Remember What You Want to Forget: Algorithms for Machine Unlearning

Ayush Sekhari
Cornell University
as3663@cornell.edu

Jayadev Acharya[*]
Cornell University
acharya@cornell.edu

Gautam Kamath[*]
University of Waterloo
g@csail.mit.edu

Ananda Theertha Suresh[*]
Google Research, NY
theertha@google.com

Abstract

We study the problem of unlearning datapoints from a learnt model. The learner first receives a dataset S drawn i.i.d. from an unknown distribution, and outputs a model $\widehat{w}$ that performs well on unseen samples from the same distribution. However, at some point in the future, any training datapoint $z \in S$ can request to be unlearned, thus prompting the learner to modify its output model while still ensuring the same accuracy guarantees. We initiate a rigorous study of generalization in machine unlearning, where the goal is to perform well on previously unseen datapoints. Our focus is on both computational and storage complexity.

For the setting of convex losses, we provide an unlearning algorithm that can unlearn up to $O(n/d^{1/4})$ samples, where d is the problem dimension. In comparison, in general, differentially private learning (which implies unlearning) only guarantees deletion of $O(n/d^{1/2})$ samples. This demonstrates a novel separation between differential privacy and machine unlearning.

1 Introduction

Many organizations and companies employ user data to train machine learning models for a wide array of applications, ranging from movie recommendations to health care. While some of these organizations allow users to withdraw their consent from their data being used (at which point the organization will delete the user's data), less savory businesses might covertly retain user data. Given the potential for misuse, legislators worldwide have wisely introduced laws that mandate user data deletion upon request. These include the European Union's General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), and Canada's proposed Consumer Privacy Protection Act (CPPA).

There is some natural ambiguity present in these guidelines. Is it sufficient to simply delete the user's data, or must one also take action on machine learning systems that used this data for training? Indeed, by now, privacy researchers are well-aware that user data may be extracted from trained machine learning models (e.g., Shokri et al. [2017], Carlini et al. [2019]). In a potentially landmark decision, the Federal Trade Commission recently ordered a company to delete not only data from users who deleted their accounts, but also models and algorithms derived from this data [Federal Trade Commission, 2021]. This suggests that organizations have an obligation to retrain any machine learning models after excluding users whose data has been deleted.

[*]Authors in alphabetical order.

However, naïvely retraining models after every deletion request would be prohibitively expensive: training modern machine learning models may take weeks, and use resources of value in the millions. One could instead imagine more careful methods, which attempt to excise the required datapoints from the model: crucially, *without* incurring the cost of retraining from scratch. This notion is called *machine unlearning*. The goal would be to obtain a model which is *identical* to the alternative model that would be obtained when trained on the dataset after removing the points that need to be forgotten. This requirement is rather strong: Ginart et al. [2019] proposed a relaxed notion of deletion, in which the model must only be *close* to the alternative, where closeness is defined in a way reminiscent of differential privacy [Dwork et al., 2006a,b] (our variant of this notion is described in Definition 2). This relaxation has inspired the design of several efficient algorithms for data deletion from machine learning models [Guo et al., 2020, Izzo et al., 2021, Neel et al., 2021, Ullah et al., 2021].

As mentioned before, one naïve strategy involves retraining the model from scratch, sans the deleted datapoints. When the training dataset is large, this approach is undesirable for several reasons. First, it is computationally very expensive. Even iteration over the training data can be too costly, let alone training a new model on it. Second, preserving the entire training dataset consumes a significant amount of storage.[2]

Another straightforward approach involves model checkpointing, in which the learner preemptively stores backup models in which certain points have been excluded. While this strategy makes it easy to quickly return an appropriate backup model upon receiving a deletion request, the downside is that one typically has to store a number of additional models which scales with the training data size, which may be prohibitively large. As we can see from these examples, computational and storage complexity are two vital metrics when designing a machine unlearning algorithm.

Finally, while there has recently been a wealth of results in machine unlearning, all of it has focused on the core problem of *empirical risk minimization*, where the goal is to minimize the training loss. However, to fulfil the promise of machine learning, we desire algorithms that can *generalize* to previously unseen test data. Motivated simultaneously by all of these concerns, our goal is to address the following question:

How do we design resource-efficient machine unlearning algorithms which generalize?

Our contributions. We initiate a new line of inquiry in machine unlearning:

- We investigate generalization properties of unlearning algorithms, in particular asking: how many samples can we unlearn while still ensuring good performance on unseen test data? In comparison, prior work focused on the empirical training loss only.
- We consider machine unlearning simultaneously under storage constraints as well as the previously studied computation constraints. Unlike prior work, our algorithms do not require the training data to be available to the unlearning algorithm when deleting samples.
- A clean approach for unlearning is to ignore which particular samples are being unlearnt and directly apply known algorithms and guarantees from differential privacy (DP). We show a strict separation between DP and machine unlearning.

  In particular, algorithms based on DP can delete at most $\widetilde{\Theta}(n/\sqrt{d})$ samples while still retaining test loss performance, where d denotes the dimension of the problem. On the other hand, we provide efficient unlearning algorithms that take into account the particular samples to be unlearnt and show that we can delete up to $\widetilde{O}(n/d^{1/4})$ samples, thus giving a quadratic improvement in terms of dependence of d over DP. Our results apply to both strongly convex and convex loss functions.

1.1 Related work

Cao and Yang [2015] introduced the term "machine unlearning," and gave efficient deterministic algorithms for exact unlearning in certain settings. This definition requires an algorithm to have identical outputs on a dataset after deleting a point, and if that point was never inserted. However, their

[2]Orthogonal to storage constraints, an additional issue is that government regulations may restrict the learner from storing raw user data for extended periods of time due to privacy concerns. However, we focus on storage as it captures undesirability of a wider range of unsatisfactory solutions.

algorithms are restricted to very structured problems only. Bourtoule et al. [2021] provide unlearning algorithms using a sharding-based strategy, though in a weaker unlearning model (requiring only that it be possible that the output may have arisen), and without error guarantees.

Ginart et al. [2019] introduced the probabilistic notion of unlearning, inspired by differential privacy [Dwork et al., 2006b,a]. Their definition requires the output distribution of the unlearning algorithm to be similar to the output distribution obtained by running the learning algorithm on the dataset without the deleted points. Several recent works [Guo et al., 2020, Izzo et al., 2021, Neel et al., 2021, Ullah et al., 2021] provide theoretical error guarantees for various problem settings under this probabilistic notion of unlearning. While our unlearning setup is closely related to that of Ginart et al. [2019] and in the related works, there are two major differences.

First, the prior work focuses on empirical risk minimization [Guo et al., 2020, Izzo et al., 2021, Neel et al., 2021, Ullah et al., 2021]. In their setup, the goal of the unlearning algorithm is to find approximate minimizers of the empirical loss on the remaining training dataset after deleting samples. In comparison, our focus in this paper is on the test loss, and we wish to understand how many samples can be deleted from a learnt model while still ensuring that the updated model performs well on unseen examples (i.e., the generalization error). As we discuss in Section 3.1, the goal of minimizing the training loss is qualitatively different from that of minimizing the test loss.

Second, the prior work focuses exclusively on the computational cost of unlearning, without concern for associated storage requirements. This has led to approaches which involve memory-intensive checkpointing data structures, which enables fast processing of deletion requests, but consumes potentially impractical amounts of storage. In contrast, we are additionally concerned with memory usage, which highlights the drawbacks of such approaches. Unlike prior work, our algorithms do not require the training data to be available to the unlearning algorithm when deleting samples, and only rely on some cheap-to-store data statistics.

The most closely related work to ours is the *certified data removal framework* of Guo et al. [2020] which provides efficient data deletion algorithms for generalized linear models (linear and logistic regression). While our deletion algorithm is similar to the Newton update removal mechanism considered in their work, there are some important technical differences. First, their unlearning setup requires access to the entire training dataset for deleting samples; we do not require this. Second, they provide theoretical guarantees in terms of the norm of the empirical gradient being small after data removal. In comparison, our guarantees are for the test loss. Third, their unlearning definition requires the learning algorithm to be randomized, and this leads to worse performance guarantees due to added noise. In comparison, we do not need to randomize the learning algorithm. Finally, our guarantees hold for arbitrary convex loss functions and are thus broader in scope.

Several other models of unlearning have been considered. Garg et al. [2020] give an alternative perspective on machine unlearning, grounded in cryptography. Other works in this space focus on exploring privacy risks [Chen et al., 2020] and verification [Sommer et al., 2020] in machine unlearning settings. For specific learning models like SVMs, exact unlearning has been considered under the framework of decremental learning [Cauwenberghs and Poggio, 2001, Tveit et al., 2003, Karasuyama and Takeuchi, 2010, Romero et al., 2007]. However, the primary motivation in these works is to use the framework of decremental learning to estimate the leave-one-out error in order to provide generalization guarantees for the learnt model. Finally, there has also been recent empirical and theoretical work in developing definitions and algorithms for machine unlearning with deep neural networks for application domains in computer vision [Du et al., 2019, Golatkar et al., 2020a,b, Nguyen et al., 2020].

2 Preliminaries

Let $\mathcal{D}$ be a distribution over an instance space $\mathcal{Z}$ and $\mathcal{W} \subseteq \mathbb{R}^d$ be the parameter space of a hypothesis class. Let $f\colon \mathcal{W} \times \mathcal{Z} \to \mathbb{R}$ be a loss function. The goal is to minimize the test loss population risk (test loss), given by

$$F(w) := \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)], \tag{1}$$

where $f(w, z)$ is the loss of the hypothesis corresponding to $w \in \mathcal{W}$ on the instance $z \in \mathcal{Z}$. Let $F^* = \min_{w \in \mathcal{W}} F(w)$ be the value of this minimum and w^* be a corresponding minimizer. Since the distribution $\mathcal{D}$ is often unknown, we are restricted to rely on samples to find a small test loss model.

Given $S = (z_1, z_2, \dots, z_n)$, a set of n samples drawn independently from $\mathcal{D}$, standard learning algorithms minimize the empirical loss given by

$$\widehat{F}_n(w) := \frac{1}{n} \sum_{i=1}^{n} f(w, z_i). \tag{2}$$

2.1 Learning

Let $A : \mathcal{Z}^n \to \mathcal{W}$ be a learning algorithm that takes the dataset S and returns a hypothesis $A(S) \in \mathcal{W}$. The quality of A is measured in terms of the difference between the population risk of the hypothesis $A(S)$ and the risk of the best hypothesis w^* in $\mathcal{W}$, i.e., the excess risk

$$\mathbb{E}[F(A(S))] - F^*,$$

where the expectation is over the randomness in A and S. This gives a natural notion of sample complexity.

Definition 1 (Sample complexity of learning)**.** *The γ-sample complexity of a problem is defined as*

$$n_\gamma := \min\{n \mid \exists A \text{ s.t. } \mathbb{E}_{S\sim\mathcal{D}^n}[F(A(S))] - F^* \leq \gamma \quad \textit{for all } \mathcal{D}\},$$

the fewest number of samples with which a γ-suboptimal minimizer of the population loss $F(w)$ can be achieved for any distribution over the data samples.

For comparing different algorithms throughout the paper, we set $\gamma = 0.01$ (or any other small arbitrary constant), and require that the provided learning algorithms learn with population risk sub-optimality of at most 0.01. Standard results in learning theory [Bubeck, 2014, Theorem 6.1] show that for convex and strongly convex losses,

$$n_{0.01} = O(1), \tag{3}$$

where the hidden constant depends on the properties of f such as its Lipschitzness, but is independent of the dimension d of the parameter space $\mathcal{W} \subseteq \mathbb{R}^d$.

2.2 Unlearning

Suppose a learning algorithm A over S outputs the model $A(S)$. An unlearning algorithm $\bar{A}$ takes as input the model $A(S)$ and a set $U \subset S$ of data samples that are to be deleted, and is required to output a new model $\widehat{w} \in \mathcal{W}$. Besides the set U and the model $A(S)$, the unlearning algorithm $\bar{A}$ can also access some additional data statistics $T(S) \in \mathcal{T}$. We emphasize that the unlearning algorithm does not have access to the entire original dataset S in this resource-constrained setting and hence cannot retrain from scratch.

This set of statistics $T(S)$ captures the additional storage required by the algorithm to support unlearning. Thus, one of our goals is to minimize $|T(S)|$, in particular aiming for memory requirements which are independent of the training data size n. This precludes strategies which involve storing and reusing the entire training set, or aggressive model checkpointing. On the other hand, it permits storage of simple statistics such as the empirical mean or average gradient of training data points, which may prove useful when unlearning. At the same time, we are still concerned with our unlearning algorithm's time complexity. This goes hand in hand with the storage complexity: for most natural algorithms, the two are likely to be polynomially related. Augmented by this set of data statistics $T(S)$, an unlearning algorithm is a mapping $\bar{A} : \mathcal{Z}^m \times \mathcal{W} \times \mathcal{T} \to \mathcal{W}$.

To illustrate our unlearning setup, consider the following toy example: Suppose we train a model $\widehat{w}$ with four datapoints $S = \{[0.0, 0.1], [2.0, 2.3], [4.0, 0.5], [3.0, 1.3]\}$. After training the model, the learner only keeps $\widehat{w}$ and a cheap-to-store sufficient statistic $T(S)$, and deletes the dataset S from the memory. At this point the learner does not have access to any datapoints from S anymore. Now, when a delete request comes, the request contains just the sample U to be deleted e.g. $[4.0, 0.5]$. The unlearning algorithm at this point will unlearn this sample, using the information $\widehat{w}$, $T(S)$, and $U = [4.0, 0.5]$ only. Here, note that the data sample $[4.0, 0.5]$ is provided to the unlearning algorithm by the user who owns this data and requests for its deletion.

We now define a notion of unlearning, which is motivated by the definition of differential privacy [Dwork et al., 2006a].

Definition 2 ((ε, δ)-unlearning)**.** *For all S of size n and delete requests $U \subseteq S$ such that $|U| \leq m$, and $W \subseteq \mathcal{W}$, a learning algorithm A and an unlearning algorithm $\bar{A}$ is (ε, δ)-unlearning if*

$$\Pr\big(\bar{A}(U, A(S), T(S)) \in W\big) \leq e^{\varepsilon} \cdot \Pr\big(\bar{A}(\emptyset, A(S \setminus U), T(S \setminus U)) \in W\big) + \delta,$$

and

$$\Pr\big(\bar{A}(\emptyset, A(S \setminus U), T(S \setminus U)) \in W\big) \leq e^{\varepsilon} \cdot \Pr\big(\bar{A}(U, A(S), T(S)) \in W\big) + \delta,$$

where $\emptyset$ denotes the empty set and $T(S)$ denotes the data statistics available to $\bar{A}$.

The above states that with high probability, an observer cannot differentiate between the two cases (i) the model is trained on the set S and then a set U of m points are deleted by the unlearning algorithm using statistics $T(S)$ and (ii) the model is trained on the set $S \setminus U$ and no points are deleted thereafter by the unlearning algorithm (using statistics $T(S \setminus U)$). For simplicity, throughout the paper, we assume that $\varepsilon \leq 1$.

While being similar, the above notion of unlearning is slightly different from the one considered in Ginart et al. [2019]. Specifically, their definition compares the output of the unlearning algorithm after deleting m samples, to the output of the learning algorithm that only operates on $S \setminus U$. Due to this, they require the learning algorithm to be randomized, even in the situation when there would be no delete requests in the future. Thus, the output of the learning algorithm will suffer a degradation in its performance guarantees due to this added noise. On the other hand, our definition does not require the learning algorithm to be randomized since we only compare the output of the unlearning algorithms in the two scenarios—with and without the delete requests. Furthermore, our definition is more general than that of Ginart et al. [2019], as we can simulate their comparison in our definition by considering the unlearning algorithms for which $\bar{A}$ simply adds noise to the output of $A(S \setminus U)$ when $U = \emptyset$.

Our definition of unlearning leads to the following natural definition of the deletion capacity that formalizes how many samples can be deleted while still ensuring good test loss guarantees.

Definition 3 (Deletion capacity)**.** *Let $\varepsilon, \delta \geq 0$. Let S be a dataset of size n drawn i.i.d. from $\mathcal{D}$, and let $f(w, z)$ be a loss function. For a pair of learning and unlearning algorithms $A, \bar{A}$ that are (ε, δ)-unlearning, the deletion capacity $m^{A,\bar{A}}_{\varepsilon,\delta}(d, n)$ is defined as the maximum number of samples U that can be unlearnt, while still ensuring an excess population risk of* 0.01. *Specifically,*

$$m^{A,\bar{A}}_{\varepsilon,\delta}(d, n) := \max\Big\{m \mid \mathbb{E}\Big[\max_{U \subseteq S : |U| \leq m} F(\bar{A}(U, A(S), T(S))) - F^*\Big] \leq 0.01\Big\},$$

where the expectation above is with respect to $S \sim \mathcal{D}^n$ and output of the algorithms A and $\bar{A}$.

We are primarily interested in unlearning algorithms for which $T(S)$ is small, and in particular does not grow with the dataset size n (which can potentially be very large).

2.3 Unlearning via retraining from scratch

The most naïve yet natural baseline for unlearning is to simply retrain the model from scratch using the remaining data. That is, we let $\bar{A}(U, A(S), T(S)) = A(S \setminus U)$. However, the straightforward method to implement this approach would require us to set $T(S)$ to contain the entire training dataset S, and thus $|T(S)| \geq n$. However, recall that, we aim to provide unlearning algorithms for which $T(S)$ is independent of n. Furthermore, retraining from scratch is computationally expensive – merely reading all the data takes $\Omega(n - m)$ time, not accounting for the cost of actually running the algorithm. These drawbacks lead one to explore more efficient methods for unlearning.

3 Our results

Prior works consider unlearning from an optimization perspective, focusing on minimizing the empirical risk, and do not discuss the implications of unlearning on test loss. As we show in the next section, the two could significantly different objectives even for some of the simplest learning problems.

3.1 Population risk vs empirical training risk

We first provide a simple example motivating our study of population risk over empirical risk, quantified rigorously in Theorem 1. Consider the following mean estimation problem. Let $d = 1$, $\mathcal{Z} = \mathbb{R}$, and the loss function $f(w, z) = (w - z)^2$. The empirical risk of n points $z_1, z_2, \ldots, z_n$ is minimized by the average $\frac{1}{n}\sum_{i=1}^{n} z_i$. For this problem there is a simple unlearning algorithm that minimizes empirical risk and also unlearns U exactly. We store the average of the points $w_1 = \sum_{i=1}^{n} z_i/n$, and upon receiving a deletion request of a set U of m samples, subtract those samples and renormalize to compute the minimizer of the empirical loss on the remaining training samples, i.e., we output

$$w_2 = \frac{n}{n-m}\Big(w_1 - \frac{1}{n}\sum_{z \in U} z\Big). \tag{4}$$

The above update rule requires $T(S)$ to be of size $O(d)$ and completely deletes the samples U satisfying the unlearning guarantee with $\varepsilon = \delta = 0$. The returned solution w_2 is the exact minimizer of the empirical loss on left over data points.

However, w_2 may not perform well on fresh samples drawn from the test distribution. Consider the same setting as above, but where the points z_i are drawn i.i.d. from Bernoulli(1/2). Thus, the optimal parameter w^* that minimizes the test loss is given by $1/2$. However, consider the scenario where all of the m delete requests correspond to points with value 1. In this case, the minimizer of the updated empirical loss would be smaller by an additive factor of m/n than the previous estimate, and would thus have worse test loss.

In other words, in the unlearning framework, while minimizing training loss might be a good algorithm, just providing guarantees on the training loss can be vacuous if we want the model to generalize to unseen test samples. Furthermore, focusing on the test loss inherently limits the deletion capacity as noted in example discussed above. We further formalize this intuition in Theorem 1 and show that even if the unlearning algorithm has access to all undeleted samples $S \setminus U$, there is a non-trivial limit on the deletion capacity.

Theorem 1. *Let $\delta \leq 0.005$ and $\varepsilon = 1$. There exists a 4-Lipschitz, 1-strongly convex function f and distribution $\mathcal{D}$, such that for any learning algorithm A and unlearning algorithm $\bar{A}$, which even has access to undeleted samples $S \setminus U$, the deletion capacity*

$$m_{\varepsilon,\delta}^{A,\bar{A}}(d, n) \leq cn,$$

where c depends on the properties of function f and is strictly less than 1.

Prior work [Shalev-Shwartz et al., 2009a, Feldman, 2016] suggests that even in the learning setting there are problems for which empirical risk minimizer solution fails to perform well on fresh samples, and regularization does not help [Kale et al., 2021]. Our situation is worse, since the delete requests U can be adversarially chosen from S [Lai et al., 2016, Diakonikolas et al., 2019]. In fact, the proof of Theorem 1 relies on showing the existence of an adversary that can only delete points and can change the empirical loss considerably. We defer full details to Appendix B.1.

3.2 Strict separation between unlearning and differential privacy

Given the strong resemblance between differential privacy and our definition for unlearning, a natural approach would be to use tools from differential privacy (DP) for machine unlearning. The simplest way is to ignore the particular set of delete requests U and construct an unlearning algorithm $\bar{A}$ that only depends on the learning algorithm $A(S)$. More formally, such an unlearning algorithm is of the form $\bar{A}(U, A(S), T(S)) = \bar{A}(A(S))$ and satisfies:

$$\Pr\big(\bar{A}(A(S)) \in W\big) \leq e^{\varepsilon}\Pr\big(\bar{A}(A(S \setminus U)) \in W\big) + \delta,$$
$$\Pr\big(\bar{A}(A(S \setminus U)) \in W\big) \leq e^{\varepsilon}\Pr\big(\bar{A}(A(S)) \in W\big) + \delta.$$

Note that any such pair of algorithms would be differentially private with respect to the original dataset S, where the notion of neighboring datasets is for datasets with edit distance of m. The above guarantee is stronger than the distribution-free unlearning guarantee in Definition 2, and thus it suffices to satisfy it. The key observation is that any DP algorithm A, which is private for datasets with edit distance m, automatically unlearns any m data samples. Thus, the standard performance guarantees for DP learning yields the following bound on deletion capacity:

Lemma 1 (Unlearning via DP)**.** There exists a polynomial time learning algorithm A and unlearning algorithm $\bar{A}$ of the form $\bar{A}(U, A(S), T(S)) = A(S)$ such that the deletion capacity

$$m^{A,\bar{A}}_{\varepsilon,\delta}(d, n) \geq \widetilde{\Omega}\Big(\frac{n\varepsilon}{\sqrt{d\log(e^{\varepsilon}/\delta)}}\Big), \tag{5}$$

where the constant in the Ω-notation above depends on the properties of the loss function f.

The above result raises an immediate question of whether this particular dependence on d and n is necessary on the deletion capacity, and if it can be improved further. The following lemma shows that there exist problem instances for which any unlearning algorithm that ignores the samples U, can not improve over the factor of $\sqrt{d}$ in the denominator of the deletion capacity bound in (5).

Lemma 2 (Bassily et al. [2019], Section C)**.** For any learning algorithm A and an unlearning algorithm $\bar{A}$ that does not use U, i.e., $\bar{A}(U, A(S)) = \bar{A}(A(S))$, there exists a 1-strongly convex function and $O(1)$-Lipschitz loss function f, and a distribution $\mathcal{D}$ such that we can not unlearn even a single sample point, if

$$n \leq c \cdot \frac{\sqrt{d}}{\varepsilon},$$

where c depends on the properties of the function f.

Given that the $\sqrt{d}$ dependence on dimension is unavoidable for algorithms that directly use DP, it is natural to wonder whether this factor may be bypassed using other techniques. Our main contribution in this work a positive answer in this direction. As we show in the next section, when the loss function is convex, there exist unlearning algorithms which can delete up to $n/d^{1/4}$ sample points while still retaining the performance guarantee with respect to the test loss.

3.3 Unlearning for convex loss functions

In this section, we provide an unlearning algorithm $\bar{A}$ for convex losses that can delete more points than unlearning algorithms that simply use DP.

Theorem 2. *There exists a learning algorithm A and an unlearning algorithm $\bar{A}$ such that for any convex (and hence strongly convex), L-Lipschitz, and M-Hessian-Lipschitz loss f and distribution $\mathcal{D}$,*

$$m^{A,\bar{A}}_{\varepsilon,\delta}(d, n) \geq c \cdot \frac{n\sqrt{\varepsilon}}{(d\log(1/\delta))^{1/4}},$$

where the constant c depends on the Lipschitz constants L and M. Furthermore, for the unlearning algorithm $\bar{A}$ has running time $O(d^{\omega})$ where $\omega \in [2, 2.38]$ is the exponent of matrix multiplication, and space complexity for $T(S) = O(d^2)$.

Theorem 2 and Lemma 2 together show that there exist problem settings, where the deletion capacity in unlearning and DP is different by a multiplicative $\Theta(d^{1/4})$. In particular, when learning with convex loss functions, we can delete $O(n/d^{1/4})$ samples while still retaining good performance on the unseen test loss, whereas DP only guarantees deletion of $\Theta(n/d^{1/2})$ samples. Hence, our algorithm is at least quadratically better in terms of dependence on d in deletion capacity that standard DP algorithms. Besides improving the dependence on d, our algorithm also enjoys better dependence on ε and $\log(1/\delta)$ in the deletion capacity, by at least a quadratic factor.

Our learning algorithm stores additional statistics of the dataset S in order to delete the set U in the unlearning algorithm. The extra memory we need for these statistics is independent of n. Furthermore, our algorithm uses the samples in U during the unlearning phase. This paradox of storing and using information in order to delete it, motivates the name of the paper: *Remember what you want to forget*.

Characterizing the entire set of problems for which unlearning and differential privacy are different remains an interesting open question. Theorem 2 yields an improved upper bound, but it is not clear if this dependence on d or n in the deletion capacity is tight or if it can be improved even further. Resolving this question would be a fascinating future research direction.

4 Unlearning algorithms

In the following, we provide learning and unlearning algorithms when the loss function $f(\cdot, z)$ is λ-strongly convex. The unlearning algorithms for convex losses follows by appealing to the strongly convex case after adding regularization. We defer the algorithms and proofs for the convex case to Appendix D. Throughout this section, we make the following assumption:

Assumption 1. For any $z \in \mathcal{Z}$, the function $f(w, z)$ is λ-strongly convex, L-Lipschitz and M-Hessian Lipschitz with respect to w.

Learning algorithm. We denote our learning algorithm by A_{sc}. When given a dataset S of n points sampled i.i.d. from some distribution $\mathcal{D}$, the algorithm A_{sc} computes the point $\widehat{w}$ by minimizing the empirical loss $\widehat{F}_n(w)$, i.e.

$$\widehat{w} \leftarrow \operatorname{argmin} \widehat{F}_n(w) := \frac{1}{n} \sum_{z \in S} f(w, z). \tag{6}$$

A_{sc} then returns the point $\widehat{w}$ and the statistics $T(S) := \{\nabla^2 \widehat{F}(\widehat{w})\}$ containing the Hessian of $\widehat{F}(w)$ evaluated at the output point $\widehat{w}$. We provide the pseudocode for A_{sc} in the appendix.

Unlearning algorithm. We denote our unlearning algorithm by $\bar{A}_{sc}$ and provide the pseudocode in Algorithm 1. $\bar{A}_{sc}$ receives as input the set of delete requests U, the point $\widehat{w}$ and the data statistics $T(S)$. Using these inputs, $\bar{A}_{sc}$ first estimates the matrix $\widehat{H}$ that denotes the Hessian of the empirical function on the dataset $S \setminus U$ when evaluated at the point $\widehat{w}$. Then, $\bar{A}_{sc}$ computes the point $\bar{w}$ by removing the contribution of the deleted points U from $\widehat{w}$ using the update in (8). Finally, $\bar{A}_{sc}$ perturbs $\bar{w}$ with noise ν drawn from $\mathcal{N}(0, \sigma^2 \mathbb{I}_d)$ and returns the perturbed point $\widetilde{w}$.

Algorithm 1 Unlearning algorithm ($\bar{A}_{sc}$)

```
Input: Delete requests: $U = \{z_j\}_{j=1}^m \subseteq S$, output of $A_{sc}(S)$: $\widehat{w}$, additional statistic $T(S)$ :
    $\{\nabla^2 \widehat{F}(\widehat{w})\}$, loss function: $f(w, z)$.
1: Set $\gamma = \frac{2Mm^2L^2}{\lambda^3 n^2}$, $\sigma = \frac{\gamma}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.
2: Compute
       $\widehat{H} = \frac{1}{n-m}\big(n\nabla^2 \widehat{F}(\widehat{w}) - \sum_{z \in U} \nabla^2 f(\widehat{w}, z)\big).$   (7)
3: Define
       $\bar{w} = \widehat{w} + \frac{1}{n-m}(\widehat{H})^{-1} \sum_{z \in U} \nabla f(\widehat{w}, u).$   (8)
4: Sample $\nu \in \mathbb{R}^d$ from $\mathcal{N}(0, \sigma^2 \mathbb{I}_d)$.
5: Return $\widetilde{w} := \bar{w} + \nu$.
```

Our main technical insight that leads to improvements in deletion capacity over differential privacy is the following observation. For loss functions that satisfy Assumption 1, when deleting m samples, we can approximate the empirical minimizer on the dataset $S \setminus U$ up to a precision of $O(m^2/n^2)$ by the point $\bar{w}$ computed in (8). This implies that we only need to add noise of the scale of $\sigma \propto O(m^2/n^2)$ to get the desired unlearning guarantee. This noise is smaller than the amount of noise typically added for DP learning [Dwork and Roth, 2014] by a quadratic factor; hence giving us a quadratic improvement in the deletion capacity.

Lemma 3. Suppose the loss function f satisfies Assumption 1. Let $S \sim \mathcal{D}^n$ be a set of n samples, and $U \subseteq S$ denote the set of m delete requests. Define the point $\widehat{w}'$ as the empirical minimizer over $S \setminus U$, i.e. $\widehat{w}' \in \operatorname{argmin}_w \sum_{z \in S \setminus U} f(w, z)/(n - m)$. Then,

$$\|\widehat{w}' - \bar{w}\| \leq \frac{2ML^2m^2}{\lambda^3 n^2},$$

where the point $\bar{w}$ is defined in (8) in Algorithm 1.

The following theorem provides performance guarantees for the algorithms A_{sc} and $\bar{A}_{sc}$, and show that A_{sc} and $\bar{A}_{sc}$ are (ε, δ)-unlearning.

Theorem 3. *Suppose the loss function f satisfy Assumption 1 and let the dataset $S \sim \mathcal{D}^n$. Then,*

(*a*) *The point $\widehat{w}$ returned by running A_{sc} on S satisfies*

$$\mathbb{E}_{S\sim\mathcal{D}^n}[F(\widehat{w}) - \min_{w\in\mathcal{W}} F(w)] \leq \frac{4L^2}{\lambda n}. \tag{9}$$

(*b*) *For any set $U \subseteq S$ of m delete requests, the point $\widetilde{w}$ returned by $\bar{A}_{sc}$ satisfies*

$$\mathbb{E}_{S,\nu}[F(\widetilde{w}) - \min_{w\in\mathcal{W}} F(w)] = O\Big(\frac{\sqrt{d}Mm^2L^3}{\lambda^3 n^2 \varepsilon}\sqrt{\ln(1/\delta)} + \frac{4mL^2}{\lambda n}\Big). \tag{10}$$

(*c*) *The learning algorithm A_{sc} and the unlearning algorithm $\bar{A}_{sc}$ are (ε, δ)-unlearning.*

Note that our guarantees in the above theorem are in terms of the test loss performance, while our algorithms compute minimizers of the training loss. We defer the proof to the Appendix C.2. This gives a lower bound on the number of samples m that can be deleted while still ensuring the desired excess risk guarantee (deletion capacity). Specifically, from the performance guarantee for $\bar{A}_{sc}$ in (10), we observe that we can delete

$$m \geq c \cdot \frac{n\sqrt{\varepsilon}}{\left(d\log(1/\delta)\right)^{1/4}},$$

samples from the set S (with size n) while still ensuring that an excess risk guarantee of $\gamma = 0.01$. Here, c depends on the constants M, K and λ for the function f. This proves Theorem 2 for strongly convex loss functions.

Memory. We do not need to store the entire dataset S for the unlearning algorithm $\bar{A}_{sc}$. We note that the data statistic $T(S)$ that is passed as an input to $\bar{A}_{sc}$ is given by $T(S) = \{\nabla^2 \widehat{F}(\widehat{w})\}$. Clearly, $T(S)$ needs $O(d^2)$ memory and thus $|T(S)|$ is independent of n or m.

Computation. For the sake of exposition above, our learning algorithm A_{sc} computes the exact minimizer for the empirical loss in (6). However, as we show in Appendix C.2, our theoretical guarantees hold even when the empirical minimizer is computed approximately up to a precision of $O(1/n^2)$. When the domain $\mathcal{W}$ is convex, such a minimizer can be efficiently computed using standard optimization algorithms like accelerated gradient descent, SGD, clipped-SGD, etc. For example, for the λ-strongly convex case, Nesterov's accelerated GD algorithm can find a $O(1/n^2)$ approximate minimizer in time $\widetilde{O}(nd/\sqrt{\lambda})$ [Bubeck, 2014, Nemirovski and Yudin, 1983]. Furthermore, A_{sc} takes $O(nd^2)$ time to compute $T(S)$.

On the other hand, the running time for the unlearning algorithm $\bar{A}_{sc}$ scales as $O(md^2 + d^\omega)$, the time taken to compute the matrix $\widehat{H}$ and to invert it. Here, $\omega \in [2, 2.38]$. Note that our unlearning time is independent of the (potentially large) dataset size n. Furthermore, for problems such as linear SVMs where the Hessian is a diagonal matrix, A_{sc} takes time $O(nd)$ and $\bar{A}_{sc}$ takes time $O(d)$.

Comparison to retraining from scratch. Retaining from scratch requires access to the entire training dataset S during unlearning, and is thus not a feasible unlearning algorithm in the resource constrained setting that we consider in this work where the size of the data statistics $T(S)$ should not scale with the size of S. However, ignoring the memory issues, there are regimes for the value of m for which retraining from scratch is computationally more efficient that implementing Algorithm 1, and the returned solution enjoys the same test loss performance guarantee. In particular, when $\max(m^5d^2, m^{4+\omega}) > \lambda^4 n^2 \varepsilon$, retraining from scratch by solving (6) using stochastic gradient descent is more efficient that computing the matrix $\widehat{H}$ and its inverse in Algorithm 1. On the other hand, for large datasets (when $n \gg d$), we expect our algorithm to be computationally more efficient. We defer the exact characterization of memory / computation tradeoffs in machine unlearning for future work.

Algorithms for convex losses. Our unlearning algorithms when the loss function is convex are based on reductions to the strongly convex setting discussed above. Give the convex loss function $f(\cdot, z)$, we define the function $\widetilde{f}(\cdot, z)$ as

$$\widetilde{f}(w, z) = f(w, z) + \frac{\lambda}{2}\|w\|^2.$$

The key observation is that the function $\widetilde{f}(w, z)$ is λ-strongly convex, $(L + \lambda\|w\|)$-Lipschitz, $(H + \lambda)$-smooth and M-Hessian Lipschitz, and thus we can run algorithms A_{sc} and $\bar{A}_{sc}$ on $\widetilde{f}$. We defer the algorithmic implementation and theoretical analysis for the convex loss setting to Appendix D.

5 Conclusion

We initiated a new study on machine unlearning with a focus on population risk minimization, in comparison to previous works that focus on empirical risk minimization. For the case of convex loss functions, we provide a new unlearning algorithm that improves over the deletion capacity, by at least a quadratic factor in d, than using an out of the box differentially private algorithm for unlearning. Proving a dimension dependent information theoretic lower bound on the deletion capacity is an interesting future research direction. Another exciting direction of future research is to provide efficient unlearning algorithms for finite / discrete hypothesis class, and for non-convex loss functions. Finally, in this work, we considered the problem of batch deletion where the delete request U all arrive at the same time; Extending our algorithms for the online case is another interesting research direction that we are excited to pursue in future research.

Acknowledgements

We thank Robert Kleinberg, Mehryar Mohri, and Karthik Sridharan for helpful discussions. JA is supported in part by the grant NSF-CCF-1846300 (CAREER), NSF-CCF-1815893, and a Google Faculty Fellowship. GK is supported by an NSERC Discovery Grant and a University of Waterloo startup grant.

Funding Transparency Statement

Funding in direct support of this work: NSF-CCF-1846300 (CAREER), NSF-CCF-1815893, a Google Faculty Fellowship, an NSERC Discovery Grant, and a University of Waterloo startup grant.

References

General Data Protection Regulation. Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016, 2016.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, SP '17, pages 3–18, Washington, DC, USA, 2017. IEEE Computer Society.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, USENIX Security '19, pages 267–284. USENIX Association, 2019.

Federal Trade Commission. California company settles ftc allegations it deceived consumers about use of facial recognition in photo storage app, January 2021.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pages 3518–3531. Curran Associates, Inc., 2019.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006a. Springer.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International*

Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT '06, pages 486–503, Berlin, Heidelberg, 2006b. Springer.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, ICML '20, pages 3832–3842. JMLR, Inc., 2020.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models: Algorithms and evaluation. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, AISTATS '21. JMLR, Inc., 2021.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, ALT '21. JMLR, Inc., 2021.

Enayat Ullah, Tung Mai, Anup Rao, Ryan Rossi, and Raman Arora. Machine unlearning via algorithmic stability. *arXiv preprint arXiv:2102.13179*, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proceedings of the 36th IEEE Symposium on Security and Privacy*, SP '15, pages 463–480, Washington, DC, USA, 2015. IEEE Computer Society.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *proceedings of the 42nd IEEE Symposium on Security and Privacy*, SP '21, Washington, DC, USA, 2021. IEEE Computer Society.

Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '20, pages 373–402, Berlin, Heidelberg, 2020. Springer.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. *arXiv preprint arXiv:2005.02205*, 2020.

David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247*, 2020.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, pages 409–415, 2001.

Amund Tveit, Magnus Lie Hetland, and Håavard Engum. Incremental and decremental proximal support vector classification using decay coefficients. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 422–429. Springer, 2003.

Masayuki Karasuyama and Ichiro Takeuchi. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks*, 21(7):1048–1059, 2010.

Enrique Romero, Ignacio Barrio, and Lluís Belanche. Incremental and decremental learning for linear support vector machines. In *International Conference on Artificial Neural Networks*, pages 209–218. Springer, 2007.

Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. Lifelong anomaly detection through unlearning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1283–1297, 2019.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020a.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. *arXiv preprint arXiv:2012.13431*, 2020b.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33, 2020.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, 2009a.

Vitaly Feldman. Generalization of erm in stochastic convex optimization: The dimension strikes back. *arXiv preprint arXiv:1608.04414*, 2016.

Satyen Kale, Ayush Sekhari, and Karthik Sridharan. Sgd: The role of implicit regularization, batch-size and multiple-epochs. *arXiv preprint arXiv:2107.05074*, 2021.

Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.

Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Guha Thakurta. Private stochastic convex optimization with optimal rates. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pages 11282–11291. Curran Associates, Inc., 2019.

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

Arkadij Semenovič Nemirovski and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, 2009b.

A Additional Notation

We recall the following standard definitions for the loss function $f(w; z)$.

Definition 4 (Lipschitzness). *The function $f(w, z)$ is L-Lipschitz in the parameter w if for all $z \in \mathcal{Z}$, and all $w_1, w_2 \in \mathcal{W}$,*

$$|f(w_1, z) - f(w_2, z)| \leq L\|w_1 - w_2\|.$$

Definition 5 (Strong convexity). *The function $f(w, z)$ is λ-strongly convex, if for all $z \in \mathcal{Z}$, and all $w_1, w_2 \in \mathcal{W}$,*

$$f(w_1, z) \geq f(w_2, z) + \langle \nabla f(w_2, z), w_1 - w_2\rangle + \frac{\lambda}{2}\|w_1 - w_2\|^2.$$

Definition 6 (Hessian-Lipschitzness). *The function $f(w, z)$ is said to be M-Hessian Lipschitz if for all $z \in \mathcal{Z}$, and all $w_1, w_2 \in \mathcal{W}$,*

$$\|\nabla^2 f(w_1, z) - \nabla^2 f(w_2, z)\| \leq M\|w_1 - w_2\|,$$

or equivalently, that $\|\nabla^3 f(w, z)\| \leq M$ for all w.

B Missing proofs from Section 3

B.1 Proof of Theorem 1

We first develop some technical results, which we will use to prove Theorem 1. For a distribution $\mathcal{D}$, define $\mu(\mathcal{D}) := \mathbb{E}_{Z\sim\mathcal{D}}[Z]$.

Lemma 4. There exists two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over $[0, 1]$ such that $\|\mathcal{D}_1 - \mathcal{D}_2\|_1 \leq \alpha$ and $|\mu(\mathcal{D}_1) - \mu(\mathcal{D}_2)| \geq \alpha/2$.

Proof. Let $\mathcal{D}_1$ be the uniform distribution over $[0.25 + \alpha/2, 0.75 + \alpha/2]$ and $\mathcal{D}_2$ be the uniform distribution over $[0.25, 0.75]$. We note that $\mathbb{E}_{z\sim\mathcal{D}_1}[z] = 0.5 + \alpha/2$ and $\mathbb{E}_{z\sim\mathcal{D}_2}[z] = 0.5$ and hence $|\mathbb{E}_{z\sim\mathcal{D}_1}[z] - \mathbb{E}_{z\sim\mathcal{D}_2}[z]| = \alpha/2$. However, the ℓ_1 distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is bounded by α. □

Lemma 5. Given two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over the domain $\mathcal{Z}$, let the distribution $\bar{\mathcal{D}}$ be defined such that for any $z \in \mathcal{Z}$, $\bar{\mathcal{D}}(z) \propto \min(\mathcal{D}_1(z), \mathcal{D}_2(z))$. Then, for any $\mathcal{D}_1$ and $\mathcal{D}_2$ such that $\|\mathcal{D}_1 - \mathcal{D}_2\|_1 = \alpha$, and n samples $\{z_i\}_{i=1}^n$ drawn iid from $\mathcal{D}_1$, there exists an adversary that deletes at most $2n\alpha$ samples and outputs the dataset $\{z_j\}_{j=1}^{n'}$ with a distribution $\widetilde{\mathcal{D}}^{n'}$ and $n' \leq n - 2n\alpha$ such that

$$\|\widetilde{\mathcal{D}}^* - \bar{\mathcal{D}}^*\|_1 \leq e^{-n\alpha/6},$$

where the superscript denotes distribution over all sequences over the domain $\mathcal{Z}$.

Proof. Let $m = 2n\alpha$. Our proof uses two adversaries A' and A. We first define A'. Given n samples $\{z_i\}_{i=1}^n$, it deletes sample z_i with probability $\frac{[\mathcal{D}_1(z_i)-\mathcal{D}_2(z_i)]_+}{\mathcal{D}_1(z_i)}$, where $[x]_+ := \max\{x, 0\}$. First observe that for any sample z, probability that z is retained is given by

$$\mathcal{D}_1(z) \cdot \frac{\min\{\mathcal{D}_1(z), \mathcal{D}_2(z)\}}{\mathcal{D}_1(z)} = \min\{\mathcal{D}_1(z), \mathcal{D}_2(z)\}.$$

Thus the distribution of samples outputted by A' is exactly $\bar{\mathcal{D}}$. However, it can delete more than m samples. Next, we consider another adversary A, which is same as A', except it stops after deleting m samples. Hence, for sequences of length $> n - m$, the output of $\mathcal{A}'$ and $\mathcal{A}$ are the same. Hence,

$$\|\widetilde{\mathcal{D}}^* - \bar{\mathcal{D}}^*\|_1 \leq \Pr(N \geq m), \tag{11}$$

where N is the number of deleted samples. In the rest of the proof, we bound this probability.

Let $X_i \in \{0, 1\}$ be the random variable that takes the value 1 if the sample i is deleted. Hence $N = \sum_i X_i$. Furthermore, $\mathbb{E}[X_i] = \int_z [\mathcal{D}_1(z), \mathcal{D}_2(z)]_+ dz = \|\mathcal{D}_1 - \mathcal{D}_2\|_2/2 = m/4n$. By the Chernoff bound, we have that

$$\Pr\Big(\sum_i X_i \geq m\Big) \leq e^{-m/12}.$$

Using the above with (11) implies the desired statement. □

We now have all the tools to prove Theorem 1.

Proof of Theorem 1. The proof for small values of m follows from known bounds for sample complexity of learning [Bubeck, 2014, Shalev-Shwartz and Ben-David, 2014]. In the following, we provide an information-theoretic lower bound for $m \geq 100$ by constructing two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ and show that no single learning-unlearning pair $A, \bar{A}$ can perform well on both of them.

Let $\mathcal{W} = [0, 1]$ and $\mathcal{Z} = R$. Further, let the loss function be $f(w, z) = (w - z)^2$. Our proof consists of two main parts: we first provide a reduction from learning to mean estimation, and then give the lower bound by a reduction from mean estimation to hypothesis testing.

Reduction from learning to mean estimation: We first show that for any distribution $\mathcal{D}$ for the population loss given by $F(w) := \mathbb{E}_{z\sim D}[f(wz)]$ satisfies,

$$F(w) - F(w^*) = (w - w^*)^2.$$

To observe this note that

$$\begin{aligned}
F(w) &= \mathbb{E}[(w - z)^2] \\
&= \mathbb{E}[(w - w^* + w^* - z)^2] \\
&= \mathbb{E}[(w - w^*)^2] + 2\ \mathbb{E}[(w - w^*)(w^* - z)] + \mathbb{E}[(w^* - z)^2] \\
&= (w - w^*)^2 + 2(w - w^*)\,\mathbb{E}[(w^* - z)] + F(w^*) \\
&= (w - w^*)^2 + F(w^*),
\end{aligned}$$

where the last inequality uses the fact that $\mathbb{E}[z] = w^*$ for our loss function. Thus, in order to bound the learning error, it suffices to bound the error in estimating the mean of the underlying distribution w^*.

From mean estimation to the lower bound: Since $\mathcal{Z}$ is unbounded and having more information only helps, we assume that the unlearner has access to the entire sample set S i.e., the passed data statistics $T(S) = \{S\}$. Since the output $A(S)$ can be derived from S, it suffices to consider unlearning algorithms $\bar{A}$ of the form $\bar{A}(U, S)$. Let w denote the output $\bar{A}(U, S)$. By the definition of forgetting rule,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_1}(\bar{A}(U, A(S)) - w_1^*)^2 &= \int_w \Pr_{\mathcal{D}_1}(\bar{A}(U, A(S)) = w)(w - w_1^*)^2 dw \\
&\geq e^{-\varepsilon} \int_w \Pr_{\mathcal{D}_1}(\bar{A}(\phi, A(S \setminus U)) = w)(w - w_1^*)^2 dw - \delta,
\end{aligned}$$

where the last inequality uses the fact that $\mathcal{W} = [0, 1]$. Let $\tilde{A} = \bar{A}(\phi, A(\cdot))$. Let $\alpha = m/2n$ and let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distributions such that $\|\mathcal{D}_1 - \mathcal{D}_2\|_1 = \alpha$. Let the set of delete requests U be chosen by the adversary in Lemma 5. For these set of samples U, we have that

$$\int_w \Pr_{\mathcal{D}_1}(\bar{A}(\phi, A(S \setminus U)) = w)(w - w_1^*)^2 dw \geq \int_w \Pr_{\bar{\mathcal{D}}}(\bar{A}(\phi, A(S \setminus U)) = w)(w - w_1^*)^2 dw - e^{-m/12},$$

where $\bar{\mathcal{D}}$ is defined in Lemma 5. Similarly, we have that

$$\mathbb{E}_{\mathcal{D}_2}(\bar{A}(U, A(S)) - w_1^*)^2 \geq e^{-\varepsilon} \int_w \Pr_{\bar{\mathcal{D}}}(\bar{A}(\phi, A(S \setminus U)) = w)(w - w_1^*)^2 dw - e^{-m/12} - \delta.$$

The sum of errors the unlearner makes on either of the errors is at least

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}_1}[F(w)] - F^*_{\mathcal{D}_1} + \mathbb{E}_{\mathcal{D}_2}[F(w)] - F^*_{\mathcal{D}_2} \\
&\qquad\geq e^{-\varepsilon} \int_w (\Pr_{\bar{\mathcal{D}}}(\tilde{A}(S \setminus U) = w)((w - w_1^*)^2 + (w - w_2^*)^2) dw - 2\delta - 2e^{-m/12} \\
&\qquad\geq \frac{e^{-\varepsilon}}{2} \int_w (\Pr_{\bar{\mathcal{D}}}(\tilde{A}(S \setminus U) = w)((w_1^* - w_2^*)^2) dw - 2\delta - 2e^{-m/12} \\
&\qquad= \frac{e^{-\varepsilon}(w_1^* - w_2^*)^2}{2} - 2e^{-m/12} - 2\delta, \qquad (12)
\end{aligned}$$

where the inequality in the second last line follows from the fact that $a^2 + b^2 \geq (a+b)^2/2$ for any $a, b \in \mathbb{R}$.

Plugging in the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ as given in Lemma 4 with $\alpha = m/2n$ in the error bound (12) implies that

$$\mathbb{E}_{\mathcal{D}_1}[F(w)] - F^*_{\mathcal{D}_1} + \mathbb{E}_{\mathcal{D}_2}[F(w)] - F^*_{\mathcal{D}_2} \geq \frac{e^{-\varepsilon} m^2}{32n^2} - 2e^{-m/12} - 2\delta.$$

Since $\delta \leq 0.005$, $\varepsilon \leq 1$ and $m \geq 100$, for the above quantity to be less than 0.01, we have that $m \leq cn \cdot e^{\varepsilon}$, where $ce^{\varepsilon} < 1$. □

B.2 Proof of Lemma 1

Proof. We let A to be a DP algorithm which is private for datasets with edit distance m. Our unlearning algorithm $\bar{A}$ simply returns the input point $A(S)$ without making any changes to it, i.e. $\bar{A}(U, A(S), T(S)) = A(S)$. Clearly, the unlearning algorithm $\bar{A}$ does not require any additional data statistics and thus $T(S) = \emptyset$.

We set the DP algorithm A as the mini-batch noisy SGD method from Bassily et al. [2019]. The learning guarantee for A from Bassily et al. [2019, Theorem 3.2] together with the group privacy property of differential privacy [Vadhan, 2017, Lemma 2.2] implies that:

$$F(A(S)) - F^* \leq 10BL\Big(\frac{1}{\sqrt{n}} + \frac{m\sqrt{d\log(me^{\varepsilon}/\delta)}}{\varepsilon n}\Big). \tag{13}$$

Furthermore, $A(S)$ is (ε, δ)-DP for datasets with edit distance m, i.e. for any set $U \subseteq S$ of m samples:

$$\begin{aligned}\Pr(A(S) \in W) &\leq e^{\varepsilon}\Pr(A(S \setminus U) \in W) + \delta, \\ \Pr(A(S \setminus U) \in W) &\leq e^{\varepsilon}\Pr(A(S) \in W) + \delta.\end{aligned}$$

Since $T(S) = \emptyset$ and $\bar{A}(A(S), U, T(S)) = A(S)$ for any $U \subseteq S$ such that $|U| = m$, we can rewrite the DP guarantee as:

$$\begin{aligned}\Pr\big(\bar{A}(U, A(S), T(S)) \in W\big) &\leq e^{\varepsilon} \cdot \Pr\big(\bar{A}(\emptyset, A(S \setminus U), T(S \setminus U)) \in W\big) + \delta, \\ \Pr\big(\bar{A}(\emptyset, A(S \setminus U), T(S \setminus U)) \in W\big) &\leq e^{\varepsilon} \cdot \Pr\big(\bar{A}(U, A(S), T(S)) \in W\big) + \delta,\end{aligned}$$

implying that the pair $(A, \bar{A})$ is (ε, δ)-unlearning for U of size m.

We next bound the deletion complexity. The bound in the right hand side of (13) implies that we can delete

$$m = \widetilde{\Omega}\Big(\frac{0.01\varepsilon}{\sqrt{\log(e^{\varepsilon}/\delta)}} \cdot \frac{n}{\sqrt{d}}\Big)$$

samples while still ensuring that the excess risk is bounded by $\gamma = 0.01$. The above implies the desired lower bound on the deletion capacity. □

B.3 Proof of Theorem 2

The following lower bound on the deletion capacity is based on the excess risk guarantees for our learning and unlearning algorithms given in Theorem 3 and Theorem 4 (in Appendix D) for strongly convex and convex loss setting respectively.

Proof. We consider the strongly convex loss and convex loss setting separately below.

Strongly convex loss setting. Our learning algorithm A_{sc} and the unlearning algorithm $\bar{A}_{sc}$ are given in Algorithm 2 and Algorithm 1 respectively. Theorem 3 implies that the learning algorithm A_{sc} and the unlearning algorithm $\bar{A}_{sc}$ are (ε, δ)-unlearning. Furthermore, we have that

$$\mathbb{E}[F(\widehat{w}) - F^*] \leq \frac{4L^2}{\lambda n},$$

and

$$\mathbb{E}[F(\widetilde{w}) - F^*] = O\Big(\frac{\sqrt{d}Mm^2L^3}{\lambda^3 n^2 \varepsilon}\sqrt{\ln(1/\delta)} + \frac{4mL^2}{\lambda n}\Big),$$

where $\widehat{w}$ denotes the output point $A_{sc}(S)$ and $\widetilde{w}$ denotes the output point $\bar{A}_{sc}(U, A_{sc}(S), T(S))$.

The above upper bound on the excess risk implies that we can delete at least

$$m = c \cdot \frac{n\sqrt{\varepsilon}}{(d\log(1/\delta))^{1/4}},$$

samples while still ensuring an excess risk guarantee of $\gamma = 0.01$. Here, the constant c depends on the constants M, L and λ for the function f. This gives us the desired lower bound on the deletion capacity $m_{\varepsilon,\delta}^{A_{sc},\bar{A}_{sc}}(d,n)$.

Convex loss setting. Our learning algorithm A_c and the unlearning algorithm $\bar{A}_c$ are given in Algorithm 3 and Algorithm 4 respectively. Lemma 13 implies that the learning algorithm A_c and the unlearning algorithm $\bar{A}_c$ are (ε,δ)-unlearning. Furthermore, as a consequence of Corollary 2, we note that setting λ as in (25) implies that:

$$\mathbb{E}[F(\widehat{w}) - F^*] = O\Big(c_1\sqrt{\frac{m}{n}} + c_2\Big(\frac{d\log(1/\delta)}{\varepsilon^2}\Big)^{1/8}\sqrt{\frac{m}{n}}\Big)$$

and

$$\mathbb{E}[F(\widetilde{w}) - F^*] = O\Big(c_1\sqrt{\frac{m}{n}} + c_2\Big(\frac{d\log(1/\delta)}{\varepsilon^2}\Big)^{1/8}\sqrt{\frac{m}{n}}\Big),$$

where $\widehat{w}$ denotes the output point $A_c(S)$ and $\widetilde{w}$ denotes the output point $\bar{A}_c(U, A_c(S), T(S))$, and the constants c_1 and c_2 depend on the properties of the function f.

The above upper bound on the excess risk implies that we can delete at least

$$m = c \cdot \frac{n\sqrt{\varepsilon}}{(d\log(1/\delta))^{1/4}},$$

samples while still ensuring an excess risk guarantee of $\gamma = 0.01$. Here, the constant c depends on the constants M, L and B for the function f. This gives us the desired lower bound on the deletion capacity $m_{\varepsilon,\delta}^{A_c,\bar{A}_c}(d,n)$.

□

C Missing details from Section 4

C.1 Proof of Lemma 3

Lemma 6. The points $\widehat{w}$ and $\widehat{w}'$, defined in Lemma 3, satisfy the following guarantee

$$\|\widehat{w} - \widehat{w}'\| \le \frac{2mL}{\lambda n}.$$

Proof. Define the functions $\widehat{F}_1$ and $\widehat{F}_2$ as

$$\widehat{F}_1(w) := \frac{1}{n}\sum_{z\in S} f(w,z) \qquad \text{and,} \qquad \widehat{F}_2(w) := \frac{1}{n-m}\sum_{z\in \bar{S}} f(w,z),$$

where the set $\bar{S} := S \setminus U$. Note that $\widehat{w} = \operatorname{argmin}_w \widehat{F}_1(w)$ and, $\widehat{w}' = \operatorname{argmin}_w \widehat{F}_2(w)$. We first observe that

$$n\big(\widehat{F}_1(\widehat{w}') - \widehat{F}_1(\widehat{w})\big) = \sum_{z\in S} f(\widehat{w}', z) - \sum_{z\in S} f(\widehat{w}, z)$$

$$
\begin{aligned}
&= \sum_{z \in \bar{S}} f(\widehat{w}', z) - \sum_{z \in \bar{S}} f(\widehat{w}, z) + \sum_{z \in U} f(\widehat{w}', z) - \sum_{z \in U} f(\widehat{w}, z) \\
&= (m-n)\big(\widehat{F}_2(\widehat{w}') - \widehat{F}_2(\widehat{w})\big) + \sum_{z \in U} f(\widehat{w}', z) - \sum_{z \in U} f(\widehat{w}, z) \\
&\overset{(i)}{\leq} \sum_{z \in U} f(\widehat{w}', z) - \sum_{z \in U} f(\widehat{w}, z) \overset{(ii)}{\leq} mL\|\widehat{w}' - \widehat{w}\|,
\end{aligned} \tag{14}
$$

where the equality in the second line above follows from the fact that $\bar{s} = S \setminus U$ and the equality in the third line holds from the definition of the function $\widehat{F}_2$. The inequality (i) holds because $\widehat{w}'$ is the minimizer of the function $\widehat{F}_2(w)$ and the inequality (ii) is due to the fact that the function f is L-lipschitz. Next, note that the function $\widehat{F}_1$ is λ-strongly convex. Thus,

$$
\widehat{F}_1(\widehat{w}') - \widehat{F}_1(\widehat{w}) \geq \frac{\lambda}{2}\|\widehat{w} - \widehat{w}'\|^2. \tag{15}
$$

Using (14) and (15), we get that

$$
\frac{\lambda n}{2}\|\widehat{w} - \widehat{w}'\|^2 \leq mL\|\widehat{w} - \widehat{w}'\|,
$$

which implies that $\|\widehat{w} - \widehat{w}'\| \leq \frac{2mL}{\lambda n}$. □

Proof of Lemma 3. Given the function f that satisfies Assumption 1, define the functions $\widehat{F}_1$ and $\widehat{F}_2$ as

$$
\widehat{F}_1(w) := \frac{1}{n}\sum_{z \in S} f(w, z) \qquad \text{and,} \qquad \widehat{F}_2(w) := \frac{1}{n-m}\sum_{z \in \bar{S}} f(w, z),
$$

where the set $\bar{S} := S \setminus U$. Using the Taylor's expansion for $\nabla \widehat{F}_2(\widehat{w}')$ around the point $\widehat{w}$, we get that

$$
\|\nabla \widehat{F}_2(\widehat{w}') - \nabla \widehat{F}_2(\widehat{w}) - \nabla^2 \widehat{F}_2(\widehat{w})[\widehat{w}' - \widehat{w}]\| \leq \frac{M}{2}\|\widehat{w} - \widehat{w}'\|^2,
$$

where M denotes the Hessian-Lipschitz constant for the function $f(\cdot, z)$, i.e. $\|\nabla^3 \widehat{F}_2(\widehat{w})\| \leq M$. Since $\widehat{w}'$ is a minimizer of $\widehat{F}_2$, and $\widehat{F}_1$ is smooth, we have that $\nabla \widehat{F}_2(\widehat{w}') = 0$. Plugging this in the above bound, we get

$$
\|\nabla \widehat{F}_2(\widehat{w}) + \nabla^2 \widehat{F}_2(\widehat{w})[\widehat{w}' - \widehat{w}]\| \leq \frac{M}{2}\|\widehat{w} - \widehat{w}'\|. \tag{16}
$$

Also note that

$$
\begin{aligned}
\nabla \widehat{F}_2(\widehat{w}) &= \frac{1}{n-m}\sum_{z \in \bar{S}} \nabla f(\widehat{w}, z) \\
&= \frac{1}{n-m}\sum_{z \in S} \nabla f(\widehat{w}, z) - \frac{1}{n-m}\sum_{z \in U} \nabla f(\widehat{w}, z) \\
&= \frac{n}{n-m}\nabla \widehat{F}_1(\widehat{w}) - \frac{1}{n-m}\sum_{z \in U} \nabla f(\widehat{w}, z) \\
&= -\frac{1}{n-m}\sum_{z \in U} \nabla f(\widehat{w}, z)
\end{aligned}
$$

where the equality in the second line above holds because $\bar{S} = S \setminus U$, the third line follows by using the definition of the function $\widehat{F}_1(w)$, and the last line is due to the fact that $\widehat{w}$ is the minimizer for the function $\widehat{F}_1(w)$ and hence $\nabla \widehat{F}_1(\widehat{w}) = 0$. Plugging the above in (16), we get that

$$
\|-\frac{1}{n-m}\sum_{z \in U} \nabla f(\widehat{w}; u) + \nabla^2 \widehat{F}_2(\widehat{w})[\widehat{w}' - \widehat{w}]\| \leq \frac{M}{2}\|\widehat{w} - \widehat{w}'\|^2. \tag{17}
$$

Algorithm 2 Learning algorithm (A_{sc})

```
Input: Dataset $S : \{z_i\}_{i=1}^n \sim \mathcal{D}^n$, loss function: $f$.
1: Compute
       $\widehat{w} \leftarrow \text{argmin } \widehat{F}_n(w) := \frac{1}{n}\sum_{i=1}^n f(w, z_i).$
2: Return $(\widehat{w}, \nabla^2 \widehat{F}(\widehat{w}))$.
```

Now, let us define the vector v such that

$$\widehat{w}' = \widehat{w} + \frac{1}{n-m}(\nabla^2 \widehat{F}_2(\widehat{w}))^{-1} \sum_{z \in u} \nabla f(\widehat{w}; z) + v. \tag{18}$$

Plugging the above relation in (17), we get that

$$\|\nabla^2 \widehat{F}_2(\widehat{w})v\| \leq \frac{M}{2}\|\widehat{w} - \widehat{w}'\|^2. \tag{19}$$

Since, the function $\widehat{F}_2$ is λ-strongly convex, we have that $\|\nabla^2 \widehat{F}_2(\widehat{w})v\| \geq \lambda\|v\|$ for any vector v. Using this fact in (19), we get that

$$\|v\| \leq \frac{M}{2\lambda}\|\widehat{w} - \widehat{w}'\|^2.$$

Finally, an application of Lemma 6 implies that $\|\widehat{w} - \widehat{w}'\| \leq \frac{2mL}{\lambda n}$, using which in the above bound, we get that

$$\|v\| \leq \frac{2Mm^2L^2}{\lambda^3 n^2}.$$

Plugging in the definition of the vector v from (18), we get that

$$\left\|\widehat{w}' - \widehat{w} - \frac{1}{n-m}(\nabla^2 \widehat{F}_2(\widehat{w}))^{-1} \sum_{z \in u} \nabla f(\widehat{w}; z)\right\| \leq \frac{2Mm^2L^2}{\lambda^3 n^2}.$$

The desired bound follows by setting $\widehat{H} := \frac{1}{n-m}\sum_{z \in S \setminus U} \nabla^2 f(\widehat{w}, z) = \nabla^2 \widehat{F}_2(\widehat{w})$. $\square$

C.2 Proof of Theorem 3

Before we delve into the proof of Theorem 3, we first provide in Algorithm 2, the pseudocode for the learning algorithm A_{sc}. We also recall the following technical result that provides excess risk guarantees for the empirical risk minimizer when the loss is strongly convex and Lipschitz.

Lemma 7 (Claim 6.2 in Shalev-Shwartz et al. [2009b]). For any $z \in \mathcal{Z}$, let $f(w, z)$ be a L-Lipschitz and λ-strongly convex function in the variable w. Given any distribution $\mathcal{D}$, let $S = \{z_i\}_{i=1}^n$ denote a dataset of n samples drawn independently from $\mathcal{D}$. Let the point $\widehat{w}$ be defined as $\widehat{w} := \text{argmin}_w \frac{1}{n}\sum_{i=1}^n f(w, z_i)$. Then,

$$\mathbb{E}[F(\widehat{w}) - F(w^*)] \leq \frac{4L^2}{\lambda n},$$

where the function $F(w) := \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)]$ and $w^* \in \text{argmin}_w F(w)$.

We are now ready to prove the statements of Theorem 3. We prove each part in a separate lemma below. The following result provides performance guarantee for the output of the learning algorithm A_{sc}.

Lemma 8 (Learning guarantee for A_{sc}). For any distribution $\mathcal{D}$, the output $\widehat{w}$ of running Algorithm 2 on the dataset $S \sim \mathcal{D}^n$ satisfies

$$\mathbb{E}_{S \sim \mathcal{D}^n}[F(\widehat{w})] - F^* \leq \frac{4L^2}{\lambda n},$$

where F^* denotes $\min_{w \in \mathcal{W}} F(w)$.

Proof of Lemma 8. We note that the point $\widehat{w}$ is given by the empirical risk minimizer on the dataset S, i.e.

$$\widehat{w} \leftarrow \operatorname*{argmin}_w \frac{1}{n} \sum_{z \in S} f(w, z).$$

Since the function $f(w, z)$ is λ-strongly convex and L-Lipschitz, the desired performance guarantee for the ERM point follows from Lemma 7. $\square$

Next, we provide performance guarantees for the output of the unlearning algorithm $\bar{A}_{sc}$.

Lemma 9. For any dataset S, output $\widehat{w}$ of $A_{sc}(S)$ and set U of m delete requests, the point $\widetilde{w}$ returned by Algorithm 1 satisfies

$$\mathbb{E}[F(\widetilde{w}) - F^*] = O\Big(\frac{\sqrt{d} M m^2 L^3}{\lambda^3 n^2 \varepsilon} \sqrt{\ln(1/\delta)} + \frac{4mL^2}{\lambda n}\Big),$$

where the expectation above is taken with respect to the dataset S and noise ν.

Proof of Lemma 9. We recall that

$$\widetilde{w} = \widehat{w} + \frac{1}{n-m} (\widehat{H})^{-1} \sum_{z \in u} \nabla f(\widehat{w}, z) + \nu, \tag{20}$$

where the vector $\nu \in \mathbb{R}^d$ is drawn independently from $\mathcal{N}(0, \sigma^2 \mathbb{I}_d)$ with σ given by $\sqrt{2 \ln(\frac{1.25}{\delta})} \cdot \frac{2Mm^2L^2}{\lambda^3 n^2 \varepsilon}$. Thus,

$$\begin{aligned} \mathbb{E}[F(\widetilde{w}) - F(w^*)] &= \mathbb{E}[F(\widetilde{w}) - F(\widehat{w}) + F(\widehat{w}) - F(w^*)] \\ &= \mathbb{E}[F(\widetilde{w}) - F(\widehat{w})] + \mathbb{E}[F(\widehat{w}) - F(w^*)] \leq \mathbb{E}[L \|\widetilde{w} - \widehat{w}\|] + \frac{4L^2}{\lambda n}, \end{aligned} \tag{21}$$

where the inequality in the last line follows from the fact that the function $F = \mathbb{E}[f(w,z)]$ is L-Lipschitz, and by using Lemma 8. Further, from the relation in (20), we have that

$$\begin{aligned} \mathbb{E}[\|\widetilde{w} - \widehat{w}\|] &= \mathbb{E}[\| \frac{1}{n-m} (\widehat{H})^{-1} \sum_{z \in U} \nabla f(\widehat{w}, z) + \nu \|] \\ &\overset{(i)}{\leq} \mathbb{E}[\sum_{z \in U} \| \frac{1}{n-m} (\widehat{H})^{-1} \nabla f(\widehat{w}, z) \|] + \mathbb{E}[\|\nu\|] \\ &\overset{(ii)}{\leq} \sum_{z \in U} \frac{1}{(n-m)\lambda} \mathbb{E}[\|\nabla f(\widehat{w}, z)\|] + \sqrt{\mathbb{E}[\|\nu\|^2]}, \end{aligned}$$

where the inequality in (i) follows from an application of the triangle inequality, and the inequality (ii) holds because the function $F(w)$ is λ-strongly convex which implies that $\nabla^2 F(\widehat{w}) \succcurlyeq \lambda \mathbb{I}_d$, and by an application of Jensen's inequality to bound $\mathbb{E}[\|\nu\|]$. Next, using the fact that the vector $\nu \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_d)$, we get that

$$\begin{aligned} \mathbb{E}[\|\widetilde{w} - \widehat{w}\|] &\leq \frac{1}{(n-m)\lambda} \mathbb{E}[\sum_{z \in U} \|\nabla f(\widehat{w}, z)\|] + \sqrt{d}\sigma \\ &\leq \frac{mL}{(n-m)\lambda} + \sqrt{d}\sigma, \end{aligned}$$

where the last line holds because $f(w, z)$ is L-Lipschitz. Using the above bound in (21), we get

$$\mathbb{E}[F(\widetilde{w}) - F(w^*)] \leq \frac{mL^2}{(n-m)\lambda} + \sqrt{d}\sigma L + \frac{4L^2}{\lambda n}.$$

Our final guarantee follows by plugging in the value of σ and using the fact that $n = \Omega(m)$. $\square$

Finally, we show that the algorithms A_{sc} and $\bar{A}_{sc}$ are (ε, δ)-unlearning.

Lemma 10 (Unlearning guarantee)**.** For any distribution $\mathcal{D}$, dataset S and set of delete requests $U \subseteq S$, the algorithms A_{sc} and $\bar{A}_{sc}$ satisfy the following guarantees for any set $W \subseteq \mathbb{R}^d$,

(a) $\Pr(\bar{A}_{sc}(U, A_{sc}(S), T(S)) \in W) \le e^{\varepsilon} \Pr(\bar{A}_{sc}(\emptyset, A_{sc}(S \setminus U), T(S \setminus U)) \in W) + \delta$, and

(b) $\Pr(\bar{A}_{sc}(\emptyset, A_{sc}(S \setminus U), T(S \setminus U)) \in W) \le e^{\varepsilon} \Pr(\bar{A}_{sc}(U, A_{sc}(S), T(S)) \in W) + \delta$.

Proof of Lemma 10. The proof follows along the lines of the proof of the differential privacy guarantee for the Gaussian mechanism (see e.g., Dwork and Roth [2014, Appendix A]).

Let $\widehat{w}$ denote the output of the learning algorithm A_{sc} when run on dataset S, and let $\widetilde{w}$ denote the corresponding output of the unlearning algorithm $\bar{A}_{sc}$ when run with delete requests U, the input model $\widehat{w}$ and data statistics $T(S)$, i.e. $\widehat{w} = A_{sc}(S)$ and $\widetilde{w} = \bar{A}_{sc}(U, \widehat{w}, T(S))$. Additionally, let $\bar{w}$ be the local variable defined in line 3 of $\bar{A}_{sc}$ (see Algorithm 1) when computing $\widetilde{w}$.

Similarly, let $\widehat{w}'$ denote the output of the learning algorithm A_{sc} when run on dataset $S \setminus U$, and let $\widetilde{w}'$ denote the corresponding output of the unlearning algorithm $\bar{A}_{sc}$ when run with delete requests $\emptyset$, the input model $\widehat{w}'$ and data statistics $T(S \setminus U)$, i.e. $\widehat{w}' = A_{sc}(S \setminus U)$ and $\widetilde{w}' = \bar{A}_{sc}(\emptyset, \widehat{w}', T(S \setminus U))$. Additionally, let $\bar{w}'$ be the local variable defined in line 3 of $\bar{A}_{sc}$ for this case.

Note that in the algorithm A_{sc}, the points $\widehat{w}$ and $\widehat{w}'$ are computed as:

$$\widehat{w} = \operatorname*{argmin}_w \frac{1}{n} \sum_{z \in S} f(w, z) \qquad \text{and} \qquad \widehat{w}' = \operatorname*{argmin}_w \frac{1}{n-m} \sum_{z \in S \setminus U} f(w, z).$$

An application of Lemma 3 thus gives us the bound

$$\left\| \widehat{w}' - \widehat{w} - \frac{1}{n-m} (\widehat{H})^{-1} \sum_{z \in U} \nabla f(\widehat{w}, z) \right\| \le \frac{2Mm^2L^2}{\lambda^3 n^2},$$

where the matrix $\widehat{H}$ is defined in (7). Using the relation in (8) for the points $\bar{w}$ and $\bar{w}'$, and observing that $\widehat{w}' = \bar{w}'$ since $U = \emptyset$ in the calculation of $\widehat{w}'$, we get that

$$\|\bar{w}' - \bar{w}\| \le \frac{2Mm^2L^2}{\lambda^3 n^2} =: \gamma. \tag{22}$$

Next, note that in the algorithm $\bar{A}_{sc}$, the points $\widetilde{w}$ and $\widetilde{w}'$ are computed as $\widetilde{w} = \bar{w} + \nu$ and $\widetilde{w}' = \bar{w}' + \nu$ respectively, where the noise $\nu \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_d)$ with $\sigma = (\gamma/\varepsilon) \cdot \sqrt{2\ln(1.25/\delta)}$. Thus, following the same proof as Dwork and Roth [2014, Theorem A.1], with the bound (22), we get that for any set W,

$$\Pr(\widetilde{w} \in W) \le e^{\varepsilon} \Pr(\widetilde{w}' \in W) + \delta,$$

and

$$\Pr(\widetilde{w}' \in W) \le e^{\varepsilon} \Pr(\widetilde{w} \in W) + \delta,$$

giving us the desired unlearning guarantee. □

Proof of Theorem 3. The statements of the theorem follow immediately from Lemma 8, Lemma 9 and Lemma 10 respectively. □

D Unlearning algorithms for convex loss function

In this section, we provide learning and unlearning algorithms when f is convex (but not necessarily strongly convex). Similar to the strongly convex setting, we assume that

Assumption 2. For any $z \in \mathcal{Z}$, the function $f(w, z)$ is convex, L-Lipschitz and M-Hessian Lipschitz with respect to w.

Additionally, we also assume the following about some minimizer of the population loss $F(w) = \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)]$.

Assumption 3. There exists a $w^* \in \operatorname{argmin}_{w \in \mathcal{W}} F(w)$ such that $\|w^*\| \le B$.

Algorithm 3 Learning algorithm (A_c)

```
Input: Dataset $S : \{z_i\}_{i=1}^n \sim \mathcal{D}^n$, loss function: $f$, regularization parameter: $\lambda$.
1: Define
       $\widetilde{f}(w, z) := f(w, z) + \frac{\lambda}{2}\|w\|^2.$
2: Run the algorithm $A_{sc}$ on the dataset $S$ with loss function $\widetilde{f}$.
3: return $(\widehat{w}, \widetilde{H}) \leftarrow A_{sc}(S; \widetilde{f})$, where $\widetilde{H} := \frac{1}{n}\sum_{z \in S} \widetilde{f}(\widehat{w}, z)$.
```

Algorithm 4 Unlearning algorithm ($\bar{A}_c$)

```
Input: Delete requests: $U = \{z_j\}_{j=1}^m \subseteq S$, output of $A_c$: $\widehat{w}$, additional statistic $T(S) : \widetilde{H}$, loss
    function: $f$, regularization parameter: $\lambda$.
1: Define
       $\widetilde{f}(w, z) := f(w, z) + \frac{\lambda}{2}\|w\|^2.$
2: Run the algorithm $\bar{A}_{sc}$ for the delete request $U$ with input model $\widetilde{w}$ and with loss function $\widetilde{f}$.
3: return $\widetilde{w} \leftarrow \bar{A}_{sc}(S, \widehat{w}, T(S); \widetilde{f})$.
```

Our algorithms for convex losses are based on algorithms for the strongly convex setting. Given the convex function $f(\cdot, z)$, define the function $\widetilde{f}(\cdot, z)$ as

$$\widetilde{f}(w, z) = f(w, z) + \frac{\lambda}{2}\|w\|^2.$$

The key observation is that at any $w \in \mathbb{R}^d$, the function $\widetilde{f}(w, z)$ is λ-strongly convex, $(L + \lambda\|w\|)$-Lipschitz, $(H + \lambda)$-smooth and M-Hessian Lipschitz in w for any z. Clearly, the function $\widetilde{f}$ satisfies Assumption 1 whenever w is such that $\|w\| \leq L/\lambda$ (see Lemma 14), and thus we can run algorithms A_{sc} and $\bar{A}_{sc}$ respectively on the function $\widetilde{f}$.

Our learning and unlearning algorithms for the convex loss f simply invoke the algorithms A_{sc} and $\bar{A}_{sc}$ on the function $\widetilde{f}$ with an appropriate choice of λ. We provide the pseudocode in Algorithm 3 and Algorithm 4 respectively.

In order to avoid confusion in this section, for any algorithm A, we use the notation $A(S; f)$ to denote the fact that A is run on the loss function f (similarly for $\bar{A}$). Whenever clear from context, we will drop the argument f from the notation. Additionally, we also define $\widetilde{F}$ to denote the population loss w.r.t. the loss function $\widetilde{f}$, i.e. $\widetilde{F}(w) := \mathbb{E}_{z\sim\mathcal{D}}[\widetilde{f}(w, z)]$.

Theorem 4. *Suppose the loss function f satisfy Assumption 2 and Assumption 3. Let the dataset $S \sim \mathcal{D}^n$. Then,*

(a) The point $\widehat{w}$ returned by running A_c on S satisfies

$$\mathbb{E}_{S\sim\mathcal{D}^n}[F(\widehat{w}) - \min_{w\in\mathcal{W}} F(w)] \leq \frac{\lambda B^2}{2} + \frac{16L^2}{\lambda n}. \tag{23}$$

(b) For any set $U \subseteq S$ of m delete requests, the point $\widetilde{w}$ returned by $\bar{A}_c$ satisfies

$$\mathbb{E}_{S,\nu}[F(\widetilde{w}) - \min_{w\in\mathcal{W}} F(w)] = O\Big(\frac{\lambda B^2}{2} + \frac{\sqrt{d}Mm^2L^3}{\lambda^3 n^2 \varepsilon}\sqrt{\ln(1/\delta)} + \frac{mL^2}{\lambda n}\Big). \tag{24}$$

(c) The learning algorithm A_c and the unlearning algorithm $\bar{A}_c$ are (ε, δ)-unlearning.

Corollary 1. Suppose we did not have any unlearning requests, and only cared about the performance of the output point $\widehat{w}$ for the learning algorithm A_c. Then, setting $\lambda = L/B\sqrt{n}$, the performance guarantee for the point $\widehat{w}$ given in Theorem 4 implies

$$\mathbb{E}[F(\widehat{w}) - F^*] \leq \frac{BL}{\sqrt{n}}.$$

The above rate is tight for learning with Lipschitz convex losses (see Bubeck [2014, Theorem 6.1]).

Corollary 2. Suppose that we have m delete requests and thus care about the performance guarantee of both the point $\widehat{h}$, output of the learning algorithm A_c, and the point $\widetilde{w}$, output of the unlearning algorithm $\bar{A}_c$. In this case, we set the regularization parameter λ as:

$$\lambda = \max\Big\{\frac{L}{B}\sqrt{\frac{m}{n}}, \Big(\frac{\sqrt{d}Mm^2L^3}{B^2n^2\varepsilon}\sqrt{\ln(1/\delta)}\Big)^{1/4}\Big\}. \tag{25}$$

Plugging the above values of λ in Theorem 4, we get that

$$\mathbb{E}[F(\widehat{w}) - F^*] = O\Big(c_1\sqrt{\frac{m}{n}} + c_2\Big(\frac{d\log(1/\delta)}{\varepsilon^2}\Big)^{1/8}\sqrt{\frac{m}{n}}\Big)$$

and

$$\mathbb{E}[F(\widetilde{w}) - F^*] = O\Big(c_1\sqrt{\frac{m}{n}} + c_2\Big(\frac{d\log(1/\delta)}{\varepsilon^2}\Big)^{1/8}\sqrt{\frac{m}{n}}\Big),$$

where the constant $c_1 \propto BL$ and $c_2 \propto \left(\frac{ML^3}{B^2}\right)^{1/4}$.

D.1 Proof of Theorem 4

We are now ready to prove the statements of Theorem 4. We prove each part in a separate lemma below. The following provides performance guarantee for the output of the learning algorithm A_c.

Lemma 11 (performance guarantee for A_c)**.** For any $\lambda > 0$ and $S \sim \mathcal{D}^n$, the point $\widehat{w}$ returned by Algorithm 3 satisfies

$$\mathbb{E}[F(\widehat{w}) - F^*] \le \frac{\lambda B^2}{2} + \frac{16L^2}{\lambda n}.$$

Proof. First, note that an application of Lemma 14 implies that for any dataset S, the empirical minimizer $\widehat{w}$ (returned by A_c) satisfies: $\|\widehat{w}\| \le L/\lambda$. Thus, our domain of interest is $\mathcal{W} := \{w \mid \|w\| \le L/\lambda\}$. Over the set $\mathcal{W}$, the function $\widetilde{f}$ is $2L$-Lipschitz, and thus, an application of Lemma 8 implies that the returned point $\widehat{w}$ satisfies

$$\mathbb{E}[\widetilde{F}(\widehat{w})] \le \widetilde{F}(\widetilde{w}^*) + \frac{16L^2}{\lambda n},$$

where $\widetilde{w}^*$ denotes the minimizer of $\widetilde{F}(w)$. We can further upper bound the right hand side above as

$$\mathbb{E}[\widetilde{F}(\widehat{w})] \le \widetilde{F}(w^*) + \frac{16L^2}{\lambda n},$$

where w^* denotes a minimizer of the population loss $F(w)$ that satisfies Assumption 3. Plugging in the form of the function $\widetilde{F}(w)$ in the above, we get

$$\begin{aligned}\mathbb{E}[F(\widehat{w})] &\le F(w^*) + \frac{\lambda}{2}\|w^*\|^2 + \frac{16L^2}{\lambda n} \\ &\le F(w^*) + \frac{\lambda B^2}{2} + \frac{16L^2}{\lambda n},\end{aligned}$$

where the last line holds due to Assumption 3. □

Lemma 12 (performance guarantee for $\bar{A}_c$)**.** For any $\lambda > 0$, dataset $S \sim \mathcal{D}^n$, output $\widehat{w}$ of $A_c(S)$ and set U of m delete requests, the point $\widetilde{w}$ returned by Algorithm 4 satisfies

$$\mathbb{E}[F(\widehat{w}) - F^*] = O\Big(\frac{\lambda B^2}{2} + \frac{\sqrt{d}Mm^2L^3}{\lambda^3n^2\varepsilon}\sqrt{\ln(1/\delta)} + \frac{mL^2}{\lambda n}\Big).$$

Proof. Let $w^* \in \operatorname{argmin}_w F(w)$. We note that

$$\begin{aligned}\mathbb{E}[F(\widetilde{w}) - F(w^*)] &= \mathbb{E}[F(\widetilde{w}) - F(\widehat{w}) + F(\widehat{w}) - F(w^*)] \\ &= \mathbb{E}[F(\widetilde{w}) - F(\widehat{w})] + \mathbb{E}[F(\widehat{w}) - F(w^*)] \\ &\leq \mathbb{E}[L\|\widetilde{w} - \widehat{w}\|] + \mathbb{E}[F(\widehat{w}) - F(w^*)], \qquad (26)\end{aligned}$$

where the inequality in the last line holds because the loss function $f(w, z)$, and thus the function $F(w)$, is L-Lipschitz. We next note that $\bar{A}_c$ computes the point $\widetilde{w}$ by running the algorithm $\bar{A}_{sc}$ with inputs U, $\widehat{w}$ on the loss function $\widetilde{f}$. Thus, we have that

$$\mathbb{E}[\|\widetilde{w} - \widehat{w}\|] = \mathbb{E}\Big[\Big\|\frac{1}{n-m}(\widehat{H})^{-1}\sum_{z \in U} \nabla \widetilde{f}(\widehat{w}, z) + \nu\Big\|\Big],$$

where $\widehat{H} := \frac{1}{n-m}\sum_{z \in S \setminus U} \nabla^2 \widetilde{f}(w, z)$. Since, the function $\widetilde{f}$ is λ-strongly convex, we note that $\widehat{H} \succcurlyeq \lambda \mathbb{I}_d$. Using this fact with the above relation implies

$$\begin{aligned}\mathbb{E}[\|\widetilde{w} - \widehat{w}\|] &\leq \frac{1}{\lambda(n-m)}\|\sum_{z \in U} \nabla \widetilde{f}(\widehat{w}, z)\| + \mathbb{E}\|\nu\| \\ &\leq \frac{1}{\lambda(n-m)}\sum_{z \in U}\|\nabla \widetilde{f}(\widehat{w}, z)\| + \mathbb{E}\|\nu\| \leq \frac{2mL}{\lambda(n-m)} + \sigma, \qquad (27)\end{aligned}$$

where the last line follows from the fact that $\|\widehat{w}\| \leq \frac{L}{\lambda}$, and thus

$$\|\nabla \widetilde{f}(\widehat{w}, z)\| \leq \|\nabla f(\widehat{w}, z)\| + \lambda\|\widehat{w}\| \leq 2L.$$

Finally, using (27) and Lemma 11 in (26), and by plugging in the value of σ implies the desired performance guarantee. □

Finally, the algorithms A_c and $\bar{A}_c$ are (ε, δ)-forgetting, as a consequence of Lemma 10.

Lemma 13 (Forgetting guarantee)**.** For any distribution $\mathcal{D}$, dataset S, set of delete requests $U \subseteq S$, the algorithms A_c and $\bar{A}_c$ satisfy the following guarantees for any set $W \subseteq \mathbb{R}^d$,

(a) $\Pr(\bar{A}_c(U, A_c(S), T(S)) \in W) \leq e^{\varepsilon} \Pr(\bar{A}_c(\emptyset, A_c(S \setminus U), T(S \setminus U)) \in W) + \delta$, and

(b) $\Pr(\bar{A}_c(\emptyset, A_c(S \setminus U), T(S \setminus U)) \in W) \leq e^{\varepsilon} \Pr(\bar{A}_c(U, A_c(S), T(S)) \in W) + \delta$.

Proof of Theorem 3. The desired statements follow immediately from Lemma 11, Lemma 12 and Lemma 13 respectively. □

D.2 Supporting technical results

The following lemma gives a bound on the regularized empirical risk minimizer point for the loss function $f(w, z)$ for any dataset S. This gives us a bound on the domain of interest, and thus allows us to bound the Lipschitz constant for loss function over this domain.

Lemma 14. Let $f(w, z)$ be a L-Lipschitz function in the variable w, and let $\widetilde{f}$ be defined as

$$\widetilde{f}(w, z) = f(w, z) + \frac{\lambda}{2}\|w\|^2.$$

Given a dataset $S = \{z_i\}_{i=1}^n$, define $\widehat{G}(w) = \frac{1}{n}\sum_{i=1}^n \widetilde{f}(w, z_i)$, and let $\widehat{w}$ denote the empirical risk minimizer of the loss function $\widetilde{f}$ on dataset S, i.e. $\widehat{w} \in \operatorname{argmin}_w \widehat{G}(w)$. Then, the point $\widehat{w}$ satisfies $\|\widehat{w}\| \leq \frac{L}{\lambda}$.

Proof. Since, $\widehat{w} \in \operatorname{argmin}_w \widehat{G}(w)$, we have

$$\nabla \widehat{G}(\widehat{w}) = \frac{1}{n}\sum_{z \in S} \nabla \widetilde{f}(\widehat{w}, z) = 0.$$

Plugging in the definition of the function $\widetilde{f}$ in the above implies that

$$\frac{1}{n}\sum_{z\in S}\nabla f(\widehat{w},z)+\lambda\widehat{w}=0.$$

Rearranging the terms, we get

$$\|\lambda\widehat{w}\|=\|\frac{1}{n}\sum_{z\in S}\nabla f(\widehat{w},z)\|\overset{(i)}{\leq}\frac{1}{n}\sum_{z\in S}\|\nabla f(\widehat{w},z)\|\overset{(ii)}{\leq}L,$$

where the inequality in (i) follows from an application of the Triangle inequality, and the inequality in (ii) holds because the function f is L-Lipschitz in the variable w, and thus $\|\nabla f(w,z)\|\leq L$ for all $w\in\mathcal{W}$ and $z\in\mathcal{Z}$. $\square$