Appendix

A Design of LSH Functions:

In practice, we could realize $d(x, y)$ with cosine similarity for dense vectors. In this case, the LSH function family $\mathcal{H}$ should have the following form.

Definition A.1 (Sign Random Projection (SRP) Hash [56]). *We define a function family $\mathcal{H}$ as follow: Given a vectors $x \in \mathbb{R}^d$, any $h : \mathbb{R}^d \to \{0, 1\}$ that is in the family $\mathcal{H}$, we have*

$$h(x) = \textsf{sign}(Ax),$$

where $A \in \mathbb{R}^{d\times 1}$ is a random matrix that every entry of A is sampled from normal distribution $\mathcal{N}(0, 1)$, $\textsf{sign}$ is the sign function that set positive value to 1 and others to 0. Moreover, for two vectors $x, y \in \mathbb{R}^d$ we have

$$\Pr[h(x) = h(y)] = 1 - \pi^{-1} \arccos \frac{x^\top y}{\|x\|_2 \|y\|_2}.$$

As shown in Definition A.1, the SRP hash is an LSH function built upon random Gaussian projections, which is a fundamental technique in ML [57, 58, 59, 60] If two vectors are close in terms of cosine similarity, their collision probability would also be high.

The SRP hash is usually designed for dense vectors. If both $x \in \{0, 1\}^d$ and $y \in \{0, 1\}^d$ are high dimensional binary vectors with large d. Their Jaccard similarity [61] is also an important measure for search [62, 63, 64] and learning tasks [65, 66]. There also exists a family of LSH functions for Jacacrd similarity. We define this LSH function as:

Definition A.2 (MinHash [67]). *A function family $\mathcal{H}$ is a MinHash family if for any $h \in \mathcal{H}$, given a vectors $x \in \{0, 1\}^d$, we have*

$$h(x) = \arg\min(\Pi(x))$$

where Π is a permutation function on the binary vector x. The $\arg\min$ operation takes the index of the first non-zero value in $\Pi(x)$. Moreover, given two binary vectors $x, y \in \{0, 1\}^d$, we have

$$\Pr[h(x) = h(y)] = \frac{\sum_{i=1}^d \min(x_i, y_i)}{\sum_{i=1}^d \max(x_i, y_i)},$$

where the right term represents the Jaccard similarity of binary vectors x and y.

Following Definition A.2, MinHash serves as a powerful tool for Jaccard similarity estimation [68, 69, 70]. We will use both SRP hash and MinHash in the following section to build a sketch for data distribution.

In this paper, we take a kernel view of the collision probability of LSH (see Definition 3.2). This view aligns with a series of research in efficient kernel decomposition [71, 72, 73], kernel density estimation [34] and kernel learning [74].

B More Algorithms

In this section, we introduce the client selection algorithm with our one-pass sketch.

Algorithm 3 Client Selection with Distribution Sketch

Input: Clients $\mathcal{C} = \{c_1, \cdots, c_n\}$, Number of rounds T, Step size η, LSH function family $\mathcal{H}$, Hash range B, Rows R, Number of active clients L, Number of Selected Clients K, Epochs E, Random Seed s
Output: Global model w^T.
Initialize: Global model w^1, global sketch S_g, random seed s.
for $c \in \mathcal{C}$ **do**
  Get client data $\mathcal{D}_c$ from client c.
  Compute sketch S_c using Algorithm 1 with parameter $\mathcal{D}_c$, $\mathcal{H}$, B, R and s.
  Send S_c to server
  $S_g \leftarrow S_g + S_c$
end for
$S_g \leftarrow S_g/n$ ▷ Generate global sketch on server
for $i \in [T]$ **do**
  L clients $\{c_1, \cdots, c_L\} \subset \mathcal{C}$ are activated at random.
  for $j \in [L]$ **do**
    $p_j = 1/\|S_g - S_j\|_2$ ▷ S_j is the sketch for client c_j
  end for
  for $j \in [L]$ **do**
    $p_j = \frac{\exp p_j}{\sum_{l=1}^{L} \exp p_l}$
  end for
  Sample K clients out of $\{c_1, \cdots, c_L\}$ without replacement, client c_j is selected with probability p_j.
  Server send w^i to the selected clients.
  Each selected client c_k updates w^i to w^i_k by training on its data for E epochs with step size η.
  Each selected client sends w^i_k back to the server.
  $w^{i+1} = \sum_{k=1}^{K} w^i_k$
end for
return w^T

C Discussion

Limitations. To make the sketching process faster and more efficient, we need a powerful GPU for performing matrix multiplication. However, we still need to work on developing a hardware-friendly implementation for sketching in the future.

Potential Negative Societal Impacts. Our work assesses how different the data is among clients without sharing the data. While the calculations are fast, they could still release carbon emissions, particularly when using GPUs as hardware.

D Proofs

To formally prove the Theorems in the paper. We start with introducing a query algorithm to the one-pass distribution sketch.

Next, we introduce the formal statements in the paper as below.

Theorem D.1 (Formal version of Theorem 3.3). *Let $P(x)$ denote a probability density function. Let $\mathcal{D} \underset{\text{iid}}{\sim} P(x)$ denote a dataset. Let $k(x, y)$ be an LSH kernel (see Definition 3.2). Let S define the function implemented by Algorithm 0. We show that*

$$S(x) \underset{\text{i.p.}}{\rightarrow} \frac{1}{N} \sum_{x_i \in \mathcal{D}} k(x_i, q)$$

with convergence rate $O(\sqrt{\log R}/\sqrt{R})$.

Algorithm 4 Query to the Distribution Sketch

Input: Query $q \in \mathbb{R}^d$, LSH functions $h_1, \ldots, h_R$, Dataset $\mathcal{D}$, Sketch $S \in \mathbb{R}^{R \times B}$ built on $\mathcal{D}$ with Algorithm 1
Output: $S(q) \in R$
Initialize: $S(q) \leftarrow 0$
$A \leftarrow \emptyset$
for $i = 1 \rightarrow R$ **do**
  $A \leftarrow A \cup \{S_{i,h_i(q)}\}$
end for
$S(q) \leftarrow \mathsf{median}(A)$
return $S(q)$

Proof. Let $\kappa(x) = \sum_{x_i \in \mathcal{D}} \sqrt{k(x, x_i)}$ be the (non-normalized) kernel density estimate. Theorem 3.4 of [75] provides the following inequality for any δ, where $\widetilde{\kappa}(x) = \sum_{x_i \in \mathcal{D}} \sqrt{k(x, x_i)}$:

$$\Pr\left[|NS(x) - \kappa(x)| > \left(32 \frac{\widetilde{\kappa}^2(x)}{R} \log 1/\delta\right)^{1/2}\right] < \delta$$

This is equivalent to the following inequality, which we can obtain by dividing both sides of the inequality inside the probability by N.

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(32 \frac{\widetilde{\kappa}^2(x)}{N^2 R} \log 1/\delta\right)^{1/2}\right] < \delta$$

We want to show that the error $\left|S(x) - \frac{1}{N}\kappa(x)\right|$ converges in probability to zero, because this directly proves the main claim of the theorem. To do this, we must show that for any $\Delta > 0$,

$$\lim_{R \to \infty} \Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \Delta\right] = 0$$

This can be done by setting $\delta = \frac{1}{R}$, which yields the following inequality:

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(32 \frac{\widetilde{\kappa}^2(x)}{N^2 R} \log R\right)^{\frac{1}{2}}\right] < \frac{1}{R}$$

Because $\widetilde{\kappa} < N$, the following (simpler, but somewhat looser) inequality also holds:

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(32 \frac{\log R}{R}\right)^{\frac{1}{2}}\right] < \frac{1}{R}$$

This implies that $S(x) \underset{\text{i.p.}}{\rightarrow} \frac{1}{N}\kappa(x)$ by considering R large enough that $\sqrt{\log R/R} < \Delta$. □

Theorem D.2 (Formal version of Theorem 3.4)**.** *Let S be an ϵ-differentially private distribution sketch of a dataset $\mathcal{D} = \{x_1, \ldots x_N\}$ with $\mathcal{D} \underset{\text{iid}}{\sim} P(x)$ and let $k(x, y)$ be an LSH kernel (see Definition 3.2). Let S define the function implemented by Algorithm 0. Then*

$$S(x) \underset{\text{i.p.}}{\rightarrow} \frac{1}{N} \sum_{x_i \in \mathcal{D}} k(x_i, x)$$

with convergence rate $O(\sqrt{\log R/R} + \sqrt{R \log R}/(N\epsilon)$ when $R = \omega(1)$ (e.g. $R = \log N$).

Proof. As before, let $\kappa(x) = \sum_{x_i \in \mathcal{D}} \sqrt{k(x, x_i)}$ be the (non-normalized) kernel density estimate and let $\widetilde{\kappa}(x) = \sum_{x_i \in \mathcal{D}} \sqrt{k(x, x_i)}$. For the ϵ-differentially private version of the algorithm, Theorem 3.4 of [75] provides the following inequality for any $\delta > 0$.

$$\Pr\left[|NS(x) - \kappa(x)| > \left(\left(\frac{\widetilde{\kappa}^2(x)}{R} + 2\frac{R}{\epsilon^2}\right) 32 \log 1/\delta\right)^{1/2}\right] < \delta$$

Again, we divide both sides of the inner inequality by N, and we also loosen (and simplify) the inequality with the observation that $\widetilde{\kappa}(x)/N < 1$.

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(\left(\frac{1}{R} + 2\frac{R}{N^2\epsilon^2}\right) 32\log 1/\delta\right)^{1/2}\right] < \delta$$

We want to show that the error $\left|S(x) - \frac{1}{N}\kappa(x)\right|$ converges in probability to zero, because this directly proves the main claim of the theorem. To do this, we must show that for any $\Delta > 0$,

$$\lim_{R\to\infty} \Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \Delta\right] = 0$$

As before, we choose $\delta = \frac{1}{R}$, which yields the following inequality:

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(\left(\frac{1}{R} + 2\frac{R}{N^2\epsilon^2}\right) 32\log R\right)^{1/2}\right] < \frac{1}{R}$$

This is the same as:

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > \left(32\frac{\log R}{R} + 64\frac{R}{N^2\epsilon^2}\log R\right)^{1/2}\right] < \frac{1}{R}$$

Here, we will loosen the inequality again (also for the sake of presentation). Because $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, the following inequality is also satisfied:

$$\Pr\left[\left|S(x) - \frac{1}{N}\kappa(x)\right| > 4\sqrt{2}\sqrt{\frac{\log R}{R}} + 8\frac{\sqrt{R\log R}}{N\epsilon}\right] < \frac{1}{R}$$

Here, the convergence rate is $O(\sqrt{\log R/R} + \sqrt{R\log R}/(N\epsilon)$. For us to have the error converge in probability to zero, we need for

$$\sqrt{\frac{\log R}{R}} + \sqrt{2}\frac{\sqrt{R\log R}}{N\epsilon} \to 0$$

as $N \to \infty$. A simple way to achieve this is for R to be weakly dependent on N (i.e. choose $R = \omega(1)$). For example, choosing $R = \log N$ satisfies the conditions. $\square$