PAC-Private Algorithms

Mayuri Sridhar
mayuri@mit.edu
MIT CSAIL
Cambridge, Massachusetts, USA

Hanshen Xiao
hsxiao@mit.edu
MIT CSAIL
Cambridge, Massachusetts, USA

Srinivas Devadas
devadas@mit.edu
MIT CSAIL
Cambridge, Massachusetts, USA

ABSTRACT

Provable privacy typically requires involved analysis and is often associated with unacceptable accuracy loss. While many empirical verification or approximation methods, such as Membership Inference Attacks (MIA) and Differential Privacy Auditing (DPA), have been proposed, these do not offer rigorous privacy guarantees. In this paper, we apply recently-proposed Probably Approximately Correct (PAC) Privacy to give formal, mechanized, simulation-based proofs for a range of practical, black-box algorithms: K-Means, Support Vector Machines (SVM), Principal Component Analysis (PCA) and Random Forests. To provide these proofs, we present a new simulation algorithm that efficiently determines *anisotropic* noise perturbation required for any given level of privacy. We provide a proof of correctness for this algorithm and demonstrate that anisotropic noise has substantive benefits over isotropic noise.

Stable algorithms are easier to privatize, and we demonstrate privacy amplification resulting from introducing regularization in these algorithms; meaningful privacy guarantees are obtained with small losses in accuracy. We also propose new techniques in order to *canonicalize* algorithmic output and convert intractable geometric stability verification into efficient deterministic stability verification. Thorough experiments are included, and we validate our provable adversarial inference hardness against state-of-the-art empirical attacks.

KEYWORDS

PAC Privacy; Differential Privacy; Black-box Security Proof; Inference Hardness; Membership Attack.

1 INTRODUCTION

The expansion of data collection and increasing complexity of data processing are happening at unprecedented rates. Concerns on information leakage are receiving increasing attention, while privacy preservation is simultaneously challenged by fast-paced and sophisticated advancements. Efficient and widely-applicable risk quantification has become a fundamental and urgent problem in privacy research.

Most existing provable privacy analyses of data processing require strong algorithmic assumptions. For example, Differential Privacy (DP) [11] requires bounded sensitivity [1], which can only be tightly computed in a few simple applications such as aggregation or linear queries; Maximal Leakage (MaxL) [18] requires the knowledge of the likelihood function produced by each input selection, and thus is not applicable to a continuous or infinite input space. Moreover, to ensure these *input-independent* indistinguishability guarantees, artificial modifications are typically required to decompose most algorithms into multiple simpler and analyzable components, such as mean estimation or majority voting to enable tractable analysis; Differentially-Private Stochastic Gradient Descent (DP-SGD) [1] and PATE [25] are representative examples. Unfortunately, artificial modifications usually come with limits on algorithms and data structures, and often with a significant compromise on utility.

As a consequence, the lack of powerful risk quantification tools heavily restricts the study and design of defensive methods for leakage control, as the privacy implications of many operations are not well-understood. Even for perturbation, the most popular and straightforward privacy-preserving technique, the minimal noise to produce required security parameters, largely remains open for most practical algorithms. In addition, the definition of sensitive information varies across different processing tasks and different individual preferences. For example, for image data, people may worry about whether the adversary can reconstruct sensitive face features; for health data, the privacy objective can be the relationship between certain associations between patients and diseases; and in anonymous communication, identities are sensitive. Universal risk quantification is thus highly desirable to capture diverse and customized concerns.

Besides provable analyses, there is also a long line of works focusing on empirical defenses against adversarial inference. Privacy verification has been extensively studied, in particular, for membership inference attacks (MIA) [5, 17, 31]. For example, many operations such as regularization [24, 31], data augmentation [20, 38], and model compression [33] are empirically shown to resist certain kinds of attacks. However, qualitative analysis for those strategies is challenging and largely remains open, especially in involved data processing algorithms. Though carefully-designed empirical simulations can provide meaningful approximation of the lower bound of privacy risks with respect to *specific* adversarial strategies, a rigorous proof is desired to show worst-case guarantees against *arbitrary* adversaries. Closing this gap remains a key and open problem in security and privacy research.

One recent effort to technically address the risk quantification for black-box data processing is PAC Privacy [34]. From a statistical inference perspective, Xiao and Devadas [34] develop a new framework to semantically interpret privacy risk as concrete inference hardness for a computationally-unbounded adversary to recover sensitive information satisfying a certain criterion, which can be arbitrarily selected. A set of new tools are also established in Xiao and Devadas [34] to provably convert the objective inference hardness into simulatable quantities, which enables high-confidence estimation from end-to-end black-box simulations to provide a privacy proof. However, as a theoretical solution to conduct privacy analysis for a black-box processing, there are two important aspects of PAC Privacy which have not been systematically explored. First, how

[1] In the context of DP, sensitivity captures the worst-case influence of an individual on the output, which is in general NP-hard to compute [37].

can we *efficiently* determine the (near-)optimal *anisotropic* noise[2] to add to each exposed output, and provide an associated privacy proof? Second, how can we *stabilize* a black-box data processing algorithm to provably produce a stronger privacy guarantee or a sharpened utility-privacy tradeoff?

In this paper, we contribute an initial comprehensive study to answer these questions, as summarized below.

(1) **Novel algorithm for efficient simulation proofs**: We present an algorithm in Section 4 that adds anisotropic noise, but is more computationally efficient than the algorithm (Algorithm 1) in [34] which requires running Singular Value Decomposition (SVD) on the *entire* output dimension, and can be prohibitively expensive. Efficiency is further enhanced through faster convergence; our algorithm only needs to accurately estimate variance of each output, as opposed to converging on a covariance matrix as in [34]. We prove the correctness of our algorithm.
(2) **Efficient privatization for black-box algorithms**: We implement PAC-private versions of several classic algorithms. We provide noise estimates and utility tradeoffs, demonstrating that these algorithms can generally achieve meaningful privacy with a small losses in utility. We show that adding anisotropic noise has significant utility benefits over adding isotropic noise using l_2-norm estimation.
(3) **Sharpening privacy-utility tradeoffs**: We first characterize the root of instability in these classic algorithms, and separate them into two large categories – superficial and intrinsic. Then, we provide novel canonicalization techniques to improve upon superficial instability, while exploring classic and novel techniques, based on regularization and data augmentation techniques, to improve intrinsic instability. In particular, we show how the use of a random unitary matrix in Principal Component Analysis (PCA) can essentially eliminate superficial instability.
(4) **Empirical verification for end-to-end privacy**: Finally, we provide experimental support, based on simulated attacks to validate our privacy guarantees. We convert the theoretical mutual information guarantees into posterior guarantees and demonstrate that our privatized algorithms more than satisfy these guarantees against state-of-the-art attacks.

2 BACKGROUND

We first introduce the PAC Privacy model to describe information leakage and privacy risk in general. Let X denote the sensitive input, which is randomly generated from a (possibly black-box) distribution D, and $\mathcal{M}$ denote a (possibly black-box) processing mechanism, where the output, $\mathcal{M}(X)$, is released and observed by an adversary. We challenge the adversary as to whether they can return some estimation $\tilde{X}$ satisfying a certain criterion, which can be described by some indicator function ρ. Such an inference challenge can be used to capture arbitrary privacy concerns and customized leakage control that a user is comfortable with. For example, to capture a membership inference attack [31], ρ can be selected as $\rho(X, \tilde{X}) = 1$ if $\tilde{X}$ predicts the membership of some particular datapoint u_0 correctly in X; ρ may also capture data reconstruction [3, 16] and we may set $\rho(X, \tilde{X}) = 1$ iff $\|X - \tilde{X}\|_2 \leq 1$, i.e., the adversary can recover the input with error in l_2-norm smaller than 1. For side-channel attacks on a cryptographic protocol [15], where X corresponds to the secret key, ρ can capture the colliding bits between X and $\tilde{X}$.

Now, given the data entropy, determined by D, and the objective inference task, we can define the optimal *a priori* success rate $(1 - \delta_o^\rho)$ that an adversary can return a satisfied estimation before they observe the release $\mathcal{M}(X)$, i.e.,

$$\delta_o^\rho = \min_{\tilde{X}_o} \Pr_{X \sim \mathrm{D}} (\rho(X, \tilde{X}_o) \neq 1).$$

Similarly, we can define the posterior success rate $(1 - \delta)$ to capture the probability for an adversary to return a satisfied estimation after observing the release. With the above preparation, we can now formally define PAC Privacy.

Definition 1 ($(\delta, \rho, \mathrm{D})$ PAC Privacy [34]). *For a processing function $\mathcal{M} : \mathcal{X}^* \to \mathcal{O}$, some data distribution* D*, and an inference criterion function $\rho(\cdot, \cdot)$, we say $\mathcal{M}$ satisfies $(\delta, \rho, \mathrm{D})$-PAC Privacy if the following experiment is impossible:*

A user generates data X from distribution D *and sends $\mathcal{M}(X)$ to an informed adversary. The adversary who knows* D *and $\mathcal{M}$ is asked to return an estimation $\hat{X}$ on X such that with probability at least $(1 - \delta)$, $\rho(\hat{X}, X) = 1$.*

Equivalently, $\mathcal{M}$ can be defined as $(\Delta_f \delta, \rho, \mathrm{D})$ PAC-advantage private if the posterior advantage measured in f-divergence satisfies

$$\Delta_f \delta = \mathcal{D}_f(\mathbf{1}_\delta \| \mathbf{1}_{\delta_o^\rho}) = \delta_o^\rho f(\frac{\delta}{\delta_o^\rho}) + (1 - \delta_o^\rho) f(\frac{1 - \delta}{1 - \delta_o^\rho}),$$

where $(1 - \delta_o^\rho)$ represents the optimal prior success rate,

$$\delta_o^\rho = \min_{X' \in \mathcal{X}^*} \Pr_{X \sim \mathrm{D}} (\rho(X', X) \neq 1),$$

and $\mathbf{1}_\delta$ and $\mathbf{1}_{\delta_o^\rho}$ represent two Bernoulli distributions of parameters δ and δ_o^ρ, respectively. Here, $\mathcal{D}_f$ is some f-divergence.

In [34], $\mathcal{D}_f$ is selected to be the KL-divergence and it is shown that,

$$\Delta_{KL} \delta = \mathcal{D}_{KL}(\mathbf{1}_\delta \| \mathbf{1}_{\delta_o^\rho}) \leq \mathsf{MI}\big(X; \mathcal{M}(X)\big), \tag{1}$$

where $\mathsf{MI}(\cdot, \cdot)$ represents *mutual information* and $\mathcal{D}_{KL}(\mathbf{1}_\delta \| \mathbf{1}_{\delta_o^\rho}) = \delta \ln(\frac{\delta}{\delta_o^\rho}) + (1 - \delta) \ln(\frac{1-\delta}{1-\delta_o^\rho})$.

We now define the standard Membership Inference Attack (MIA) [31], formalized to match PAC Privacy below.

Definition 2 (Membership Inference Attack). *Given a finite data pool $\mathsf{U} = \{u_1, u_2, \cdots, u_N\}$ and some processing mechanism $\mathcal{M}$, X is an n-subset of* U *randomly selected. An informed adversary is asked to return an n-subset $\hat{X}$ as the membership estimation of X after observing $\mathcal{M}(X)$. We say $\mathcal{M}$ is resistant to $(1 - \delta_i)$ individual membership inference for the i-th datapoint u_i, if for an arbitrary adversary, $\Pr_{X \leftarrow \mathsf{U}, \tilde{X} \leftarrow \mathcal{M}(X)}(\mathbf{1}_{u_i \in X} = \mathbf{1}_{u_i \in \hat{X}}) \leq 1 - \delta_i$. Here, $\mathbf{1}_{u_i \in X}$ $(\mathbf{1}_{u_i \in \hat{X}})$ is an indicator which equals 1 if u_i is in X $(\hat{X})$.*

In this paper, we will use PAC Privacy to provably and automatically measure the privacy risk. We will also qualitatively (and occasionally quantitatively) compare our results to prior work with Differential Privacy (DP); its formal definition is presented below.

[2] Noise varying across output dimensions

Definition 3 ($(\epsilon, \bar{\delta})$ Differential Privacy [11]). *Given a data universe $\mathcal{X}^*$, we say that two datasets $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{X}^*$ are adjacent, denoted as $\mathcal{S} \sim \mathcal{S}'$, if $\mathcal{S} = \mathcal{S}' \cup s$ or $\mathcal{S}' = \mathcal{S} \cup s$ for some additional datapoint s. A randomized processing function $\mathcal{M}$ is said to be $(\epsilon, \bar{\delta})$-differentially-private (DP) if for any pair of adjacent datasets $\mathcal{S}, \mathcal{S}'$ and any set o in the output space O of $\mathcal{M}$, it holds that* $\Pr(\mathcal{M}(\mathcal{S}) \in o) \leq e^{\epsilon} \cdot \Pr(\mathcal{M}(\mathcal{S}') \in o) + \bar{\delta}$.

We can interpret DP in a context of the posterior success rate for successful membership inference. In the same setup of Definition 2, if $n = \frac{N}{2}$ [3], i.e., each datapoint is included in X with probability $1/2$, and $\mathcal{M}$ satisfies $(\epsilon, \bar{\delta})$-DP, then by [19], the posterior success rate $(1 - \delta_i)$ is upper bounded by

$$1 - \delta_i \leq 1 - \frac{1 - \bar{\delta}}{1 + e^{\epsilon}}. \tag{2}$$

3 AUTOMATIC PRIVATIZATION

3.1 A template for provable privacy

In this section, we present a formal template for privatizing black-box algorithms using PAC Privacy. The key steps of this technique are summarized in Figure 1.

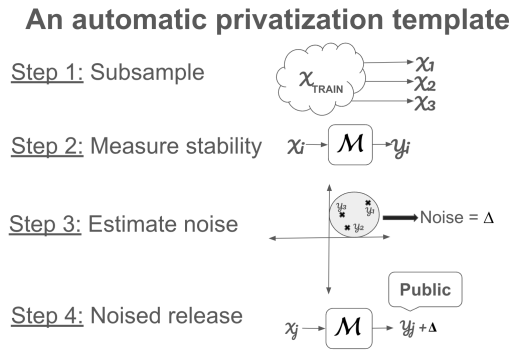

Figure 1: A simple 4-step process to automatically privatize a black-box algorithm $\mathcal{M}$. We first measure the stability of $\mathcal{M}$ by computing $Y_i = \mathcal{M}(X_i)$ on varying subsets of data X_i. We then use the variance of the output distribution Y_i to estimate the required noise necessary to privatize $\mathcal{M}$. Finally, we release a noisy version of the learned vector.

In particular, we consider any black-box algorithm $\mathcal{M}$. The goal of our template is to release $Y = \mathcal{M}(X)$ for a **secret** input X. We want to bound the *posterior* advantage the adversary gains upon observing $Y = \mathcal{M}(X)$ by adding noise to Y. Y is an arbitrary learned statistic about the input X that is **exposed** to the adversary. We denote X_{train} as the complete training dataset which X is sampled from. We then subsample $X_1 \cdots X_m \subset X_{train}$, which are independent and identically distributed subsets of X_{train} and $|X_i| = |X|$. We denote

$$r := \frac{|X_i|}{|X_{train}|}$$

as the subsampling rate of the privatization procedure. For our experiments, we choose $r = 0.5$, or 50%. In our privacy analysis, we make the conservative assumption that the adversary observes X_{train}, which is not typically true in the real world. Importantly, the computed posterior advantage will hold for this adversary or a weaker one with only partial or no knowledge of X_{train}. Note that the **sampled X is hidden from the adversary**, and therefore the participation of a particular data element in X is unknown to the adversary.

We represent each $x \in X_i$ as a d-dimensional vector, with its l_2-norm bounded by a known constant. In order to provide a private representation of $Y = \mathcal{M}(X)$, we follow the framework of PAC Privacy [34] to determine the minimal noise we must add to the vector Y. We aim to minimize the noise in order to maximize the *utility* of the output vector Y. Following the steps described in Figure 1, we can use the stability of $\mathcal{M}$ on distinct X_i's in order to determine the required noise to provide privacy for $\mathcal{M}$.

To do this, we first evaluate $\mathcal{M}$ on distinct subsampled datasets $X_1 \ldots X_m$, producing output vectors $Y_1 \ldots Y_m$. We can then use the (co)variance of the Y_i's (along with appropriate security parameters) to estimate the minimum noise required to add to the output of $\mathcal{M}$, in order to provide a meaningful bound on the mutual information, which in turn bounds the posterior advantage.

The posterior advantage holds for an *arbitrary* inference task ρ on the input dataset X. The classic membership attack [31] defines a specific ρ, where the goal is to determine whether a fixed sample x was included in X; that is $\rho(x, X) = 1$ if the adversary correctly guesses if $x \in X$. Other attacks may include reconstruction attacks (recovering X), norm estimation, or several more [3, 16].

We make a few key observations about this template:

(1) $\mathcal{M}$ is treated as a black-box. The magnitude of the added noise depends solely on the output distribution of Y_i's. This allows for us to generate a privacy template for complex *black-box* algorithms, in an instance-specific (i.e., specific to X_{train}) manner.
(2) We observe that the magnitude of noise added only impacts the *posterior* advantage of the inference task. We make *no* assumptions on the specific inference task of the adversary; rather, PAC privacy allows us to bound the mutual information between the output Y and the secret input X, bounding the maximal posterior advantage. We further discuss the relationship between mutual information and the posterior advantage for specific membership inference attacks in Section 8.
(3) In order to meaningfully measure the (co)variance across Y_i's, the outputs on varying inputs X_i must lie on the same output space. In particular, we must *canonicalize* our outputs. That is, if we assume Y_i is a learned vector, it must remain in the same order and even simpler, the same length. For certain tasks, like regression, this appears obvious, since a learned weight vector has fixed dimension and order. However, for unsupervised learning tasks or even certain classification algorithms, this becomes non-trivial.

We can then use Equation (1) to convert the mutual information guarantee to arbitrary inference maximal posterior advantage. We can expand Equation (1) below as

$$p_o \ln\left(\frac{p_o}{p}\right) + (1 - p_o) \ln\left(\frac{1 - p_o}{1 - p}\right) \leq MI(X_i; Y_i) \tag{3}$$

[3] For the general case, one can perform similar reasoning by solving a constrained linear program with respect to Type I and Type II errors as described by Eqn. (1) in [19].

Mutual Information	Posterior Success Rate (p_o)	
	Prior p = 50%	Prior p = 1%
1/64	58.815%	3.213%
1/32	62.434%	4.364%
1/16	67.490%	6.200%
1/8	74.464%	9.171%
1/4	83.789%	14.057%
1/2	95.181%	22.177%
1	100%	35.729%
2	100%	58.103%
4	100%	92.582%

Table 1: Mutual information can be related to the theoretical maximal posterior success rate for different prior success rates using Equation (3).

where p is the prior success rate and p_o is the posterior success rate. Table 1 provides the theoretical maximal posterior success rates for two different prior success rates of 50% and 1%. The prior success rate p for a subsampling rate r equals $\max(r, 1-r)$ for an *individual* membership inference task; we choose $r = 0.5$ to minimize p to 50%.[4] However, p can be much lower for a generalized membership inference task for the same r (e.g., 1%) (cf. Section 8.2).

We can use Equation (2) to interpret $(\epsilon, \bar{\delta})$-DP parameters as posterior success rate. For example, a (0.36,0)-DP ((2.98,0)-DP) corresponds to a posterior success rate of 58.815% (95.181%) for a prior of 50%. This is useful in calibrating mutual information in Table 1 with a DP ϵ.

3.2 Privacy vs. utility

In this section, we focus on a technique we denote as *canonicalizing* the output distributions. That is, we first classify varying causes of instability in the output distributions for a black-box algorithm $\mathcal{M}$. In general, we can consider two major causes of instability for an algorithm:

(1) **Intrinsic instability:** We denote an algorithm's intrinsic instability as instability that cannot be reduced without *semantically* changing the output of the algorithm.
(2) **Superficial instability:** We denote an algorithm's superficial instability as an instability in the output that does *not* reflect a semantic difference in the output.

In this work, we explore techniques to reduce both types of instability in a broad set of widely-used algorithms.

We first consider a simple example of superficial instability in unsupervised learning algorithms. In general, unsupervised learning algorithms provide a mechanism for *clustering*. However, by definition, these clusters do not have labels. Thus, an algorithm could return the same set of clusters in any order; while the ordered vector appears very different, the true result is semantically the same. In this case, reducing the instability of the algorithm is near-trivial; we can simply assign arbitrary labels to each cluster and choose labels to minimize the variance across Y_i's.

We now consider an example of intrinsic instability. In this, we consider the random forest algorithm. The goal of this algorithm is to classify different classes within a dataset, by constructing several decision trees. Each decision tree chooses a subset of features to train on; then, each level of the tree splits the dataset into subsets in order to minimize entropy or Gini impurity [4]. These algorithms are known to be unstable, since small changes to the input dataset can lead to significant changes in the threshold values. In Section 5, we discuss how we modify this algorithm to provide meaningful guarantees in our framework.

Finally, we note that in the classic non-private setting for these algorithms, stability is useful primarily as a proxy for understanding the generalizability of these algorithms. However, in our setting, stability directly affects the utility of the privatized algorithm, since it is inversely correlated with the total added noise. This implies that efficiently privatizing these algorithms involves an inner optimization problem, similar to the hyperparameter search typically done using cross-validation. We discuss heuristic strategies for this search and empirical results in Section 6.

[4]The prior success rate of positive identification of individual membership equals r.

4 EFFICIENTLY COMPUTING ANISOTROPIC NOISE

In this section, we formally describe a "best of both worlds" algorithm that is as efficient as the isotropic noise addition algorithm of [34] while computing anisotropic noise.

We then prove that the noise mechanism satisfies the mutual information guarantees. The algorithm in full is described in Algorithm 1. We denote n as the number of input elements, A as a unitary projection matrix, m as the number of trials, and d as the output dimension. After computing $\Sigma_{\mathcal{B}}$, we add Gaussian noise $\mathcal{B} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathcal{B}} A^T)$ to each element of the output $\mathcal{M}(X)$. For our experiments, we choose $A = \mathbf{I}_d$.

Algorithm 1 Anisotropic Noise Determination of Deterministic Mechanism $\mathcal{M}$

Input: A deterministic mechanism $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}^d$, data distribution $\mathcal{D}$, mutual information requirement β, number of trials m, $d \times d$ unitary projection matrix A.

(1) **for** $k = 1, 2, \ldots, m$:
  (a) Generate data $X^{(k)}$ from $\mathcal{D}$.
  (b) Record $y^{(k)} = \mathcal{M}(X^{(k)})$.
(2) For each $y^{(k)}$, calculate $g_i^{(k)} = y^{(k)} \cdot A_i$.
(3) Calculate the empirical variance σ_i for each g_i.
(4) Calculate the required noise in each direction i as

$$e_i := \frac{\sqrt{\sigma_i} \sum_{i=1}^{d} \sqrt{\sigma_i}}{2\beta}.$$

(5) Return a diagonal matrix $\Sigma_{\mathcal{B}}$, where $\Sigma_{\mathcal{B}}[i][i] = e_i$.

Theorem 1. *For an arbitrary deterministic mechanism $\mathcal{M}$, a public unitary matrix A, and Gaussian noise of the form $\mathcal{B} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathcal{B}} A^T)$, where $\sigma_i = Var(\mathcal{M}(X) \cdot A_i)$ and Σ_B is a diagonal matrix with entries*

$$e_i := \frac{\sqrt{\sigma_i} \sum_{i=1}^{d} \sqrt{\sigma_i}}{2\beta},$$

the output $M(X) + \mathcal{B}$ satisfies

$$MI(X; \mathcal{M}(X) + \mathcal{B}) \leq \beta.$$

Proof. We first recall Theorem 3 of Xiao and Devadas [34]; this theorem states that,

$$MI(X; \mathcal{M}(X) + \mathcal{B}) \leq \frac{1}{2} \ln \det(\mathbf{I}_d + \Sigma_{\mathcal{M}(X)} \Sigma_{\mathcal{B}}^{-1}).$$

We then note that

$$MI(X; \mathcal{M}(X) + \mathcal{B}) = MI(X; \mathcal{M}(X)A + \mathcal{B}A),$$

since A is unitary and public.

By Hadamard's inequality, since $\Sigma_{\mathcal{M}(X)A}$ is positive semi-definite,

$$\det(\Sigma_{\mathcal{M}(X)A}) \leq \det(\mathrm{diag}(\Sigma_{\mathcal{M}(X)A})),$$

where $\mathrm{diag}(\Sigma_{\mathcal{M}(X)A})$ is the diagonal matrix with i'th element σ_i. By construction, $\mathcal{B}A$ has variance $\Sigma_{\mathcal{B}}$, which is a diagonal matrix with elements e_i. Thus,

$$\begin{aligned}
MI(X; \mathcal{M}(X)A + \mathcal{B}A) &\leq \frac{1}{2} \ln \det(\mathbf{I}_d + \Sigma_{\mathcal{M}(X)A} \Sigma_{\mathcal{B}}^{-1}) \\
&\leq \frac{1}{2} \ln \det(\mathbf{I}_d + \mathrm{diag}(\Sigma_{\mathcal{M}(X)A}) \Sigma_{\mathcal{B}}^{-1}) \\
&= \frac{1}{2} \ln \prod_i (1 + \sigma_i \frac{2\beta}{\sqrt{\sigma_i} \sum_i \sqrt{\sigma_i}}) \\
&= \frac{1}{2} \sum_i \ln(1 + \frac{2\beta\sqrt{\sigma_i}}{\sum_i \sqrt{\sigma_i}}) \\
&\leq \frac{1}{2} \sum_i \frac{2\beta\sqrt{\sigma_i}}{\sum_i \sqrt{\sigma_i}} \\
&\leq \beta,
\end{aligned}$$

where the fifth inequality uses the fact that $\ln(1 + x) \leq x$.

□

The primary advantage of our algorithm is that it avoids building the covariance matrix and subsequent SVD (as in Algorithm 1 of [34]), while determining sufficient anisotropic noise for privacy. The optimal A for minimal noise can be determined by estimating the covariance matrix and using SVD. Using the identity matrix and thus only estimating the variance of each output further results in requiring substantially fewer trials (lower m) for convergence (cf. Section 7).

5 ALGORITHMS

In this section, we discuss several classic algorithms and the required modifications to automatically privatize them.

For all the algorithms, we first normalize our data and separate it into a training dataset and a test dataset. We then measure "accuracy" (also referred to as utility) on the test dataset.

5.1 Clustering: K-Means

The K-Means clustering algorithm, originally developed by Lloyd in 1982, aims to partition an input set X into K non-overlapping subsets or *clusters* [2, 23]. Each subset $i \in [1, K]$ is defined by its centroid, μ_i. The objective is to minimize the sum-of-squares within each cluster, over all the clusters, i.e.,

$$\arg\min_{\mu} \sum_{i=0}^{n} \min_{\mu_j} \left\| x_i - \mu_j \right\|^2.$$

That is, the classic algorithm outputs a list of centroids, corresponding to each cluster. We observe that K-Means requires minimal changes to fit into our PAC privacy framework. The x_i's are the **secret** input, and the learned centroids μ_i's are the **exposed** output.

In order to canonicalize the output, we simply order these centroids by inferring appropriate cluster labels. For supervised learning, we do this by choosing the label that is best associated with each cluster. We then measure test accuracy by comparing the inferred cluster label with the true class label on the test dataset.

5.2 Classification: SVM

Consider the multi-class support vector machine algorithm [6, 13]. The linear support vector machine problem solves the following optimization problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b)). \tag{4}$$

Here, the x_i's are the features, and the y_i's are the labels; these both correspond to the **secret** inputs. The learned weight vector w, b is the **exposed** output.

We use the regularization weight C to trade off between the hinge loss and the norm of the learned weight vector. Without any modification, the standard value of C used is 1. To accommodate multi-class strategies, we consider a one-versus-rest classification strategy. That is, we train K classifiers for K distinct classes [28].

After the weight vector has been trained, we can use it to compute a "per-class" score for a new point x_i. The predicted label $\hat{y}_i$ is the class with the highest score. Similar to K-Means, we measure the accuracy of the test dataset by computing the class label predicted by SVM to the true label.

We note that this algorithm may or may not have a lot of superficial instability. That is, the learned weight vectors are inherently ordered by the labels of their corresponding classes; it thus requires almost no modification to fit into the PAC privacy framework. However, there may be several near-optimal solutions with no obvious ordering when regularization is not applied appropriately. Strong regularization (low values of C) can reduce the algorithm's intrinsic instability, though it may come with a utility tradeoff.

5.3 Dimensionality Reduction: PCA

Consider the classic dimensionality reduction algorithm, principal component analysis (PCA) [27]. PCA is used to decompose a multivariate dataset into orthogonal components that explain the most variance.

Unlike the other algorithms considered, PCA is not independently used for a regression or classification task; rather, it is typically a subroutine. We consider an initial data matrix $X \in \mathbb{R}^{m \times d}$, with m samples of dimension d, where $X \subset X_{train}$ is **secret**. We then reduce the dimensionality of X to be in $\mathbb{R}^{m \times d'}$ using PCA;

that is, we compute the top d' principal components and denote them as a matrix $S \in \mathbb{R}^{d \times d'}$. S is the **exposed** output.

We observe that PCA has significant superficial instability. In particular, we consider the subspace defined by the basis vectors $[0, 1]$ and $[1, 0]$; this subspace is $\mathbb{R}^2$. However, there are an infinite number of basis vectors with the same span; in fact, any two linearly-independent vectors span $\mathbb{R}^2$. This implies that two calls to the PCA algorithm can return the *same* subspace, represented by *significantly* different basis vectors.

In order to canonicalize the basis vectors, we consider two instances of the PCA algorithm and denote the returned basis vectors as S_1 and S_2, where $|S_i| = d'$ for $i = 1, 2$. We observe that we can choose a unitary matrix M and compute MS_2 as an equivalent representation of the basis chosen by S_2. The goal is now to choose M in order to minimize the distances between S_1 and S_2; we use this formulation and the properties of singular value decomposition (SVD) to compute the optimal M. We note that the SVD decomposition is unique up to the sign of the right and left singular vectors.

Consider the following optimization problem:

$$\min_{M; M^T M = I} \quad \|A - MB\|_F^2,$$

where A, B are matrices of basis vectors, with dimensionality $d' \times d$ and M is any unitary matrix. We first observe that this models our PCA problem *exactly*; that is, PCA returns to us a set of d' basis vectors with dimensionality d. We can freely optimize over the matrix M as long as it remains unitary, since M is simply a linear map.

Claim 1. *The optimal choice for M is of the form $M = C[0 : d', 0 : d']$, where*

$$C = V_A V_B^T,$$

and V_A, V_B are the right singular vectors of A and B, respectively.

Proof. We prove this directly from the optimization problem. That is, M is chosen to minimize

$$\min_{M; M^T M = I} \quad \|A - MB\|_F^2,$$

for fixed matrices A and B of dimension $d' \times d$. We suppress the unitary requirement on M for succinctness in the remainder of this argument. We first denote the SVDs of A and B as $U_A \Sigma_A V_A$ and $U_B \Sigma_B V_B$, respectively. We observe that $U_A \in \mathbb{R}^{d' \times d'}$, $\Sigma_A \in \mathbb{R}^{d' \times d}$, $V_A \in \mathbb{R}^{d \times d}$. We further note that Σ_A has d' real values on the diagonal and the remaining entries are 0, since the underlying rank of A is assumed to be d'. We can thus denote Σ_A^{-1} as the inverse of Σ_A where $\Sigma_A^{-1} \Sigma_A = [I_{d'} | 0]$. The same constraints follow for B.

We then observe that

$$\begin{aligned}
&\min_M \quad \|A - MB\|_F^2 \\
&= \min_M \quad \|U_A \Sigma_A V_A - M U_B \Sigma_B V_B\| \\
&= \min_M \quad \left\| \Sigma_A^{-1} U_A^T U_A \Sigma_A V_A - \Sigma_A^{-1} U_A^T M U_B \Sigma_B V_B \right\| \\
&= \min_{M'} \quad \left\| [I_{d'} | 0] - M' [I_{d'} | 0] V_B V_A^T \right\|.
\end{aligned}$$

We note that the fourth equality switches from optimizing over M to optimizing over a different matrix M' after factoring out the relevant components of the SVD of B; however, since M and M' are both free variables, this does not affect correctness. The optimal solution for this is $M = V_A V_B^T$, truncated to the first d' rows and columns. □

By canonicalizing our output, we reduce the superficial instability; "nearby" subspaces are represented by "nearby" basis vectors. Without the appropriate canonicalization, the PCA algorithm is very unstable and difficult to privatize.

For a given input X_{test}, after running PCA, the projection of X_{test} is computed as $X_{test} S$, representing the best projection of X_{test} into the learned rank-d' subspace. We can then "restore" X_{test} into the original rank-d subspace by computing $X_{test} S S^T$, also known as the PCA inverse transform; we denote this matrix as X' and calculate the restoration error as

$$\text{Restoration error (RE)} := \frac{\|X' - X_{test}\|}{\|X_{test}\|}. \tag{5}$$

We use the restoration error as a proxy of our accuracy metric for other algorithms, since low RE would imply high success rate on any secondary task.

5.4 Boosting: Random Forest

As mentioned in Section 3, random forest algorithms typically involve both superficial *and* intrinsic instability, making them an interesting case study for our template.

We first describe the classic random forest algorithm [4] and then describe our modifications for canonicalization. The classic random forest algorithm is an ensemble learning technique which combines several weak classifiers (decision trees) to make an ensemble model which performs better than any of the individual trees. Typically, these decision trees are trained on subsets of the provided dataset and the final classification is the plurality vote of the individual trees. For each tree, the algorithm chooses a feature (or subset of features) to split on. Then, the "value" to split on is chosen to minimize a metric – in our case, we use the metric of weighted entropy. The provided dataset is the secret input, and the learned trees are the exposed output: a number of trees with corresponding structures and weights.

In our setting, we require that the trees all have the same *structure*. To simplify this, we ensure that all the trees are complete and split on the same order of features. Thus, our random forest algorithm has two hyperparameters: the number of trees (a classic requirement for an ensemble model) and the *ordered list* of the features to split on for each tree, denoted here as L. The latter is unique to our setting and is required for us to measure the variance across different trials. The structure of a tree is fully determined from the ordered set of features; that is, if there are d features, then the tree will have exactly 2^d leaf nodes. Each node at level i will split on feature $L[i]$; the exact value of the split threshold is determined by computing the minimum weighted entropy across all possible values of the feature $L[i]$.

In particular, each possible "split" on the feature $L[i]$ at value v splits the dataset into two sets $S_r(v)$ and $S_l(v)$. We can calculate the entropy of each split as

$$H(S) = \sum_i -p_i \log p_i,$$

where p_i is the empirical probability of element i (the frequency of item i in S divided by $|S|$). The total entropy of a split can be calculated as

$$H_v := |S_l(v)|H(S_l(v)) + |S_r(v)|H(S_r(v)).$$

We then choose the split that minimizes the weighted entropy.

We consider the choice of ordered features akin to early work in bagging schemes, where subsets of features were chosen for each tree. We pass in all the data to each decision tree and output a simple majority vote of the trees as the final decision of our random forest. We note that classic regularization schemes on decision trees (or random forests) focus on pruning the depth of the tree or allowing the tree to split on a maximum number of features at each level [4]. Neither of these are consistent with our framework. In particular, the former does not allow for an efficient canonicalization since the trees will have different structures. The latter is irrelevant for us, since our trees split on a single feature at each level.

In this setting, the features x_i and the labels y_i for our training data represent our **secret** input. The coefficients of the learned trees represent the **exposed** output. As with the prior algorithms, after the trees are exposed, we can measure the test accuracy by comparing the learned classification of a test data point to its true label.

We use two techniques intended to increase the stability of the random forest algorithm, following the form of regularization and data augmentation defenses suggested in Nasr et al. [24] and Kaya and Dumitras [20]. First, we define a *data augmentation* defense. That is, we first discretize the possible split values of each level. Thus, the possible split values of a feature $L[i]$ are in the range $[0, 1]$, evenly divided into $1/p$ segments of length p, where p is a tunable hyperparameter (typically 0.01). Then, rather than just calculating the entropy of the split, we calculate our final split value as

$$v := \arg\min_v \quad (1 - w_1)H_v + w_1(H_{v-p} + H_{v+p}).$$

We denote the tuple (w_1, p) as our augmentation regularization parameter. If the entropies of the neighbors ($v - p$ and $v + p$) are also low, then this suggests that the split value v is robust to small amounts of perturbation. Increasing the weight w_1 forces the algorithm to choose a split that is more robust.

Second, we add l_1 *regularization*, which adds a penalty of the form $|w_2 v|$ for any split value v. This follows the classic l_1 regularization scheme, where we encourage sparsity in the learned split vector. This is especially important in our setting; intuitively, we only want the complete tree to learn a non-degenerate split if the change in entropy is *significant* and thus, the learned split is stable. Thus, our overall regularization parameter is of the form (w_1, p, w_2).

6 EXPERIMENTS

6.1 Datasets

Iris dataset. The Iris dataset is available in the UC Irvine Machine Learning Repository [14]. This is a classic dataset used in machine learning for both supervised and unsupervised learning tasks. It is a small dataset, with 50 instances of each class and 4 features, and its size makes privatization difficult. The goal is to classify the class of irises; there are three classes and there is no class imbalance. We use 100 datapoints as our training set and 50 as the test dataset.

Rice dataset. The Rice dataset is available in the UC Irvine Machine Learning Repository [10]. This dataset contains 3,810 instances of rice, from two distinct species: Osmancik and Cammeo; the goal is to classify the species of rice. Each example contains 7 features such as area, perimeter and eccentricity. We use 70% of the dataset for training and the remaining 30% as the test dataset.

Dry Bean dataset. The Dry Bean dataset is available in the UC Irvine Machine Learning Repository. This dataset contains seven different types of dry beans; there are 13,611 instances of data with 16 features each [21]. Example features include area, perimeter and eccentricity. We use 70% of the dataset for training and the remaining 30% as the test dataset.

CIFAR-10 dataset. Finally, we consider the CIFAR-10 dataset [22]. This dataset consists of 60,000 images across 10 classes. The classes represent varying objects (e.g., "cat" or "deer") and there are 6,000 images per class. Each image is represented as a length-3072 vector, where each element of the vector contains information about the RGB values about a given pixel. We use 50,000 images as the training dataset and the remaining 10,000 as the test dataset. Due to the large size of the dataset, we only use CIFAR-10 for the PCA algorithm.

6.2 Experimental Design

For each of our experiments, we follow the template from Figure 1. We first choose our required privacy guarantee, represented by an upper bound on the mutual information (MI) between the input and output to our algorithm, $\mathcal{M}$. In our experiments, we vary MI between $\frac{1}{64} = 2^{-6}$ and $4 = 2^2$. We then estimate the stability of $\mathcal{M}$, on the training data X_{train}. To do this, we repeatedly randomly sample $X_i \subset X_{train}$ where each X_i from $i = 1 \dots m$ satisfies $|X_i| := 0.5|X_{train}|$. We denote m as the number of simulation trials. We then compute the stability of $\mathcal{M}$ as a function of the covariance of $\mathcal{M}(X_i)$ over all subsets $X_1 \dots m$. We use this to compute the noise required to privatize $\mathcal{M}$, which we denote as $\Delta(\mathcal{M}, MI)$.

For all of our algorithms (Mean, K-Means, SVM, PCA, and Random Forest), we implement the estimation algorithm of Section 4 to determine additive noise. For all of the experiments, we choose a trial complexity m, such that each output estimate converges to a precision level of 10^{-6}; more details are provided in Section 7.

We measure the *utility* of the baseline and both the isotropic and anisotropic privatized versions of $\mathcal{M}$. In particular, we first run $\mathcal{M}$ on the entire X_{train} and calculate the accuracy of $\mathcal{M}(X_{train})$ on the *test* dataset X_{test}. This provides our accuracy metric for the baseline non-private algorithm; we denote this as the "baseline accuracy" of $\mathcal{M}$. Then, we construct two privatized algorithms by, respectively, adding anisotropic noise and isotropic Gaussian noise (with mean 0 and variance $\Delta(\mathcal{M}, MI)$) to each element of the trained vector $\mathcal{M}(X_j)$, following the template of Figure 1, and using Algorithm 1. This creates two privatized trained vectors, $\mathcal{M}_P(X_j)$ (anisotropic noise) and $\mathcal{M}_Q(X_j)$ (isotropic noise); we then compute the accuracy of $\mathcal{M}_P(X_j)$ and $\mathcal{M}_Q(X_j)$ on X_{test}; these are, respectively, the anisotropic and isotropic "privatized accuracy" of $\mathcal{M}$, averaged over 1000 releases for each setting.

We now provide results across varying datasets and algorithms. All code used is provided at https://github.com/mayuri95/pac_algs.

6.3 Warmup: Estimating the Mean

Mean estimation is simple enough that we can provide a quantitative, head-to-head comparison between PAC and DP, since DP does not require significant changes beyond l_2-norm clipping for bounded sensitivity. For our experiments, we do a search to find the optimal clipping threshold to minimize the overall distance between the privatized mean estimate and the true mean. It is noted that for mean estimation with DP guarantees, the underlying error is twofold, the bias introduced due to artificially clipping the individual records [36] for a bounded sensitivity and the additional noise. For a given clipping threshold C and dataset size n, the global sensitivity for the mean estimate is C/n. The required noise to provide an ϵ-DP guarantee is then a Laplacian with scale $\frac{C}{n\epsilon}$ [12]. This suggests that the expected l_2-norm of such a noise is $O(\frac{\sqrt{d}}{n\epsilon})$ scaling up with $\sqrt{d}$, where d is the dimensionality of the samples and, indeed, such a curse of dimensionality is unavoidable [35].

In contrast to DP, which can use the entire dataset, PAC requires an input distribution, which we derive from subsampling. PAC does not require clipping. We chose the subsampling rate $r = 0.5$ to minimize prior success rate for an individual membership inference attack. (Any $0 < r < 1$ can be used, with different privacy-utility tradeoffs; we do not explore those here.)

To make a meaningful comparison, we compare DP and PAC *fixing* the posterior success rate. A given posterior success rate for membership inference can be translated to a particular ϵ-DP guarantee using Equation (2). Similarly, mutual information bounds and posterior success rates are related by Equation (3) (also see Table 1). We can therefore compare the l_2 distance between DP and PAC estimated means and the true means for the same success rates in Table 2. In general, the mean is a stable quantity and therefore neither DP nor PAC require significant noise addition. Often, a significant fraction of the PAC l_2-distance is due to the distance after subsampling. Using PAC with isotropic noise increases the l_2 distance after privatization by up to 1.7×.

We observe that the resulting privatized distances for DP and PAC are comparable for these datasets. It is important to note that DP and PAC privacy guarantees are not equivalent in the semantic sense. PAC provides instance-based worst-case guarantees rather than the input-independent worst-case guarantees of DP. For purposes of this comparison, we focus on the specific membership inference task on a single data point x; thus, the PAC noise estimates use individual mutual information in order to provide the tightest guarantees. In this setup, we observe that most of the distance in PAC is due to the noise from subsampling and the noise for privatization is negligible.

6.4 K-Means

As previously discussed, we expect K-Means to be easily compatible with the PAC Privacy framework; results are provided in Figure 2.

We observe that the baseline accuracy on our test set is above 90% for the Iris and Rice datasets. On small datasets like Iris, we observe a significant gap between the baseline and privatized algorithms for $MI < 2^{-3}$. On the Rice dataset, the centroids are quite stable and thus, we see no meaningful difference between privatized accuracy and the non-private baseline. In the Dry Bean dataset, the underlying baseline accuracy is quite low (≈ 75%). The anistropic privatized accuracy nearly matches it at MI ≥ 1, but the isotropic accuracy remains significantly worse.

Overall, our K-Means results match our expectations; that is, it is practical to provide meaningful privacy guarantees for the K-Means algorithm, with minimal utility losses, using anisotropic noise.

Dataset	Metric	$\epsilon = 1.64$; $1-\delta = 0.84$; MI = 1/4	$\epsilon = 0.73$; $1-\delta = 0.67$; MI = 1/16	$\epsilon = 0.36$; $1-\delta = 0.59$; MI = 1/64
Iris	DP	(0.004, 0.02)	(0.012, 0.05)	(0.012, 0.09)
Iris	PAC	(0.043, 0.043)	(0.046, 0.046)	(0.043, 0.048)
Rice	DP	(0.002, 0.002)	(0.002, 0.004)	(0.002, 0.007)
Rice	PAC	(0.008, 0.008)	(0.007, 0.007)	(0.008, 0.008)
Dry Bean	DP	(0, 0.001)	(0, 0.002)	(0, 0.004)
Dry Bean	PAC	(0.005, 0.005)	(0.005, 0.005)	(0.005, 0.005)

Table 2: Quantitative comparison of DP vs. PAC Privacy for private mean estimation. DP uses $\bar{\delta} = 0$ for varying posterior success probabilities, $(1-\delta)$. PAC noise estimates use *individual* mutual information rather than global guarantees. DP cells provide l_2 distance after clipping and after clipping and privatization; PAC cells provide l_2 distance after subsampling and after subsampling and anistropic privatization using Algorithm 1. All results are averaged over 1000 releases.

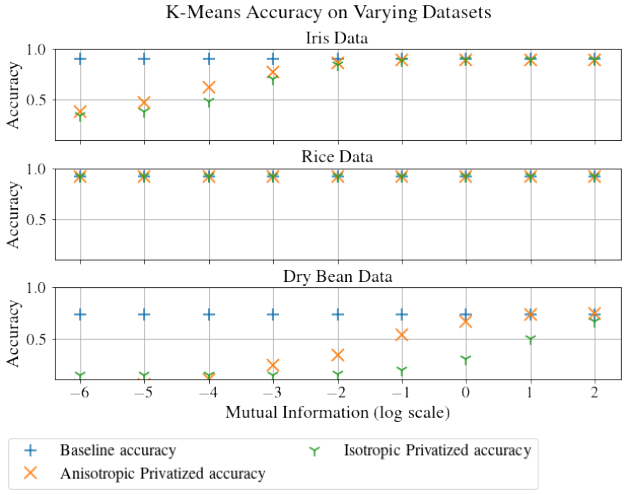

Figure 2: We plot the accuracy of the K-Means algorithm without privatization in blue. We then show the anisotropic privatization in orange and isotropic privatization in green. The accuracy is measured across mutual information varying from 2^{-6} to 2^2. As expected, we observe better utility using anisotropic noise across all datasets and mutual information values. The Rice dataset is the easiest to privatize, while the Dry Bean dataset is the hardest.

6.5 Support Vector Machines (SVM)

Our initial results on SVM, without any additional regularization ($C = 1.0$), are summarized in Figure 3[5].

[5]The Dry Bean SVM experiments are run to a precision of 10^{-5} for computational efficiency.

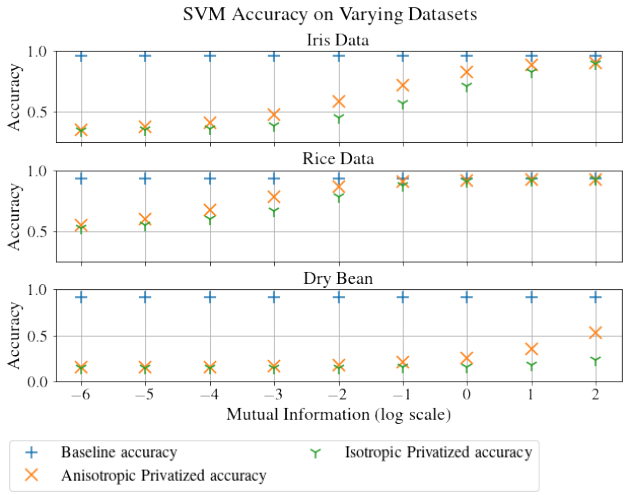

Figure 3: Without additional regularization, we observe that it is difficult to privatize the Iris dataset (significant utility loss for $MI \leq 2^0$) and nearly impossible to privatize the Dry Bean dataset. The Rice dataset is easier to privatize and shows minimal utility losses for $MI \geq 2^{-2}$.

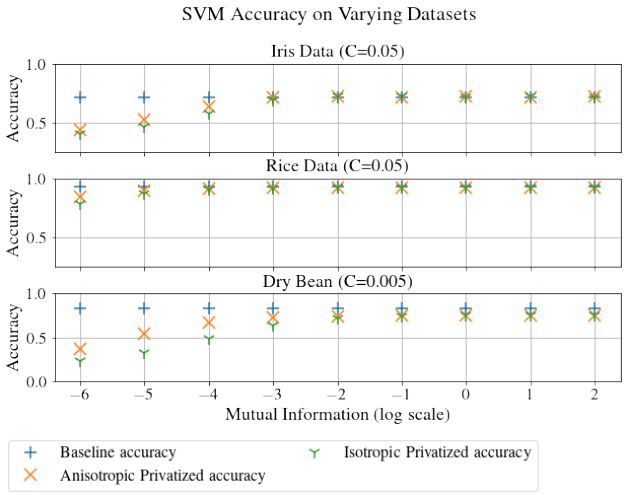

Figure 4: Regularization for SVM affects our datasets in significantly different ways. In the Iris dataset, the baseline algorithm suffers a significant utility loss due to the strong regularization (from $> 90\%$ to $\approx 75\%$). However, the gap between the privatized and baseline accuracy decreases, suggesting that strong regularization is optimal for tight mutual information guarantees. We observe similar results on the Dry Bean data set which achieves stability with small utility losses for $MI \geq 2^{-3}$ at $C = 0.005$. In the Rice dataset, regularization removes instability from the algorithm without a significant loss in baseline utility.

We first consider the Iris dataset. For sufficiently large MI (> 1), the utility loss due to privatization is minimal. However, when we tighten the mutual information guarantee, the magnitude of required noise increases until the utility impact is quite severe – at an MI guarantee of 2^{-4}, the privatized algorithm has a utility $\approx 41\%$. The Dry Bean dataset is a more stark example of this phenomenon — the noise added is so large that the privatized utility does not achieve $> 50\%$ until $MI > 2$. While the gap is not large, the anisotropic utility is consistently better than the isotropic counterpart.

We observe a similar trend on the Rice dataset for low MI values. However, the Rice dataset achieves stability at $MI \approx 2^{-1}$, where both baseline and privatized algorithms achieve accuracy of >90%.

There are many possible reasons for the difference in performance between the baseline and privatized algorithms. We consider two main cases, corresponding to superficial and intrinsic instability, respectively. In particular, we observe that many sources of superficial instability can be resolved by *regularization*. That is, regularization provides a technique to *order* multiple solutions which provide similar utility, by simply choosing the simplest one (lowest norm). However, increasing regularization too much can interfere with the baseline results – intuitively, we can prioritize simple solutions over those with higher utility. Thus, this cannot successfully resolve issues where the underlying algorithm is unstable due to *inherent* instability, without a significant utility impact.

We experiment with the stability of the SVM algorithm by increasing the regularization. We vary the regularization parameter C from Equation (4) and our results are in Figure 4.

We first consider the Iris dataset; the results here are plotted for $C = 0.05$. We first observe that the *non-private* version of the algorithm shows a decrease in accuracy, across all possible mutual information bounds – that is, the baseline accuracy drops from $> 90\%$ to $\approx 75\%$. This shows that the regularization is strong enough to overpower the loss in utility; that is, for the chosen value of C, the optimization problem prefers a stable low-norm solution more than the $\approx 15\%$ increase in accuracy. However, we observe that the *privatized* version of the algorithm shows a significant increase in accuracy, with minimal utility losses for MI as low as 2^{-4}. We observe similar results with the Dry Bean dataset, although stronger regularization is required. That is, we return to the privacy versus utility discussion, touched upon in Section 3. In the non-private setting, $C = 0.05$ for Iris represents regularization that is too strong since the error on the test dataset is *higher* than with $C = 1.0$.

We observe our best results on the Rice dataset. In this setting, we choose a regularization parameter of $C = 0.05$ – the baseline accuracy *increases* by 0.1% due to the regularization. Additionally, the stability of our algorithm is improved significantly and we can achieve negligible utility loss for the privatized version. This suggests that even complex algorithms with sufficiently large and representative datasets can achieve stability with appropriate regularization techniques.

6.6 Principal Component Analysis

We then consider the principal component analysis algorithm for dimensionality reduction. For this algorithm, we evaluate its performance by measuring the distance between the reconstructed test matrix X' and the original test matrix X_{test} as defined in Equation (5). We first explore the underlying rank of the datasets.[6] These results are summarized in Figure 5.

[6]We do not use the Iris dataset due to its small dimension.

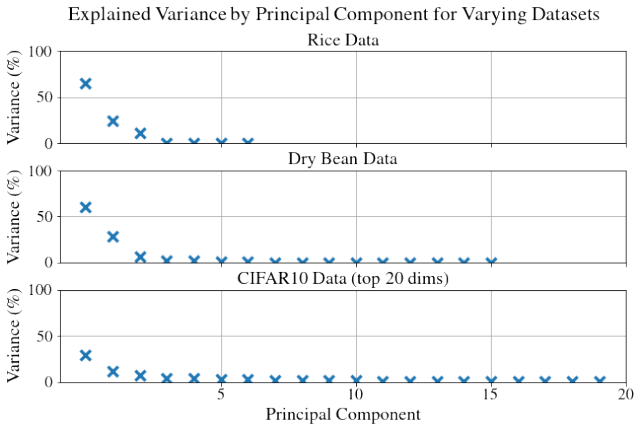

Figure 5: We measure the percentage of explained variance by the top principal components for each dataset. The Rice dataset has a total of 7 features, while the Dry Bean dataset has 16 features. The CIFAR-10 dataset has 3072 features — we only plot the explained variance for the top 20 dimensions, which account of 70% of the total variance.

In general, we expect that the PCA algorithm will be inherently unstable at tight MI guarantees when there are several principal components with similar "importance". That is, if there are two principal components that both explain $\approx 1\%$ of the underlying variance, we expect that either could be returned arbitrarily, even for extremely similar datasets. We first investigate $d = 1$ in Figure 6.

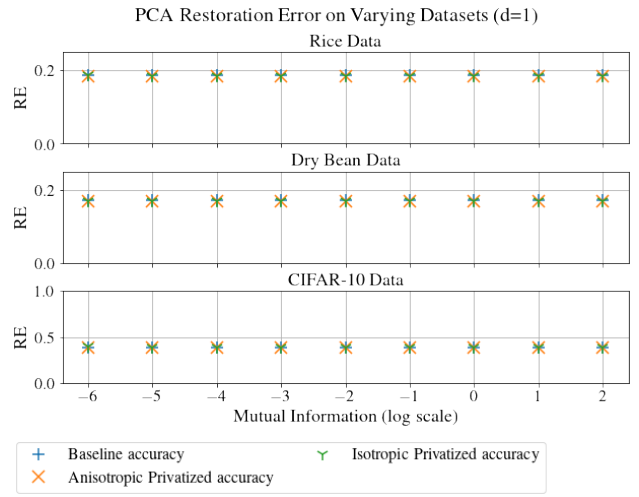

Figure 6: We observe that our algorithm is stable on all the datasets for all MI values. We observe a relatively low RE ($<$ 20%) for the Rice and Dry Bean datasets due to the significance of the top eigenvector. Meanwhile, CIFAR-10 has a larger RE ($\approx$ 40%), but similar stability guarantees.

As observed in Figure 5, most of the variance in the Rice and Dry Bean dataset are explained in the first component. Thus, we observe in Figure 6 that the restoration error is $<$ 20% for all mutual information values and we can largely recover the original matrix. In contrast, less than 50% of the variance of the CIFAR-10 dataset is explained by the first component. Thus, this shows a much higher restoration error $\approx$ 40%. Across all the datasets, we observe negligible changes in RE for all mutual information values, indicating that the algorithm is stable in identifying the top eigenvector.

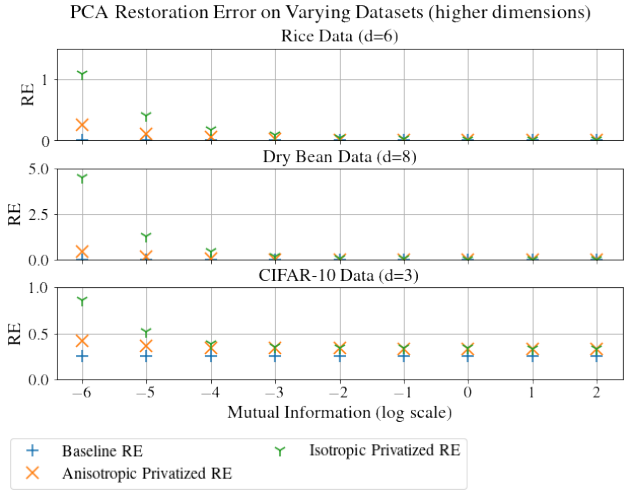

Figure 7: We run PCA with varying numbers of components (d in the plots) for the different datasets. With large d, the baseline restoration error drops to near zero for Rice and Dry Bean. This indicates that we can privatize these algorithms with negligible impact. In contrast, we choose $d = 3$ for CIFAR-10 in order to provide meaningful privatized utility; however, the baseline RE remains high due to the relatively low dimension. Increasing d further significantly reduces the stability, making the privatized algorithm unusable.

We then consider the same algorithm with higher dimensions, as seen in Figure 7. Here, we observe that all the datasets show a significant decrease in restoration error for the non-private baseline; this is expected since we are increasing the number of dimensions kept and thus, capturing more of the variance in the original matrix.

For the Rice dataset, we observe that the anisotropic noise consistently provides privatized RE $\leq 5\%$, which is a significant improvement, indicating the stability of the algorithm. The Dry Bean dataset shows similar results, with both converging to a negligible change in RE by $MI \approx 2^{-3}$. This suggests that we can privatize PCA on such large datasets with large enough dimension to capture most of the variance. In both of these cases, we observe the benefit of anisotropic noise — the corresponding isotropic algorithm often has much worse results in higher dimensions.

The CIFAR-10 dataset, in contrast, can only be privatized for $d = 3$. In particular, the eigenvectors for $d > 3$ have similar "importance" and the stability of the algorithm drops significantly (the l_1 norm of the noise added from $d = 3$ to $d = 4$ is an increase of 100×). The anisotropic algorithm RE for $d = 3$ varies from $\approx 43\%$ for $MI = 2^{-6}$ to $\approx 33\%$ for $MI = 2^2$, which is a small improvement over $d = 1$.

6.7 Random Forest

Finally, we consider the random forest algorithm. As discussed in Section 5, the random forest algorithm is known to be unstable and is quite difficult to adapt to our framework.

We first test the naïve algorithm with no additional regularization on the Iris and Rice datasets.[7] Our results are summarized in Figure 8. We use a single tree for the Iris dataset with depth 3. We use 3 trees for the Rice dataset, with depth 3. In each iteration of the algorithm, the chosen features for each tree are randomly sampled and the variance across the threshold values (and the classified values at each leaf) is measured.

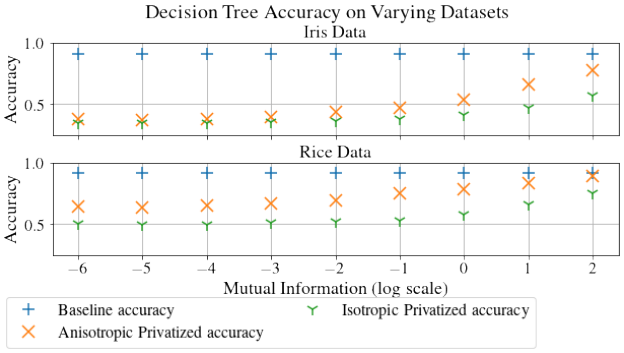

Figure 8: The naïve random forest algorithm shows significant instability on the Iris and Rice datasets. The Iris dataset achieves over 90% accuracy in the non-private case, but shows a dramatic loss in utility (down to $\approx$ 50% for $MI \leq 2^{-1}$) after privatization. The Rice dataset shows better results, with $\approx$ 91% accuracy in the baseline and $\approx$ 70% accuracy after anisotropic privatization at $MI = 2^{-2}$.

As expected, the privatized version of random forest without additional regularization shows significant instability for our datasets. Further investigation shows that there are several possible causes for the instability within a tree:

- When there are a small number of samples that are in a path, the optimal "threshold" value to split on is unstable. We resolve this by providing regularization penalties, to ensure that nodes with few samples split on a default value of 0.
- The exact threshold value to split on can be noisy due to the exact set of points that are observed. For instance, a feature with values ranging from 0 to 1 can split on 0.5 or 0.55 and achieve the same entropy. Initially, we chose to split only on values that were observed in the samples. To improve stability, we instead considered a fixed set of threshold values with finite precision.
- Finally, the threshold values are sometimes unstable due to the non-uniform spread of the feature values. To address this, we calculate a weighted average of the entropy.

We choose these three techniques in order to address the issue that the trees cannot be pruned while maintaining a canonical ordering that can be compared across iterations. We now experiment with adding regularization of the form (w_1, p, w_2) (cf. Section 5) for the Iris and Rice datasets.

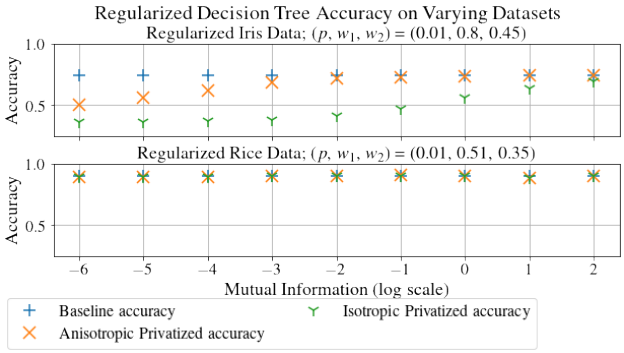

Figure 9: On the Iris dataset, we observe a significant drop in the baseline accuracy due to heavy regularization from 90% to 75%. Note that the best non-private accuracy is obtained when only 50% of the dataset is used, else overfitting worsens results. However, the gap between the baseline and privatized accuracy is much lower and we achieve over 70% privatized utility for $MI \geq 2^{-3}$. In the Rice dataset there is almost no utility loss in the baseline due to regularization (accuracy remains $\approx$ 90%). However, the privatized accuracy *increases* to achieve similar utility due to the improvements in stability.

We first consider the Iris dataset. As seen in Figure 9, we must add significant regularization to improve our privatized accuracy. As in the SVM setting, this corresponds to a decrease in the baseline accuracy. This regularization improves the privatized algorithm's utility to $\geq$ 70% for $MI \geq 2^{-3}$. We observe that the anisotropic noise provides a significant utility benefit over the isotropic setting, which provides $\leq$ 50% accuracy at $MI < 1$.

In contrast, we analyze the Rice dataset. For this dataset, we suffer no utility loss due to our increased regularization. However, the improvement in stability is significant; there is thus a negligible loss in utility between the privatized and baseline algorithm after the increased regularization for all MI values. This indicates that even unstable algorithms can be privatized with little utility cost when the dataset is sufficiently large and stable.

7 CONVERGENCE OF ALGORITHM 1

In this section, we discuss our empirical convergence guarantee. We observe that Theorem 1 provides a mutual information bound when the variances are estimated exactly. For practical guarantees, we choose sample complexity large enough such that each element of our variance estimate converges with very small precision ($p = 10^{-6}$). In particular, we run our noise estimation algorithm and estimate our variance vector after every 10 instances. We converge when none of the estimates in our output vector have changed by more than p. We choose p sufficiently small such that the impact of adding noise in the order of p is negligible. Sample results are shown in Figure 10.

Figure 10 provides the change in the first element of our variance vector for varying algorithms on the Iris dataset. We observe that the sample complexity varies across algorithms and datasets. Stable

[7] For this algorithm, we do not use the Dry Bean dataset since the algorithm takes a long time even for a single trial.

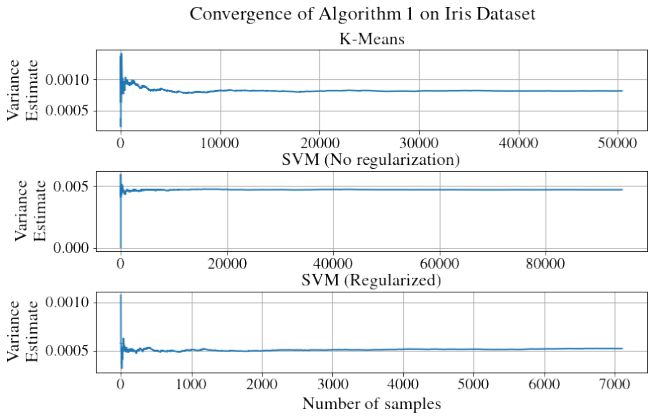

Figure 10: We choose our sample complexity m for Algorithm 1 by measuring the change in our variance estimate in each direction g_i. When all of the directions are stabilized within 10^{-6}, we return the current variance estimate.

algorithms (e.g., SVM with strong regularization) converge more than 10× faster than SVM without regularization.

Convergence on the covariance matrix would require an order of magnitude more trials.

8 EMPIRICAL PRIVACY ESTIMATION

8.1 Empirical membership inference attacks

Although PAC Privacy allows us to provably bound the posterior advantage for *any* attack, in this section, we focus on membership inference attacks for concreteness and validation. The objective of membership inference attacks (MIA) is defined in Shokri et al. [31] as follows: given a machine learning model and a single datapoint x, the goal is to determine whether x was used to train the model.

For our purposes, we define our machine learning model as the trained output vector Y_i, representing some statistic about our input data $X_i \subset X_{train}$. In K-Means, this vector corresponds to the list of centroids; for SVM, the vector represents the list of hyperplanes separating the classes.

In this section, we vary the mutual information bound and focus on a specific membership attack on the Iris dataset. We observe that our model is trained on a subset X_i where $|X_i| = 0.5(|X_{train}|)$ in all of our experiments. This indicates that for any particular datapoint x, the prior $\Pr[x \in X_i] = 0.5$. Table 1 gives a theoretical maximal posterior advantage, which can be compared to the empirical advantage observed. We further discuss how generalized membership attacks have much lower priors in Section 8.2.

We consider the Likelihood-Ratio Attack (LIRA) as described in Carlini et al. [5] and adapt it to the K-Means and SVM algorithms. In particular, LIRA exploits the idea that when a model is trained on a particular point x, its "confidence" on classifying x into a particular class or cluster will be higher than on a point it is not trained on. The original work by Carlini et al. [5] exploits this by framing a membership attack as a hypothesis test. In particular, they approximate the distribution $Q(X_i)$ observed when $x \notin X_i$. Then, they compute $\phi(\mathcal{M}(x))$, denoting the confidence of $\mathcal{M}$ on x. Finally, they explore a varying set of thresholds t where the attack concludes $x \in X_i$ if and only if $\phi(\mathcal{M}(x)) \geq t$. Each threshold t has corresponding true and false positive rates, and we simply consider the maximum accuracy over all thresholds t, which represents the maximum posterior advantage achievable by LIRA.

Unlike Carlini's work, we note we do not directly produce confidence values from our algorithms. Thus, we instead translate our output vectors to approximate confidence values. For the K-Means algorithm, we compute a confidence metric $\phi(x) := 1 - d(x)$, where $d(x)$ represents the normalized distance to the cluster that x is assigned to. For SVM, we use Platt calibration to translate the distance from the point to the hyperplane into a confidence metric [29], i.e.,

$$\phi(x, i) = \frac{1}{1 + \exp(-d(x, i))},$$

where $d(x, i)$ represents the distance between x and the hyperplane for class i. Then, $\phi(x) = \max_i \phi(x, i)$. We observe that the distribution of $\phi(x)$ is not always Gaussian. Hence, rather than approximating them with Gaussians, we use 1,000 trials to approximate the CDF $Q(X_i)$.

Our results on the Iris dataset are summarized in Figure 11. PAC Privacy is necessarily conservative; in Figure 11, the empirical posterior advantages (denoted as p_e) for the privatized algorithms are significantly lower than the upper bounds given by Table 1 across all mutual information bounds.

We observe a decrease in the privatized posterior advantage across all the algorithms. In the K-Means algorithm, we observe the most significant change, from $\approx$ 11% to 1.7% at $MI = 1/64$. In the SVM algorithm, the baseline advantage of the non-private algorithm is significantly lower at $\approx$ 3%. For both $C = 1.0$ and $C = 0.05$, we observe a small decrease of $\approx 0.5\%$ in p_e.

8.2 Generalized membership attacks

We observe that the prior guarantee of 0.5 is specific to membership attacks where we are trying to identify the membership of a *single* datapoint x. Thus, we first consider a generalization of the membership attack where there is a fixed number of points k that must be correctly identified.

As before, we have a data distribution represented by the total set of points X_{train} and our model is trained on $X_i \subset X_{train}$, where $|X_i| = 0.5|X_{train}|$. However, our inference task is to guess a set X' of size $0.5|X_{train}|$, such that at least k points in X' are classified correctly. That is, there exists a subset $X^* \subseteq X'$, such that every x in X^* is correctly classified and $|X^*| \geq k$.

We observe that for any point x, our prior remains at 50% for correctly identifying whether $x \in X_i$. However, for any fixed size k, our prior probability becomes

$$p := 1 - \frac{\sum_{k'=0}^{k-1} \binom{n/2}{k'}^2}{\binom{n}{n/2}}. \tag{6}$$

As k increases, this indicates that our prior success probability drops, as seen in Figure 12. In fact, when we consider $n = 100$ and $k = 32$, our prior probability drops below 1%, providing meaningful posterior bounds for large values of MI as discussed in Table 1. For stronger reconstruction attacks, where we consider k as a function

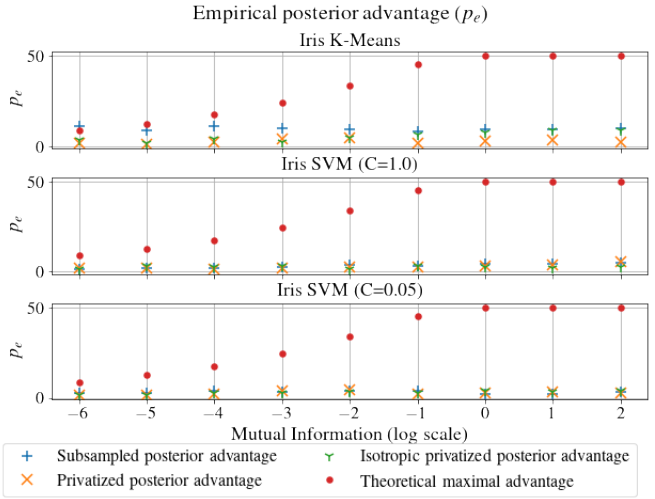

Figure 11: Empirical posterior advantage from LIRA over 250 trials. The posterior advantage of the subsampled algorithms are all at most 11%. For the privatized algorithms, the empirical advantages are always below the theoretical posterior advantages of Table 1. The K-Means algorithm shows the largest reduction in posterior advantage due to privatization, where the baseline algorithm has average $p_e \approx 9.7\%$ and the anisotropic privatized algorithm has average $p_e \approx 2.7\%$ over all values of MI.

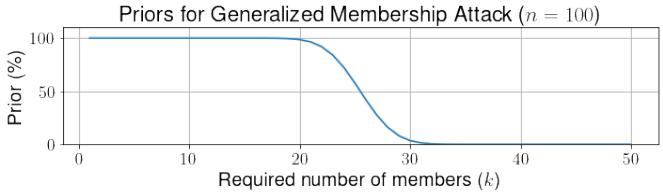

Figure 12: The prior from the generalized membership attack (Equation (6)) drops below 5% for $k \geq 30$. This indicates that loose MI guarantees can be meaningful for harder adversarial inference tasks.

of n, this allows for loose MI guarantees to still be meaningful. Table 1 gives meaningful posterior success guarantees for mutual information bounds of up to 4 for a prior of 1%. For instance, a generalized membership attack with $k = 35$ and a mutual information guarantee of 1, provides a $\leq 14.56\%$ posterior success guarantee, by solving Equation (3) given the appropriate prior.

9 RELATED WORK

Quantitative comparisons to DP for generic mean estimation were provided in Section 6.3. For more complex algorithms, small ϵ-DP guarantees are harder to provide and often involve significant changes to algorithm implementation. Even with white-box changes, the resulting algorithm often requires a large dataset with small data dimension to provide meaningful utility guarantees. In contrast, PAC provides instance-specific guarantees with reasonable utility loss, even when subsampling small datasets of $\approx$ 1,000 datapoints with large dimension. PAC can learn the output distribution, and add sufficient anisotropic noise in each dimension. Given the substantial algorithmic differences between DP white-boxed algorithms and the PAC black-box approach, and the semantic difference in privacy guarantees, we restrict ourselves to qualitative comparisons between our PAC-privatized algorithms and state-of-the-art DP algorithms for the various problems.

We first consider K-Means. Early work by Su et al. [32] developed DPLloyd, a DP version of Lloyd's algorithm for K-Means clustering. Intuitively, DPLloyd adds Laplacian noise each time the approximate centroids are computed. However, DPLloyd requires a fixed number of iterations to converge (this must be pre-computed to approximate the noise required per iteration). We observe that even for a simple algorithm, providing DP required significant changes to the algorithmic structure, e.g., fixing the number of iterations. Further, we observe that K-Means is *known* to be sensitive to the initial centroids chosen; thus, DPLloyd requires a new private initialization procedure as well.

We then consider SVM classifiers with differential privacy. There has been a long history of developing SVMs with DP guarantees [7, 30]. In general, these techniques tried to use the SVM algorithm to compute the optimal weight vector and add appropriate noise to provide DP guarantees. However, as observed by Zhang et al. [39], large training sets led to large weight vectors with increased noise. Moreover, often there were strong restrictions on the objective function (e.g., convexity) to enable tight bounds on the noise. Zhang et al. [39] suggests a novel method in order to solve the dual problem of SVM, which approaches the non-private SVM accuracy for sufficiently large training sets. However, note that this is still a white-box mechanism to achieve privacy, i.e., Laplacian noise was added in *each* iteration and in each iteration, an inner loop is required to choose the pairs of dual variables to update.

Random forests with differential privacy have been explored less extensively. Patil and Singh [26] suggests that differentially-private random forests can be constructed by allocating a privacy budget across trees, and then across levels of each tree. Each tree is "complete" when either all the features are used, the remaining samples all belong to the same class or when a maximum height is reached. Consul and Williamson [9] expands this work and constructs differentially-private median forests, which also improve the stability of the data structure. However, they still observe significant utility losses for sufficiently small ϵ.

Finally, we consider DP for identifying principal components. Chaudhuri et al. [8] constructs a near-optimal technique for identifying principal components while providing DP. Their technique requires datasets with a large number of datapoints, but the noise scales with the original dimension, making it impractical for datasets with large dimension, even if the true rank is constant. Lastly, we observe that their resulting utility guarantees are upon a secondary classification task, rather than the restoration of the original matrix task that we evaluate on. In practice, we expect the latter to be a stronger guarantee since a perfectly restored matrix would provide the best utility guarantee on *any* secondary task.

10 CONCLUSIONS

We have shown how PAC Privacy can be applied to privatize black-box algorithms, by giving a template that can be applied to virtually any algorithm. Using Algorithm 1 to add anisotropic noise is critical to improving privacy-utility tradeoffs.

Another exciting aspect of PAC Privacy that is demonstrated by our results is the potential win-win situation in the algorithm tradeoff space. Stability with respect to input changes is a desirable feature of algorithms, because stable algorithms generalize better to new inputs, and have better worst cases. Concomitantly, stable algorithms require less additive noise on their outputs for privatization.

Future work includes using the compositional properties of mutual information to tackle unstable algorithms such as Stochastic Gradient Descent (SGD).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. *Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms* 8, 1027–1035. https://doi.org/10.1145/1283383.1283494

[3] Borja Balle, Giovanni Cherubin, and Jamie Hayes. 2022. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1138–1156.

[4] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[5] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 2, 3, Article 27 (may 2011), 27 pages. https://doi.org/10.1145/1961189.1961199

[7] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially Private Empirical Risk Minimization. *J. Mach. Learn. Res.* 12, null (jul 2011), 1069–1109.

[8] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. 2012. Near-optimal Differentially Private Principal Components. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/f770b62bc8f42a0b66751fe636fc6eb0-Paper.pdf

[9] Shorya Consul and Sinead A. Williamson. 2021. Differentially Private Random Forests for Regression and Classification. https://api.semanticscholar.org/CorpusID:235365701

[10] Ilkay Cınar and Murat Koklu. 2019. Classification of Rice Varieties Using Artificial Intelligence Methods. *International Journal of Intelligent Systems and Applications in Engineering* (2019). https://api.semanticscholar.org/CorpusID:208105752

[11] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[12] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (aug 2014), 211–407. https://doi.org/10.1561/0400000042

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9, 61 (2008), 1871–1874. http://jmlr.org/papers/v9/fan08a.html

[14] R. A. Fisher. 1988. Iris. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76.

[15] Daniel Gruss, Julian Lettner, Felix Schuster, Olya Ohrimenko, Istvan Haller, and Manuel Costa. 2017. Strong and efficient cache {Side-Channel} protection using hardware transactional memory. In *26th USENIX Security Symposium (USENIX Security 17)*. 217–233.

[16] Jamie Hayes, Saeed Mahloujifar, and Borja Balle. 2023. Bounding Training Data Reconstruction in DP-SGD. *arXiv preprint arXiv:2302.07225* (2023).

[17] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.

[18] Ibrahim Issa, Aaron B Wagner, and Sudeep Kamath. 2019. An operational approach to information leakage. *IEEE Transactions on Information Theory* 66, 3 (2019), 1625–1657.

[19] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The composition theorem for differential privacy. In *International conference on machine learning*. PMLR, 1376–1385.

[20] Yigitcan Kaya and Tudor Dumitras. 2021. When does data augmentation help with membership inference attacks?. In *International conference on machine learning*. PMLR, 5345–5355.

[21] Murat Koklu and Ilker Ali Özkan. 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. *Comput. Electron. Agric.* 174 (2020), 105507. https://api.semanticscholar.org/CorpusID:219762890

[22] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[23] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[24] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 634–646.

[25] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908* (2018).

[26] Abhijit Patil and Sanjay Singh. 2014. Differential private random forest. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2623–2630. https://doi.org/10.1109/ICACCI.2014.6968348

[27] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572. https://doi.org/10.1080/14786440109462720 arXiv:https://doi.org/10.1080/14786440109462720

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[29] John Platt. 2000. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Adv. Large Margin Classif.* 10 (06 2000).

[30] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. 2012. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning. *Journal of Privacy and Confidentiality* 4, 1 (Jul. 2012). https://doi.org/10.29012/jpc.v4i1.612

[31] 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.

[32] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. 2016. Differentially Private K-Means Clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy* (New Orleans, Louisiana, USA) *(CODASPY '16)*. Association for Computing Machinery, New York, NY, USA, 26–37. https://doi.org/10.1145/2857705.2857708

[33] Yijue Wang, Chenghong Wang, Zigeng Wang, Shanglin Zhou, Hang Liu, Jinbo Bi, Caiwen Ding, and Sanguthevar Rajasekaran. 2021. Against Membership Inference Attack: Pruning is All You Need. In *International Joint Conference on Artificial Intelligence*.

[34] Hanshen Xiao and Srinivas Devadas. 2023. PAC Privacy: Automatic Privacy Measurement And Control Of Data Processing. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part II* (Santa Barbara, CA, USA). Springer-Verlag, Berlin, Heidelberg, 611–644. https://doi.org/10.1007/978-3-031-38545-2_20

[35] Hanshen Xiao, Jun Wan, and Srinivas Devadas. 2023. Geometry of Sensitivity: Twice Sampling and Hybrid Clipping in Differential Privacy with Optimal Gaussian Noise and Application to Deep Learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2636–2650.

[36] Hanshen Xiao, Zihang Xiang, Di Wang, and Srinivas Devadas. 2023. A theory to instruct differentially-private learning via clipping bias reduction. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2170–2189.

[37] Xiaokui Xiao and Yufei Tao. 2008. Output perturbation with query relaxation. *Proceedings of the VLDB Endowment* 1, 1 (2008), 857–869.

[38] Yu Yin, Ke Chen, Lidan Shou, and Gang Chen. 2021. Defending privacy against more knowledgeable membership inference attackers. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2026–2036.

[39] Yaling Zhang, Zhifeng Hao, and Shangping Wang. 2019. A Differential Privacy Support Vector Machine Classifier Based on Dual Variable Perturbation. *IEEE Access* 7 (2019), 98238–98251. https://doi.org/10.1109/ACCESS.2019.2929680

Received 29 April 2024