A One-Pass Distributed and Private Sketch for Kernel Sums with Applications to Machine Learning at Scale

Benjamin Coleman
brc7@rice.edu
Rice University
Houston, Texas, USA

Anshumali Shrivastava
anshumali@rice.edu
Rice University
Houston, Texas, USA

ABSTRACT

Differential privacy is a compelling privacy definition that explains the privacy-utility tradeoff via formal, provable guarantees. In machine learning, we often wish to release a function over a dataset while preserving differential privacy. Although there are general algorithms to solve this problem for any function, such methods can require hours to days to run on moderately sized datasets. As a result, most private algorithms address task-dependent functions for specific applications. In this work, we propose a general purpose private sketch, or small summary of the dataset, that supports machine learning tasks such as regression, classification, density estimation, and more. Our sketch is ideal for large-scale distributed settings because it is simple to implement, mergeable, and can be created with a one-pass streaming algorithm. At the heart of our proposal is the reduction of many machine learning objectives to kernel sums. Our sketch estimates these sums using randomized contingency tables that are indexed with locality-sensitive hashing. Existing alternatives for kernel sum estimation scale poorly, often exponentially slower with an increase in dimensions. In contrast, our sketch can quickly run on large high-dimensional datasets, such as the 65 million node Friendster graph, in a single pass that takes less than 20 minutes, which is otherwise infeasible with any known alternative. Exhaustive experiments show that the privacy-utility tradeoff of our method is competitive with existing algorithms, but at an order-of-magnitude smaller computational cost. We expect that our sketch will be practically useful for differential privacy in distributed, large-scale machine learning settings.

CCS CONCEPTS

• **Security and privacy** → **Domain-specific security and privacy architectures**; • **Theory of computation** → **Sketching and sampling**.

KEYWORDS

differential privacy; scaling; sketching; machine learning; locality-sensitive hashing

ACM Reference Format:
Benjamin Coleman and Anshumali Shrivastava. 2021. A One-Pass Distributed and Private Sketch for Kernel Sums with Applications to Machine Learning at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3460120.3485255

1 INTRODUCTION

Large-scale data management is an integral component of the modern data analysis pipeline. The success of machine learning and data mining algorithms critically depend on the quantity and quality of the input data. Although vast amounts of data are available, the information is often of a personal or sensitive nature. Models can leak substantial information about individual participants, even if we only release predictions, outputs or descriptive statistics [19, 20, 35]. Privacy is, therefore, an important design factor when designing systems that analyze and store sensitive data.

To protect against diverse and sophisticated attacks, ϵ-differential privacy has emerged as a theoretically rigorous definition of privacy with robust guarantees [15]. Informally, an algorithm is differentially private if the inclusion (or exclusion) of any specific data record cannot substantially alter the output. Differentially private algorithms are useful because they guarantee a minimum level of protection, no matter what side information an attacker might have. The quality of the protection is measured using the privacy budget ϵ. A small value of ϵ implies that the algorithm can leak very little information about any individual record in the database. In machine learning, most tasks can be reduced to optimizing specific functions over the dataset. As a result, it is critically important to evaluate and optimize such functions while preserving ϵ-differential privacy.

Private Function Release: In this paper, we consider the task of privately releasing a function $f_{\mathcal{D}}(q)$ that applies a pairwise operation $k(x, q)$ to a query and every record in a dataset. Given a query q a dataset $\mathcal{D} = \{x_1, \dots x_N\}$, our objective is to compute an ϵ-differentially private version of the sum:

$$f_{\mathcal{D}}(q) = \sum_{x \in \mathcal{D}} k(x, q) \tag{1}$$

Although one could directly evaluate $f_{\mathcal{D}}(q)$ and release the result with the exponential mechanism, this would cause each query to leak additional information about the dataset. With this method, we can only evaluate $f_{\mathcal{D}}(q)$ a finite number of times before the privacy budget runs out. In practice, this is undesirable because we often require many interactions with the dataset for machine learning or exploratory analysis.

This limitation leads us to consider the private function release problem. Rather than incrementally consume the privacy budget to interactively answer queries, we instead use the privacy budget to release a private summary $\mathcal{S}_{\mathcal{D}}$ of $f_{\mathcal{D}}$ [31]. Since the summarization algorithm itself is private, queries made using the summary do not consume any more of the privacy budget. All downstream analysis tasks are protected by the robustness property of ϵ-differential privacy, and we may issue an unlimited number of queries (provided that we use the summary and not the original dataset) [22]. This problem is related to private database release, but we seek strong error bounds for all queries[1], not just the global minimum as with empirical risk minimization [8] or a finite set of linear queries as in [23].

Application to Private Machine Learning: The function release task is relevant to problems where it is difficult or impossible to place up-front limits on the number of times we must query $f_{\mathcal{D}}(q)$. This situation often happens when we deploy a machine learning system that computes $f_{\mathcal{D}}(q)$ over a private dataset when it processes a query. Many useful functions for data analysis take the form of Equation 1, and the ability to evaluate $f_{\mathcal{D}}(q)$ is critical for a large and important class of statistical models. For example, when $k(x,q)$ is a kernel function, $f_{\mathcal{D}}$ is the kernel density estimate - a convenient way to approximate the likelihood function in a classification model and an important quantity for other applications such as non-i.i.d. federated learning [28]. It is not difficult to find data analysis pipelines that query a likelihood function each time they are used. Without private function release, we can only use the system a finite number of times, rendering it useless in production.

A similar situation arises in exploratory data analysis, where an analyst might want to visualize a dataset by repeatedly querying $f_{\mathcal{D}}$. Another application is data modeling, where $k(x,q)$ is a loss function and $f_{\mathcal{D}}$ is an objective we wish to minimize. A practical method to release $f_{\mathcal{D}}$ would allow many existing algorithms for density estimation, clustering, classification and regression to be made private. As a result, function release has received a considerable amount of theoretical attention.

Practical Limitations: There are elegant, general and powerful techniques to privately release essentially any function of interest [3, 22, 31, 45]. However, private function release has not yet been widely adopted in practice because existing methods fail to scale beyond small, low-dimensional datasets. The practical utility of function release is plagued by issues such as quadratic runtime and exponential memory requirements. For instance, many algorithms release $f_{\mathcal{D}}$ via function approximation over an interpolation lattice, but the size of the lattice grows exponentially with dimensions [3]. Our experiments show that even for low-dimensional data ($d = 3$), this process can take *days*. Other methods require the eigenvalue decomposition of large kernel matrices or hundreds of Monte Carlo integrations for each query [31, 45]. As a result, not many methods can scale beyond a few thousand records or $d > 3$ dimensions. Although private function release is a promising technique, existing algorithms do not satisfy the practical demands of large-scale systems.

Scalable Function Release with Sketches: In this work, we propose a scalable approach to function release using streaming algorithms and compressed sketches. For differential privacy to have a practical impact in big data settings, we require simple algorithms that can operate on internet-scale problems in a single pass through the data [14, 18, 30]. Private sketches are sought after by practitioners because they can efficiently scale to massive datasets [38]. We show how to construct a sketch $\mathcal{S}_{\mathcal{D}}$ that satisfies these requirements for a large class of functions.

Our function release method is an extension of the RACE sketch, a recent development in (non-private) data streaming algorithms [12]. RACE sketches consist entirely of a small array of integers (~4 MB) that are indexed using hash functions. Our private version inherits many properties that are critical for large-scale distributed implementations, such as mergeability and low communication overhead, while also preserving ϵ-differential privacy. We derive pointwise error bounds for our approximation to $f_{\mathcal{D}}$ which show that RACE is competitive with existing methods for function release, and we provide experiments to show that our algorithm has a substantially smaller computation cost. Our experiments show that RACE easily scales to datasets with hundreds of dimensions and millions of entries.

Sketches for Kernel Compositions: In this work, we trade flexibility for computational efficiency. Our sketch restricts the choice of $f_{\mathcal{D}}$ to a specific class of kernels $k(x,q)$ known as *locality sensitive hash (LSH) kernels.* This restriction allows us to obtain fast streaming algorithms, but not all functions can be expressed in terms of LSH kernels. So far, the (non-private) sketch literature has considered only a few specific instances of $f_{\mathcal{D}}$ [11, 12]. However, we argue that RACE is actually capable of performing a more general set of machine learning tasks. We show how to express classification, linear regression, kernel density estimation (KDE), anomaly detection and mode finding in terms of LSH kernel compositions. We conduct an exhaustive set of experiments with KDE, classification and linear regression. Our experiments show that RACE can cheaply release useful functions for many tasks with a competitive privacy-utility tradeoff. While other algorithms can potentially provide better accuracy, RACE is often the only method capable of running on large, high-dimensional datasets in practice.

Our Contribution: We make the following specific contributions. First, we propose a private version of the RACE sketch for scalable function release and prove competitive error bounds on our estimate of $f_{\mathcal{D}}(q)$. We reduce several classes of important machine learning techniques to LSH kernel sum approximation and discuss other methods to extend the RACE framework. We conduct an exhaustive set of experiments for three applications: density estimation, regression and classification. Finally, we obtain an order-of-magnitude speedup for large-scale kernel sum tasks. In particular, kernel density tasks which take hours for existing methods [3, 5, 31] can be completed in seconds using the private RACE sketch.

2 BACKGROUND

We consider a dataset $\mathcal{D}$ of N points in $\mathbb{R}^d$. Although our analysis naturally extends to any metric space, we restrict our attention to $\mathbb{R}^d$ for the sake of presentation.

2.1 Differential Privacy

We use the well-established definition of differential privacy [15].

[1]By "all queries", we mean all values of q. There is no polynomial algorithm for all general queries [40].

Method	Error Bound	Runtime	Comments
Bernstein polynomials [3]	$d^{\frac{d}{d+H}} \left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)^{\frac{H}{d+H}}$	$O(dNM^d)$	$M \geq 2$. Memory is also exponential in d.
PFDA [31]	$\frac{2}{\epsilon}\sqrt{\log \frac{2}{\beta} \log \frac{1}{\delta} \frac{C}{\phi}}$	$O(dN^2)$	C and ϕ are task-dependent (ϵ, β)-differential privacy
MWEM [23]	$N^{\frac{2}{3}} \left(\frac{\log N \log \lvert Q \rvert}{\epsilon}\right)^{1/3}$	$O(dN\lvert Q \rvert)$	Q is a set of query points. Holds with probability $1 - 1/\text{poly}(\lvert Q \rvert)$
Trigonometric polynomials [45]	$\frac{1}{\epsilon} N^{\frac{2d}{2d+H}}$	$O(dN^{1+\frac{d}{2d+H}})$	The result holds with probability $1 - \delta$ for $\delta \geq 10e^{-\frac{1}{5}N^{d/2d+K}}$
This work	$\left(\frac{N}{\epsilon} \log \frac{1}{\delta}\right)^{\frac{1}{2}}$	$O(dN)$	Applies only for LSH kernels. Efficient streaming algorithm.

Table 1: Summary of related methods to release the kernel sum $f_{\mathcal{D}} = \sum_{\mathcal{D}} k(\mathrm{x}, \mathrm{q})$ for an N point dataset $\mathcal{D}$ in $\mathbb{R}^d$. Unless otherwise stated, the error is attained with probability $1 - \delta$ and ϵ-differential privacy. We hide constant factors and adjust results to estimate $f_{\mathcal{D}}$ rather than the KDE ($N^{-1} f_{\mathcal{D}}(\mathrm{q})$) when necessary. H is a kernel smoothness parameter.

Definition 2.1. Differential Privacy [15] A randomized function A is said to provide (ϵ, β)-differential privacy if for all neighboring databases $\mathcal{D}$ and $\mathcal{D}'$ (which differ in at most one element) and all subsets S in the codomain of A,

$$\Pr[A(\mathcal{D}) \in S] \leq e^{\epsilon} \Pr[A(\mathcal{D}') \in S] + \beta$$

The parameter ϵ is the privacy budget, and it acts as a limit to the amount of information that $A(\mathcal{D})$ can leak about any individual element of $\mathcal{D}$. If $\beta > 0$, then $A(\mathcal{D})$ could potentially leak more information, but this only happens with probability up to β. In this paper, we set $\beta = 0$, which is simply called "ϵ-differential privacy." The Laplace mechanism [15] is a general method to satisfy ϵ-differential privacy. By adding zero-mean Laplace noise to a real-valued function, we obtain differential privacy if the noise is scaled based on the sensitivity of the function (Definition 2.2).

Definition 2.2. Sensitivity [15] For a function $A : \mathcal{D} \to \mathbb{R}^d$, the L1-sensitivity of A is

$$\Delta = \sup ||A(\mathcal{D}) - A(\mathcal{D}')||_1$$

where the supremum is taken over all neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$

Theorem 2.3. *Laplace Mechanism [15] Let $A : \mathcal{D} \to \mathbb{R}^d$ be a non-private function with sensitivity Δ and let $\mathbf{z} \sim \text{Lap}(\Delta/\epsilon)^d$ (d-dimensional i.i.d Laplace vector). Then the function $A(\mathcal{D}) + \mathbf{z}$ provides ϵ-differential privacy.*

2.2 Related Work

There are several existing techniques to release the kernel sum $f_{\mathcal{D}}$. Table 1 compares these methods based on approximation error and computational complexity. A common approach is to decompose $f_{\mathcal{D}}$ into a set of weighted basis functions using standard methods from function approximation theory [46]. To create a private representation of the function, we truncate the basis expansion to m terms and represent $f_{\mathcal{D}}$ as a set of weights in $\mathbb{R}^m$. The weights can then be released via the Laplace mechanism and used to construct a private version of $f_{\mathcal{D}}$. The main theoretical difficulty lies in bounding the sensitivity of the weights; once this is done, the algorithm is straightforward.

Basis Expansion: Each basis term in the representation increases the quality of the approximation but degrades the quality of the weights, since the privacy budget ϵ is shared among the m weights. This is a bias-variance tradeoff: we trade variance in the form of Laplace noise with bias in the form of truncation error. The Fourier basis [22], Bernstein basis [3], trigonometric polynomial basis [45] and various kernel bases have all been used for private function release. Because they all internally rely on function approximation, all basis expansion mechanisms share similar properties. For example, they are most effective when $f_{\mathcal{D}}$ is a smooth function because fewer basis terms are needed to approximate smooth curves. They also tend to be computationally infeasible in high dimensions because function approximation suffers from the curse of dimensionality.

Functional Data Analysis: An alternative set of techniques rely on either functional data analysis [31] or synthetic databases [5, 23]. In [31], the authors use densities over function spaces to release a smoothed approximation to $f_{\mathcal{D}}$. By representing the sum $f_{\mathcal{D}}$ as an average over a set of kernels $k(x, q)$ drawn from a distribution, we turn the task of representing $f_{\mathcal{D}}$ into the well-established problem of privately releasing distribution statistics. The key insight from [31] is that we can efficiently release the mean with (ϵ, β)-differential privacy.

Synthetic Databases: The main idea of [23] and [5] is to release a set of weighted synthetic points that can be used in place of the original dataset to estimate $f_{\mathcal{D}}$. While many synthetic database algorithms exist, we focus on ones that are designed to minimize the error of the privately released function $f_{\mathcal{D}}$. For example, [23] provides theoretical error bounds that hold for a finite number of evaluations of $f_{\mathcal{D}}$. The kernel mean embedding algorithm from [5] attempts to release synthetic points with minimal distance to the true dataset in a kernel embedding space. Optimizing the kernel embedding distance is very similar to preserving the kernel density, since the kernel mean embedding is a smoothed version of the KDE. Synthetic databases may also be generated by privately training a machine learning model, such as a generative adversarial network (GAN), to reproduce the data. Methods such as DP-GAN [48] and DP-cGAN [39] can be effective in practice, but require many iterations through the data to train and do not provide statistical guarantees on the quality of the generated data.

Private Sketches: Finally, sketching is a well-established approach to both scalable machine learning and differential privacy. A large class of sketching algorithms deal with numerical linear algebra problems, where the sketch is created by projecting the dataset with a random matrix [47]. These techniques form the basis for differentially private linear regression [34], low rank approximation [41], and many other linear problems. Another class of private sketches solve discrete problems, such as identifying frequent items from a data stream [38].

The practice of adding noise to a sketch to obtain privacy is a very well-established idea that has been applied to the Count-Min Sketch, the HyperLogLog sketch, the AMS sketch, and many other popular sketches. Clever extensions to these algorithms have enabled numerous advances in areas such as federated learning. For example, the algorithm in [27] uses a private Count-Min Sketch to estimate the average gradient over a set of users. The FetchSGD algorithm uses a Count Sketch for a similar purpose [33]. However, sketching algorithms such as the Count Sketch deal with highly specific tasks that are insufficient for the general problem of private function release. It should be noted that there is no private sketch that can address this problem. Existing sketches do not preserve enough information to answer an unlimited set of function queries, and existing solutions for private function release do not provide the computational efficiency afforded by a sketching algorithm.

2.3 Locality-Sensitive Hashing

LSH Functions: An LSH family $\mathcal{F}$ is a family of functions $l(\mathbf{x}) : \mathbb{R}^d \to \mathbb{Z}$ with the following property: Under $l(\mathbf{x})$, similar points have a high probability of having the same hash value. We say that a collision occurs whenever two points have the same hash code, i.e. $l(\mathbf{x}) = l(\mathbf{y})$. In this paper, we use a slightly different definition than the original [24] because we require the collision probability at all points.

Definition 2.4. We say that a hash family $\mathcal{F}$ is locality-sensitive with collision probability $k(\cdot, \cdot)$ if for any two points $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^d$, $l(\mathbf{x}) = l(\mathbf{y})$ with probability $k(\mathbf{x}, \mathbf{y})$ under a uniform random selection of $l(\cdot)$ from $\mathcal{F}$.

LSH Kernels: When the collision probability $k(\mathbf{x}, \mathbf{y})$ is a monotone decreasing function of the distance metric $\text{dist}(\mathbf{x}, \mathbf{y})$, one can show that k is a positive semidefinite kernel function [12]. We say that a kernel function $k(\mathbf{x}, \mathbf{y})$ is an *LSH kernel* if it forms the collision probability for an LSH family. For a kernel to be an LSH kernel, it must obey the conditions described in [9]. A number of well-known LSH families induce useful kernels [21]. For example, there are LSH kernels that closely resemble the cosine, Laplace and multivariate Student kernels [12].

2.4 RACE Sketch

LSH kernels are interesting because there are efficient algorithms to estimate the quantity

$$f_{\mathcal{D}}(\mathbf{q}) = \sum_{\mathbf{x} \in \mathcal{D}} k(\mathbf{x}, \mathbf{q})$$

when $k(\mathbf{x}, \mathbf{q})$ is an LSH kernel. In [12], the authors present a one-pass streaming algorithm to estimate kernel density sums, a special case of $f_{\mathcal{D}}$. The algorithm constructs a RACE (Repeated Array of Count Estimators) sketch $\mathcal{S}_{\mathcal{D}} \in \mathbb{Z}^{R \times W}$, a 2D array of integers that we index using LSH functions. This array is sufficient to report $f_{\mathcal{D}}(\mathbf{q})$ for any query $\mathbf{q} \in \mathbb{R}^d$. We begin by constructing R functions $\{l_1(\mathbf{x}), \dots l_R(\mathbf{x})\}$ from an LSH family $\mathcal{F}$ with the desired collision probability. When an element $\mathbf{x}$ arrives from the stream, we hash $\mathbf{x}$ to get R hash values, one for each row of $\mathcal{S}_{\mathcal{D}}$. We increment row i at location $l_i(\mathbf{x})$ and repeat for all elements in the dataset. To approximate $f_{\mathcal{D}}(\mathbf{q})$, we return the mean of $\mathcal{S}_{\mathcal{D}}[r, l_r(\mathbf{q})]$ over the R rows. This streaming algorithm produces an efficient mergeable summary that approximates the kernel density estimate for all possible queries in a single pass.

To prove rigorous error bounds, observe that each row of $\mathcal{S}_{\mathcal{D}}$ is an unbiased estimator of $f_{\mathcal{D}}$. The main theoretical result of [12] is stated below as Theorem 2.5.

Theorem 2.5. *Unbiased RACE Estimator[12] Suppose that X is the query result for one of the rows of $\mathcal{S}_{\mathcal{D}}$. That is, $X = \mathcal{S}_{\mathcal{D}}[r, l_r(\mathbf{q})]$*

$$\mathbb{E}[X] = f_{\mathcal{D}}(\mathbf{q}) \qquad \text{var}(X) \leq \left(\tilde{f}_{\mathcal{D}}(\mathbf{q})\right)^2 = \left(\sum_{\mathbf{x} \in \mathcal{D}} \sqrt{k(\mathbf{x}, \mathbf{q})}\right)^2$$

3 PRIVATE SKETCHES WITH RACE

We propose a private version of the RACE sketch.[2] We obtain ϵ-differential privacy by applying the Laplace mechanism to each count in the RACE sketch array. Algorithm 1 introduces a differentially private method to release the RACE sketch, illustrated in Figure 1. It is straightforward to see that Algorithm 1 only requires $O(NR)$ hash computations. Assuming fixed R, we have $O(dN)$ runtime. Algorithm 2 shows how to query the sketch. The query algorithm uses the fact that for row r of the array, the value of column $l_r(\mathbf{q})$ is an unbiased estimator of $f_{\mathcal{D}}(\mathbf{q})$. Algorithm 2 simply averages R of these estimators. Since many practical applications divide $f_{\mathcal{D}}$ by N, the number of elements in the private dataset, we also estimate N directly from the private sketch. Note that this normalization step is optional and does not affect privacy. To provide a theoretical comparison with the methods in Table 1, we omit the normalization from our analysis. We also analyze the median-of-means estimate [4] rather than the mean, because the median-of-means procedure allows us to prove exponential concentration around $f_{\mathcal{D}}$ without any assumptions.

Although the Laplace mechanism is a standard and well-known method to achieve differential privacy for count data, the error bounds and practical performance of Algorithms 1 and 2 are comparable to those of specialized privacy mechanisms such as the Bernstein mechanism [3]. To better understand the tradeoffs of our method, we provide a precise (non-asymptotic) theoretical quantification of the estimation error in terms of the privacy budget and the characteristics of $f_{\mathcal{D}}$.

3.1 Privacy

We begin by proving that the sketch returned by Algorithm 1 is indeed ϵ-differentially private. We view Algorithm 1 as a function

[2] From this point onward, we use the term "RACE" to refer to our private version.

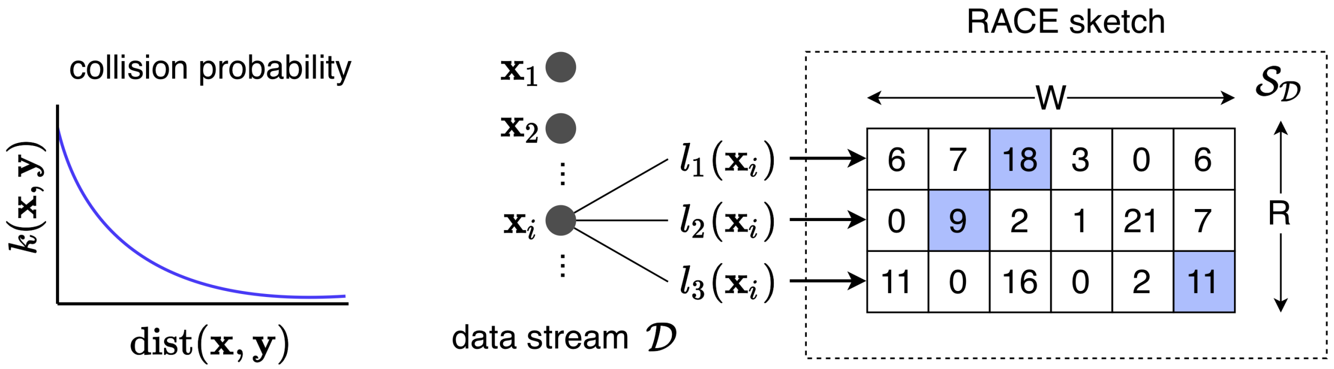

Figure 1: Illustration of Algorithm 1 for $\mathcal{S}_\mathcal{D} \in \mathbb{Z}^{3\times 6}$. We hash each element in the stream with LSH functions $\{l_1, l_2, l_3\} \in \mathcal{F}$ having collision probability $k(\mathbf{x},\mathbf{y})$. In this example, $l_1(\mathbf{x}_i) = 3$, $l_2(\mathbf{x}_i) = 2$ and $l_3(\mathbf{x}_i) = 6$. We increment the highlighted cells. The addition of the Laplace noise is not shown in the figure, but is done by perturbing each count in $\mathcal{S}_\mathcal{D}$.

Algorithm 1 Private RACE sketch

```
Input: Dataset $\mathcal{D}$, privacy budget $\epsilon$, LSH family $\mathcal{F}$, dimensions $R \times W$
Output: Private sketch $\mathcal{S}_\mathcal{D} \in \mathbb{Z}^{R\times W}$
Initialize: $R$ independent LSH functions $\{l_1, ..., l_R\}$ from the LSH family $\mathcal{F}$
$\mathcal{S}_\mathcal{D} \leftarrow \mathbf{0}^{R\times W}$
for $\mathbf{x} \in \mathcal{D}$ do
  for $r$ in 1 to $R$ do
    Increment $\mathcal{S}_\mathcal{D}[r, l_r(\mathbf{x})]$
  end for
end for
$\mathcal{S}_\mathcal{D} = \lfloor \mathcal{S}_\mathcal{D} + Z \rfloor$ where $Z \overset{\text{iid}}{\sim} \text{Lap}(R\epsilon^{-1})$
```

Algorithm 2 RACE query

```
Input: Sketch $\mathcal{S}_\mathcal{D}$, query $\mathbf{q}$, the same $R$ LSH functions from Algorithm 1
Output: Estimate of $N^{-1} f_\mathcal{D}(\mathbf{q})$
$\widehat{N} \leftarrow R^{-1} \sum_{i,j} \mathcal{S}_\mathcal{D}[i,j]$ (optional normalization step)
$\hat{f}_\mathcal{D} \leftarrow 0$
for $r$ in 1 to $R$ do
  $\hat{f}_\mathcal{D} = \hat{f}_\mathcal{D} + \frac{1}{R}\mathcal{S}_\mathcal{D}[r, l_r(\mathbf{q})]$
end for
Return: $\hat{f}_\mathcal{D}/\widehat{N}$
```

$A : \mathcal{D} \to \mathbb{R}^{R\times W}$ that takes in a dataset and outputs an $R \times W$ RACE sketch. The codomain of A is the set of all RACE sketches with R rows and W columns.

Proof Sketch: The sketch $\mathcal{S}_\mathcal{D}$ consists of R independent rows. We can think of each row as a differentially private histogram over the set of hash values for the dataset [49]. Lemma 3.1 shows that we can privately release *one* of the rows by adding Laplace noise with variance ϵ^{-1} (i.e. sensitivity $\Delta = 1$). Then, we repeatedly apply Lemma 3.1 with a budget of ϵ/R to release *all* the rows, proving the theorem.

Lemma 3.1. *Consider one row of the RACE sketch, and add independent Laplace noise* Lap$(1/\epsilon)$ *to each counter. The row can be released with ϵ-differential privacy.*

Proof. Consider one of the W counters in the row. This counter has sensitivity $\Delta = 1$ because adding or removing a single element from the dataset can only create a change of ± 1 in the counter. Therefore, we can add Laplace noise Lap$(1/\epsilon)$ to release the count with ϵ-differential privacy. Suppose we independently do this to release all W counters. The parallel composition theorem [17] states that if each counter is computed over a disjoint subset of the dataset, then we can release all of the counters with ϵ-differential privacy. This is indeed the case because each element in the dataset maps to exactly one of the W counters under the LSH function. Thus, the W noisy counters are computed on disjoint subsets and we can add Laplace noise Lap$(1/\epsilon)$ to release the row. □

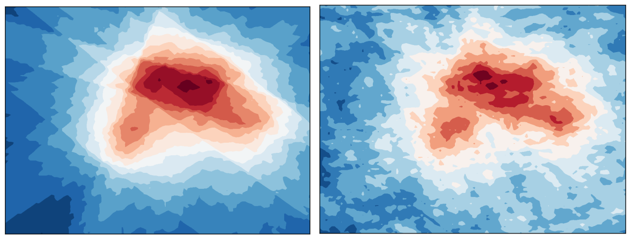

Figure 2: Visualization of non-private (left) and private (right) sketch-based KDE. Note the distinct boundaries between regions that map to different locations and counts in the array. In $\mathbb{R}^d$, RACE approximates functions with a piecewise-constant spline over random partitions. The Laplace mechanism perturbs the value of each partition.

Theorem 3.2. *For any $R > 0$, $W > 0$, and LSH family $\mathcal{F}$, the output of Algorithm 1, or the RACE sketch $\mathcal{S}_\mathcal{D}$, is ϵ-differentially private.*

Proof. The sketch is composed of R independent rows. The sequential composition theorem [17] states that given R mechanisms $\mathcal{M}_1, \ldots \mathcal{M}_R$ (one to release each row) which are independently ϵ-differentially private, we can compute all the mechanisms (i.e. release all rows) with $R\epsilon$-differential privacy. To construct the sketch, we apply Lemma 3.1 to each row with ϵ/R differential privacy by adding independent Laplace noise $\mathrm{Lap}(R/\epsilon)$ to each counter. Therefore, the sketch is ϵ-differentially private by the sequential composition theorem. □

3.2 Utility

Given a sufficiently good estimate of $f_{\mathcal{D}}$, one can easily construct many different learning algorithms [3]. Therefore, we focus on utility guarantees that bound the pointwise error of the private RACE estimate for $f_{\mathcal{D}}(\mathbf{q})$. Since the sketch is a simple collection of R unbiased estimators, our proof strategy is to bound the deviation between the estimate and the mean. This is done by bounding the relative variance of the estimate and applying the median-of-means concentration technique. We briefly reproduce the technique here. Note that Lemma 3.3 is a special case[3] of Theorem 2.1 from [4].

Lemma 3.3. *Let $X_1, \ldots X_R$ be R i.i.d. random variables with mean $\mathbb{E}[X] = \mu$ and variance $\leq \sigma^2$. To get the median of means estimate $\hat{\mu}$, break the R random variables into k groups with $m = R/k$ elements in each group.*

$$\hat{\mu} = \text{median}\left(\frac{1}{m}\sum_{i=1}^{m} X_i, \ldots, \frac{1}{m}\sum_{i=(k-1)m+1}^{km} X_i\right)$$

Put $k = 8\log(1/\delta)$ and $m = R/k$. Then with probability at least $1-\delta$, the deviation of the estimate $\hat{\mu}$ from the mean μ is

$$|\hat{\mu} - \mu| \leq \sqrt{32\sigma^2 \frac{\log(1/\delta)}{R}}$$

Lemma 3.3 requires a bound on the variance of the private RACE estimator, which we obtain by adding the independent Laplace noise variance $2R^2\epsilon^{-2}$ to the bound from Theorem 2.5. Theorem 3.4 follows.

Theorem 3.4. *Let $\hat{f}_{\mathcal{D}}(\mathbf{q})$ be the median-of-means estimate using an ϵ-differentially private RACE sketch with R rows and $\tilde{f}_{\mathcal{D}}(\mathbf{q}) = \sum_{\mathcal{D}} \sqrt{k(\mathbf{x},\mathbf{q})}$. Then with probability $1-\delta$,*

$$|\hat{f}_{\mathcal{D}}(\mathbf{q}) - f_{\mathcal{D}}(\mathbf{q})| \leq \left(\frac{\tilde{f}^2_{\mathcal{D}}(\mathbf{q})}{R} + \frac{2}{\epsilon^2}R\right)^{1/2} \sqrt{32\log 1/\delta}$$

Proof. To use median-of-means, we add the (independent) Laplace noise variance $2R^2\epsilon^{-2}$ to the variance bound in Theorem 2.5.

$$\mathrm{var}(X_r) = \sigma^2 \leq \left(\tilde{f}_{\mathcal{D}}\right)^2 + 2\frac{R^2}{\epsilon^2}$$

Substituting this variance bound into Lemma 3.3 proves the theorem. □

[3]For simplicity, we suppose k and m are integers that evenly divide R. If this is not the case, the results do not change substantially. However, the statement of the lemma becomes more complicated. See [4] for the complete analysis.

Theorem 3.4 suggests a tradeoff for which there is an optimal value of R. This may be thought of as a bias-variance problem, similar to the one that arises with the orthogonal series estimators from [3, 22, 45]. The privacy budget ϵ is shared by the R rows of the sketch, and increasing R improves the quality of the median-of-means estimate. If we increase R, we improve the estimator but must add more Laplace noise to each row to preserve ϵ-differential privacy. If we decrease the number of rows, we require less Laplace noise but start with a worse estimator. To get our main utility guarantee, we choose an optimal R that minimizes the error bound. Surprisingly, the optimal R produces a bound with an *asymptotically better* $O(\epsilon^{-1/2})$ dependence on ϵ than the $O(\epsilon^{-1})$ factor obtained with any other of value of R.

Corollary 3.5. *Put $R = \lceil \frac{1}{\sqrt{2}} \tilde{f}_{\mathcal{D}}(\mathbf{q})\epsilon \rceil$. Then the approximation error bound is*

$$|\hat{f}_{\mathcal{D}}(\mathbf{q}) - f_{\mathcal{D}}(\mathbf{q})| \leq 16\left(\frac{\tilde{f}_{\mathcal{D}}(\mathbf{q})}{\epsilon}\log 1/\delta\right)^{1/2} \leq 16\sqrt{\frac{N}{\epsilon}\log 1/\delta}$$

Proof. Take the derivative of $\sqrt{a/R + bR}$ with respect to R to find that $R = \sqrt{a/b}$ minimizes the bound. Put $a = \tilde{f}^2_{\mathcal{D}}(\mathbf{q})$ and $b = 2/\epsilon^2$. The corollary is obtained by substituting $R = \frac{1}{\sqrt{2}}\tilde{f}_{\mathcal{D}}(\mathbf{q})\epsilon$ into Theorem 3.4. We may replace $\tilde{f}_{\mathcal{D}}(\mathbf{q})$ with N in the inequality because

$$\tilde{f}_{\mathcal{D}}(\mathbf{q}) = \sum_{\mathbf{x}\in\mathcal{D}} \sqrt{k(\mathbf{x},\mathbf{q})} \leq \sum_{\mathbf{x}\in\mathcal{D}} 1 = N$$

□

3.3 Practical Implications

Although we use an average rather than the median-of-means estimator in practice (Algorithm 2), our theory still has direct consequences for applications. In this section, we interpret the practical implications of our utility theorem.

Low-Density Queries: Corollary 3.5 suggests that small values of $f_{\mathcal{D}}$ are difficult to estimate accurately. If we divide both sides of Corollary 3.5 by $f_{\mathcal{D}}$, we have a bound on the relative (or percent) error rather than the absolute error. The percent error bound has a $\tilde{f}_{\mathcal{D}}(\mathbf{q})^{-1/2}$ factor that can be very large if $f_{\mathcal{D}}(q)$ is small. This agrees with our intuition about how the KDE should behave under differential privacy guarantees. Fewer individuals make heavy contributions to $f_{\mathcal{D}}$ in low-density regions than in high-density ones, so the effect of the noise is worse for sparsely-populated portions of the histogram. This can be seen in Figure 2, where the low-density regions appear to be noisier.

Hyperparameter Tuning: In practice, we must spend a portion of the privacy budget to evaluate each hyperparameter combination, so it is important to select good parameters to avoid wasting resources. Our sketch has three hyperparameters: W, R, and the kernel $k(\mathbf{x},\mathbf{q})$. In many cases, the kernel is known and the value of W can be determined from the kernel or else set to a large constant value [12]. The main challenge, then, is to select a good value of R. Corollary 3.5 states that the optimal value of R is $R^* = \lceil \frac{1}{\sqrt{2}}\tilde{f}_{\mathcal{D}}(\mathbf{q})\epsilon \rceil$, which implies that $1 \leq R^* \leq \frac{1}{\sqrt{2}}N\epsilon$. Unfortunately, we cannot directly compute R^* unless we have apriori knowledge of the expected value $\mathbb{E}_q[\tilde{f}_{\mathcal{D}}(q)]$ over the queries issued by a specific application.

However, we can identify a good value of R by evaluating a few choices from the interval $[1, \frac{1}{\sqrt{2}}N\epsilon]$. A principled and efficient way to select R is to use binary search over this interval.

4 APPLICATIONS

The ability to release a pairwise sum is broadly useful for many problems in machine learning. We present private algorithms for density estimation, classification, regression, mode finding, anomaly detection and sampling using our sketch. All of these algorithms are derived by releasing a useful instance of the LSH kernel sum $f_{\mathcal{D}}$. Given a sufficiently good estimate of $f_{\mathcal{D}}$, the following applications are possible:

Kernel Density Estimation: Kernel density estimation is a classical nonparametric method to directly estimate a distribution from a dataset. To use RACE for KDE, we select one or more LSH kernels from the options described by [12]. We require a separate sketch for each kernel and bandwidth setting. Figure 2 shows a visualization of our private density estimation method.

Mode Finding: Given access to the probability density, we can locate the modes of the data distribution. Gradient-free optimization over the sketch works surprisingly well, but in general the KDE is a non-convex function. An alternative (but more expensive) approach is to intersect the hash partitions to identify the mode as a point from the partition with the largest count values [11].

Naive Bayes Classification: The KDE is a convenient way to estimate likelihood functions in statistical estimation. Using kernel density classification, a well-developed result from statistical hypothesis testing [25], we can construct classifiers with RACE under both the maximum-likelihood and maximum a posteriori (MAP) decision rules. To make this example more concrete, suppose we are given a training set $\mathcal{D}$ with M classes $C_1, \dots C_M$ and a query $\mathbf{q}$. We can represent the empirical likelihood $\Pr[\mathbf{q}|C_i, \mathcal{D}]$ with a sketch of the KDE for class i. Algorithm 2 returns an estimate of this probability, which may be used directly by a naive Bayes classifier or other type of probabilistic learner.

Anomaly Detection / Sampling: Anomaly detection can be cast as a KDE classification problem. If Algorithm 2 reports a low density for $\mathbf{q}$, then the training set contains few elements similar to $\mathbf{q}$ and thus $\mathbf{q}$ is an outlier. This principle is behind the algorithms in [29] and [10].

Empirical Risk Minimization: The applications presented so far have only used one type of function $f_{\mathcal{D}}$, the KDE sum. KDE sums are simple to estimate because most LSH functions have positive symmetric collision probabilities. However, LSH kernel sums can be combined to estimate a much more diverse set of functions than positive semidefinite kernel sums. Using asymmetric LSH [36], we can obtain asymmetric kernels by applying different hashes to x and q. We may also add and subtract LSH kernels by incrementing $\mathcal{S}_{\mathcal{D}}$ using multiple hash functions or incrementing by values other than 1, although the sensitivity in Theorem 3.2 increases for each additional function. To maintain ϵ-differential privacy, we must properly scale the noise so that the sensitivity is correct.

This flexibility allows RACE to perform empirical risk minimization (ERM). In ERM problems, we are given a dataset $\mathcal{D} = \{\mathbf{z}_1 \dots \mathbf{z}_N\}$ of training examples and a loss function $L(\theta, \mathbf{z})$, where θ is a parameter that describes a predictive model that we wish to train over the dataset. The task is to find a model θ that minimizes the mean loss over the training set. Using the sketch, we can privately approximate the loss sum when $L(\theta, \mathbf{z})$ can be expressed in terms of LSH collision probabilities.

Optimization over Sketches: Given a model parameter θ, we can estimate the empirical risk by querying the sketch with (possibly some transformation of) θ. Although we cannot analytically find the gradient of the RACE count values, black-box access to the loss function is sufficient to optimize the model with derivative-free optimization [13]. The idea is to query the sketch using random points surrounding θ to approximate the gradient. Since the sketch has already been released with differential privacy, we can iterate until θ converges to the optimal model without consuming the privacy budget. Our experiments (Figure 5) show that derivative-free optimization is highly effective over count-based sketches, validating results from [11].

Constructing Losses from Kernel Sums: To train a specific ERM model using the RACE sketch, we must approximate the loss function $L(\theta, \mathbf{z})$ using LSH kernels. Our goal is to construct a hash function whose collision probability forms a *surrogate loss* for the model of interest. For example, consider the linear regression loss $\mathcal{L}([\mathbf{x}, y], \theta) = (\langle [\mathbf{x}, y], [\theta, -1] \rangle)^2$. We wish to use LSH kernels to build a surrogate loss with the same minimum as $\mathcal{L}$. Note that the SRP kernel from [12] is a monotone increasing function of the inner product $\langle [\mathbf{x}, y], [\theta, -1] \rangle$. We may also obtain a kernel that is a monotone *decreasing* function of the inner product by multiplying the query by -1 (i.e. hashing $-[\theta, -1]$ rather than $[\theta, -1]$). If we query the sketch with with both $\pm[\theta, -1]$ and add the results, we estimate the quantity:

$$f_{\mathcal{D}}(\theta) = \sum_{[\mathbf{x}, y] \in \mathcal{D}} k_{\text{SRP}}(\langle [\mathbf{x}, y], [\theta, -1] \rangle) + k_{\text{SRP}}(-\langle [\mathbf{x}, y], [\theta, -1] \rangle)$$

One can show that $f_{\mathcal{D}}(\theta)$ is a convex surrogate loss for linear regression [11]. By minimizing $f_{\mathcal{D}}(\theta)$ with derivative-free optimization, we may train a private regression model from the sketch. One can use a similar process to design a surrogate loss for linear SVMs. For details, see [11], but note that some of the modifications proposed in their work prevent the sketches from being merged or released with differential privacy. For a discussion of techniques to extend this framework beyond linear regression and classification, see Section 7.1.

5 EXPERIMENTS

We perform an exhaustive comparison for KDE, classification and regression. Table 2 presents the datasets used in our experiments. For KDE, we estimate the density of the salaries for New York City (NYC) and San Francisco (SF) city employees in 2018, as well as high-dimensional densities for the skin, codrna and covtype UCI datasets. The purpose of our one-dimensional experiment is to visualize the functions released by various algorithms, while the other datasets are included in our evaluation to benchmark function release methods on large and high-dimensional datasets. Note that for the covtype KDE task, we were unable to run existing methods due to the size of the data. We needed to apply sampling and dimensionality reduction to compare against baselines. We

Dataset	N	d	σ	Description	Task
NYC	25k	1	5k	NYC salaries (2018)	KDE
SF	29k	1	5k	SF salaries (2018)	
skin	241k	3	5.0	RGB skin tones	
codrna	57k	8	0.5	RNA genomic data	
covtype	580k	55	20	Cartographic features for forestry	
nomao	34k	26	0.6	User location data	Classification
occupancy	17k	5	0.5	Building occupancy	
pulsar	17k	8	0.1	Pulsar star data	
airfoil	1.4k	9	-	Airfoil parameters and sound level	Regression
naval	11k	16	-	Frigate turbine propulsion	
gas	3.6k	128	-	Gas sensor, different concentrations	

Table 2: Datasets used for KDE and classification experiments. Each dataset has N entries with d features. σ is the kernel bandwidth.

	PFDA	Bernstein	KME	RACE	PrivBayes
Sketch	> 3 days	2.3 days	6 hr	15 sec	12 sec
Query	-	6.2 ms	1.2 ms	0.4 ms	0.5 sec

Table 3: Computation time for KDE on the skin dataset. PFDA was unable to finish on this dataset within 3 days.

also include experiments specifically designed to test the scaling capacity of RACE in Section 6.

For the regression and classification experiments, we use UCI datasets of moderate size ($N > 1k$) and dimensionality ($3 \leq d \leq 200$). We preprocess the nursery and occupancy datasets by using integer coding for the categorical attributes. We also scale the features of our linear regression datasets so that the training data lies in the unit sphere, as this is required for some of our baseline algorithms to have good performance. Here, our main objective is to show that these machine learning tasks can be performed using only the private RACE summary. To show that the private RACE sketch can be used to obtain reasonable machine learning models, we compare the performance on a test set with well-known methods to privately train linear and logistic models.

Hyperparameters: Our goal is to provide a fair comparison that shows the performance of each algorithm under high-performing hyperparameter settings. For kernel density estimation and max-likelihood classification, we use the p-stable Euclidean LSH kernel, which closely resembles the exponential kernel. We chose the kernel bandwidth based on the scale of the features in the dataset. For example, we selected σ = \$5k as a reasonable level at which to distinguish annual salaries and $\sigma = 5$ as a significant difference in the distance between RGB color tones. We provide detailed information about hyperparameter tuning for each algorithm below. It should be noted that in practice, the privacy loss incurred by the hyperparameter tuning process affects the *end-to-end* differential privacy budget. However, an end-to-end evaluation is outside the scope of our experiments. Although we do limit the number of hyperparameter settings for each algorithm (i.e. we avoid exhaustive grid search, which is utterly infeasible in practice), our goal is to compare algorithm performance and computation costs under optimal parameter configurations. Therefore, the values of ϵ reported in our experiments are the privacy loss of the best-performing configuration rather than the total privacy loss for all configurations that were considered.

5.1 Baseline Algorithms

We consider the following algorithms in our evaluation. To give a fair comparison, we profiled and optimized each of our baselines. Unless otherwise specified, we implemented each algorithm in Python.

Spectral Approximation: We approximate the kernel density function with a Fourier series, truncate the series expansion, and add Laplace noise to each coefficient. We release the noisy coefficients as the function summary. This algorithm is otherwise known as the orthogonal series estimator [22, 46]. Due to the high-dimensional Bravais lattice used for the Fourier Transform, the memory footprint of spectral approximation scales poorly. As a result, we were only able to apply this method to our one-dimensional salary datasets. Spectral approximation requires that the dataset lie within the interval $[0, 1]$ - we scale the salaries by 250k to satisfy this condition. The method takes one hyperparameter M, the number of basis functions, which we select from $[2, 4, 8, 10, 20]$.

Bernstein Mechanism: This method is also based on function approximation, but uses Bernstein polynomials rather than the Fourier basis [3]. The coefficients of the Bernstein interpolation are released on a high-dimensional grid via the Laplace mechanism, where the number of grid points M is user-specified. While M scales exponentially with dimensions, the coefficients can be computed in parallel using linear memory. We consider $M \in \{2, 4, 8, 10, 20, 40, 100\}$, but were unable to evaluate $M > 20$ for the skin and codrna datasets due to computational limitations. The Bernstein mechanism requires that the dataset lie within the unit hypercube, so we scale the datasets accordingly.

Private Functional Data Analysis (PFDA): We use the PFDA algorithm from [31], to release an estimator for the KDE. PFDA guarantees (ϵ, δ) differential privacy, as it uses the Gaussian mechanism. For all experiments, we use $\delta = 0.01$. PFDA uses a smoothing hyperparameter ϕ, which we set to 0.01 as suggested by the paper. We used the R implementation from [31] but were unable to run the algorithm on high-dimensional datasets.

Kernel Mean Embedding (KME): This baseline method uses the kernel mean embedding to generate a synthetic database. Data release via KME involves weighting the synthetic database so that the kernel mean embedding of the synthetic public database is close to the embedding of the private database. We use the Python code released by [5] in our experiments. The method has two hyperparameters: the distribution of the synthetic points and the number M of synthetic points. Where possible, we use information about the dataset to inform the distribution of synthetic points because this can substantially improve the performance of KME. For example, all skin tone value are between 0 and 255, so we adjust the mean and standard deviation of the public synthetic dataset to cover this range. We choose M from [5k, 10k, 20k, 40k, 100k].

PrivBayes: PrivBayes [50] is a recent synthetic database release algorithm that draws samples from a private Bayesian network that models the correlations between features in the data. We use PrivBayes to release synthetic datasets for our KDE and regression experiments. However, we were unable to evaluate PrivBayes on the 128-dimensional gas dataset, as it took > 6 hours to construct the high-dimensional conditional distributions and release the dataset for each value of ϵ. We use the code provided by [50] and optimize two hyperparameters: the degree D of the Bayesian network and the number M of synthetic points. We consider $D = [2, 3, 4, 5]$ as suggested by the paper and $M =$ [5k, 10k, 20k, 40k, 100k].

Synthetic Data from Histograms: For our salary datasets, we constructed a simple baseline method that draws samples from a Laplace-perturbed histogram. We begin by releasing a differentially private histogram using standard techniques [32], and construct a weighted synthetic dataset from the histogram bin centers and values. This algorithm is not effective for high dimensional data due to limitations that are inherent to histogram-based methods.

Objective Perturbation: Objective perturbation is a general technique for differentially private empirical risk minimization [8]. The algorithm first perturbs the loss function to preserve privacy, then trains a model using the modified version of the loss. The model can then be released with ϵ-differential privacy. We use the technique to train a regularized logistic classifier for our classification experiments and to release linear regression models for our regression experiments. For our logistic classifier, we select the regularization hyperparameter from $\lambda = [0.01, 0.1, 1.0, 2.0]$ based on test set performance.

SSP and AdaSSP: Sufficient statistics perturbation and the new adaptive version of the algorithm are specialized algorithms for linear regression which privately release a set of *sufficient statistics* to compute the linear model. The algorithms essentially release private projections of $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}\mathbf{y}$. We use the MATLAB implementations provided by [43].

OPS and AdaOPS: Posterior sampling (OPS) [44] and the adaptive version proposed by [43] are algorithms which privately sample from a Bayesian posterior and use the sample to obtain the model. We use the MATLAB implementations provided by [43]. Note that the AdaSSP and AdaOPS methods do not require hyperparameter tuning.

RACE: We implemented RACE using Python and NumPy to batch-process the hash calculations. For our KDE experiments, we used $W = 1000$ and selected R using the procedure described in Section 3.3. In practice, the hyperparameter search process amounted to building RACE sketches with $R = [100, 200, 500, 1000, 2000]$ and selecting the best sketch. For our classification experiments, we construct max-likelihood classifiers using RACE to estimate the likelihood functions. We used $W = 100$ for all sketches and selected R from $R = [10, 20, 50, 100, 200]$ because these datasets have smaller N. These experiments were mostly done with binary-coded categorical data on the unit hypercube. Therefore, we selected the bandwidth parameter via binary search of the interval $[0, 1]$. For our regression experiments, we use the surrogate loss described in Section 4 with $R = 1000$ and $W = 16$ (for this loss function, W is determined by the hash function - see [11] for details).

5.2 Results

Figure 3 shows the results of our KDE experiments. On our salary datasets, we evaluate performance using the L2 function norm between the approximate KDE and the ground-truth KDE. We query the SF and NYC datasets on a grid of 200 points ranging from \$10k to \$250k. We integrate the squared error over this grid to approximate the L2 function error. Figure 3 shows the relationship between the error and the privacy budget. To visualize the functions released by various methods, we also include a qualitative comparison at the $\epsilon = 1.0$ privacy budget. For high-dimensional datasets like skin, covtype and codrna, it is too computationally expensive to integrate the density over the entire domain. Instead, we query these datasets using a test set of 2000 query points and report the average relative (percent) error. Figure 7 shows the results of KDE on the covtype dataset. While RACE ran in under 2 minutes, we needed to apply sampling to get the baselines to run in less than 6 hours. Figure 4 shows the results of our classification experiments, and Figure 5 shows our linear regression experiments. In both sets of experiments, we privately train the model and evaluate performance on a held-out test set.

Computation: Table 3 displays the computation time needed to construct a useful function release. Note that Bernstein release can run faster if we use fewer interpolation points (see Table 1), but we still required at least 12 hours of computation for competitive results. The Bernstein mechanism requires many binomial coefficient evaluations, which were expensive even when we used optimized C code. KME required a large-scale kernel matrix computation, and PFDA required several days for an expensive eigenvalue computation. Although PrivBayes ran well for skin ($d = 3$ dimensions), the runtime of the algorithm rapidly degrades in high dimensions. In particular, it took several hours to release synthetic databses for the naval and gas regression datasets.

6 LARGE SCALE EXPERIMENTS

To test the scaling capacity of our algorithm, we consider kernel density problems on the graphs from the Stanford Network Analysis Project [26]. In the context of social networks, the Jaccard kernel is a measure of the local edge density near the query node [37]. Since well-connected nodes have high density, a differentially private Jaccard sum reveals whether a query node is part of a community without disclosing the presence of any particular data node.

Parallel Construction: We use OpenMP in C++ to distribute the sketching algorithm over 90 threads and merge the sketches. We implement the algorithm as a data-parallel operation, where

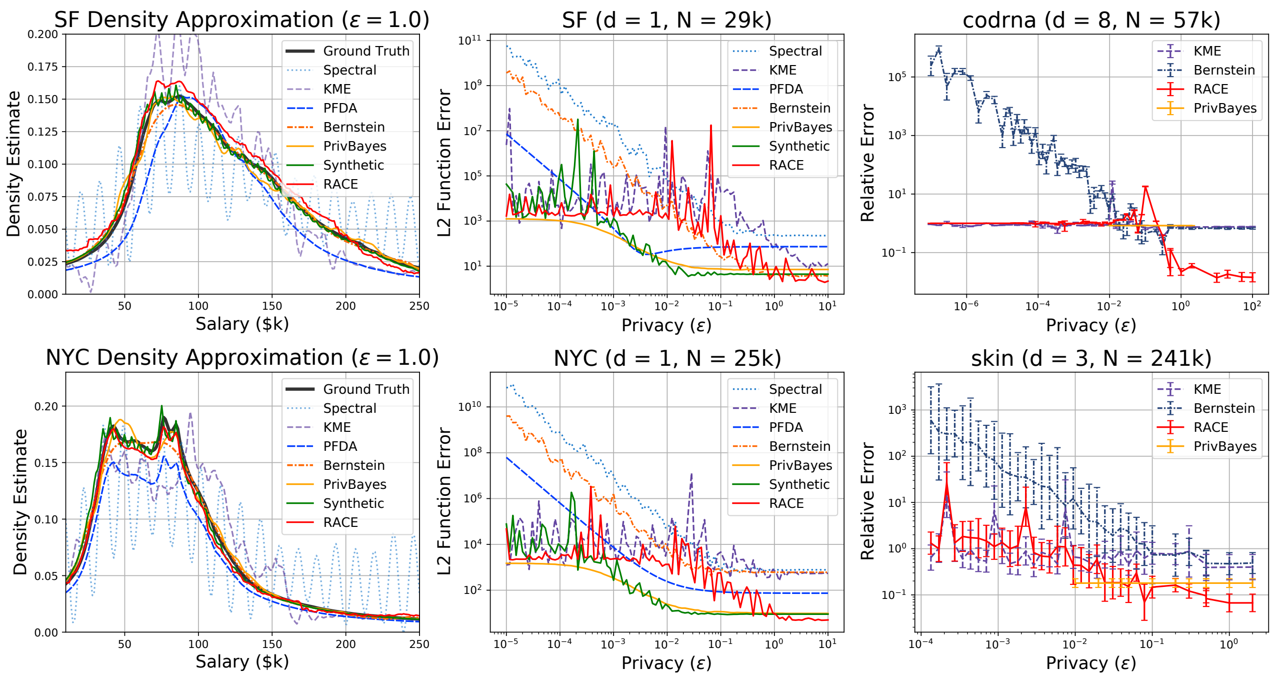

Figure 3: Privacy-utility tradeoff for private function release methods. We report the L2 function error and the mean relative error for 2000 held-out queries.

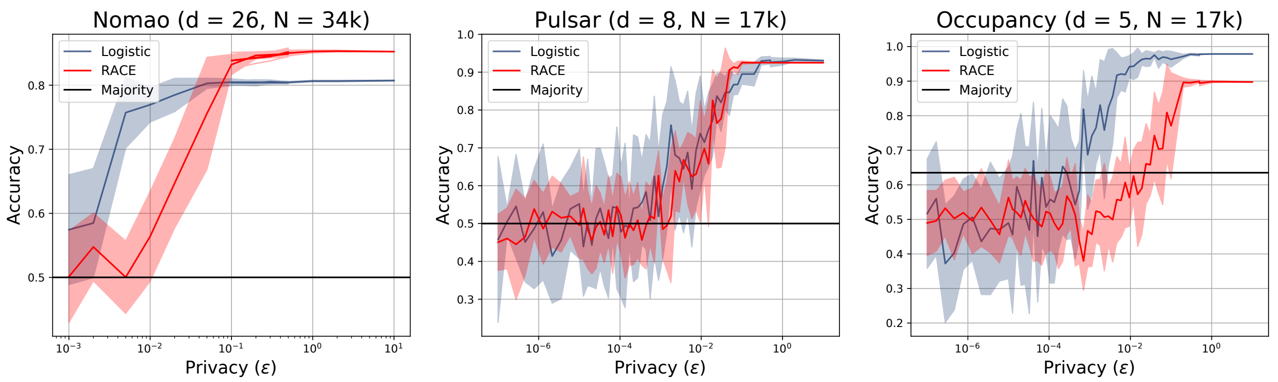

Figure 4: Binary classification experiments. We show the privacy-utility tradeoff for a private logistic regression classifier and the RACE max-likelihood classifier. Average over 10 repetitions.

we divide the dataset evenly among the 90 threads. We merge the sub-sketches by adding the corresponding integer counters. We add the Laplace noise Lap($R\epsilon^{-1}$) to the sketch of the complete dataset after merging is finished.

Results: Figure 6 shows the mean error between our sketch and $f_{\mathcal{D}}$. It should be noted that computing the ground truth sums took several weeks for the Friendster graph, while our sketching algorithm runs in a mere 18 minutes with microsecond queries (Table 4). Unfortunately, we were unable to compare against baseline methods due to the dimensionality and size of the graphs. For example, each row of the Friendster graph (N = 65M) is a sparse binary vector with 65 million dimensions. Even if we were able to project this vector to a subspace embedding of 100 dimensions, the Bernstein mechanism and KME release would be infeasible because they require many kernel sum calculations.

7 EXTENSIONS

The RACE framework can be extended to accommodate private distributed sketching, exotic LSH kernel compositions, and the use of public data to improve the private algorithm.

7.1 Learned Hash Constructions

Until now, we have only considered collision probabilities $k(x, q)$ that have an analytic closed-form expression. We focused on simple LSH functions because they allow us to derive concise theoretical expressions for the estimation error of $f_{\mathcal{D}}$. However, the space of LSH kernels contains a wide variety of exotic kernels.

There are two ways to extend the RACE framework to work for more values of $k(x, q)$. The first method is to construct a new kernel by combining existing LSH functions, using the concatenation

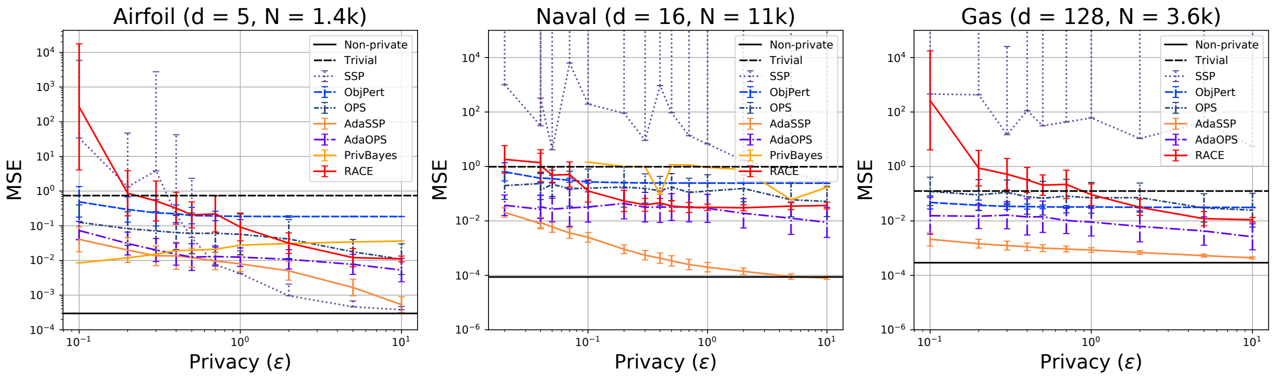

Figure 5: Linear regression experiments. We show the privacy-utility tradeoff for RACE and several other linear regression methods. Average over 10 repetitions.

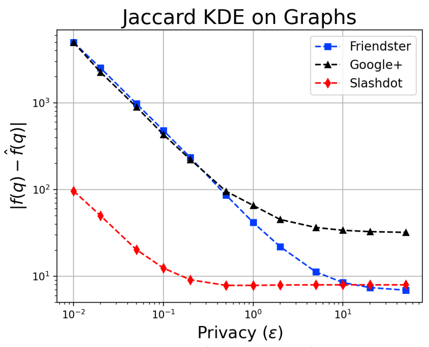

Figure 6: Approximating the Jaccard KDE over large social network graphs. We use $R = 2$k and $W = 100$k except for Friendster, where we use $W = 500$k.

Graph	N	Edges	Sketch	Query	Baselines
Friendster	65M	1.8B	18.1 min	164 μs	> 1 week
Google Plus	72.3K	13M	31 sec	238 μs	> 1 week
Slashdot	82.2K	948K	25 sec	157 μs	> 1 week

Table 4: Large-scale datasets used for graph experiments. The sketch time is reported for our parallel sketch implementation. Query times are an average over ≈2k queries. We were unable to run baseline methods on problems of this size, but based on rough calculations we estimate that it would take weeks to perform function release on these datasets with baseline algorithms.

techniques discussed in [11]. One can obtain a private RACE sketch for any $k(x, q)$ that is a linear combination or product of LSH kernels. To identify the specific components of the new hash function, one could perform an exhaustive search, use mixed-integer linear programming over the space of kernel functions, or apply standard techniques for basis function decomposition.

However, another way to construct the hash function is to adopt methods from the *learning to hash* literature [42]. It is widely known in the deep learning community that hash functions can be learned to improve performance in recommender systems and information retrieval. For example, many algorithms learn embeddings of documents, images, users or words. If these embeddings are later hashed using an LSH function, the RACE sketch estimates a *transformed* version of the collision probability:

$$\Pr[l(g(x)) = l(g(y))] = k(g(x), g(y))$$

Here, $g(x)$ is a transformation that alters the shape of the LSH kernel. We could design $g(x)$ explicitly, but we can also learn $g(x)$ from arbitrary inputs by considering a regression problem that fits $k(g(x), g(y))$ to the desired function.

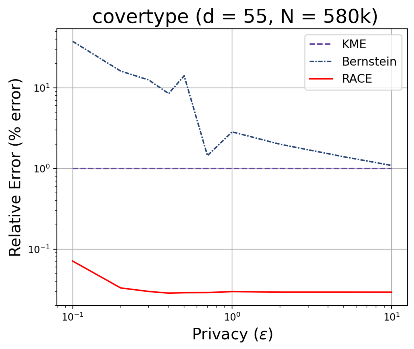

Figure 7: RACE can approximate high-dimensional, large kernel sums. To run Bernstein or KME, we must sub-sample the data to $N = 100$k and apply dimensionality reduction to $d < 10$.

7.2 Access to Public Data

A recent trend in differential privacy is to consider the situation where we have free access to a (small) public dataset and wish to answer queries about a (large) private dataset. In practice, this situation may arise if some users voluntarily disclose information publicly. Several recent works address the question of whether it is possible to improve our private queries using the publicly-available information. For example, the *Public-Assisted Private* framework of [6] provides a way to characterize the number of public samples that are required to improve private learning algorithms whose hypothesis classes obey specific constraints.

The KME paper considers a similar problem, where we are asked to release a synthetic database but are allowed to freely use a *public subset* of the data. For the RACE sketch, public data can be exploited by learning the hash function on the public data points. For example, we can improve our max-likelihood classifier by learning a hash function that causes samples from the same class to collide more frequently with each other than with other classes. There are many standard techniques to learn this type of hash function; see [42] for a comprehensive review.

7.3 Private Distributed Sketching

From a computational perspective, our sketch already possesses the three critical properties that make sketches ideal for distributed settings:

(1) The memory required by our sketch depends mainly on the desired approximation error and is sublinear with respect to N, the number of entries in the dataset.
(2) The sketch can be efficiently constructed in a single pass through the data, without auxiliary data structures or additional storage.
(3) The sketches are *mergeable* - given a sketch $\mathcal{S}_1$ of dataset $\mathcal{D}_1$ and $\mathcal{S}_2$ of $\mathcal{D}_2$, we can obtain a sketch of the combined dataset $\mathcal{D}_1 \cup \mathcal{D}_2$ by simply adding the counters of $\mathcal{S}_1$ and $\mathcal{S}_2$ [2].

It is well-known that these properties are sufficient to enable fast, distributed implementations of the algorithm [2]. However, these properties also enable privacy in the distributed setting. Thanks to federated learning, which requires private aggregation of gradients, a large number of solutions have recently been proposed for the private aggregation problem. Our sketch is compatible with most of these solutions. There are two approaches of interest: multi-party protocols and local noise addition.

Local Noise Addition: This scenario is very well-studied because federated learning requires private centralized aggregation of gradient updates. In the federated learning setting, users add noise to their gradient updates locally before sending the private update to an aggregator. This may be done by adding Gaussian or Binomial noise to obtain local (ϵ, β)-differential privacy as in [1]. These approaches directly apply to our sketch in the same way that they have been applied to the Count Sketch by [27].

Secure Aggregation: If we wish to perform aggregation without a trusted central server, we can also use multi-party protocols to securely compute the sum of count values [7]. However, we must generate noise via a distributed privacy mechanism such as the one proposed in [16] to ensure that these sums preserve privacy (in addition to being computed securely). Recent work by Choi et. al. addresses the specific case of multi-party based aggregation for sketches and is directly applicable to our sketch. To ensure that the count values are protected by ϵ-differential privacy, Choi et. al. add quantized gamma noise to each count before running the MPC protocol so that the sum of gamma noises is Laplace-distributed, then run the MPC protocol to merge the sketches. Our sketches fit into this framework because we only need to run the MPC protocol $R \times W$ times, once for each count.

In summary, our sketch is compatible with a large amount of existing work on private sum aggregation. If RACE sketches are constructed in distributed settings, we do not lose the ability to release private sketches.

8 DISCUSSION

Our experiments show that RACE can privately release useful function summaries for many machine learning problems. Our sketch beats interpolation-based methods for private function release when the dataset has more than a few dimensions and when $f_{\mathcal{D}}$ is not smooth. In our experiments (Figure 3), the Bernstein mechanism outperforms RACE on the SF dataset but fails to capture the nuances of the NYC salary distribution, which has sharp peaks. RACE preserves the details of $f_{\mathcal{D}}$ because the Laplace noise can only make local changes to each hash partition of the RACE structure. For example, if we were to generate an unusually large Laplace noise, only queries in the problematic partition are affected. If we perturb one or two of the most important weights of a series estimator, the changes propagate to all queries.

Although RACE can only estimate LSH kernels, the space of LSH kernels is large enough to be useful for machine learning. For example, RACE performed well on classification tasks for $\epsilon > 10^{-1}$, providing a competitive utility tradeoff for practical privacy budgets. For linear regression, RACE outperforms objective perturbation, a well-established and general method for private learning [8], but not AdaSSP or AdaOPS, which are specialized methods that only work for linear regression but deliver the best possible performance [43]. Therefore, RACE can perform regression and classification with good (though not necessarily state-of-the-art) accuracy.

Function Release at Scale: Our most important result is that the private RACE sketch is orders of magnitude faster than competing algorithms for function release. Although PFDA and the Bernstein mechanism have the strongest theoretical error bounds, they required days to produce the experimental results in Figure 3 while RACE and PrivBayes required a few seconds. This is a serious barrier in practice - it would require $> 2^{128} \approx 10^{38}$ computations and over *one billion exabytes of memory* to run the Bernstein mechanism on the UCI gas dataset. As a result, we were unable to run these methods on the covtype dataset without sampling or dimensionality reduction. The graphs in Table 4, which are much larger, are utterly infeasible for all methods other than RACE, which only requires a few minutes to run. Our sketch has a small memory footprint, inexpensive streaming updates and a fast distributed implementation.

Our sketch is also convenient to use and deploy in a production environment because it is simple to implement and construct. Unlike other algorithms, which require complex construction processes that model correlations between features or release functions on high-dimensional lattices, RACE only requires a 2D array of integers and a hash function. Simplicity is a strong practical advantage because it substantially reduces the likelihood that an incorrect algorithm implementation will lead to a breach of privacy. The simplicity of the sketch also enables straightforward ways to perform hyperparameter selection, since we only need to choose the width W and height R of the 2D array. Although we discuss principled methods to select R in Section 3.3, we found in our experiments that these choices are not critical. Any $R \in [100, 1k]$ with $W > 1k$ will provide good results for many function release problems. Finally, the private RACE sketch is the only function release method that can be constructed in parallel and distributed settings. Although one could possibly merge synthetic point sets or compute Bernstein coefficients in parallel, neither operation has the error-stable merging property enjoyed by our sketch. Therefore, we believe that the private RACE sketch can make private function release a viable tool for real-world applications.

9 CONCLUSION

We have presented a differentially private sketch for a variety of machine learning tasks. RACE is competitive with the state of the art on tasks including density estimation, classification, and linear regression. The sketches can be constructed in the distributed one-pass streaming setting and are highly computationally efficient. At the same time, they offer good performance and make an efficient use of the privacy budget. Given the utility, simplicity and speed of the algorithm, we expect that RACE will enable private machine learning in large-scale settings.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation IIS-1652131, BIGDATA-1838177, AFOSR-YIP FA9550-18-1-0152, ONR DURIP Grant, and the ONR BRC grant on Randomized Numerical Linear Algebra.

REFERENCES

[1] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*. 7564–7575.

[2] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. 23–34.

[3] Francesco Aldà and Benjamin IP Rubinstein. 2017. The bernstein mechanism: Function release under differential privacy. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[4] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences* 58, 1 (1999), 137–147.

[5] Matej Balog, Ilya Tolstikhin, and Bernhard Schölkopf. 2018. Differentially Private Database Release via Kernel Mean Embeddings. In *International Conference on Machine Learning*. 414–422.

[6] Raef Bassily, Albert Cheu, Shay Moran, Aleksandar Nikolov, Jonathan Ullman, and Steven Wu. 2020. Private Query Release Assisted by Public Data. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal DaumÃĺ III and Aarti Singh (Eds.). PMLR, 695–703. http://proceedings.mlr.press/v119/bassily20a.html

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[8] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, Mar (2011), 1069–1109.

[9] Flavio Chierichetti and Ravi Kumar. 2012. LSH-Preserving Functions and Their Applications. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms* (Kyoto, Japan) *(SODA âĂŹ12)*. Society for Industrial and Applied Mathematics, USA, 1078âĂŞ1094.

[10] Benjamin Coleman, Benito Geordie, Li Chou, RA Leo Elworth, Todd J Treangen, and Anshumali Shrivastava. 2019. Diversified RACE Sampling on Data Streams Applied to Metagenomic Sequence Analysis. *bioRxiv* (2019), 852889.

[11] Benjamin Coleman, Gaurav Gupta, John Chen, and Anshumali Shrivastava. 2020. STORM: Foundations of End-to-End Empirical Risk Minimization on the Edge. *arXiv preprint arXiv:2006.14554* (2020).

[12] Benjamin Coleman and Anshumali Shrivastava. 2020. Sub-linear RACE Sketches for Approximate Kernel Density Estimation on Streaming Data. In *Proceedings of the 2020 World Wide Web Conference*. International World Wide Web Conferences Steering Committee.

[13] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. 2009. *Introduction to derivative-free optimization*. Vol. 8. Siam.

[14] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*. 1655–1658.

[15] Cynthia Dwork. 2006. Differential Privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)* (33rd international colloquium on automata, languages and programming, part ii (icalp 2006) ed.) *(Lecture Notes in Computer Science, Vol. 4052)*. Springer Verlag, 1–12. https://www.microsoft.com/en-us/research/publication/differential-privacy/

[16] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[17] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[18] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.

[20] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 17–32.

[21] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, Vol. 99. 518–529.

[22] Rob Hall, Alessandro Rinaldo, and Larry Wasserman. 2013. Differential privacy for functions and functional data. *Journal of Machine Learning Research* 14, Feb (2013), 703–727.

[23] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*. 2339–2347.

[24] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 604–613.

[25] George H. John and Pat Langley. 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (Montréal, Qué, Canada) *(UAIâĂŹ95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 338âĂŞ345.

[26] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[27] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. 2019. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972* (2019).

[28] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[29] Chen Luo and Anshumali Shrivastava. 2018. Arrays of (locality-sensitive) count estimators (ACE): Anomaly detection on the edge. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1439–1448.

[30] Ashwin Machanavajjhala, Xi He, and Michael Hay. 2017. Differential privacy in the wild: A tutorial on current practices & open challenges. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1727–1730.

[31] Ardalan Mirshani, Matthew Reimherr, and Aleksandra Slavković. 2019. Formal Privacy for Functional Data with Gaussian Perturbations. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, Long Beach, California, USA, 4595–4604. http://proceedings.mlr.press/v97/mirshani19a.html

[32] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

[33] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*. PMLR, 8253–8265.

[34] Or Sheffet. 2017. Differentially private ordinary least squares. In *International Conference on Machine Learning*. 3105–3114.

[35] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[36] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*. 2321–2329.

[37] Alexander Strehl and Joydeep Ghosh. 2000. Impact of similarity measures on web-page clustering. In *AAAI-2000: Workshop of Artificial Intelligence for Web Search*.

[38] Apple Differential Privacy Team. 2017. Learning with privacy at scale. *Apple Machine Learning Journal* 1, 8 (2017).

[39] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2019. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.

[40] Jonathan Ullman. 2013. Answering N2+o(1) Counting Queries with Differential Privacy is Hard. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (Palo Alto, California, USA) *(STOC 2013)*. Association for Computing Machinery, New York, NY, USA, 361âĂŞ370. https://doi.org/10.1145/2488608.2488653

[41] Jalaj Upadhyay. 2018. The price of privacy for low-rank factorization. In *Advances in Neural Information Processing Systems*. 4176–4187.

[42] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 769–790.

[43] Yu-Xiang Wang. 2018. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain *(Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence)*. Monterey, California USA, 93–103.

[44] Yu-Xiang Wang, Stephen Fienberg, and Alex Smola. 2015. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In *International Conference on Machine Learning*. 2493–2502.

[45] Ziteng Wang, Kai Fan, Jiaqi Zhang, and Liwei Wang. 2013. Efficient algorithm for privately releasing smooth queries. In *Advances in Neural Information Processing Systems*. 782–790.

[46] Larry Wasserman and Shuheng Zhou. 2010. A statistical framework for differential privacy. *J. Amer. Statist. Assoc.* 105, 489 (2010), 375–389.

[47] David P Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. *Theoretical Computer Science* 10, 1-2 (2014), 1–157.

[48] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739* (2018).

[49] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. 2013. Differentially private histogram publication. *The VLDB Journal* 22, 6 (2013), 797–822.

[50] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.