CS5234 Algorithms at Scale

Notes

AY2024/25 Semester 1 · Prepared by Tian Xiao *@snoidetx*

1 Sampling

Probability bounds:

- Markov bound: Let X be a **non-negative** r.v., then for any $t > 0$,
$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$
- Chebychev bound: Let X be a r.v. For any $t > 0$,
$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathrm{Var}(X)}{t^2}.$$
- Chernoff bound: Let $X_1, \cdots, X_t$ be independent r.v. $\in \{0, 1\}$, $X = \sum_i X_i$ and $\mu = \mathbb{E}[X]$, then
$$\Pr[X > (1+\epsilon)\mu] \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mu} \text{ for any } \epsilon > 0;$$
$$\Pr[X < (1-\epsilon)\mu] \leq \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{(1-\epsilon)}}\right)^{\mu} \text{ for any } \epsilon \in (0, 1).$$
  - ▷ Simplified Chernoff bounds:
$$\Pr[X \geq (1+\epsilon)\mu] \leq e^{-\frac{\epsilon^2\mu}{3}} \text{ for any } \epsilon \in (0, 1);$$
$$\Pr[X \leq (1-\epsilon)\mu] \leq e^{-\frac{\epsilon^2\mu}{2}} \text{ for any } \epsilon \in (0, 1);$$
$$\Pr[|X - \mu| \geq \epsilon\mu] \leq 2e^{-\frac{\epsilon^2\mu}{3}} \text{ for any } \epsilon \in (0, 1).$$
- Hoeffding bound: Let $X_1, \cdots, X_t$ be independent r.v., where X_i takes values from $[a_i, b_i]$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X]$, then
$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{2t^2}{\sum_i (b_i - a_i)^2}} \text{ for any } t > 0.$$

Median Approximation

Given a set of numbers $S = \{x_1, x_2, \cdots, x_m\}$, define $\mathrm{rank}(x) = |\{x_i \in S \mid x_i \leq x\}|$. Find a number $x \in S$ s.t. $\frac{m}{2} - \epsilon m \leq \mathrm{rank}(x) \leq \frac{m}{2} + \epsilon m$.

- Randomly pick one: W.p. $2\epsilon + \frac{1}{m}$.
- Median trick: Sample t and use their median:
  1. Fails when at least $\frac{t}{2}$ samples are from S_L or S_U;
  2. X_k denotes the Bernoulli of k-th sample from S_L or S_U;
  3. $\sum_k X_k$ can apply Chernoff bound;
  4. Set $t = \Theta\left(\epsilon^{-2} \log\left(\frac{2}{\delta}\right)\right)$.

Reservoir Sampling

Find a uniform sample s from a stream $x_1 x_2 \cdots x_m$ and we do not know m. Each $x_i \in [n]$.

1. Initialize $s \leftarrow x_1$;
2. On the arrival of each x_i, $s \leftarrow x_i$ w.p. $\frac{1}{i}$.

- Space: $O(\log n)$.
  - ▷ t uniform samples without replacement: $O(t \log n)$.

2 Distinct Elements

Streaming model: A sequence of tokens $\sigma_1\sigma_2\cdots$ where each $\sigma \in [n]$.

- Represented by frequency vector $(f_1, f_2, \cdots, f_n)$, whree f_i is number of occurences of i.
- Turnstile model: Each token $\in [n] \times \{-L, \cdots, L\}$.
  - ▷ Each token (i, c) updates $f_i \leftarrow f_i + c$.
  - ▷ $|f_1| + |f_2| + \cdots + |f_n| = m$.
  - ▷ Strict turnstile: Each $f_i \geq 0$ at any point of time.
  - ▷ Cash register: Each $c > 0$ (no deletion).
- General aim: Use sublinear space, best $O(\log n + \log m)$ space.

k-Universal hashing: For $h \in H$ picked randomly, for any $x \neq x'$, the probability of them having the same hashing $\leq \frac{1}{|Y|^k}$.

Distinct Elements

Find the value of $d(\sigma) = \sum_i f_i^0$.

Goal: Find an (ϵ, δ)-estimation:
$$\Pr[|A(\sigma) - d(\sigma)| > \epsilon \cdot d(\sigma)] \leq \delta.$$

- Algorithm 1:
  1. Take a perfectly random hash function $h : [n] \to [n]$; $z \leftarrow 0$;
  2. For each token $(i, *)$,
     - ▷ Let zeros$(h(i))$ be the maximum j such that 2^j divides $h(j)$.
     - ▷ If zeros$(h(i)) > z$, $z \leftarrow$ zeros$(h(i))$.
  3. Output $2^{z+\frac{1}{2}}$.
- Algorithm 1 + median trick (improves probability)
- Algorithm 1 + median trick + 2-universal hashing (improves space)
  - ▷ Guarantee: $\frac{d}{3} \leq \hat{d} \leq 3d$ w.p. at least $1 - \delta$ when $t = \Theta(\log \frac{1}{\delta})$.
  - ▷ Space: $O(\log \frac{1}{\delta} \log n)$.

3 Frequency Moment

Frequency Moment

Problem: Find the value of $F_k(\sigma) = \sum_i f_i^k$.

Estimation: Find an (ϵ, δ)-estimation:
$$\Pr[|A(\sigma) - F_k(\sigma)| > \epsilon \cdot F_k(\sigma)] \leq \delta.$$

- AMS estimator:
  1. Pick a token J uniformly at random using reservoir sampling from a stream of length m;
  2. Maintain a counter to count m;
  3. Computer $r := |\{p \geq J \mid \sigma_p = \sigma_J\}|$;
  4. Output $X = m\left(r^k - r^{k-1}\right)$.
     - ▷ Analyzed using Chebyshev bound (depending on variance).
- AMS estimator + median of mean trick.

Sketching: Let σ_1, σ_2 be streams and $\sigma_1 \circ \sigma_2$ be their concatenation. A data structure sk() is called *a stretch* if
$$\mathrm{COMB}(\mathrm{sk}(\sigma_1), \mathrm{sk}(\sigma_2)) = \mathrm{sk}(\sigma_1 \circ \sigma_2).$$

- Linear sketch: sk() is a linear function of the frequency vector.

F_2 Estimation

Problem: Find the value of $F_2(\sigma) = \sum_i f_i^2$.

Estimation: Find an (ϵ, δ)-estimation:
$$\Pr[|A(\sigma) - F_2(\sigma)| > \epsilon \cdot F_2(\sigma)] \leq \delta.$$

- Another AMS sketch for turnstile models:
  1. Pick a hash function $h \to \{-1, +1\}$ uniformly at random from a 4-universal family; $z \leftarrow 0$;
  2. For each token (i, c), $z \leftarrow z + c \cdot h(i)$;
  3. Output z^2.
- Another AMS sketch + median of mean trick

4 Heavy Hitter and Sparse Recovery

Heavy Hitters

For an insertion only model, find and output every item with frequency $> \epsilon m$.

Goal: Count: $f_i - \epsilon m \leq \mathrm{count}(i) \leq f_i + \epsilon m$. Heavy hitter: return every item with frequency $> 2\epsilon m$ and no item with frequency $< \epsilon m$.

- Misra-Gries algorithm (deterministic):
  1. Item-count pairs $L \leftarrow \{\}$;
  2. For each token i:
     - ▷ If $i \in L$ then increment its count; else add $\langle i, 1\rangle$ to L;
     - ▷ If $|L| > k$, decrement count of each stored item;
     - ▷ Remove all items with count $= 0$;
  3. For query j, if $j \in L$ then report the corresponding count; else return 0.
  - ▷ Space: $O(\epsilon^{-1}(\log n + \log m))$.
  - ▷ Guarantee:
    - ∗ Count: $f_i - \epsilon m \leq \mathrm{count}(i) \leq f_i$.
    - ∗ Heavy hitters: Return every item with frequence $\geq 2\epsilon m$; no item with frequency $< \epsilon m$.
- Count sketch algorithm:
  1. Initialize an empty array $C[1 \cdots t][1 \cdots k]$. Set $k = 3/\epsilon^2$, $t = \Theta(\log(1/\delta))$;
  2. Choose t independent $h_1, \cdots, h_t : [n] \to [k]$ from a 2-universal family;
  3. Choose t independent $g_1, \cdots, g_t : [n] \to [-1, +1]$ from a 2-universal family;
  4. For each token (i, c):
     - ▷ For $r = 1, \cdots, t$, $C[r][h_r(i)] \leftarrow C[r][h_r(i)] + cg_r(i)$;
  5. For query j, return $\widehat{f_j} = \mathrm{median}_{1 \leq r \leq t}\, g_r(j) C[r][h_r(j)]$.
  - ▷ Space: $O(\epsilon^{-2} \log(1/\delta)(\log n + \log m))$.

▷ Guarantee: (ϵ, δ)-approximation.

> **1-Sparse Recovery**
> For a turnstile model, define the *support* of frequency vector f as $\text{supp}(f) := \{i \in [n] : f_i \neq 0\}$. We say f is s-sparse if $|\text{supp}(f)| \leq s$. Maintain a sketch sk of the stream s.t. if f is 1-sparse, recover f from sk; else detect non-sparsity.

- Algorithm:
  ① $\ell, z, p \leftarrow 0$ and pick a random $r \in \mathbb{F}$ where $n^3 < |\mathbb{F}| \leq 2n^3$;
  ② For each token (j, c):
    ▷ $\ell \leftarrow \ell + c$;
    ▷ $z \leftarrow z + cj$;
    ▷ $p \leftarrow p + cr^j$;
  ③ If $\ell = z = p = 0$, output f is a 0-vector; else if $z/\ell \notin [n]$, output f is not 1-sparse; else if $p \neq \ell r^{z/\ell}$, output f is not 1-sparse; else output $\tilde{f}$ by setting $\tilde{f}_i = \ell$ if $i = z/\ell$ and 0 otherwise.
  ▷ Space: $O(\log n)$.
  ▷ s-sparse recovery can be reduced to 1-sparse recovery by using hash functions to split the stream.

5 Lower Bound

Lower bound using reduction: Let Q be some streaming problem, P be some communication problem that uses at least L bits of space. In P, Alice has x, Bob has y and they want to compute $P(x, y)$. Suppose there is a reduction $x \to \sigma_x$ and $y \to \sigma_y$ such that knowing $Q(\sigma_x \circ \sigma_y)$ solves $P(x, y)$, then we get that Q also requires at least L bits of space.

- Indexing: Alice gets an n-length binary string x and Bob gets an index $j \in [n]$. Bob needs to determine $x[j]$ w.p. $\geq 9/10$. Then Alice must send at least $\Omega(n)$ bits.
- Equality: Both Alice and Bob has an n-length binary string x, y and Bob needs to decide whether $x = y$ deterministically. Then Alice must send at least $\Omega(n)$ bits.
- Gap-Hamming distance: Both Alice and Bob has an n-length binary string x, y and Bob needs to estimate $H(x, y)$ up to an additive $\sqrt(n)$ factor, i.e.,
$$H(x, y) - \sqrt{n} \leq \Delta(x, y) \leq H(x, y) + \sqrt{n}.$$
Then Alice must send at least $\Omega(n)$ bits.
- Self-disjointness: Alice and Bob have two subsets $X, Y \subseteq [n]$ and Bob needs to decide whether $X \cap Y = \emptyset$. Then Alice must send at least $\Omega(n)$ bits.

6 Dimensionality Reduction

Johnson-Lindenstrauss lemma: For any $\epsilon \in (0, 1)$ and any $X \subseteq \mathbb{R}^d$ where $|X| = n$, there exists $f : X \to \mathbb{R}^m$ for $m = O(\epsilon^{-2} \log n)$ s.t.
$$\forall x, y \in X \ \left[(1 - \epsilon)\|x - y\|_2^2 \leq \|f(x) - f(y)\|_2^2 \leq (1 + \epsilon)\|x - y\|_2^2\right].$$

- f is a linear map.
- m is independent of the original dimension d.

Metric space: An ordered pair (X, dist) is a metric space if
① $\text{dist}(x, x) = 0$;
② For any two $x \neq y$, $\text{dist}(x, y) > 0$;
③ Symmetry: $\text{dist}(x, y) = \text{dist}(y, x)$;
④ Triangle inequality: $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$.

Local sensitive hashing (LSH): Consider a metric space (X, dist). Let $S \subseteq X$ be the input set of points. We say that a family of functions $H = \{h : X \to \mathbb{Z}\}$ (r, cr, p_1, p_2)*-LSH* if for all $x, y \in S$,
① $\text{dist}(x, y) \leq r \Rightarrow \Pr_{h \in_R H}[h(x) = h(y)] \geq p_1$;
② $\text{dist}(x, y) > cr \Rightarrow \Pr_{h \in_R H}[h(x) = h(y)] \leq p_2$.

- Example: Consider the family $H = \{h_1, \cdots, h_d\}$, where $h_i(x) = x_i$. Then H is a $(r, cr, e^{-r/d}, e^{-cr/d})$-LSH.

> **Approximate Nearest Neighbor (ANN)**
> Given a subset $S \subseteq \{0, 1\}^d$ of size n, build a data structure s.t. upon receiving a query $q \in \{0, 1\}^d$, we can return $p^* \in S$ that minimizes $\text{dist}(p, q)$ over $p \in S$.

- ANN search using LSH:
  ▷ Preprocessing:
    ① Draw $k\ell$ hash functions $h_{11}, \cdots, h_{\ell k}$ from LSH family H;
    ② Construct ℓ hash tables: for all $i \in [\ell]$, store in the i-th table $f_i(p) = (h_{i1}(p), \cdots, h_{ik}(p))$ (along with p) for each $p \in S$.
  ▷ Query(q): For each $i = 1, \cdots, \ell$,
    ① Compute $f_i(q)$;
    ② For all points p with $f_i(p) = f_i(q)$, check if $\text{dist}(p, q) \leq cr$ and if so output p.
  ▷ Space: $\tilde{O}(n^{1+\rho})$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$.
  ▷ Time: $\tilde{O}(n^\rho)$.

7 Clustering

> **k-Median Clustering**
> Given points $P = \{p_1, \cdots, p_n\}$, find points $\mathcal{C} = \{c_1, \cdots, c_k\}$ in P that minimizes
> $$D(P, \mathcal{C}) = \sum_{i=1}^{n} \min_{c_j \in \mathcal{C}} d(p_i, c_j).$$
> **Goal**: Find a $\mathcal{C}$ s.t. $D(P, \mathcal{C}) \leq \gamma D(P, \mathcal{C}^*)$.

- Linear programming formulation:
$$\begin{aligned} \min \quad & \sum_{i,j} x_{ij} d(p_i, p_j) \\ \text{s.t.} \quad & \forall i \left[\sum_j x_{ij} = 1; \sum_j y_j \leq k\right] \\ & \forall i, j \ [x_{ij} \leq y_j] \\ & \forall i, j \ [x_{ij}, y_j \in \{0, 1\}]. \\ \text{Relaxation:} \quad & \forall i, j \ [0 \leq x_{ij}, y_j \leq 1]. \end{aligned}$$
  ▷ x_{ij} represents whether p_i is assigned to center p_j.
  ▷ y_j represents whether p_j is a center.
  ▷ $C_i := \sum_j x_{ij} d(p_i, p_j)$ is the *cost* of p_i.
  ▷ $V(j) := \{p_i : \exists q \ [d(p_i, q) \leq 2C_i; d(p_j, q) \leq 2C_j]\}$ is the vicinity of p_j.
- Rounding algorithm:
  ① $S \leftarrow \{\}$;
  ② Repeat until all points are deleted:
    ▷ Let p_j be the remaining point with minimum C_j;
    ▷ Add p_j to S;
    ▷ Delete all points in $V(j)$;
  ③ Return S.
  ▷ Guarantee: A $(2, 4)$-approximation: At most $2k$ points with $\mathcal{C}$ s.t. $D(P, \mathcal{C}) \leq 4D(P, \mathcal{C}^*)$.

> **k-Median Clustering in Streams**
> Points $S = s_1, \cdots, s_n$ come in an insertion-only stream.
> **Goal**: $(2, O(1))$-approximation.

- Core-set algorithm:
  ① $S \leftarrow \{\}$;
  ② Repeat $\sqrt{n/k}$ times:
    ▷ Let P = next $\sqrt{nk}$ points;
    ▷ Find $(2, 4)$-approximate clustering on P;
    ▷ Add $2k$ new cluster centers to S. Weight each cluster center with number of points attached to it;
    ▷ Empty P;
  ③ Return $(2, 4)$-approximate (weighted) clustering on S.
  ▷ Space: $O(\sqrt{nk})$.
  ▷ Guarantee: A $(2, 80)$-approximation.
- Hierarchical core-set algorithm: Whenever we see $m = n^\epsilon$ points in a level, add the $(2, 4)$-approximation to the next level.
  ▷ Space: $O(kn^\epsilon/\epsilon)$.
  ▷ Guarantee: A $(2, O(8^{1/\epsilon}))$-approximation.

8 Graph Stream

Counting Triangles
Given undirected graph $G = (V, E)$ with n nodes and m edges, count the number of triangles.
Goal: (ϵ, δ)-approximation.

- Simple yet elegant idea: Consider $\binom{n}{3}$-dimensional vector x where each element is indexed by a triplet $T = \{u, v, w\}$. Count the number of 3's: $N_3 = 0.5F_2 - 1.5F_1 + F_0$.
  - ▷ Guarantee: $(\alpha, 1/20)$-approximation.

Connected Components
Given undirected graph $G = (V, E)$ with n nodes and m edges, count the number of connected components.

- Must use $\Omega(n)$ space.
  - ▷ Reduce from self-disjointness: Alice holds set X and Bob holds set Y. For a graph with vertex set $\{s, t, v_1, \cdots, v_n\}$, Alice constructs edge set $A = \{(s, v_i) : i \neq X\}$ and Bob constructs edge set $B = \{(v_i, t) : i \neq Y\} \cup \{(s, t)\}$. Then the graph G with edge set $A \cup B$ is connected if and only if $X \cap Y = \emptyset$.
- Maintaining a spanning forest:
  1. Forest $F \leftarrow \{\}$;
  2. For each edge e in stream:
     - ▷ If $F \cup \{e\}$ has no cycle then add e to F;
  3. Return number of components in F.
  - ▷ Space: $O(n \log n)$.
  - ▷ Update cost: $O(\alpha(n, n))$.

Graph Bipartiteness
Given undirected graph $G = (V, E)$ with n nodes and m edges, check if the graph is bipartite.

- Maintaining a spanning forest:
  1. Forest $F \leftarrow \{\}$;
  2. For each edge e in stream:
     - ▷ If $F \cup \{e\}$ has no cycle then add e to F;
     - ▷ If $F \cup \{e\}$ has odd cycle then return NO;
  3. Return YES.
  - ▷ Space: $O(n \log n)$;
  - ▷ Time: $O(\alpha(n, n))$.

Shortest Path
Given undirected graph $G = (V, E)$ with n nodes and m edges, find a shortest path from u to v.
Goal: Find a *spanner* $H \subseteq G$ s.t. $d_G(u, v) \leq d_H(u, v) \leq \alpha d_G(u, v)$.

- Spanner construction (k):
  1. Subgraph $H \leftarrow \{\}$;
  2. For each edge $e = (u, v)$ in the stream:
     - ▷ If $d_H(u, v) > 2k - 1$ then add e to H;
  3. Return H.
  - ▷ Girth: girth(G) = size of smallest cycle in G.
    - ∗ If girth(G) $> 2k$, then it has $O(n^{1+1/k})$ edges.
  - ▷ H has at most $O(n^{1+1/k})$ edges.

Matching
Given undirected graph $G = (V, E)$ with n nodes and m edges, find a shortest path from u to v.
Goal: Find a maximum sized matching M^* or an approximation s.t. $|M| \geq |M^*|/c$.

- Greedy algorithm:
  1. $M \leftarrow \{\}$;
  2. For each edge $e = (u, v)$ in the stream:
     - ▷ If u and v are not matched then add e to M;
  3. Return M.
  - ▷ Guarantee: 2-approximation.

Weighted Matching
Given undirected graph $G = (V, E)$ with n nodes and m edges, find a shortest path from u to v.
Goal: Find a maximum weight matching M^* or an approximation s.t. $|M| \geq |M^*|/c$.

- Less greedy algorithm:
  1. $M \leftarrow \{\}$;
  2. For each edge $e = (u, v)$ in the stream:
     - ▷ Let C be edges adjacent to u and v in M;
     - ▷ If $w(e) > (1 + \gamma)w(C)$:
       - ∗ Remove C from M;
       - ∗ Add e to M.
  3. Return M.
  - ▷ Guarantee: 5.83-approximation.