دانشکده علوم ریاضی
دانشگاه صنعتی شریف

الگوریتم‌های خلاصه‌سازی برای مه‌داده

محمدهادی فروغمنداعرابی
پاییز ۱۳۹۹

الگوریتم‌های جویباری برای مسائل هندسی

جلسه‌ی بیست و یکم و بیست و دوم

نگارنده: شایان رنجبرزاده

۱ مروری بر مباحث گذشته

با استفاده از ایده‌هایی که در گذشته دیدیم به بیان الگوریتم‌های جویباری برای چند مسأله هندسی معروف می‌پردازیم. در این مسائل جریانی از نقاط داریم که حذف و اضافه می‌شوند و هدف ما یافتن تخمینگرهای مناسب است. برای سادگی فرض می‌کنیم که مختصات نقاط ورودی کراندار است یعنی : $p \in [\Delta]^۲$

۲ تخمین قطر

هدف ما یافتن تخمینگر D' برای بیشترین فاصله با نرم $L_۱$ روی مجموعه نقاط P هست به طوری که اگر تعریف کنیم

$$D := max_{p,p' \in P} D(p,p')$$

داشته باشیم :

$$D \leq D' \leq (۱ + \theta(\epsilon))D$$

۱.۲ الگوریتمی برای جریان های درجی

در این حالت فرض می‌کنیم تنها عملیاتی که داریم اضافه کردن نقطه است و از حذف نقاط خبری نیست. به سادگی یک الگوریتم دو تقریب می‌دهیم. کافیست بیشترین فاصله نقاط تا نقطه اول را در D' نگه داریم و هر وقت بیشترین فاصله از ما خواسته شد همان را خروجی می‌دهیم. پس D' را

برای شروع ۰ می‌گذاریم و وقتی نقطه i اضافه می‌شود D' اینگونه به‌روزرسانی می‌شود.

$$D' = max(D(p_1, p_i), D')$$

چون در حافظه فقط نقطه اول را نگه می‌داریم حافظه این الگوریتم از $o(1)$ است. حال ادعا می‌کنیم یک ۲ـ تقریب به ما می‌دهد.

اثبات. فرض کنید بیشترین فاصله بین دو نقطه i و j باشد. طبق نامساوی مثلث داریم :

$$D = D(p_i, p_j) \leq D(p_i, p_1) + D(p_1, p_j) \leq 2D'$$

همچنین

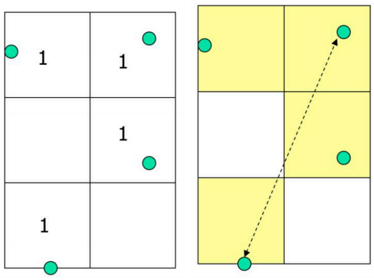
$$D' \leq D$$

زیرا مجموعه‌ای که D روی آن ماکسیمم می‌گیرد شامل مجموعه‌ای است که D' روی آن ماکسیمم می‌گیرد. □

حال به مسأله اصلی برمی‌گردیم.

۲.۲ ایده

ابتدا اندازه مناسبی برای شبکه‌بندی صفحه پیدا می‌کنیم طوری که آنقدر زیاد نباشد که خطای زیادی ایجاد کند و آنقدر کم نباشد که تعداد خانه‌های پر زیاد شود. سپس با استفاده از ساختمان داده بازیابی k تنک خانه‌های پر را پیدا می‌کنیم و بیشینه فاصله میان مرکز آن‌ها را گزارش می‌کنیم.



۳.۲ الگوریتم

```
Algorithm 1 Diameter approximation
n_p^i(c) := |{p | p ∈ c ∧ p ∈ P}| , ∀c ∈ G_i, ∀i ∈ [m]
function APPROXIMATED(P)
    G_0, ..., G_m ← square grids with diameter (1+ε)^0, (1+ε)^1, (1+ε)^2, ..., 2Δ
    for p ∈ P do
        Maintain linear sketch of n_P^i, ∀i ∈ [m]
    i* ← min_{i∈[m]}{i} such that ||n_P^i||_0 ≤ k = O(1/ε^2)      ▷ ||n_P^i||_0 from linear sketch
    Recover the set S of non-zero cells in n_p^{i*}                ▷ using k-sparse recovery of a vector
    return (1+ε)^{i*} D(S)                    ▷ D(S) is the diameter of the set S (grid coordinates)
```

چندین شبکه بندی مختلف در نظر می‌گیریم. یکی به ضلع ۱ یکی به ضلع $(1+\epsilon)$ یکی به ضلع $(1+\epsilon)^2$ تا یکی به ضلع Δ. همه آن‌ها را هم با حذف و اضافه نقاط به‌روز می‌کنیم. برای هر شبکه‌بندی یک ساختمان داده بازیابی k تنک و نرم صفر آن را (که همان تعداد خانه های پر آن است) نیز نگه می‌داریم. هربار که خواستیم بیشینه فاصله بین نقاط را محاسبه کنیم از بزرگترین شبکه‌بندی شروع می‌کنیم، اخرین شبکه‌بندی‌ای که تعداد خانه‌های پر آن کمتر از k بود را در نظر می‌گیریم و فاصله بیشینه بین مرکز خانه‌های پر آن را گزارش می‌کنیم (k همان کرانی است که برای ساختمان داده‌مان داریم؛ اگر خانه‌های پر کمتر از آن باشند ساختمان داده ما جواب درس می‌دهد؛ یعنی جای تمام خانه‌های پر را می‌دهد وگرنه جواب درستی نمی‌دهد).

اثبات. k را $\theta(1/\epsilon^2)$ می‌گذاریم. حال شبکه‌بندی به ضلع $(1+\epsilon)^i$ را در نظر بگیرید به طوری که

$$(1+\epsilon)^i \leq \epsilon D \leq (1+\epsilon)^{i+1}$$

چون برای بیشینه فاصله بین نقاط کران داریم برای تعداد خانه‌های پر در شبکه‌بندی i هم کران خواهیم داشت.

$$D \leq \frac{1}{\epsilon}(1+\epsilon)^{i+1} \Rightarrow D \leq \frac{1+\epsilon}{\epsilon}(1+\epsilon)^{i}$$

و طول ضلع هر خانه $(1+\epsilon)^i$ است پس دو خانه پر با فاصله بیشینه به اندازه $\frac{2}{\epsilon} \leq \frac{1+\epsilon}{\epsilon}$ تا خانه در شبکه‌بندی i فاصله خواهند داشت و اگر کل مربعی که قطرش این دو خانه هستند پر باشد، باز هم تعداد خانه‌های پر کران بالا از $\theta(1/\epsilon^2)$ دارد که چون k را همین مقدار قرار داده بودیم ساختمان داده این شبکه‌بندی خانه‌های پر را درست خروجی می‌دهد. خطای جواب الگوریتم ما حداکثر به اندازه قطر یک خانه خواهد بود که می‌شود

$$2(1+\epsilon)^i \leq 2\epsilon D = \theta(\epsilon D)$$

و این اثبات درستی الگوریتم را به پایان می‌رساند. □

۴.۲ تحلیل حافظه

تعداد شبکه‌بندی‌هایی که نگه داشتیم برابر

$$\log_{1+\epsilon} \Delta = \frac{\log \Delta}{\log(1+\epsilon)}$$

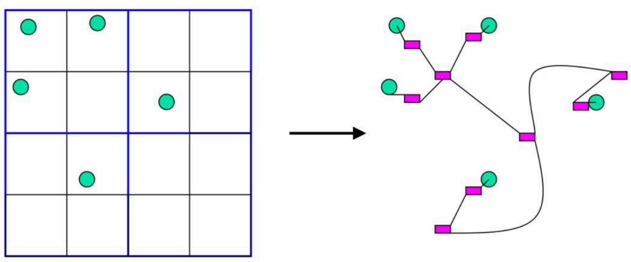
بود و هر کدام یک بازیابی k تنک هستند که حافظه $\mathcal{O}(\frac{1}{\epsilon^2}polylog(n))$ را می‌گیرند که در آن n تعداد به‌روزرسانی‌هاست(مجموع تعداد حذف‌ها و درج‌ها). پس در کل حافظه $\mathcal{O}(\frac{1}{\epsilon^2}polylog(n+\Delta))$ می‌شود.

۳ درخت فراگیر کمینه

قبل از آنکه به بیان تخمینگری برای مسأله درخت فراگیر کمینه بپردازیم ساختمان داده‌ای را معرفی می‌کنیم که هدفش این است که درختی بسازد که فواصل را تقریبا حفظ کند.

۱.۳ درخت چهارتایی

شبکه‌بندی زیر را که $[\Delta]^2$ را به مربع هایی به ضلع‌های $\Delta, \ldots, 2^0, 2^{-1}$ افراز می‌کند در نظر بگیرید. مربع اولیه به ضلع Δ را به ۴ قسمت برابر تقسیم می‌کنیم، هر ناحیه ناتهی ایجاد شده را به ۴ مربع مساوی تقسیم می‌کنیم و همین روند را دامه می‌دهیم تا به مربع هایی شامل یک نقطه برسیم. حال از روی این شبکه‌بندی یک درخت می‌سازیم. به این صورت که به هر مربعی که به ۴ قسمت تقسیم می‌شود یک ررس در نظر می‌گیریم و به رأس های مربع‌های درونش در صورت وجود وصل می‌کنیم و وزن این یال را برابر طول ضلع مربع قرار می‌دهیم و در انتها هر مربع شامل یک نقطه را به نقطه‌ای که درونش هست وصل می‌کنیم.



تذکر ۱. *طبق فرض* $p \in [\Delta]^2$ *ارتفاع درخت کران بالای* $\mathcal{O}(\log \Delta)$ *دارد.*

حال فرض کنید شبکه‌بندی اولیه را به صورت تصادفی و کاملا یکنواخت شیفت بدهیم (ابتدا شبکه بندی را به کل صفحه تعمیم دهید)،

قضیه ۲. *اگر فاصله دو نقطه p،q در درخت را تعریف کنیم* $D_T(p,q)$ *خواهیم داشت :*

$1. \|p-q\|_1 \leq D_T(p,q)$
$2. E[D_T(p,q)] \leq \|p-q\|_1 O(log(\Delta))]$

اثبات. کوچکترین مربع شامل p،q را در نظر بگیرید؛ فرض کنید طول ضلع آن ۲^i باشد. چون هر دو نقطه درون این مربع هستند فاصله $L_۱$ آن‌ها کراندار است.

$$\Rightarrow \|p-q\|_۱ \leq ۲ \times ۲^i$$

همچنین فاصله آن ها در دخت برابر است با فاصله p تا جد مشترک به علاوه فاصله q تا جد مشترک که چون مربع به ضلع ۲^i کوچکترین مربع شامل آن ها بود داریم :

$$\Rightarrow D_T(p,q) = (۱+۱+۲+...+۲^{i-۱}) + (۱+۱+۲+...+۲^{i-۱}) = ۲ \times ۲^i$$

که حکم بخش اول قضیه را نتیجه می‌دهد. یک دسته از خانه‌های شبکه‌بندی به ضلع ۲^i مثل $c = (c_x, c_y)$ را در نظر بگیرید. حال احتمال اینکه دو نقطه $p = (p_x, p_y)$ و $q = (q_x, q_y)$ در یک خانه از این دسته قرار نگیرند را $P_i = P(p \in c \wedge q \in c' \neq c)$ می‌نامیم. اگر $|p_x - q_x| \leq ۲^i$ باشد؛ احتمال اینکه p،q بخاطر مؤلفه اول توسط c_x جدا شوند برابر $\frac{|p_x - q_x|}{۲^i}$ و در غیر این صورت یک است. پس در کل برابر

$$P(p_x \in c_x \wedge q_x \in c'_x \neq c_x) = min\{۱, \frac{|p_x - q_x|}{۲^i}\} \leq \frac{|p_x - q_x|}{۲^i}$$

و به طور مشابه برای مؤلفه y داریم

$$P(p_y \in c_y \wedge q_y \in c'_y \neq c_y) = min\{۱, \frac{|p_y - q_y|}{۲^i}\} \leq \frac{|p_y - q_y|}{۲^i}$$

برای احتمال جدا شدن p،q طبق کران اجتماع داریم :

$$P_i \leq P(p_x \in c_x \wedge q_x \in c'_x \neq c_x) + P(p_y \in c_y \wedge q_y \in c'_y \neq c_y) \leq \frac{|p_x - q_x|}{۲^i} + \frac{|p_y - q_y|}{۲^i} = \frac{\|p-q\|_۱}{۲^i}$$

$$\Rightarrow E[D_T(p,q)] = \Sigma_{i=۰}^{\Theta(log(\Delta))} P_i \times ۲^i \leq \Sigma_{i=۰}^{\Theta(log(\Delta))} \frac{\|p-q\|_۱}{۲^i} \times ۲^i = \Sigma_{i=۰}^{\Theta(log(\Delta))} \|p-q\|_۱ = \|p-q\|_۱ \mathcal{O}(log(\Delta))]$$

که همان حکم بخش دوم قضیه است. □

۲.۳ تخمینگر درخت فراگیر کمینه

اگر تمام نقاط را به هم وصل کنیم و یک گراف کامل بسازیم که وزن هر یال به اندازه فاصله $L_۱$ دو سرش باشد به دنبال تخمینگری برای زیردرخت فراگیر این گراف با وزن کمینه هستیم که آن را با T' نشان می‌دهیم. حال از درختی که از روی ساختمان داده درخت چهارتایی ساخته شد استفاده می‌کنیم. فرض کنید کوچکترین مربعی که تمام نقاط را شامل می‌شود ضلع ۲^L داشته باشد. حال تخمینگری که خروجی می‌دهیم زیردرختی است که رأس متناظر با این مربع و تمام فرزندانش را شامل است. اسم آن را درخت T'' می‌گذاریم. طبق تعریف L این زیردرخت فراگیر هست. ادعا می‌کنیم :

$$cost(T') \leq ۲E[cost(T'')] \leq O(log(\Delta))cost(T')$$

برای هر یال p،q در T' اگر c_p را بزرگترین مربعی بنامیم که شامل p هست و شامل q نیست و c_q را هم مشابه تعریف کنیم خواهیم داشت :

$$\|p-q\|_۱ \leq ۲D_T(c_p, c_q)$$

حال اگر این نامساوی را روی تمام یال‌های T' جمع بزنیم؛ به نامساوی زیر می‌رسیم :

$$cost(T') \leq ۲cost(T'')$$

طبق بخش دوم قضیه قبل داریم $E[D_T(p,q) \leq \|p-q\|_۱ \mathcal{O}(log(\Delta))]$. حال این رابطه را روی تمام یال های T' جمع می‌زنیم. سمت راست آن طبق تعریف برابر $\mathcal{O}(log(\Delta))cost(T')$ می‌شود. سمت چپ آن هم هر یال T'' را می‌پوشاند(چون در درخت تمام یال ها برشی هستند برای هر یالی دو رأس وجود دارند که این یال در مسیر یکتای بین آن دو رأس ظاهر شود) و ممکن است بعضی یال‌های T'' را چند بار حساب کند که نتیجه می‌دهد :

$$E[cost(T'')] \leq \mathcal{O}(log(\Delta))cost(T') \Rightarrow ۲E[cost(T'')] \leq \mathcal{O}(log(\Delta))cost(T')$$

حال به محاسبه $cost(T'')$ می‌پردازیم. $n_P^i(c)$ را تعداد نقاطی که در خانه c از سطح i شبکه‌بندی هستند تعریف می‌کنیم و برای هر i با یک خلاصه‌سازی خطی $\|n_P^i\|_0$ را در حافظه نگه می‌داریم. حال داریم :

$$E[cost(T'')] = \Sigma_{i=0}^{L-1} 2^i \Sigma_{c\in G_i}[n_P^i(c) < 0] = \Sigma_{i=0}^{L-1} 2^i \|n_P^i\|_0$$

زیرا اگر درخت را از ریشه آویزان کنیم وزن یال‌های وارد شده به طبقه i برابر 2^i است و تعداد این یال‌ها همان تعداد مربع‌های پر در سطح i شبکه‌بندی است که می‌شود $\|n_P^i\|_0$. حال اگر روی تمام طبقات تعداد یال‌ها ضرب در وزن هر کدام را جمع ببندیم وزن کل درخت را به ما می‌دهد. حال طبق ادعایی که ثابت کردیم $2E[cost(T'')]$ یک تخمینگر مناسب برای مسأله است پس الگوریتم ما $2\Sigma_{i=0}^{L-1} 2^i \|n_P^i\|_0$ را خروجی می‌دهد.

مراجع

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

[Ind04] Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*, pages 373–380, 2004.