار می‌دهیم. مشابه با استدلال‌هایی که در قضیه ؟؟ آمده بود، از آنجا که COUNTMIN هر طبقه، پارامتر $۴k$ دارد، هر نقطه‌ی وزین مقدار گزارش‌شده‌ی بزرگتر از $\frac{۳}{۴k}\|x\|_۱$ دارد. پس طبق خاصیت مسیر ریشه تا برگ که گفته شد، در L_j خواهد آمد. همچنین هر مؤلفه‌ای که مقدار گزارش شده‌اش بیش از $\frac{۳}{۴k}\|x\|_۱$ باشد، مقدار واقعی بیش از $\frac{۱}{۲k}\|x\|_۱$ دارد. پس تعداد رأس‌های L_j در هر طبقه حداکثر $۲k$. توجه کنید اگر پرسش‌های نقطه‌ای همگی به درستی پاسخ داده شوند(یعنی خطایشان قابل قبول داشته باشد)، مجموعه‌ی L_j به ازای هر طبقه، طبق استدلال‌های بالا، تمام وزین‌های طبقه را در بر می‌گیرد و مجموعه نهایی هم تمام اندیس‌های وزین را خواهد داشت. همچنین با فرض درستی پرسش‌های نقطه‌ای در هر طبقه کمتر از $۴k$ پرسش می‌کنیم و در کل $Q = ۴k \log n$ پرسش می‌کنیم (اگر تعداد پرسش‌هایمان بیش از این شد به این معنا است که یک پاسخ نقطه‌ای اشتباه بوده و می‌توانیم اعلام کنیم که پاسخ را نتوانستیم پیدا کنیم). به این ترتیب این داده ساختار طبق کران اجتماع به احتمال کمتر از $Q\eta = \delta$، پاسخ درست می‌دهد که این اثبات درستی الگوریتم را تکمیل می‌کند.

همانطور که گفته شد کمتر از $۴k \log n$ پرسش انجام می‌دهیم، پس زمان لازم برای پاسخ دادن پرسش از مرتبه‌ی $\mathcal{O}(k \log(۱/\eta) \log(n))$ برابر با $\mathcal{O}(k \log((k \log n)/\delta) \log n)$ است. همچنین زمان لازم برای تغییر $\mathcal{O}(\log n \log(۱/\eta)) = \mathcal{O}(\log n \log((k \log n)/\delta))$

۵ الگوریتم COUNTSKETCH: پرسش نقطه‌ای برای شمارنده‌ی آزاد و نُرم $\ell_۲$

۱.۵ تعاریف مربوط به $\ell_۲$

ابتدا در مورد نُرم ۲، $\ell_۲$، تغییراتی در تعریف ایجاد می‌کنیم. الگوریتم با پارامتر k، برای پرسش نقطه‌ای باید خطای کمتر از $\|x\|_۲/\sqrt{k}$ داشته باشد. همچنین برای پرسش وزین‌ها، مجموعه‌ی خروجی باید $\mathcal{O}(k)$ عضو داشته باشد و باید تمام مؤلفه‌هایی که قدرمطلقشان از $\|x\|_۲/\sqrt{k}$ بزرگتر است داخل مجموعه قرار باشند.

۲.۵ شرح الگوریتم

الگوریتم COUNTSKETCH شبیه به الگوریتم COUNTMIN است. با کمی تفاوت؛ اول این که به ازای هر سطر به جز تابع درهم‌سازی که هر اندیس از x را به یک اندیس از سطر می‌بَرَد، یک تابع در هم‌ساز ۲ــ طرفه σ در نظر می‌گیریم که به هر اندیس x عدد ± ۱ را اختصاص می‌دهیم. حالا به ازای هر تغییر $x_i \leftarrow x_i + \Delta$ قرار می‌دهیم، $C_{ah_a(i)} \leftarrow C_{ah_a(i)} + \sigma_a(i)\Delta$. تفاوت دوم هم این است که برای پرسش نقطه‌ای i، $\mathrm{median}_a\ \sigma_a(i) C_{ah_a(i)}$ را گزارش می‌دهیم. همچنین تعداد ستون‌ها $۹k$ قرار داده می‌شود.

۳.۵ اثبات درستی

ابتدا $C_{ah_a(i)}$ به ازای یک سطر تحلیل می‌کنیم. برای تحلیل پرسش نقطه‌ای i، طبق تقارن بدون از دست رفتن کلیت مسئله، فرض می‌کنیم $\sigma(i) = ۱$. مشابه تحلیل الگوریتم COUNTMIN متغیر تصادفی Z_j را متغیر مشخصه این پدیده تعریف می‌کنیم که $h(i) = h(j)$. به این ترتیب داریم:

$$C_{ah_a(i)} = x_i + \underbrace{\sum_{j \neq i} \sigma_a(j) Z_j x_j}_{E}$$

به صورت شهودی مؤلفه‌هایی که σ مثبت دارند، به احتمال مناسبی، مؤلفه‌هایی که σ منفی دارند را «خنثی» می‌کنند. برای اثبات دقیق در نظر داشته باشید که:

$$\mathbb{E}[E] = \sum_{j \neq i} \mathbb{E}[\sigma(j) Z_j x_j] = \sum \mathbb{E}[Z_j x_j \mid \sigma_j] \underbrace{\mathbb{E}[\sigma_j]}_{۰} = ۰$$

حالا کافی است یک کران بالا برای $\mathbb{E}[E^۲]$ ارائه دهیم، سپس با نامساوی مارکوف نتیجه می‌شود که E به احتمال ثابتی کوچک است:

$$\begin{aligned}
\mathbb{E}[E^۲] &= \mathbb{E}\Big[(\sum_{j\neq i}\sigma(j)Z_jx_j)^۲\Big] \\
&= \mathbb{E}\Big[\sum_{j\neq i}Z_jx_j^۲ + \sum_{j,j'\neq i}\sigma(j)Z_jx_j\sigma(j')Z'_jx'_j\Big] \\
&= \sum_{j\neq i}\mathbb{E}[Z_j]x_j^۲ + \sum_{j,j'\neq i}\underbrace{\mathbb{E}[\sigma(j)]}_{۰}\mathbb{E}[\sigma(j')]\mathbb{E}[Z_jx_jZ'_jx'_j]\Big] && \text{(طبق ۲ ـ طرفه بودن } \sigma\text{)} \\
&= \frac{۱}{۹}\frac{\|x\|_۲^۲}{k}
\end{aligned}$$

حالا طبق نامساوی ینسن داریم:

$$\mathbb{E}[|E|] \leq \sqrt{\mathbb{E}[|E|_۲^۲]} = \frac{۱}{۳}\frac{\|x\|_۲}{\sqrt{k}}$$

و طبق نامساوی مارکوف داریم:

$$\mathbb{P}r[|E| > \frac{\|x\|_۲}{\sqrt{k}}] \leq \frac{۱}{۳}$$

حالا با اجرای موازی $L = \Theta(\log(۱/\delta))$ بار از این الگوریتم، (که در واقع سطرهای ماتریس هستند) و میانه‌گیری پاسخ‌ها نتیجه می‌شود که

$$\mathbb{P}r[|\text{median}_a\ \sigma_a(i)C_{ah_a(i)} - x_i| > \frac{\|x\|_۲}{\sqrt{k}}] < \delta$$

که نتیجه می‌دهد الگوریتم درست کار می‌کند.

۴.۵　تحلیل زمان اجرا و حافظه

مشابه الگوریتم COUNTMIN زمان اجرای لازم برای اعمال تغییر و پرسش نقطه‌ای از مرتبه‌ی $\mathcal{O}(\log(n/\delta))$ است. همچنین زمان لازم برای پرسش وزین‌ها $\mathcal{O}(k\log(n/\delta))$ است. حافظه‌ی لازم هم برای دو نوع پرسش به ترتیب $\mathcal{O}(k\log(۱/\delta))$ و $\mathcal{O}(k\log(n/\delta))$ است.

۵.۵　منابع و ارجاع‌ها

جزوه‌ی درس فشرده‌سازی آقای نلسون [?].
مقاله‌ی مربوط به COUNTMIN و حالت سریع‌تر آن [?].
مقاله‌ی مربوط به COUNTSKETCH [?].