COMS E6998-15: Algorithms for Massive Data Sep 20, 2023

Lecture 5: Dynamic connectivity, Least Square Regression

Instructor: *Alex Andoni* Scribe: *Yufei Guo*

1 Recap

Dynamic sampling

Using dynamic sampling, in the generalized turnstile model, given initial vector $X \in \mathbb{Z}^n$ and updates to it, we can provide a linear random sketch of X. And at the end of the stream, we can output a random coordinate S such that $\Pr\left[S = i\right] = (1 \pm \varepsilon) \cdot \dfrac{\mathbb{1}\left[X_i \neq 0\right]}{\|X\|_0} \pm \dfrac{1}{n^3}$. Where $\|X\|_0$ is the support of vector X.

We can obtain achieve such goal using $\mathcal{O}\left(\log^2 n\right)$ space (with high probablity). We will be using this tool to solve dynamic connectivity problem.

2 Dynamic connectivity

Setting

Compared to previous setting of normal graph streaming, in this new dynamic graph stream setting, the stream could contain insertion and deletion of edges (as opposed to only have insertions).

Problem

We wish to determine the connectivity and/or spanning forest in the graph using $\mathcal{O}\left(n \cdot (\log n)^{\mathcal{O}(1)}\right)$ space.

Borůvka's algorithm

Before designing an online algorithm, we first introduce a offline greedy algorithm that also allows parallelization.

Algorithm 1: Borůvka's algorithm

```
1 Keep connected components within disconnected graph initialized as $\{1\}, \{2\}, \cdots, \{n\}$.;
2 for $t = 1, \cdots, \mathcal{O}(\log n)$ do
3     For each connected component $Q \subseteq [n]$, take any edge that cross $Q$. A edge that cross $Q$ is
        simply an edge leaves $Q$.;
4     Merge and compute the new connected components.;
```

Correctness of Borůvka's algorithm directly comes from its procedure.

Proof. Let CC_j be the number of connected components after j iterations, and let CC be the total number of connected components.
Since at each iteration we find an edge that leaves every connected component and merge them, each iteration we decrease the number of connected components by a half and $\left(CC_{j+1} - CC\right) \leq \frac{1}{2}\left(CC_j - CC\right)$.
Since the maximum number of connected components is n, and we have $t = \mathcal{O}(\log n)$ iterations, the algorithm is guaranteed to terminate. $\square$

Definition 1 (node-edge incidence vector). *$\forall v \in V$, we keep a vector $X_v \in \mathbb{R}^{\binom{n}{2}}$, where $\binom{n}{2}$ is the total number of vertex pair.*

Let $X_v(v,i) = \begin{cases} 1 & \text{if } (v,j) \in E, v < j \\ -1 & \text{if } (j,v) \in E, j < v \\ 0 & \text{otherwise} \end{cases}$

In other words, edges that goes out of v are marked 1 in X_v, where edges goes to v are marked -1. And edges goes from small index to large index.

The original graph is undirected, but such ”direction” in edges will help us in the future.

Dynamic connectivity algorithm

Streaming phase

In streaming phase we keep a sketch of what we've met in the stream, which we will use later to deduce the connectivity of the graph.

- For each $v \in V$, keep a dynamic sampling of X_v called $DS(X_v)$. Note that all X_v uses the same randomness, or same matrix A to sketch the vector X_v.
- At insertion of edge (i,j), we will need to update X_i and X_j, WLOG, assume $i < j$.
- $DS(X_i') = DS\left(X_i + e_{i,j}\right) = DS(X_i) + DS\left(e_{i,j}\right)$, where the second step comes from the linearity of dynamic sampling.
- Similarly, $DS\left(X_j'\right) = DS\left(X_j\right) - DS\left(e_{i,j}\right)$.
- Upon deletion, simply reverse the plus and minus sign.

Connectivity algorithm using sketches

This algorithm acts similarly as Borůvka's algorithm, except some minor differences on line 3.

Algorithm 2: Dynamic connectivity algorithm using sketches

```
1 Keep connected components within disconnected graph initialized as $\{1\}, \{2\}, \cdots, \{n\}$.;
2 for $t = 1, \cdots, \mathcal{O}(\log n)$ do
3     For each connected component $Q \subseteq [n]$, sample an edge that cross $Q$ using the sketches.;
4     Merge and compute the new connected components.;
```

One immediate question is, how do we do line 3, sample a random edge that cross Q?

If $Q = \{v\}$, then it's simple, we can use $DS\left(X_v\right)$ directly since the goal of dynamic sampling is to provide an random non-zero coordinate, and the non-zero coordinates in X_v are exactly edges that enter/leaves v.

Definition 2. *Take any connected $Q \subseteq [n]$, $X_Q \triangleq Q$-edge incidence vector. Defined as*

$$X_Q\left(i,j\right) = \begin{cases} 1 & \text{if } i \in Q, v \notin Q, (i,j) \in E, i < j \\ -1 & \text{if } i \in Q, v \notin Q, (j,i) \in E, j < i \\ 0 & \text{otherwise} \end{cases}$$

Definition for X_Q is very similar to vertex edge incidence vector, but in a more "combined way", since Q is a set of vertices, this intuition gives an easier way to calculate X_Q.

Claim 3. $X_Q = \sum_{v \in Q} X_v$

Proof. Consider $X_Q\left(i,j\right)$, WLOG, assume $i < j$.

$$X_Q\left(i,j\right) = \begin{cases} X_i\left(i,j\right) + X_j\left(i,j\right) = 1 + (-1) = 0 & \text{if } i \in Q, j \in Q \\ X_i\left(i,j\right) = 1 & \text{if } i \in Q, j \notin Q \\ X_j\left(i,j\right) = -1 & \text{if } i \notin Q, j \in Q \\ 0 & \text{if } i \notin Q, j \notin Q \end{cases}$$

□

By construction, X_Q is non-zero on coordinates where edges cross Q. And to sample an edge that cross Q, we can calculate $DS\left(X_Q\right) = DS\left(\sum_{v \in Q} X_v\right) = \sum_{v \in Q} DS\left(X_v\right)$, by linearity of DS.
However, one immediate issue is, recall all $DS\left(X_v\right)$ uses the same randomness (matrix A) to allow direct sum, for iteration $t \geq 2$, DS are not independent, since they are calculated using DS from previous iteration.
One easy fix is to keep $t = \mathcal{O}\left(\log n\right)$ sketches for each X_v, namely $DS_1\left(X_v\right), \cdots, DS_t\left(X_v\right)$. And at iteration k, just use $DS_k\left(X_v\right)$.

Total space

Space used is for each vertex, the number of sketches we keep and space for each DS.
Thus $n \cdot \mathcal{O}\left(\log n\right) \cdot \mathcal{O}\left(\log^2 n\right) = \mathcal{O}\left(n \log^3 n\right)$.

Necessity for keeping $t = \mathcal{O}\left(\log n\right)$ copies

Suppose we want to query a DS k times and get k samples and that $k \gg \log n$. If we consider the support of X is about k, doing k sampling will almost recover all of X. Thus it must take k space to recover at least k samples. Since one cannot compress a random vector of dimension n to a much smaller dimension.

Recent algorithms

[KKM13] provides an $\mathcal{O}\left(\log^{\mathcal{O}(1)} n\right)$ update/query time algorithm. Also check [AGM12].

3 Numerical Linear Algebra

Problem: Least Square Regression

Definition 4 (Least Square Regression)**.** *Given A be a $n \times d$ matrix, b be a $n \times 1$ column vector, find x^* such that $x^* = \underset{x \in \mathbb{R}^d}{\arg\min} \|Ax - b\|_2$.*

Think A as n datapoints in dimension d, and $d \ll n$. The closed form solution for x^* is simple when $\nabla x \|Ax - b\|_2 = 0$. Which let $x^* = \left(A^T A\right) - 1 A^T b$.
Doing direct calculation for x^* takes $\mathcal{O}\left(n^2 d + d^3\right)$, using fast matrix multiplication, it can be reduced to $\mathcal{O}\left(n d^{\omega - 1} + d^{\omega}\right)$, where ω is the fast matrix multiplication exponent (using Strassen's or Coppersmith's). One immediate question is, if we allow some approximation, how can we do this faster.

Tool: Dimension reduction

Theorem 5 (Johnson-Lindenstrauss Lemma [JL84] (Distributive JL))**.**
$\forall \varepsilon > 0, \forall k \in \mathbb{N}$, $\exists$ *distribution over linear maps* $\varphi : \mathbb{R}^n \to \mathbb{R}^k$ *such that*
$\forall x, y \in \mathbb{R}^n, \Pr_{\varphi}\left[\|\varphi(x) - \varphi(y)\|_2 \in (1 \pm \varepsilon)\|x - y\|_2\right] \geq 1 - e^{-\frac{\varepsilon^2 k}{9}}$

Corollary. *Let $k = 9 \cdot \frac{\log \frac{1}{\delta}}{\varepsilon^2}$, $\Pr\left[JL\ fails\right] \leq \delta$*

Corollary. *For fixed n points, take $k = 27 \cdot \frac{\log N}{\varepsilon^2}$, then for each pair of x, y, JL fails with probability no greater than $\frac{1}{N^3}$. By union bound, $\Pr\left[\text{All pairs are "nice"}\right] \geq \frac{1}{N}$.*

We then provide a standard construction for this linear map for distributive JL. Let $\varphi(x) = \frac{1}{\sqrt{k}} \cdot G \cdot x$, where G_{ij} is from random Gaussian $N(0,1)$. Where G is a $k \times n$ matrix.

Claim 6. *φ satisfy DJL theorem.*

Proof. Fix arbitrary $z = x - y$, then $\varphi(x) - \varphi(y) = \varphi(z)$. Then it's enough to prove for a fixed $z \in \mathbb{R}^n$ $\Pr_{\varphi}\left[\|\varphi(z)\|_2^2 \in (1 \pm \varepsilon)\|z\|_2^2\right] \geq 1 - e^{-\frac{\varepsilon^2 k}{9}}$.

$$\begin{aligned}\|\varphi(z)\|_2^2 &= \frac{1}{k}\sum_{i=1}^{k}\left(\varphi_i(z)\right)^2 \\ &= \frac{1}{k}\sum_{i=1}^{k}\left(G_i \cdot z\right)^2 \quad \text{Where } \varphi_i \text{ is essentially the } i\text{-th row of matrix } G\end{aligned}$$

Now we take a look at $G_i \cdot z = \sum_{j=1}^{n} g_{ij} z_j$

Fact 7 (Stability of Gaussian distribution)**.** $\sum_{i=1}^{n} g_i \cdot z_i \sim \|z\|_2 \cdot g$, *where $g_i, g \sim N(0,1)$*
One easy interpretation of this is the summation of variances make this equality hold.

Now by stability, we have $G_i \cdot z \sim g_i \cdot \|z\|_2$, where $g_i \sim N(0,1)$.

$$\begin{aligned}\|\varphi(z)\|_2^2 &= \frac{1}{k}\sum_{i=1}^{k}\left(G_i \cdot z\right)^2 \\ &= \frac{1}{k}\sum_{i=1}^{k}\left(g_i \cdot \|z\|_2\right)^2 \\ &= \left(\frac{1}{k}\sum_{i=1}^{k}\left(g_i\right)^2\right)\cdot\|z\|_2^2\end{aligned}$$

Then it suffice to prove $\left(\frac{1}{k}\sum_{i=1}^{k}\left(g_i\right)^2\right) \in (1 \pm \varepsilon)$

Fact 8. $\frac{1}{k}\sum_{i=1}^{k}\left(g_i\right)^2$ *is called* χ^2 *distribution with* k *degrees of freedom.*

And it's known that $\Pr\left[\frac{1}{k}\sum_{i=1}^{k}\left(g_i\right)^2 \in (1 \pm \varepsilon)\right] \geq 1 - e^{-\frac{\varepsilon^2 k}{9}}$

□

Approximation for Least square regression

Goal: Find $x' \in \mathbb{R}^d$ such that $\|Ax' - b\|_2 \leq (1+\varepsilon)\|Ax^* - b\|_2$.

Since $x^* = \arg\min_{x\in\mathbb{R}^d}\|Ax - b\|_2$. And we have $\varphi(x) = Sx$ that satisfy DJL, we should have that

$$\begin{aligned}x^* &\approx \arg\min_{x\in\mathbb{R}^d}\|\varphi(Ax-b)\|_2 \\ &= \arg\min_{x\in\mathbb{R}^d}\|\S(Ax-b)\|_2 \\ &= \arg\min_{x\in\mathbb{R}^d}\|SAx - Sb\|_2\end{aligned}$$

Since SA is an $k \times d$ matrix, and Sb is of dimension k, this should provide a speed-up compared to direct calculation.

References

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. “Analyzing Graph Structure via Linear Measurements”. In: *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2012, pp. 459–467. DOI: `10.1137/1.9781611973099.40`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611973099.40`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611973099.40`.

[JL84] William Johnson and Joram Lindenstrauss. “Extensions of Lipschitz maps into a Hilbert space”. In: *Contemporary Mathematics* 26 (Jan. 1984), pp. 189–206. DOI: `10.1090/conm/026/737400`.

[KKM13] Bruce M. Kapron, Valerie King, and Ben Mountjoy. “Dynamic graph connectivity in polylogarithmic worst case time”. In: *Proceedings of the 2013 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2013, pp. 1131–1142. DOI: `10.1137/1.9781611973105.81`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611973105.81`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611973105.81`.