Lecture 8: CountSketch, and Sketching Applications

09-18-2025 Lecturer: Ali Vakilian | Scribe: Pratibha Zunjare | Editor: Ali Vakilian

1 CountSketch

The Count sketch [Charikar et al., 2002] is a linear sketch; similar to CountMin sketch algorithm. However, inspired by AMS Sketch, instead of only adding to a counter, the CountSketch uses a second hash function to decide if a item's frequency should add to or subtract from the counter.

Algorithm 1 CountSketch Algorithm

```
let h_1 ... h_d : [n] → [w] be independent pairwise hash functions.
let g_1 ... g_d : [n] → {−1, +1} be independent pairwise hash functions.
for ℓ = 1 to d do
    initialize counters C[ℓ, j] ← 0 for all j ∈ [w].
for each stream update (i_t, Δ_t) do
    for ℓ = 1 to d do
        C[ℓ, h_ℓ(i_t)] ← C[ℓ, h_ℓ(i_t)] + g_ℓ(i_t) · Δ_t
                                                    ▷ after the stream is over
for ℓ = 1 to d do
    x̃_i^ℓ = g_ℓ(i) · C[ℓ, h_ℓ(i)]
return median{x̃_i^1, ..., x̃_i^d}
```

1.1 Description of CountSketch

The CountSketch uses two hash families. Let $\mathcal{H}$ be a pairwise-independent family $\subseteq \{[n] \to [w]\}$ and let $\mathcal{G}$ be a pairwise-independent family $\subseteq \{[n] \to \{-1, +1\}\}$. For each row $\ell \in [d]$, independently sample $h_\ell \sim \mathcal{H}$, and $g_\ell \sim \mathcal{G}$. In particular,

1. $h_\ell : [n] \to [w]$ maps each item to a bucket (column).
2. $g_\ell : [n] \to \{-1, +1\}$ is a sign hash that determines the sign of the update.

Update: For a stream update (i, Δ), for every $\ell \in [d]$ set $C_\ell[h_\ell(i)] \leftarrow C_\ell[h_\ell(i)] + g_\ell(i) \cdot \Delta$.

Query: Estimate f_i by $\tilde{f}_i = \mathrm{median}_{\ell \in [d]}\big(g_\ell(i)\, C_\ell[h_\ell(i)]\big)$.

Unlike the Count-Min sketch, which is a biased estimator that always overestimates an item's frequency, we will show that the Count-Sketch provides an unbiased estimate.

Theorem 1.1. *Consider strict turnstile streaming (i.e., $f \geq 0$ always). Let $d = \Omega(\log \frac{1}{\delta})$ and $w > \frac{3}{\epsilon^2}$. Then, for any fixed $i \in [n]$, $\mathbb{E}[\tilde{f}_i] = f_i$, and $\Pr[|\tilde{f}_i - f_i| \geq \varepsilon \|f\|_2] \leq \delta$.*

Proof. Consider a fixed row $\ell \in [d]$. For each item $j \neq i$, lets the random variable Y_j indicating whether $h_\ell(j) = h_\ell(i)$. Observe that by pairwise independence of h_ℓ, $\mathbb{E}[Y_j] = \mathbb{E}[Y_j^2] = 1/w$.

Define the random variable $Z_\ell = g_\ell(i) C[\ell, h_\ell(i)]$. By linearity of expectation,

$$Z_\ell = f_i + g_\ell(i) \cdot \sum_{j \neq i} g_\ell(j) f_j Y_j = f_i + \sum_{j \neq i} g_\ell(i) g_\ell(j) f_j Y_j.$$

Since $\mathbb{E}[g_\ell(i)g_\ell(j)] = 0$ for $j \neq i$, we have $\mathbb{E}[Z_\ell] = f_i$. moreover, the variance is

$$\begin{aligned}\mathrm{Var}[Z_\ell] &= \mathbb{E}\big[(Z_\ell - f_i)^2\big] \\ &= \mathbb{E}\Big[\big(g_\ell(i)\sum_{j\neq i} f_j g_\ell(j) Y_j\big)^2\Big] \\ &= \mathbb{E}\Big[\sum_{j\neq i} f_j^2 \cdot Y_j^2 + \sum_{j,j'\neq i} f_j f_{j'} g_\ell(j') Y_j Y_{j'}\Big] \\ &= \mathbb{E}\Big[\sum_{j\neq i} f_j^2 \cdot Y_j^2\Big] + \mathbb{E}\Big[\sum_{j,j'\neq i} f_j f_{j'} g_\ell(j') Y_j Y_{j'}\Big] = \frac{1}{w}\sum_{j\neq i} f_j^2 \leq \frac{\|f\|_2^2}{w}.\end{aligned}$$

Hence, Chebyshev's inequality bounds its failure probability:

$$\Pr\left[|Z_\ell - f_i| \geq \varepsilon \|f\|_2\right] \leq \frac{\mathrm{Var}[Z_\ell]}{\varepsilon^2 \|f\|_2^2} \leq \frac{1}{w\varepsilon^2} \leq \frac{1}{3}$$

Then, we take the median of d such independent estimates to amplify the success probability. The final median estimate is incorrect only if at least half of the row-estimates are *bad* (i.e., deviate from their expectation by more than $\varepsilon\|f\|_2$). Since the expected number of bad estimates is less than $d/3$, the Chernoff bound guarantees that the probability of observing such a large deviation (at least $d/2$ bad estimates) is exponentially small in d. Therefore,

$$\Pr\left(|\tilde{f}_i - f_i| \geq \epsilon \|f\|_2\right) \leq \delta.$$

Thus, $\mathbb{E}[\tilde{f}_i] = f_i$ and $\Pr[|\tilde{f}_i - f_i| \geq \epsilon\|f\|_2] \leq \delta$. □

Corollary 1.2. *By setting the number of rows $d = O(\log(n/\delta))$, we can guarantee that with probability at least $1-\delta$, all n frequency estimates are simultaneously correct. That is:*

$$\Pr\left[\forall i \in [n],\ |\tilde{f}_i - f_i| < \varepsilon \|f\|_2\right] \geq 1 - \delta.$$

Proof. Let E_i be the event that the estimate for a single item i is incorrect, i.e., $|\tilde{f}_i - f_i| \geq \varepsilon\|f\|_2$. From the theorem, we can set the number of rows $d = O(\log(1/\delta_0))$ to make the failure probability for one specific item $\Pr[E_i] \leq \delta_0$.

We want to bound the probability that *any* of the n estimates fail. Using the union bound:

$$\Pr[\text{any estimate fails}] = \Pr\left[\bigcup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr[E_i] \leq n \cdot \delta_0$$

To make this total failure probability at most δ, we can choose $\delta_0 = \delta/n$. Substituting this into the requirement for the number of rows: $d = O(\log(1/\delta_0)) = O(\log(1/(\delta/n))) = O(\log(n/\delta))$. For failure probability $\delta = 1/\mathrm{poly}(n)$, this simplifies to $d = O(\log n)$. □

2 Applications of CountMin and CountSketch

Next, we will explore several applications of the CountMin and CountSketch data structures.

2.1 Heavy Hitters: Point Queries

Given a parameter $\alpha \in (0, 1]$, the goal is to find all items i whose frequency f_i exceeds $\alpha\|f\|_1$. allowing approximation, the output includes any i such that $f_i \geq (\alpha - \varepsilon)\|f\|_1$ for a small error parameter ε.

First Attempt: Query All Items. A straightforward approach is to first build a CountMin sketch over the stream. After processing the stream, one could iterate through every item $i \in [n]$, query the sketch to get its estimated frequency $\tilde{f}_i$, and report any item for which $\tilde{f}_i \geq \alpha \|f\|_1$.

This approach correctly identifies items with high frequency. The CountMin guarantee states that with high probability, $\tilde{f}_i \leq f_i + \varepsilon\|f\|_1$ for all i. This allows us to lower-bound the true frequency of any reported item by $f_i \geq \tilde{f}_i - \varepsilon\|f\|_1 \geq (\alpha - \varepsilon)\|f\|_1$.

However, the significant drawback of this method is that it requires querying every possible item in the universe $[n]$. If the universe size n is very large, this query phase with runtime $\widetilde{O}(n)$ is computationally infeasible and negates the primary benefits of using a sublinear space sketch.

Solution: Hierarchy of CountMin Sketches. The core idea is to maintain a hierarchy of CountMin sketches over the universe $[n]$.

- **Levels:** The structure has $L = \lceil \log_2 n \rceil + 1$ levels, indexed from $\ell = 0$ (the root) to $\ell = L - 1$ (the leaves).
- **Intervals:** At each level ℓ, the universe $[n]$ is partitioned into 2^ℓ disjoint intervals, or *"super-items"* denoted as $e_{\ell,1}, \cdots, e_{\ell,2^\ell}$. Each interval at level ℓ corresponds to a node in the conceptual binary tree and covers a range of items.
- **Sketches:** A separate CountMin sketch, CM_ℓ, is maintained for each level. When an item i arrives in the stream, its corresponding super-item is identified at each of the L levels, and all L sketches are updated accordingly.

The Query Algorithm. The query process is an efficient top-down search through this hierarchy to find items whose frequencies exceed the threshold $\alpha\|f\|_1$. The algorithm begins with a queue Q containing just the root of the tree and repeatedly performs the following steps:

1. **Pop & Query:** Dequeue a node (ℓ, b), representing the b-th interval at level ℓ. Query its corresponding sketch CM_ℓ to get an estimate of its total frequency (i.e., the frequency of corresponding super-item to the b-th interval).
2. **Prune or Expand:**
   - If the estimated frequency is less than the threshold $\alpha\|f\|_1$, this entire branch of the tree is *pruned*, as no item in this branch could have a frequency larger than the target threshold.
   - Otherwise, when the frequency is high enough and we are not at a leaf level, add its two children nodes $(\ell + 1, 2b - 1)$ and $(\ell + 1, 2b)$ to the end of the queue Q.
3. **Identify Candidates:** If a leaf node (level $L - 1$) has an estimated frequency above the threshold, its corresponding item is added to the candidate set.

Theorem 2.1. *The Hierarchical CountMin sketch, using a total space of* $O\left(\frac{1}{\epsilon}\log n \log \frac{n}{\delta}\right)$, *can find a candidate set* $\hat{H}$ *that solves the* (α, ϵ)*-heavy hitters problem in* $O\left(\frac{1}{\alpha}\log n\right)$ *query time. With probability at least* $1 - \delta$, *the set* $\hat{H}$ *satisfies:*

(i) ***(No False Negatives)*** *Every item* i *with* $f_i \geq \alpha\|f\|_1$ *is in* $\hat{H}$.

(ii) ***(Bounded False Positives)*** *No item* i *with* $f_i < (\alpha - \epsilon)\|f\|_1$ *is in* $\hat{H}$.

Proof. The proof consists of analyzing the space complexity and then proving the two correctness properties, which rely on the overall success probability.

Algorithm 2 Dyadic Hierarchical Search for Heavy Hitters

Input: Threshold α, error parameter ε, CountMin sketches $\{\mathrm{CM}_\ell\}_{\ell=0}^{L-1}$
initialize queue $Q \leftarrow \{(0,1)\}$ ▷ start with root level and bucket
initialize candidate set $C \leftarrow \emptyset$
while Q is non-empty **do**
  pop (ℓ, b) from Q
  estimate $\widetilde{\mathrm{freq}}(e_{\ell,b})$ by querying CM_ℓ ▷ frequency estimate of super-item
  if $\widetilde{\mathrm{freq}}(e_{\ell,b}) < (\alpha - \varepsilon)||f||_1$ **then**
    prune this branch (do not expand)
  else
    if $\ell = L - 1$ **then** ▷ leaf level
      add b to candidate set C
    else
      push children $(\ell+1, 2b-1)$ and $(\ell+1, 2b)$ to Q
Return candidate set C

Space Complexity. The data structure consists of $L = O(\log n)$ levels. To ensure the guarantees hold over all queries in the hierarchy with overall success probability $1-\delta$, each of the L CountMin sketches is constructed with a width $w = O(1/\epsilon)$ and a depth of $d = O(\log(n/\delta))$ rows. The total space is the number of sketches multiplied by the size of each sketch:

$$L \times d \times w = O(\log n) \cdot O(\log(n/\delta)) \cdot O(1/\epsilon) = O\left(\frac{1}{\epsilon} \log n \log \frac{n}{\delta}\right)$$

Proof of Correctness. The proof relies on the one-sided error guarantee of the Count-Min sketch, which states that for any item j, its estimated frequency $\tilde{f}_j$ satisfies $f_j \le \tilde{f}_j \le f_j + \epsilon\|f\|_1$.

(i) **No False Negatives:** Consider a true heavy hitter i with $f_i \ge \alpha\|f\|_1$. For the algorithm to report this item, the query process must follow the path from the root of the hierarchy down to the leaf node corresponding to i. This path is never pruned.

Consider any ancestor node (ℓ, b) on this path. The frequency of its corresponding super-item, $\mathrm{freq}(e_{\ell,b})$, is the sum of frequencies of all items in its interval. Since i is in this interval, $\mathrm{freq}(e_{\ell,b}) \ge f_i \ge \alpha\|f\|_1$. As the CountMin sketch always overestimates ($\widetilde{\mathrm{freq}}(e_{\ell,b}) \ge \mathrm{freq}(e_{\ell,b})$), the estimated frequency will also be at least $\alpha\|f\|_1$. Since the estimate never falls below the threshold, the node is never pruned. This holds for all nodes on the path to i, so i will be added to the candidate set $\hat{H}$.

(ii) **Bounded False Positives:** Suppose an item i is returned in the candidate set $\hat{H}$. This means its estimated frequency at the leaf level satisfied $\tilde{f}_i \ge \alpha\|f\|_1$. From the Count-Min guarantee, we know that with high probability, $\tilde{f}_i \le f_i + \varepsilon\|f\|_1$. Combining these two inequalities gives:

$$\alpha\|f\|_1 \le \tilde{f}_i \le f_i + \varepsilon\|f\|_1$$

Rearranging, this gives us a lower bound on the true frequency of any reported item $f_i \ge (\alpha - \epsilon)\|f\|_1$. Therefore, no item with a frequency significantly smaller than the threshold is reported.

Success Probability. The correctness proofs above rely on the CountMin error bounds holding for every node that is queried. By setting the number of rows in each sketch to $d = O(\log(n/\delta))$, we can use a union bound to guarantee that, with probability at least $1-\delta$, the estimate for *every* possible item (and thus every super-item at every level) has the required precision. Since the query algorithm only inspects a small subset of these nodes, this guarantee is sufficient to ensure the entire query process succeeds with probability at least $1-\delta$. □

2.2 Range Queries via Dyadic Intervals

In many applications, the domain $[n]$ has a natural total ordering of items. For example, $[n]$ may represent a discretized timeline for a signal, with x corresponding to the signal values; in databases, $[n]$ often represents ordered numerical attributes such as age, height, or salary. In these scenarios, range queries are particularly useful.

A *range query* is an interval of the form $[i, j]$ where $i, j \in [n]$ and $i \leq j$. The goal is to compute $\sum_{i \leq \ell \leq j} f_\ell$ There are $O(n^2)$ possible range queries and the naïve approach of estimating all items in the range takes $O(j-i)$ time which can be as large as $O(n)$ in the worst case. However, as in Homework 1 (Problem 5), it is possible to support such queries in $O(\log n)$ time by only increasing the total space complexity with a $O(\log n)$ factor.

2.3 Sparse Recovery

Sparsity is a central theme in modern data analysis, referring to data that is dominated by a few significant values. This structure can be *explicit*, as seen in naturally sparse data like graphs or document-term vectors, or *implicit*, where data like images and signals become sparse after a transformation into a different basis (e.g., a Fourier or wavelet basis). Leveraging sparsity provides significant algorithmic advantages, improving the speed, memory usage, and quality of data processing, while also revealing important underlying information, such as topics in a set of documents.

The *sparse recovery problem* formalizes the goal of finding this underlying structure. Given a dense vector or signal $x \in \mathbb{R}^n$ and an integer k, the task is to find a k-sparse vector z (meaning z has at most k non-zero entries, denoted $\|z\|_0 \leq k$) that best approximates x. "Best" is typically defined as minimizing the error $\|x-z\|_p$ for a given norm, most commonly the Euclidean norm ($p = 2$).

Optimal Offline Solution. The optimal offline solution to the sparse recovery problem is a straightforward greedy algorithm. To construct the best k-sparse approximation z for a vector x:

1. Identify the k *entries in* x *with the largest absolute values.*
2. Set the corresponding entries of z equal to these k values.
3. Set the remaining $n-k$ entries of z to zero.

This approach is optimal because to minimize the error $\|x-z\|_p$, one must zero out the entries of x that contribute the least to its overall magnitude. By preserving the k largest-magnitude entries of x, we ensure that the "error vector" $x-z$ (which contains the $n-k$ smallest-magnitude entries of x) has the minimum possible norm.

References

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 693–703. Springer, 2002.