Unlearning via Simulated Oracle Matching

Kristian Georgiev*[1], Roy Rinberg*[2,3], Sung Min Park*[1], Shivam Garg*[2,4]
Andrew Ilyas[1], Aleksander Mądry[1], Seth Neel[2]

[1]MIT EECS [2]Harvard Business School [3]Harvard SEAS [4]Microsoft Research

Abstract

Machine unlearning—efficiently removing the effect of a small "forget set" of training data on a pre-trained machine learning model—has recently attracted significant research interest. Despite this interest, however, recent work shows that existing machine unlearning techniques do not hold up to thorough evaluation in non-convex settings. In this work, we introduce a new machine unlearning technique that exhibits strong empirical performance even in such challenging settings. Our starting point is the perspective that the goal of unlearning is to produce a model whose outputs are *statistically indistinguishable* from those of a model re-trained on all but the forget set. This perspective naturally suggests a reduction from the unlearning problem to that of *data attribution*, where the goal is to predict the effect of changing the training set on a model's outputs. Thus motivated, we propose the following meta-algorithm, which we call Datamodel Matching (DMM): given a trained model, we (a) use data attribution to *predict* the output of the model if it were re-trained on all but the forget set points; then (b) *fine-tune* the pre-trained model to match these predicted outputs. In a simple convex setting, we show how this approach provably outperforms a variety of iterative unlearning algorithms. Empirically, we use a combination of existing evaluations and a new metric based on the KL-divergence to show that even in non-convex settings, DMM achieves strong unlearning performance relative to existing algorithms. An added benefit of DMM is that it is a meta-algorithm, in the sense that future advances in data attribution translate directly into better unlearning algorithms, pointing to a clear direction for future progress in unlearning.[1]

1 Introduction

The goal of machine *unlearning* is to remove (or "unlearn") the impact of a specific collection of training examples from a trained machine learning model. Initially spurred by regulations such as the EU's *Right to be Forgotten* (Ginart et al., 2019), machine unlearning has found a variety of recent applications including: removing the effect of toxic, outdated, or poisoned data (Pawelczyk et al., 2024b; Goel et al., 2024); rectifying copyright infringement in generative models (Liu, 2024; Dou et al., 2024); and even LLM safety training (Li et al., 2024; Yao et al., 2024).

This plethora of potential applications has prompted a growing line of research into better *unlearning algorithms*. An unlearning algorithm takes as input a model θ (trained on a dataset S) and a "forget set" $S_F \subset S$, and outputs a model θ_{UL} that "looks like" it was trained on the so-called "retain set" $S_R := S \setminus S_F$. Of course, one valid unlearning algorithm simply ignores the trained model θ and trains a new model θ_{UL} from scratch on the retain set S_R. This algorithm clearly

*Equal contribution
[1]Code: `bit.ly/unlearning-via-simulated-oracles`.

succeeds at the task of unlearning, since the generated θ_{UL} really *is* trained only on the retain set. But as model and dataset sizes continue to increase, or unlearning requests become more frequent, this approach becomes infeasible. The goal of unlearning is thus to *approximate* this naive retraining algorithm while imposing a much lower computational burden.

For simple (e.g., convex) models, there are fast unlearning algorithms that also enjoy provable guarantees (Neel et al., 2021; Graves et al., 2021; Izzo et al., 2021b; Mahadevan & Mathioudakis, 2021; Suriyakumar & Wilson, 2022). For large neural networks, however—where efficient unlearning is arguably most relevant, given the cost of training from scratch—the situation is considerably murkier. The only methods that obtain provable guarantees tend to significantly degrade accuracy and/or require significant changes to the training pipeline (Bourtoule et al., 2020a; Li et al., 2022).

As a result, unlearning algorithms for neural networks typically rely on heuristic approaches that finetune an initial model θ into an "empirically unlearned" model $\widehat{\theta_{UL}}$. These approaches, however, have not yet led to consistently reliable unlearning algorithms, as evidenced by a variety of empirical evaluations and benchmarks (Hayes et al., 2024; Kurmanji et al., 2023; Pawelczyk et al., 2023).

A pervasive challenge (identified by Hayes et al. (2024)) for fine-tuning-based approaches is what we refer to as the *missing targets* problem. The problem can be summarized as follows: in order to "unlearn" a forget set point $x \in S_F$, fine-tuning-based methods typically employ some version of *gradient ascent* on x, starting from θ, and gradient descent on the retain set S_R in order to maintain performance. If left unrestricted, gradient ascent will continue to make the loss on x arbitrarily high—what we want, however, is to increase the loss only until it reaches its *counterfactual value*, i.e., the loss on x of a model trained on the retain set S_R. Ideally, we could terminate the algorithm when the model's loss on x reaches this "target" value, but the problem is that (a) we do not have access to the target; and (b) the optimal "stopping time" might be different for different points $x \in S_F$. The result is the well-documented phenomenon of unlearning algorithms "undershooting" and "overshooting" the loss on different examples (Hayes et al., 2024).

This work. In this paper, we present a new unlearning algorithm that sidesteps the issue discussed above, and (empirically) achieves state-of-the-art unlearning performance. Our algorithm resembles prior techniques in that we rely on fine-tuning the trained model θ. We deviate from prior work, however, through two main ideas:

1. **Oracle Matching (OM).** Consider the following thought experiment: what if we could access the *outputs* (but not the parameters) of a model trained on the retain set S_R? We show that such "oracle" access directly enables an efficient, fine-tuning-based unlearning algorithm. Rather than minimizing/maximizing loss on the retain/forget sets, this algorithm simply picks a subset of the entire dataset that includes the forget set, and minimizes the difference between model outputs and oracle outputs. Conceptually, this algorithm is "stable" in that upon convergence, the model agrees with the oracle on all the fine-tuning points, including those in the forget set—thus sidestepping the aforementioned "missing targets" problem. Empirically, we find that the fine-tuned model also *generalizes* beyond the fine-tuning points, and in some way "distills" the target model into parameters θ'.

2. **Oracle simulation.** OM on its own is not an unlearning algorithm—it relies on the very "oracle model" that it aims to replicate. Observe, however, that implementing OM does not require access to the weights of an oracle model, but only to its outputs on a fixed number of inputs. Thus, OM can be implemented efficiently given access to an efficient routine for computing such outputs. Such a routine is precisely the target of *predictive data attribution* methods (Ilyas et al., 2024), where the goal is exactly to predict how a model's outputs would change if

its training dataset were modified. This leads to our second idea: instead of fine-tuning on "oracle" outputs, we fine-tune on *simulated* outputs from a predictive data attribution method. We show that despite these methods being imperfect, applying our OM algorithm to *simulated* oracle outputs works nearly as well as using the true oracle outputs.

The resulting algorithm, *datamodel matching* (DMM), not only achieves current state-of-the-art performance (Figure 1), but also introduces a reduction from unlearning to data attribution, allowing us to translate future improvements in the latter field to better algorithms for the former.

The rest of our paper proceeds as follows. In Section 2, we formally introduce the unlearning problem, as well as the field of (predictive) data attribution. In Section 3, we strengthen existing unlearning evaluation by introducing a new metric called *KL Divergence of Margins* (KLoM). KLoM directly adapts a formal definition of unlearning (Neel et al., 2021) to be computationally and statistically tractable to estimate, and addresses some challenges faced by existing unlearning metrics. Then, in Section 4, we combine the two insights above (Oracle Matching and Oracle Simulation) into a simple algorithm called *datamodel matching* (DMM). Our algorithm achieves state-of-the-art performance across a suite of empirical evaluations. Finally, in Section 6, we provide some theoretical justification for our algorithm using a case study of underdetermined ridge (linear) regression. In particular, we show that in this simple setting, the oracle matching (OM) primitive can provably lead to faster convergence. We conclude with a discussion of limitations and directions for future work.

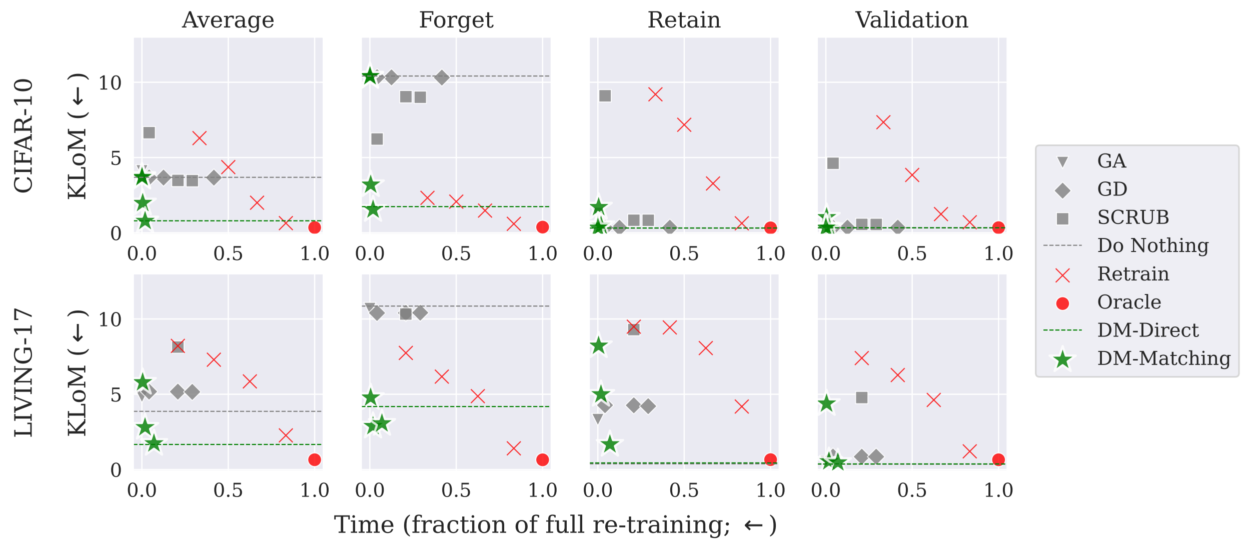


Figure 1: **Leveraging predictive data attribution enables effective unlearning.** We apply different approximate unlearning methods to trained DNNs to unlearn forget sets from CIFAR-10 and ImageNet-Living-17. We measure unlearning quality using KLoM scores (y-axis), which quantifies the distributional distance between unlearned predictions and oracle predictions (0 being perfect). To contextualize each method's efficiency, we also show the amount of compute relative to full re-training (x-axis). We evaluate KLoM values over points in the forget, retain, and validation sets to ensure that unlearning is effective across all datapoints, and report the 95th percentile in each group; we also report their weighted average (1st column). Our new methods based on data attribution (DM-DIRECT and DMM) dominate the pareto frontier of existing unlearning methods, and approach the unlearning quality of oracle models (full re-training) at a much smaller fraction of the cost.

2 Preliminaries

In this section, we introduce some preliminary notation, definitions, and results. Throughout the section, we will let $S \in \mathcal{X}^n$ be a fixed dataset drawn from an example space $\mathcal{X}$, and we define a *learning algorithm* $\mathcal{A} : \mathcal{X}^* \to \Theta$ as a (potentially random) function mapping from datasets to machine learning models θ. Finally, for an example $x \in \mathcal{X}$, we use $f_x : \Theta \to \mathbb{R}^k$ to denote a *model evaluation* on the example x (for example, this may be the k-dimensional per-class probabilities).

2.1 Machine unlearning

Consider a machine learning model $\theta \sim \mathcal{A}(S)$ trained on a dataset S. Given a "forget set" $S_F \subset S$, and a corresponding "retain set" $S_R = S \setminus S_F$, the goal of an *exact* unlearning algorithm is to compute a sample from $\mathcal{A}(S_R)$ starting from the trained model θ:

Definition 1 (Exact unlearning (Ginart et al., 2019))**.** *An unlearning algorithm $\mathcal{U} : \Theta \times 2^{|S|} \to \Theta$ is said to be an exact unlearning algorithm if, for all $S_F \subset S$, $\mathcal{U}(\mathcal{A}(S), S_F) \overset{d}{=} \mathcal{A}(S_R)$, where $\overset{d}{=}$ represents equality in distribution over models.*

Though compelling in theory, exact unlearning tends to be too stringent a criterion when applied to deep learning, often leading to computational infeasibility or degradations in accuracy (Liu, 2024). This motivates a look at *approximate unlearning*, which asks only for the distribution over unlearned models to be (ϵ, δ)-indistinguishable from re-training:

Definition 2 ((ε, δ)-unlearning (Neel et al., 2021))**.** *$\mathcal{U}$ is an (ϵ, δ)-approximate unlearning algorithm if, for all $\mathcal{O} \subset \Theta, S_F \subset S$ we have that*

$$\begin{aligned} \Pr\left[\mathcal{U}(\mathcal{A}(S), S_F) \in \mathcal{O}\right] &\leq e^{\epsilon} \Pr\left[\mathcal{A}(S_R) \in \mathcal{O}\right] + \delta, \\ \Pr\left[\mathcal{A}(S_R) \in \mathcal{O}\right] &\leq e^{\epsilon} \Pr\left[\mathcal{U}(\mathcal{A}(S), S_F) \in \mathcal{O}\right] + \delta \end{aligned} \tag{1}$$

This definition (intentionally) resembles differential privacy, and asks for the distribution of unlearned models to be statistically close to the distribution of re-trained "oracle" models. In particular, this condition guarantees than an adversary who observes the model returned by the unlearning algorithm $\mathcal{U}$ cannot draw any inferences with accuracy that is much higher than if the model was fully re-trained.

While unlearning algorithms achieving Definition 2 exist for convex models (Neel et al., 2021; Izzo et al., 2021b; Guo et al., 2019), and for non-convex models when the training process is altered or under stylized optimization conditions (Bourtoule et al., 2021a; Chien et al., 2024; Gupta et al., 2021), the bulk of ongoing work in unlearning evaluates Definition 2 *empirically*, rather than as a provable property. We return to the problem of evaluating unlearning algorithms more carefully in Section 3.

2.2 Predictive data attribution (Datamodeling)

Our work also draws on a separate line of work in machine learning called *data attribution* (Koh & Liang, 2017; Hammoudeh & Lowd, 2024; Ilyas et al., 2024). Broadly, data attribution is an area concerned with connecting training data samples to the predictions of the corresponding ML models. Of particular relevance to our work is a particular type of data attribution called *predictive data attribution* (also known as datamodeling (Ilyas et al., 2022; Park et al., 2023)).

In predictive data attribution, the goal is to produce an estimator (or *datamodel*) that takes as input a training set, and as output accurately predicts the behavior of a machine learning model trained on that training set. Using our existing notation: for an example $x \in \mathcal{X}$, a datamodel for x is a function $\hat{f} : 2^S \to \mathbb{R}$ such that, for any $S' \subset S$,

$$\hat{f}(S') \approx f_x(\mathcal{A}(S')). \tag{2}$$

In other words, $\hat{f}(S')$ directly predicts the result of applying the training algorithm $\mathcal{A}$ to the dataset S', and evaluating the function f on the resulting model. Despite the complexity of modern training algorithms $\mathcal{A}$ (e.g., training deep neural networks with stochastic gradient descent), Ilyas et al. (2022) show that *linear* datamodels often suffice to accurately predict model behavior. In other words, for an example x, one can compute a vector $\beta \in \mathbb{R}^{|S|}$ such that, for subsets $S' \subset S$,

$$\hat{f}_x(S') := \sum_{z_i \in S'} \beta_i \approx f_x(\mathcal{A}(S')).$$

To compute these coefficients, Ilyas et al. (2022) sample a variety of subsets $S_1, \ldots, S_k$ at random from S, and then solve the (regularized) regression problem

$$\beta = \min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} (w^\top \mathbf{1}_{S_i} - f_x(\mathcal{A}(S_i)))^2 + \lambda \|w\|_1. \tag{3}$$

They show that despite the datamodel being constructed using random subsets $S_i \subset S$, the function $\hat{f}$ remains remarkably accurate on non-random datasets (see Ilyas et al. (2022) for a full evaluation). Linear datamodels are particularly appealing for two reasons. First, the coefficients β_i have an intuitive interpretation as the influence of the i-th training example on a model's prediction on x (Koh & Liang, 2017; Feldman, 2021). Second, they establish a connection to a class of statistical techniques relating to influence functions, which has unlocked a suite of tools for estimating the coefficients more effectively (Park et al., 2023; Grosse et al., 2023).

3 Empirically evaluating unlearning

In Section 2, we introduced the unlearning problem, culminating in a formal definition of the problem (Definition 2). As stated, evaluating whether Definition 2 holds for a given unlearning algorithm is a difficult problem for several reasons. First, given the overparameterized nature of large-scale models, fully satisfying Definition 2 is likely impossible, and verifying it involves comparing distributions in a space with millions or billions of dimensions. Secondly, the definition generally needs to hold over arbitrary forget sets S_F, or at least across a range of forget sets S_F likely to occur in practice.

The current evaluation paradigm. To deal with these problems, one typically evaluates unlearning by focusing on model *outputs* f_x[2] rather than model parameters, and testing for the *implications* of Definition 2 rather than for the definition directly. In particular, the strongest existing unlearning evaluation for supervised learning, called `U-LiRA` (Hayes et al., 2024), takes inspiration from *membership inference attacks* (MIAs) (Carlini et al., 2022) and evaluates the ability of an adversary to distinguish between the distribution of outputs of an unlearned model on (a) validation examples and (b) unlearned examples.

Omitting a few details, the main idea is as follows: consider an unlearning algorithm $\mathcal{U}$ and a training algorithm $\mathcal{A}$. Starting from a training set S, first randomly hold out a small validation set S_V. From what remains, fix a forget set $S_F \subset S$ and sample datasets $S_i \subset S$. Train a model

[2]A common choice of model output used in prior work, which we will also use, is the *margin* of the classifier.

$\theta^* \sim \mathcal{A}(S_i)$ on each dataset, and then unlearn S_F to produce $\theta^*_{\mathcal{U}} \sim \mathcal{U}(\theta^*, S_F)$. By Definition 2, these models should be indistinguishable from models that were not trained on S_F, and thus should behave (nearly) identically on points $x \in S_F$ and on points $x \in S_V$.

Operationalizing this intuition, `U-LiRA` with probability $\frac{1}{2}$ draws either $x \in S_F$ or $x \in S_V$, and measures the output $y = f_x(\theta^*_{\mathcal{U}})$. An (optimal) adversary observes y, and tries to guess whether the corresponding x was an unlearned point or a validation point. If $\mathcal{U}$ is an (ϵ, δ)-unlearning algorithm, the two distributions $y|x \in S_F$ and $y|x \in S_V$ would be (ϵ, δ)-indistinguishable by post-processing, and so the adversary could have accuracy greater than $\frac{1}{2}e^{\epsilon} + \delta$. By the Neyman-Pearson lemma the optimal adversary performs a likelihood ratio test, and `U-LiRA` is based on approximating this likelihood by estimating the distribution of unlearned and test margins.

For a more detailed description of `U-LiRA` and overview of similar MIA-based approaches, we refer the reader to (Hayes et al., 2024), and we include the pseudocode for the computationally efficient implementation of this evaluation Efficient-ULIRA in Appendix E.2.

A more direct evaluation. Recall from Section 2 that traditionally the target of unlearning algorithms has been (ε, δ)-approximate unlearning (Definition 2). Note that in essence, Definition 2 simply asks for the distribution induced by the unlearning algorithm be "close" to a distribution of models that have never been trained on S_F. In particular, we can view it as a special case of the condition

$$\Delta_\delta(\mathcal{U}(\mathcal{A}(S), S_F), \text{safe}(S_F)) \leq \epsilon, \tag{4}$$

where Δ_δ is a statistical divergence measure (in this case the δ-approximate max divergence) parameterized by $\delta > 0$, and $\text{safe}(S_F)$ is a distribution of "safe" models (i.e., models that have not been trained on S_F). We can recover Definition 2 exactly by letting $\text{safe}(S_F)$ be exactly $\mathcal{A}(S \setminus S_F)$, i.e., the distribution of models trained on all but the forget set and setting Δ_δ appropriately.

We make two observations about (4). First, the choice of $\text{safe}(S_F)$ is somewhat arbitrary, and in particular *any* distribution that does not depend on S_F, and produces a useful model would suffice. This includes, for example, distributions $\mathcal{A}'(S \setminus S_F)$ for algorithms $\mathcal{A}' \neq \mathcal{A}$, or distributions of *ensembles* of models $\mathcal{A}(S \setminus S_F)$. Second, while the Δ_δ used in Definition 2 has an appealing privacy interpretation, it is sensible (especially given our focus on empirical evaluation) to consider other divergences that are easier to estimate. These two observations inspire a metric that we call `KLoM` for empirical unlearning evaluation. `KLoM` corresponds to Definition 2 where we (a) use $\Delta = \text{KL}$ divergence, (b) allow for an arbitrary "reference distribution" $\text{safe}(S_F)$, and (c) as in `U-LiRA`, study distributions of model *outputs* f_x rather than parameters.

Definition 3 (KL divergence of margins (`KLoM`))**.** *For an unlearning algorithm $\mathcal{U}$, reference distribution $\text{safe}(S_F)$, and input x, the KL divergence of margins (`KLoM`) is given by*

$$\texttt{KLoM}(\mathcal{U}) := D_{KL}(\textit{safe}(S_F),\ f_x(\mathcal{U}(\mathcal{A}(S), S_F)))\,.$$

We note that rather than the KL divergence, would could choose other divergences like the α-Renyi Divergence, which is also tractable to compute and has the benefit that it can be directly converted into a bound on the δ-approximate max divergence in Definition 2 Mironov (2017). While we stick with the KL divergence throughout, in Appendix F.5 we evaluate unlearning using the α-Renyi divergence for $\alpha = 2, 5, 10$, and find similar results.
Despite the arbitrariness of $\text{safe}(S_F)$, unless otherwise noted we will mirror Definition 2 and take $\text{safe}(S_F) := \mathcal{A}(S \setminus S_F)$. Throughout the rest of this work, we primarily evaluate unlearning algorithms via computing `KLoM` for different inputs x from the forget set, retain set, and validation set. We also evaluate our algorithms with `U-LiRA`, and defer these results to the Appendix.

Compared to U-LiRA, KLoM is simpler to implement, has a natural correspondence with our original Definition 2, and importantly, does not suffer from *catastrophic unlearning*: observe that an unlearning algorithm $\mathcal{U}$ that transforms its input into a random classifier will pass an U-LiRA evaluation, as the random classifier will treat unlearned points and validation points identically. In contrast, by forcing us to explicitly specify safe(S_F), KLoM explicitly compares unlearned models to a baseline whose performance we know *a priori*. We explain this further in Appendix E.3, and provide empirical evidence in Figure E.2. Crucially, both KLoM and U-LiRA evaluate unlearning algorithms using *point-specific* distributional estimates, which as observed in Hayes et al. (2024) makes these evaluations far more stringent than prior approaches.

4 DMM: Unlearning by Simulated Oracle Matching

Having defined an evaluation apparatus, we now introduce our algorithm for machine unlearning. We first motivate the algorithm by observing a common challenge in existing methods. We then, in Section 4.2, propose an effective hypothetical algorithm for unlearning, under the unrealistic assumption that we have access to outputs of the "oracle" model. In Section 4.3, we show how to accurately simulate such oracle outputs using data attribution methods. Finally, in Section 4.4, we combine these insights and present our final algorithm, datamodel matching (DMM), and demonstrate its effectiveness and efficiency.

4.1 Motivation: the missing targets problem

Recall that the goal of unlearning is to approximate an *oracle* model, i.e., a model that was never trained on a given "forget set" of data. In strongly convex settings, this oracle model is unique, since it corresponds to the minimizer of a strongly convex loss function over the complement of the forget set (called the *retain set*). Thus, running gradient descent (GD) on the retain set loss yields a provable (and in some cases, efficient (Neel et al., 2021)) unlearning algorithm.

In the context of deep neural networks, however, GD alone is insufficient. In these settings, the loss function and training data alone do not fully specify the final model. In particular, once we have already minimized loss on the forget set, applying GD on the retain set does not significantly alter forget set predictions, preventing us from recovering the oracle model. Many unlearning methods for deep neural networks (Triantafillou et al., 2023; Kurmanji et al., 2023) thus actively *increase* loss on the forget set (e.g., via gradient ascent) while *maintaining* performance on the retain set.

This general approach comes with a significant set of drawbacks, which we collectively refer to as the *missing targets* problem. First, the assumption that forget set points will increase in loss after unlearning and retain set points will not is not necessarily correct. For example, if there are duplicated points across the forget and retain sets, then loss on points in the retain set might increase, while loss on points in the forget set might not change. Second, even for a forget set point whose loss *does* increase under the oracle model, our goal is not to increase loss arbitrarily, but instead only until it reaches its "target value"—the expected loss under a perfectly retrained model. Since we lack access to these values, it is challenging to know when a given forget set point has been "unlearned." Prior work tries to deal with this problem by devising heuristic regularization schemes, e.g., via early stopping, but nevertheless often overshoot or undershoot the target loss for a given data point. Figure 2 illustrates this phenomenon for a popular unlearning algorithm called SCRUB: over iterations of the algorithm, different points are unlearned (and then subsequently "overshot") at different points in time (Hayes et al., 2024). In essence, the underlying issue is that the nature of large non-convex training compels us to enforce additional structure on the unlearning objective, but prior algorithms do not use enough information to enforce the "right" structure.

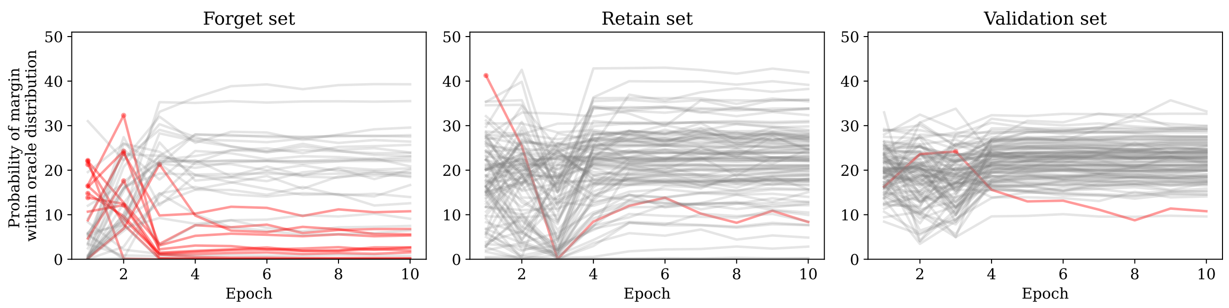


Figure 2: **The missing targets problem.** We apply the SCRUB (Kurmanji et al., 2023) algorithm to unlearn a forget set of CIFAR-10, and measure how well different (random) points are unlearned over time. To quantify how well a given point x is unlearned, we fit a Gaussian distribution to the outputs of oracle models on x, and compute the likelihood of the average outputs from unlearned models under this distribution. For many examples in the forget set (shown in red), unlearning quality is hurt by training for too long as we lack access to oracle targets.

4.2 The oracle matching algorithm

The above challenges ultimately arise from a lack of information about the behavior of the oracle model. But what if we did have access to predictions of the oracle model? In particular,

Given access to sample outputs from the oracle model (re-trained without the forget set), can we efficiently update the existing model (trained on the full dataset) to match the outputs out of sample?

While assuming access to oracle outputs is unreasonable—since our goal is to produce an oracle model in the first place—later in Section 4.4, we will replace oracle access with an efficient proxy using data attribution. For now, we simply assume we have direct access to oracle predictions, and focus on understanding whether gradient-based optimization can match predictions of the oracle. Even in this idealized setting, it is not clear how fast (if at all) gradient descent can converge to an oracle model. For one, whether we can do this efficiently with a small sample is unclear; it is possible that fine-tuning the trained model can match the oracle predictions on the sampled points, but will fail to generalize when evaluated on held-out points.

Formally, we assume access to predictions of an oracle model $f^{\text{oracle}}(x) := f_x(\mathcal{A}(S_R))$, where again S_R is the retain set and f_x is the evaluation of the model on input x (e.g., in classification settings one can take f to be the logits of the neural network). The *oracle matching* (OM) algorithm runs gradient descent on a mix of both forget and retain set points minimizing the MSE loss between the output logits from the model $f_x(\theta)$ and oracle predictions $f^{\text{oracle}}(x)$; see the pseudocode in algorithm A.1.

Evaluating oracle matching. We evaluate OM on various forget sets on two image classification tasks: ResNet-9 models trained on CIFAR-10 and ResNet-18 models trained on an ImageNet subset (Living17) (Santurkar et al., 2020). For baselines, we use gradient ascent (GA), gradient descent (GD), and SCRUB. We evaluate all methods using KLoM (Section 3): for each forget set and each unlearning method, we apply the method starting from 100 different model checkpoints, and compare the resulting distribution to 100 oracle checkpoints (see Appendix E for more details).

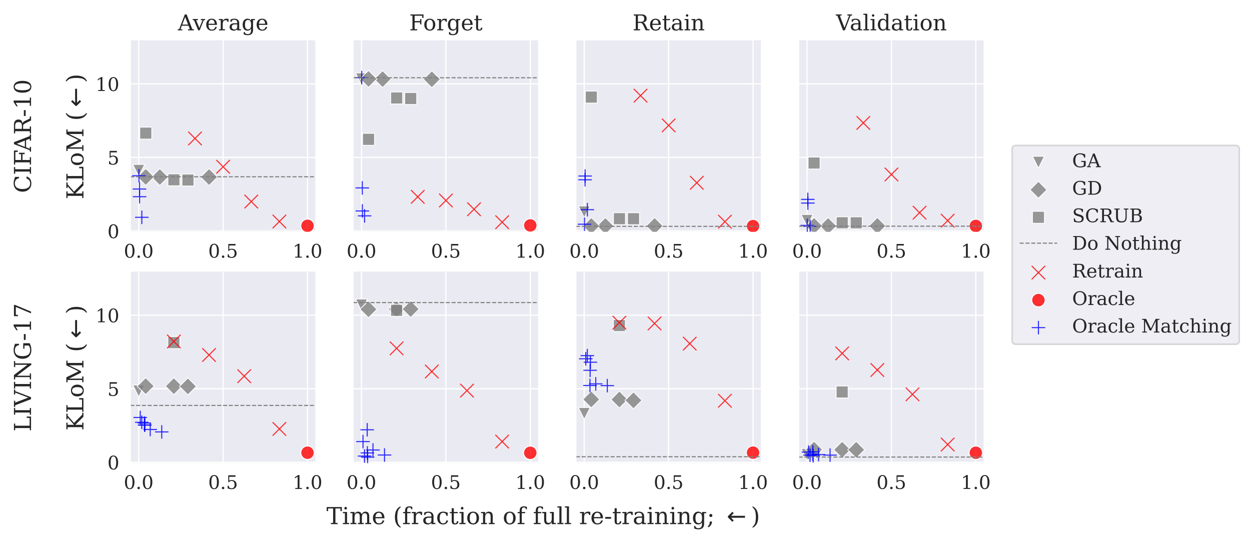


Figure 3: **Oracle matching can efficiently approximate re-training.** The `KLoM` metric (y-axis) measures the distributional difference between unlearned predictions and oracle predictions (0 being perfect). We also show the amount of compute relative to full re-training (x-axis). We evaluate `KLoM` values over points in the forget, retain, and validation sets and report the 95th percentile in each group; we also report the average across groups (1st column).

The results (Figure 3) demonstrate that OM is able to efficiently match the predictions of the oracle. Models unlearned with OM closely match the oracle distribution—as measured by `KLoM` scores—across all splits of the dataset (forget, retain, and validation sets), significantly outperforming all of the prior gradient-based approaches. Importantly, OM achieves effective unlearning while using *less than 5%* of the compute of full-retraining. This implies that despite the un-identifiability of a unique oracle in non-convex settings, for any given model θ there exists another model θ' close in parameter space that yields similar predictions as an oracle. And with sufficient guidance from the oracle, gradient descent is able to converge to θ'. We find that using a sufficiently high ratio of forget points in the fine-tuning set and a sufficiently large sample of retain points (but still much smaller than the full train set) is able to provide enough guidance (see Section 5 for ablations).

4.3 An efficient proxy for oracles: datamodels

We saw that OM is highly effective at approximating oracle outputs, but OM is not a practical algorithm as it assumes access to oracle outputs. To now turn this into a practical algorithm, we leverage methods for predictive data attribution (introduced in Section 2) to *simulate* oracle outputs.

Recall that a *datamodel* $\hat{f}_x$ predicts the counterfactual output of the model on input x when trained on an arbitrary subset $S \setminus S_F$: $\hat{f}_x(S \setminus S_F) \approx f_x(\mathcal{A}(S \setminus S_F))$. In the case of linear datamodels, we can parameterize the datamodel with a vector $\beta(x)$ so that $\hat{f}_x(S \setminus S_F) := \sum_{i \in S \setminus S_F} \beta_i(x)$. Leveraging linearity, we can re-write this as $\sum_{i \in S} \beta_i(x) - \sum_{i \in S_F} \beta_i(x)$, and we also replace the first term with the starting model output $f_x(\theta_0)$. Our general algorithm, DM-DIRECT (Appendix A.2), simulates the oracle outputs as $h(x) := f_x(\theta_0) - \sum_{i \in S_F} \beta_i(x)$.

Estimating datamodels. To estimate datamodels, we follow the approach in (Ilyas et al., 2022): we train models random subsamples of the full training set and use sparse linear regression to fit datamodel vectors $\beta(x)$ (later in Section 5.2 we explore alternative estimators).

Evaluating DM-DIRECT. In Figure 4, we compare model outputs on random forget and retain examples; the histograms show that the unlearned outputs from DM-DIRECT closely approximate the true oracle outputs. KLoM evaluations (Figure 1) show that DM-DIRECT (green line) in produces outputs close in distribution to that from oracle re-training for almost all points in the data distribution.

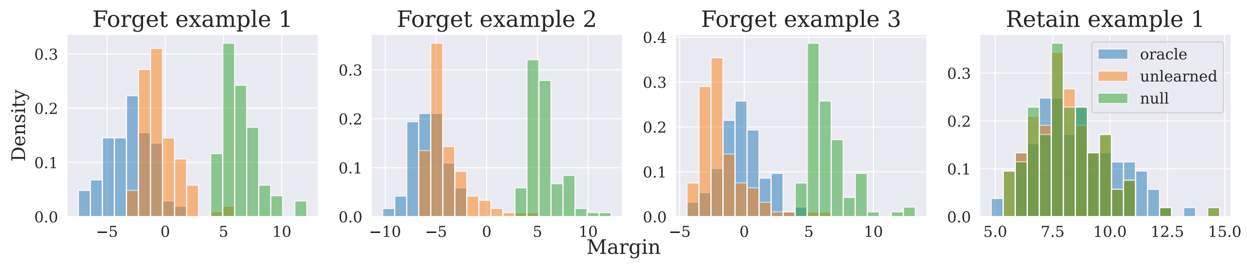


Figure 4: **Datamodels predict oracle outputs.** For random samples from the forget and retain sets, we compare the distribution (across multiple runs) of margins when evaluted on that example across three settings: i) null (model on full dataset); ii) oracle (model re-trained without forget set); and iii) unlearned (using DM-DIRECT). In every case, the predicted outputs (orange) closely match the ground-truth (oracle), demonstrating the effectiveness of datamodels as a proxy for oracle outputs.

4.4 Oracle matching with Datamodels

Now that we have an efficienty proxy for oracle outputs via datamodels, we revisit the OM algorithm from earlier. Our final algorithm, datamodel matching (DMM), first uses datamodels to generate approximations of oracle predictions on a subset of retain points and the forget points, and then runs OM on the datamodel predictions (see Appendix A.3 for pseudocode).

In Figure 1, we contextualize the performance of DMM against baselines and DM-DIRECT from earlier. DMM achieves levels of unlearning similar to that of fully retraining the model (as measured by KLoM scores), while using significantly less compute.[3] Using datamodels allow us to recover the performance of OM, and outperforms all prior gradient-based approaches. Importantly, DMM also matches the test accuracy of the oracle model and maintains accuracy on the points in the retain set (Appendix F.4), a common failure mode in prior methods.

In particular, DMM is significantly more effective than partially re-training given the same computational budget. We do not include the cost of computing datamodels as this is a *one-time* cost and hence amortized over many unlearning requests.[4] This is possible because once a predictive *datamodel* has been constructed (either via re-sampling as done here or influence function-like approximations, which we explore in Section 5.2), the datamodel generalizes well to new forget sets in practice.

To better understand the empirical success of DMM, in Section 5, we ablate different components of oracle matching and datamodel estimation. Importantly, we show that oracle matching addresses the motivating problem of missing targets, leading to stability in optimization. We also show that replacing regression-estimated datamodels with more efficient alternatives such as TRAK (Park et al., 2023) still yields effective unlearning via DMM.

[3]We only count the finetuning cost; the cost of computing datamodels is amortized across unlearning updates.
[4]The practice of not including pre-computation costs is standard in literature, e.g., Izzo et al. (2021a).

5 Understanding effectiveness of datamodel matching

We saw that datamodel matching is an effective and efficient algorithm for unlearning. Here, we aim to better understand the effectiveness of datamodel matching (DMM). Since DMM consists of i) oracle matching (the finetuning algorithm) and ii) datamodels (approximation to oracle outputs), we study each component separately. First, in Section 5.1, we analyze the stability of OM across time and to different choices of hyperparameters and show:

- **OM is stable across time**: We show that once OM unlearns an example, the example generally stays unlearned after further iterations, addressing the original "missing targets problem." As a result, OM is also much more stable than prior methods with respect to the choice of optimization hyperparameters.
- **OM generalizes from a small sample**: Though OM introduces additional design parameters (sampling ratios for forget and retain sets), we find that OM is effective as long as we include a sufficiently large sample of both. In particular, we only need to sample a small fraction of the retain set, making OM efficient.

Next, in Section 5.2, we ablate different components of the datamodel estimators to better understand necessary ingredients for DMM and show:

- **Necessity of modeling interactions between datapoints**: Datamodels (linearly) model the effect of different training examples on other inputs. We show that using only the "diagonal" entries (i.e., modeling only the self-influence) is much less effective.
- **Effectiveness of fast approximate data attribution methods**: We show that replacing regression-estimated datamodels with much faster alternatives like TRAK still yield effective unlearning algorithms, albeit with worse performance than well-estimated datamodels.

5.1 Unlearning stability of OM

To motivate our approach, we demonstrated earlier that existing fine-tuning approaches suffer from the problem of different unlearning rates due to lacking meaningful targets to converge to. In Figure 5, we find that OM no longer suffers from the same problem, since unlearning quality generally only improves over time (there is no risk of overshooting), even if points are still unlearned at different rates. Because of this, we find that OM is much more robust to the choice of optimization parameters such as learning rate and number of epochs compared to prior gradient ascent based methods (see Appendix F.1).

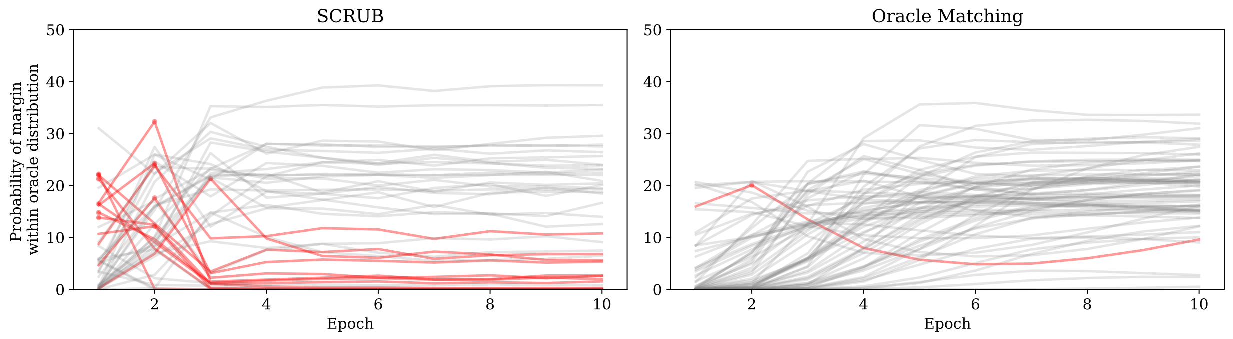


Figure 5: **Oracle matching circumvents the stopping time problem.** We revisit the earlier plot and apply the same analysis to oracle matching. The red lines highlight examples in the forget set whose unlearning quality is hurt by training longer. This happens frequently when running SCRUB, but goes away nearly entirely when using oracle matching.

Generalization of OM. OM fine-tunes on samples from both the forget and retain sets. The efficiency of OM hinges on whether it can generalize from a small sample. Ablations (Section 5.1) show i) OM succeeds as long as the ratio of forget set points in the fine-tuning set is sufficiently high, and ii) a small fraction (≥ 0.04) of retain set suffices to guide OM to converge towards an oracle on most retain set samples. That is, OM is able to effectively *generalize* from a small sample of oracle outputs. This implies that we only need to approximate oracle predictions on a small fraction of the training data, increasing the efficiency of the OM algorithm.

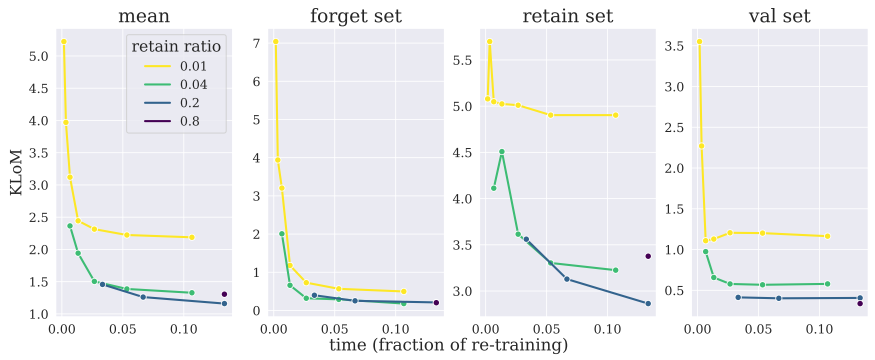


Figure 6: Varying the fraction of retain set sampled for oracle matching. A sufficiently large sample (≥ 0.04) appears to be sufficient in enabling OM to generalize to out-of-sample.

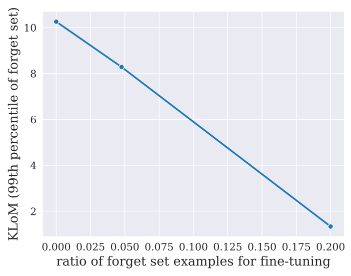


Figure 7: Varying the ratio of forget set samples in the fine-tuning set for oracle matching.

5.2 Datamodel ablations

Given the stability of OM, the success of DMM essentially only depends on the fidelity of datamodel approximation to oracle outputs. We now study which components of the datamodel are in fact necessary.

Necessity of modeling interactions between datapoints. Datamodels model the effect of different training examples on a given model output. By leveraging them, we were able to accurately simulate the oracle model outputs. Inspecting the weights of linear datamodels show that the dominant entry for the datamodel of a training input corresponding the example itself (i.e., excluding that training example has the largest negative effect on the model prediction on itself). The corresponding weight is what is also known as the *memorization score* in prior work Feldman (2021). Could memorization scores suffice to linearly model oracle outputs? In Figure 8, we evaluate the quality of oracle predictions when using the full datamodel vector vs. only the memorization scores, and find the following:

- **Insufficiency of memorization scores for unlearning non-random forget sets:** Using only the diagonal entries hurts unlearning quality for non-random forget sets (5), particularly for examples in the tail (dotted line). Intuitively, this is because the other weights in datamodels (the "non-diagonal" entries) capture important cross-example correlations (e.g., similar examples should have reinforcing effect on one another). Moreover, globally scaling the memorization scores (orange lines) cannot account for the missing non-diagonal entries. On the other hand, this is less important for random forget sets (3), as two random examples are in general unrelated to one another. Hence, to effectively unlearn forget sets that arise in practice via our approach, it seems necessary to model interactions between different datapoints.

- **Consistency of best scaling:** As an artifact, we also find that scaling down the datamodel weights globally by a factor (≈0.9 here) improves unlearning quality marginally. Fortunately, the scale seems consistent across different types of forget sets; one could calibrate this scale using a "held-out" forget as part of a pre-computation stage, and subsequently apply to all forget sets.

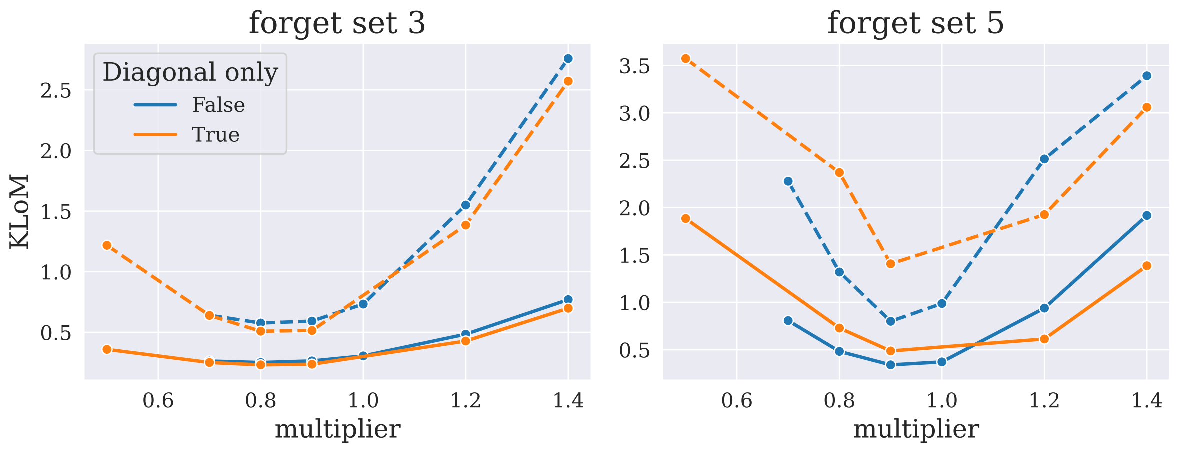


Figure 8: The effect of non-diagonal entries and re-scaling on the unlearning effectiveness of DM-DIRECT for two different types of forget set on CIFAR-10 (left is random; right is non-random). Solid and dotted lines correspond to the mean and 95%-percentile KLoM scores. Diagonal-only indicates that we only use the memorization scores (the "diagonal" of the datamodel matrix).

Efficient unlearning with TRAK. While datamodels estimated using the regression approach of Ilyas et al. (2022) are predictive, and can be pre-computed prior to unlearning, they nonetheless are expensive to compute. Since the work of Ilyas et al. (2022), follow-up works (Park et al., 2023; Grosse et al., 2023) have shown that efficient alternative methods can be quite effective with substantially lower computational costs.

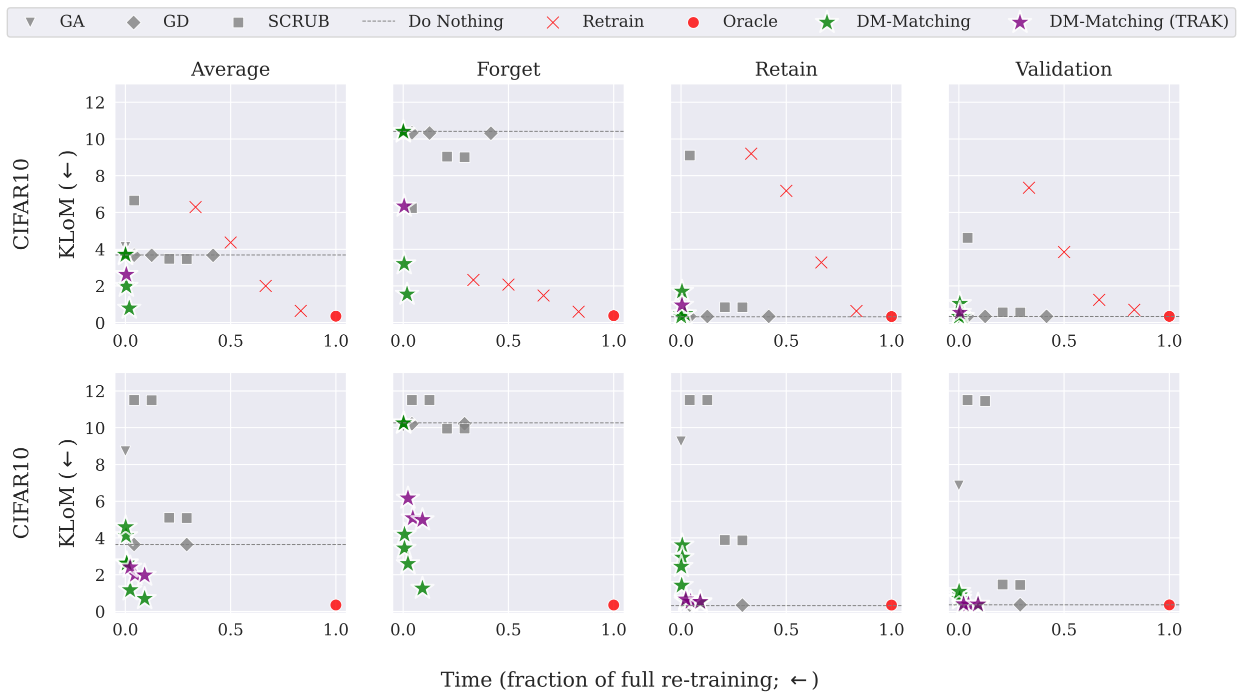


Figure 9: **Datamodel-Matching with more efficient estimators.** In the same set up as in Figure 1, we evaluate the effectivness of DM-DIRECT and DMM when a different, more efficient estimator (TRAK) for datamodels. Though the equality of unlearning degrades, our algorithms still outperform prior methods. Note that the computational savings from using TRAK is not reflected on the x-axis, as computation of datamodels considered separately from the fine-tuning cost of DMM.

Here we investigate whether DMM is still effective when datamodels are estimated with TRAK, which is based on a particular approximation to the influence function. In Figure 9 we run OM with predictions generated by TRAK estimators with x1000 less compute than the regression-based datamodels. As expected DMM with TRAK performs worse in terms of KLoM than when datamodels are used, but we find that this drastically cheaper alternative to DMM still outperforms all prior methods significantly. These results highlight that improving the accuracy of TRAK in order to close the gap in KLoM scores with datamodels represents an important direction for developing even more efficient and effective unlearning algorithms.

6 Oracle Matching for Linear Models

We have seen that empirically OM outperforms standard gradient-based unlearning methods, and we have highlighted the missing targets problem as one possible explanation. Are there other factors that contribute to the success of OM relative to prior methods gradient-based methods, and can we better understand what settings we expect OM to perform well? This motivates studying a setting where the missing targets problem is neutralized: when the objective is strongly convex. In this setting, the unlearned model is the unique empirical risk minimizer on X_R, and GD initialized at the current model is a provably effective unlearning algorithm (Neel et al., 2020). Even in the setting when GD on its own can converge, does providing "guidance" from an oracle help?

We answer this affirmatively: First, in Subsection **??**, we empirically identify two factors that influence whether OM outperforms GD: the strength of regularization and stochasticity in optimization. In Subsection 6.2 we theoretically characterize the exact convergence rates of full batch OM and GD to the unlearned model in terms of the degree of regularization and the relative eigenmass on the forget and retain sets. Unlike in the full-batch setting where both algorithms converge at a linear rate, in the stochastic setting we show that OM converges exponentially faster than SGD, which sheds light on the superior performance of stochastic OM in our empirical results.

Setting. We consider the following ridge regression algorithm, given by

$$\mathcal{A}(S) := \arg\min_{\theta} \sum_{(\boldsymbol{x}_i, y_i) \in S} \left(\theta^\top \boldsymbol{x}_i - y_i\right)^2 + \lambda \|\theta\|_2^2, \tag{5}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ are the training inputs, $y_i \in \mathbb{R}$ are the corresponding labels, and the setting is overparameterized, so $d > |S|$. Given a model $\theta_{\text{full}} = \mathcal{A}(S)$ trained on a full dataset S, our goal is to unlearn the forget set $S_F \subset S$ by obtaining a model that minimizes the objective on the retain set $S_R = S \setminus S_F$. For convenience, we use X, X_R, and X_F to denote the covariate matrices for the full dataset S, the retain set S_R, and the forget set S_F respectively. We choose the under-determined ridge regression for three reasons: (a) The objective (5) is strongly convex, and so GD on X_R is guaranteed to compute the (unique) unlearned model if ran for sufficiently many iterations; (b) the least-squares objective is amenable to theoretical analysis; and (c) the over-parameterized setting is most relevant to modern deep learning models where $d \gg n$.

Unlearning algorithms. Let $\theta_* = \mathcal{A}(S_R)$ be the minimizer of the ridge regression objective on the retain set (i.e., the unlearning target). Starting from $\theta_{\text{full}} = \mathcal{A}(S)$, we evaluate several iterative first-order unlearning algorithms in terms of their ability to recover θ_*:

(i) **Gradient Descent (GD)** minimizes the ridge regression objective on the retain set S_R using gradient descent with constant step size, starting from θ_{full};

(ii) **Gradient Descent + Ascent (GDA)** incorporates forget set points in the gradient descent updates, combining gradient descent on the retain set with gradient ascent on the forget set;

(iii) **Oracle Matching (OM)** assumes query access to an unlearned model θ^*, and uses gradient descent (with constant step size) to minimize squared error with respect to "oracle" predictions $\boldsymbol{x}_i^\top \theta^*$ on the full dataset, aiming to minimize $||X\theta - X\theta_*||^2$;

(iv) **Oracle Matching on Retain Set (OMRS)** performs gradient descent on the squared error from oracle predictions but only on the retain set, thereby minimizing the objective $\|X_R\theta^* - X_R\theta\|^2$.

We analyze both the full-batch and stochastic versions of these methods. Appendix D.3 contains more details on the exact setup of the algorithms we evaluate.

6.1 Experimental results

For our experimental analysis, we construct a synthetic setting with $n = 100$ datapoints of dimension $d = 400$. We first fix a "ground-truth" weight vector $\theta \in \mathbb{R}^{400}$ by sampling each coordinate $\theta_i \sim \mathcal{N}(0, 1)$. We then generate a dataset S consisting of 100 training points $(\boldsymbol{x}_i, y_i)$, where each $\boldsymbol{x}_i \sim N(0, \mathbf{I}_{400})$ is sampled from a 400-dimensional standard Gaussian, and each $y_i \sim \mathcal{N}(\theta^\top x_i, 4)$. We choose a forget set S_F consisting of 5 random points from S.

We obtain the starting point for unlearning θ_{full} by minimizing the ridge regression objective (5) with $\lambda = 4$. We then apply the iterative unlearning algorithms above, starting from $\theta_0 := \theta_{\text{full}}$;

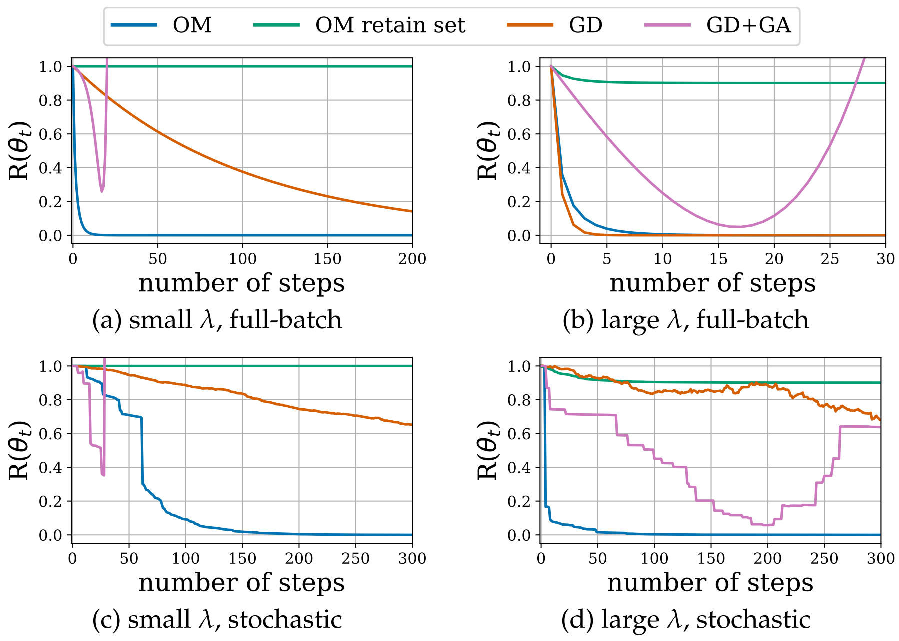


Figure 10: Comparing unlearning methods for a linear model with $d = 400, n = 100, |S_F| = 5$. The y-axis shows the relative squared distance to the optimal unlearned model $R(\theta_t) = \frac{\|\theta_* - \theta_t\|}{\|\theta_* - \theta_{\text{full}}\|}$, where θ_t is the iterate at time t, θ_{full} is the model trained on all data, and θ_* is the optimal unlearned model.

after each iteration t, we measure the relative distance from the current unlearning iterate θ_t to the unlearning target θ_*, i.e.,

$$R(\theta_t) := \frac{\|\theta_* - \theta_t\|}{\|\theta_* - \theta_{\text{full}}\|}$$

Standard setting ($\lambda = 4$, full-batch updates). Figure 10a depicts the performance of the four unlearning algorithms we consider. We observe that OM converges to the unlearned solution much faster GD, while GDA and OMRS both fail to converge even as $t \to \infty$. OM differs from GD primarily in two aspects: (i) it minimizes the squared error with respect to the target model predictions (rather than the training data labels), and (ii) its updates involve forget set points along with retain set points (rather than just retain set points). OMRS helps us isolate the impact of these differences—applying oracle matching to only the retained points makes negligible progress. This suggests that the inclusion of forget set points in the update is the primary driver of superior performance of OM.

Investigating further, we observe that during unlearning, the model parameters change the most in directions orthogonal to the retain set. Indeed, we find that despite the fact that 24% of the mass of the forget set points lies in the span of the retain set, this span actually captures less than 0.01% of the mass of the ground-truth update $\theta_* - \theta_{\text{full}}$. This explains the importance of including forget set points in the unlearning objective: OMRS cannot make any progress in directions orthogonal to the retain set, and GD only makes progress in those directions due to the the ℓ_2 regularization term. On the other hand, OM makes rapid progress in these directions due to the inclusion of forget set points.

As for GDA, this algorithm also uses the forget set points, but since it does not have access to oracle predictions on these points, it must instead introduce them into the objective in a heuristic way. As a result, the algorithm initially makes rapid progress towards the target solution θ_* (in this initial stage, the update induced by the oracle labels matches that induced by gradient ascent) but then suffers from the "stopping time" problem discussed in the previous parts and diverges. In the convex setting, we can mitigate this issue somewhat by using the gradient norm on the retain set as a stopping criterion, but there is no obvious way to transport this criterion to the more relevant non-convex case.

Large-regularization case ($\lambda = 400$) In Figure 10b, we consider the same setting as above but set λ to a much larger value of 400. Here, GD converges slightly faster than OM due to the stronger ℓ_2 regularization, which aids GD in converging along directions orthogonal to the retain set. Thus, OM converges faster than GD when λ is moderate but can be slower with large λ. Again, OMRS and GDA do not successfully converge, but GDA—guided by the heuristic use of the forget set points—initially makes significant progress towards θ_* before eventually diverging.

Stochastic case. In Figures 10c and 10d, we replicate the experiments above with stochastic variants of the unlearning algorithms. As in the non-stochastic case, we see the OM on the retain set fails to make any progress, and that GDA makes quick progress but then diverges. However, unlike in the full-batch setting, SOM outperforms SGD in both the large and small λ settings (see Appendix D.3 for a discussion). This surprising finding is characterized in Theorem 2 below, where we show that SOM converges exponentially faster than SGD.

6.2 Convergence Theory

We now turn to studying the algorithms above theoretically, aiming to formalize some of our empirical observations. We start with the full-batch case, corresponding to Figures 10a and 10b above; we then proceed to the stochastic/minibatched case (Figures 10c and 10d). In all cases, we will focus on the two convergent algorithms above: oracle matching (OM) and ridge gradient descent (GD).

Full-batch case. In Theorem 1, we provide a theoretical analysis of the convergence rates of GD and OM (see Appendix D.1 for the proof). The key takeaway from this result is that the convergence rate for both algorithms depends on both (a) the relative eigenmass of the forget and retain sets; and (b) the strength of the ridge regularization.

Theorem 1 (Proof in Appendix D.1)**.** *Let S and S_R be the full training set and the retain set respectively, with input matrices X and X_R and corresponding labels y and y_R. Additionally, let θ_{full} and θ_* denote the optima of the ridge objective (5) for the full data S and retain set S_R respectively. After t iterations of unlearning starting from θ_{full}, the iterate θ_t satisfies*

$$\theta_t - \theta_* = \begin{cases} (I - 2\eta\lambda)^t \left(I - \frac{2\eta}{1-2\eta\lambda} X_R^\top X_R\right)^t (\theta_{full} - \theta_*) & \text{for ridge gradient descent (GD).} \\ \left(I - 2\eta X^\top X\right)^t (\theta_{full} - \theta_*) & \text{for oracle matching (OM).} \end{cases}$$

Theorem 1 shows that both (full-batch) OM and GD exhibit linear convergence, albeit at different rates. Indeed, in directions orthogonal to the retain set, the middle term in the GD convergence rate disappears, and so the rate depends only on $\eta\lambda$. Thus, as long as the learning rate η is set high enough (i.e., not to cancel out the λ) higher regularization will cause GD to converge faster.

Stochastic case. In our experimental analysis, we saw that unlike the full-batch case, the stochastic version of oracle matching was *consistently* more effective than that of gradient descent. In Theorem 2 we show that, at least in the setting of under-determined ridge regression we consider here, this observation is strongly supported by theory. In particular, we show that while OM converges at a linear rate (i.e., exponentially fast in t), we can show a $\Omega(\frac{1}{t^2})$ lower bound on the convergence of SGD, giving a strong separation between the two methods.

Theorem 2 (Proof in Appendix D.2)**.** *Consider the setting of Theorem 1, where θ_t is the iterate after t steps of unlearning initialized at θ_{full}. Further let $\gamma_{min}, \gamma_{max}$ denote the minimum and maximum non-zero eigenvalues of $X^\top X$. Then, as long as the learning rate $\eta \in (0, \frac{2}{5(\gamma_{max}+\lambda)})$ and $rank(X) > 1$,*

$$\frac{\mathbb{E}\left[\|\theta_t - \theta^*\|^2\right]}{\|\theta_{full} - \theta_*\|^2} \in \begin{cases} O\left((1-\gamma_{min}\eta)^t\right) & \text{for oracle matching (OM).} \\ \Omega\left(\frac{1}{t^2}\right) & \text{for ridge gradient descent (GD).} \end{cases} \tag{6}$$

The intuition is as follows. In each update step, stochastic OM updates the model parameters only in the span of the random subset of points used for that update. In contrast, stochastic GD decays the model parameters in other directions due to the regularization term. (Recall that this is what led to GD's improved convergence in the high-regularization setting.) While this shrinkage is beneficial in the subspace orthogonal to the retain set, stochastic GD also decays the parameters along the directions spanned by retain set points that are not in the current batch.

Mathematically, this is formalized by defining what is called the *gradient disagreement* $\sigma_f^2 := \mathbb{E}_i\left[\|\nabla f_i(\theta_*) - \mathbb{E}[\nabla f_i(\theta_*)]\|^2\right] = \mathbb{E}_i\left[\|\nabla f_i(\theta_*)\|^2\right]$, which measures for a smooth function $f(x) = \sum_i f_i(x)$ that is the sum of smooth functions, a measure of the expected extent to which θ^* differs from the minimizer of each f_i.

Note that for the OM objective, θ^* is the minimizer of each f_i and so $\sigma_f^2 = 0$, whereas for GD when $\text{rk}(X) > 1, \sigma_f^2 > 0$. Standard results Garrigos & Gower (2023) show that if θ_t is the iterate after t steps of SGD, then

$$\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] \geq \left(1 - 2\eta\beta - \eta^2\beta^2\right)\|\theta_{t-1} - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2}$$

The fact that this second term is non-zero for GD is what forces the $\Omega(1/t^2)$ lower bound. A full proof is deferred to Appendix D.2.

7 Conclusion and Discussion

In this work, we showcased a general framework for reducing unlearningto the related problem of predictive data attribution. Our finegrained evaluations using `KLoM`—which directly measures the quality of unlearning in line with the standard differential privacy-inspired definition—demonstrate that our matching algorithms significantly outperform prior gradient-based unlearning methods and approaches the performance of oracle re-training. To conclude, we discuss some limitations and promising directions for future work:

Extending techniques and evaluations beyond classification. While our methods perform well in classification settings with few classes, extending them to work on settings with more classes (e.g., full ImageNet or language-modeling) would make them much more practically useful. Applying our techniques directly (directly attributing all class logits) would incur a heavy computational cost, so we expect that additional techniques will be necessary.

Improving and understanding Oracle Matching. As we saw, OM can sometimes cause a mismatch on the retain set due to "reversing the overfitting." Better understanding the dynamics of the OM algorithm and leveraging other insights (e.g., from the model distillation literature) would be valuable for making the matching part of our framework more stable. One potential direction is to understand when and how better sampling strategies (instead of random subsampling) for OM.

Reducing computational costs. Even the most efficient data attribution methods require a non-trivial computational cost (roughly on the same order as training the original model). Can we design other proxies for data attribution that we can still leverage for unlearning without the full computational cost?

Exploring alternative target model classes for unlearning. Our analysis of DM-DIRECT showed that alternate model classes may be useful for our goal of unlearning if we do not restrict ourselves to stay in the exact same model class. More broadly, finding other ways to leverage this freedom in designing unlearning algorithms may lead to more practically deployable and well performing unlearning methods.

References

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentation. *arXiv preprint arXiv:2405.12186*, 2024.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020a. URL https://arxiv.org/abs/1912.03817.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020b. URL https://arxiv.org/abs/1912.03817.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 141–159. IEEE, 2021a. doi: 10.1109/SP40001.2021.00019. URL https://doi.org/10.1109/SP40001.2021.00019.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021b.

Tamara Broderick, Ryan Giordano, and Rachael Meager. An automatic finite-sample robustness metric: when can dropping a little data make a big difference? *arXiv preprint arXiv:2011.14999*, 2020.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

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

Eli Chien, Haoyu Wang, Ziang Chen, and Pan Li. Stochastic gradient langevin unlearning, 2024.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context. *arXiv preprint arXiv:2409.00729*, 2024.

Guangyao Dou, Zheyuan Liu, Qing Lyu, Kaize Ding, and Eric Wong. Avoiding copyright infringement via machine unlearning, 2024. URL https://arxiv.org/abs/2406.10952.

Ronen Eldan and Mark Russinovich. Who's harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels, 2024.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail, 2021. URL https://arxiv.org/abs/1906.05271.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

Shashwat Goel, Ameya Prabhu, Amartya Sanyal, Ser-Nam Lim, Philip Torr, and Ponnurangam Kumaraguru. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*, 2022.

Shashwat Goel, Ameya Prabhu, Philip Torr, Ponnurangam Kumaraguru, and Amartya Sanyal. Corrective machine unlearning, 2024. URL https://arxiv.org/abs/2402.14015.

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*, pp. 9304–9312, 2020.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learing (ICML)*, 2019.

Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *ArXiv*, abs/2106.04378, 2021. URL https://api.semanticscholar.org/CorpusID:235367846.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Jamie Hayes, Ilia Shumailov, Eleni Triantafillou, Amr Khalifa, and Nicolas Papernot. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy, 2024.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *CoRR*, abs/2202.00622, 2022. URL https://arxiv.org/abs/2202.00622.

Andrew Ilyas, Kristian Georgiev, Logan Engstrom, and Sung Min (Sam) Park. Data attribution at scale: Icml 2024 tutorial, 2024. URL https://ml-data-tutorial.org/. Accessed: 2024-09-24.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021a.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models, 2021b.

Louis A Jaeckel. *The infinitesimal jackknife*. Bell Telephone Laboratories, 1972.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22691–22702, 2023.

Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *arXiv preprint arXiv:2302.09880*, 2023.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The wmdp benchmark: Measuring and reducing malicious use with unlearning, 2024. URL https://arxiv.org/abs/2403.03218.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners, 2022. URL https://arxiv.org/abs/2110.05679.

Ken Ziyu Liu. Machine unlearning in 2024, Apr 2024. URL https://ai.stanford.edu/~kzliu/blog/unlearning.

Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models, 2021. URL https://arxiv.org/abs/2106.15093.

Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pp. 263–275. IEEE Computer Society, 2017. doi: 10.1109/CSF.2017.11. URL https://doi.org/10.1109/CSF.2017.11.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning, 2020.

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)*, 2021.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale, 2023.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: Language models as few shot unlearners, 2023.

Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. *arXiv preprint arXiv:2406.17216*, 2024a.

Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks, 2024b.

Daryl Pregibon. Logistic regression diagnostics. *The annals of statistics*, 9(4):705–724, 1981.

Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pp. 18400–18421. PMLR, 2022.

Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. In *ArXiv preprint arXiv:2008.04859*, 2020.

Vinith Suriyakumar and Ashia C Wilson. Algorithms that approximate data removal: New results and limitations. *Advances in Neural Information Processing Systems*, 35:18892–18903, 2022.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 4007–4022, 2022.

Eleni Triantafillou, Fabian Pedregosa, Isabelle Guyon, Sergio Escalera, Julio C. S. Jacques Junior, Gintare Karolina Dziugaite, Peter Triantafillou, Vincent Dumoulin, Ioannis Mitliagkas, Lisheng Sun Hosoya, Meghdad Kurmanji, Kairan Zhao, Jun Wan, and Peter Kairouz. Neurips 2023 machine unlearning challenge. `https://unlearning-challenge.github.io`, 2023. Accessed: 2024-05-29.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pp. 10355–10366. PMLR, 2020.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning, 2024. URL `https://arxiv.org/abs/2310.10683`.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*, 2023.

Appendices

A Pseudocode

A.1 Oracle Matching

Algorithm A.1 Oracle Matching (OM)

```
1:  Input: Trained model $\theta_0$; oracle predictions $f^{\text{oracle}}(x)$; fine-tuning set size $r$
2:  Output: Unlearned model $\theta$
3:  for $t = \{1, ..., T\}$ do                                          ▷ $T$ epochs
4:      $S'_R \leftarrow S \setminus S_F$                     ▷ Sub-sample $r$ points from retain set
5:      $S_{\text{fine-tune}} = S_F \bigcup S'_R$
6:      for $x \sim S_{\text{fine-tune}}$ do                              ▷ mini-batch
7:          $L(\theta_t) = \|f_x(\theta_t) - f^{\text{oracle}}(x)\|^2$            ▷ Compute loss
8:          $\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta L(\theta_t)$     ▷ Perform update with gradient
9:      end for
10: end for
11: Return Model $\theta = \theta_T$
```

A.2 Datamodel Direct

Algorithm A.2 DM-DIRECT

```
1:  Input: Trained model $\theta_0$; datamodels $\beta(x)$ for each $x \in S$; forget set $S_F$
2:  Output: A predictor $h(\cdot) : S \mapsto \mathbb{R}^k$
3:  $h(x) := f_x(\theta_0) - \sum_{i \in S_F} \beta_i(x)$
4:  End
```

A.3 Datamodel Matching

Algorithm A.3 Datamodel Matching (DMM)

```
1:  Input: Trained model $\theta_0$; datamodels $\beta(\cdot)$; fine-tuning set size $r$
2:  Output: Unlearned model $\theta$
3:  $S'_R \leftarrow S \setminus S_F$                         ▷ Sub-sample $r$ points from retain set
4:  $S_{\text{fine-tune}} = S_F \bigcup S'_R$
5:  $h \leftarrow$ DM-DIRECT$(\theta_0, \beta, S_f)$              ▷ Simulate oracles with datamodels
6:  for $t = \{1, ..., T\}$ do                                          ▷ $T$ epochs
7:      for $x \sim S_{\text{fine-tune}}$ do                              ▷ mini-batch
8:          $L(\theta_t) = \|f_x(\theta_t) - h(x)\|^2$                       ▷ Compute loss
9:          $\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta L(\theta_t)$     ▷ Perform update with gradient
10:     end for
11: end for
12: Return Model $\theta = \theta_T$
```

B Related work

Data attribution. In this work, we make extensive use of the predictive data attribution framework (Ilyas et al., 2022; Park et al., 2023). More broadly, the field of data attribution has rich history dating back to (Hampel, 1974; Jaeckel, 1972; Pregibon, 1981). It has broad applications—notably, there are deep connections with economics (Ghorbani & Zou, 2019; Broderick et al., 2020; Cohen-Wang et al., 2024). On the computational side, there has been an explosion of works in recent years applying data attribution methods to deep learning systems in a computationally efficient way (Koh & Liang, 2017; Park et al., 2023; Engstrom et al., 2024; Grosse et al., 2023; Choe et al., 2024; Bae et al., 2024). For surveys on the topic, refer to Hammoudeh & Lowd (2024); Ilyas et al. (2024).

Machine unlearning methods. The field of machine unlearning (Cao & Yang, 2015) has enjoyed extensive interest. The term "unlearning" itself is overloaded—it is used both for removing the influence of particular training samples (Ginart et al., 2019; Wu et al., 2020; Neel et al., 2021; Bourtoule et al., 2021b), and unlearning "concepts" or "knowledge" (Ravfogel et al., 2022; Eldan & Russinovich, 2023; Kumari et al., 2023; Zhong et al., 2023). Further, within the data-deletion setting there are two main categories of unlearning: perfect unlearning and approximate unlearning. Where in perfect unlearning, the output of the unlearning algorithm exactly simulates what a model that never trained on the forget set would be (a fully retrained model). An example of exact unlearning is SISA Bourtoule et al. (2020b); in SISA, data is partitioned such that one data point is only in one partition and a series of models are trained on each of the partitions, then the models are aggregated as an ensemble used for inference or for labeling a new dataset — this means a data-deletion request to only retrain the relevant models that were trained on the forget set (we defer a full description to the original paper). In approximate unlearning, the output of the unlearning algorithm only approximates the fully-retrained model, generally this is specified as a statistical distance. In our work we focus on data-deletion in the approximate unlearning setting, where a specific data point or data points are to be deleted from a dataset.

Machine unlearning evaluations. We observe that the goal of unlearning very much dictates the right measures of success for unlearning, which themselves dictate what the best machine unlearning method is best. For example, machine unlearning with the purpose of removing poisoned data has a very different measure of success of unlearning for legal compliance reasons. Whereas for removing poisoned data the measure of success is if the model remains poisoned, the measure of success for privacy reasons is closer to a statistical measure about models.

In the context of deep learning, exact unlearning, i.e., algorithms that come with provable guarantees, are difficult to achieve, and often comes at the cost of accuracy (Bourtoule et al., 2020a). Thus, most unlearning algorithms for deep learning are heuristic. This approximate nature necessitates solid empirical evaluations in lieu of a provable guarantee. One line of work adapts membership inference attacks (MIAs) to evaluate machine unlearning (Golatkar et al., 2020; Goel et al., 2022; Hayes et al., 2024). Complementary to that, Pawelczyk et al. (2024a) evaluate machine unlearning methods' ability to remove backdoor attacks from the training set. Additionally, Thudi et al. (2022) argue that due to the stochastic nature of deep learning optimization, approximate evaluation of machine unlearning is only well-defined on an algorithmic level, and not on an individual model instance level.

Fine-tuning based unlearning methods. Due to challenges of developing rigorous unlearning methods in non-convex settings, typical approaches involve some form of gradient-based optimization.

In unlearning via *gradient descent*, the method works by retraining on the retain set, while ignoring the forget set. This approach is effective in convex optimization problems where the loss function has a single global minimum Neel et al. (2021). However, in non-convex settings, this approach performs poorly, empirically.

The *gradient ascent* method for unlearning takes negative gradient descent steps on the "forget set" to reverse the impact of the data to be unlearned. In this approach, instead of retraining on the retain set, the model takes steps that increase the loss specifically on the forget set by taking a step that negates a gradient descent step.

The primary method we compare against is SCRUB (and SCRUB+R), introduced in Kurmanji et al. (2023). They achieve the current state-of-the-art on strong unlearning evaluations Hayes et al. (2024). At a high level, SCRUB finetunes the original model to: i) maximize the KL divergence between the probabilities of the original model and the new model on forget points; ii) minimize the KL divergence on retain points; and iii) also minimize test loss. SCRUB+R checkpoints updates of SCRUB and returns the “best” checkpoint according to by tracking loss values on a validation set that is constructed to be similar to the forget set. We do not implement SCRUB+R directly, but instead evaluate SCRUB trained for different numbers of epochs and return the best one according to our metrics of success; which is an upper bound on how well SCRUB+R would perform.

C Experimental setup

C.1 Training setup

For CIFAR-10, we train ResNet-9 models[5] for 24 epochs with SGD with a batch size of 512, momentum 0.9, and weight decay $5e-4$. We set learning rate initially at 0.4, and a single-peak cosine schedule peaking at the 5th epoch. We use a momentum of 0.9 and a weight decay of $5e-4$.

For ImageNet Living17 (Santurkar et al., 2020), we train ResNet-18 models for 25 epochs using SGD with a batch size of 1024, momentum 0.9, and weight decay $5e-4$. Label smoothing is set to 0.1.

C.2 Constructing forget sets

We evaluate methods across various types and sizes of forget sets to test the robustness of unlearning. Our selection of unlearning scenarios span both random and non-random forgets of different sizes; that said, we view the non-random sets as practically more interesting. Compared to prior work, our target sets are harder to unlearn as we remove a small coherent subpopulation as opposed to an entire class.

On CIFAR-10, we use 9 different forget sets: sets 1,2,3 are random forget sets of sizes 10,100,1000 respectively; sets 4-9 correspond to semantically coherent subpopulations of examples (e.g., all dogs facing a similar direction) identified using clustering methods.

Specifically, we take a $n \times n$ influence matrix constructed by compute TRAK influence scores (Park et al., 2023) on a dataset of size n (for CIFAR-10 this is $50,000$). Next, we compute the top principal components (PCs) of the influence matrix and construct the following forget sets:

1. Forget set 1: 10 random samples.
2. Forget set 2: 100 random samples.
3. Forget set 3: 500 random samples.
4. Forget set 4: 10 samples with the highest projection onto the top PC of the TRAK matrix.
5. Forget set 5: 100 samples with the highest projection onto the top PC of the TRAK matrix,
6. Forget set 6: 250 samples with the highest projection onto the top PC of the TRAK matrix and 250 with lowest projection onto the top PC.
7. Forget set 7: 10 samples with the highest projection onto the 2nd PC of the TRAK matrix.
8. Forget set 8: 100 samples with the highest projection onto the second PC of the TRAK matrix.
9. Forget set 9: 250 samples with the highest projection onto the 2nd PC of the TRAK matrix and 250 with the lowest projection onto the 2nd PC.

On ImageNet Living-17, we use three different forget sets: set 1 is random of size 500; sets 2 and 3 correspond to 200 examples from a certain subpopulation (corresponding to a single original ImageNet class) within the Living-17 superclass.

We include a deterministic process for computing these in our code.

[5] https://github.com/wbaek/torchskeleton/blob/master/bin/dawnbench/cifar10.py

C.3 Hyperparameters of unlearning algorithms

Most unlearning algorithms are very sensitive to the choice of forget set; thus, so for each of our unlearning algorithms, for each forget set, we evaluate over a grid of hyperparameters and report the best `KLoM` scores, as decided by the Forget Set `KLoM` score. Below we report the hyperparameter grid that we search over for each of the unlearning algorithms.

1. **Gradient Ascent**: Our Gradient Ascent methods are trained with SGD. Learning rates: $(1e-5, 1e-3, 1e-2)$. Number of epochs: $(1, 3, 5, 7, 10)$.
2. **Gradient Descent**:Our Gradient Descent methods are trained with SGD. Learning rates: $(1e-5, 1e-3, 1e-2)$. Number of epochs: $(1, 3, 5, 7, 10)$.
3. **SCRUB**: Our SCRUB methods are trained with SGD. Forget Batch Size: $(32, 64)$. Maximization Epochs (number of epochs where gradient-ascent-based methods occurs): $(3, 5)$. Learning rates: $(5e-3, 1e-3, 5e-4, 5e-5)$. And number of total epochs $(5, 7, 10)$. We set the retain batch size to 64.
4. **Oracle Matching / Datamodel Matching**: Our Oracle Matching (derived) methods are trained with Adam. Batch Sizes: 512. Retain Multipliers: $(3., 5.)$. Learning Rates: $(1e-3, 1e-4, 1e-5, 1e-6)$. Number of Total Epochs: $(1, 2, 4, 8)$. Retain ratio (fraction of full retain set sampled): 0.01-0.8.

C.4 Datamodel estimation

Regression-based. We re-train models on random 50% subsets of the full train dataset, and use between 1,000 and 20,000 models. We use the sparse linear regression based solvers from Ilyas et al. (2022) to estimate each datamodel vector. Though our main results are computed with 20,000 models, we find that using just 1,000 models suffice effective unlearning with DMM.

TRAK. We compute TRAK scores using 300 model checkpoints and 16328 projection dimensions using the code provided in Park et al. (2023).

D Linear model analysis

D.1 Proof of Theorem 1

We restate Theorem 1 below.

Theorem 1 (Proof in Appendix D.1)**.** *Let S and S_R be the full training set and the retain set respectively, with input matrices X and X_R and corresponding labels y and y_R. Additionally, let θ_{full} and θ_* denote the optima of the ridge objective (5) for the full data S and retain set S_R respectively. After t iterations of unlearning starting from θ_{full}, the iterate θ_t satisfies*

$$\theta_t - \theta_* = \begin{cases} (I - 2\eta\lambda)^t \left(I - \frac{2\eta}{1-2\eta\lambda} X_R^\top X_R\right)^t (\theta_{full} - \theta_*) & \text{for ridge gradient descent (GD).} \\ (I - 2\eta X^\top X)^t (\theta_{full} - \theta_*) & \text{for oracle matching (OM).} \end{cases}$$

Proof. Note that by using the Taylor expansion of the ridge gradient descent objective around θ_*, we can rewrite it as follows:

$$
\begin{aligned}
&\|X_R\theta - y_R\|_2^2 + \lambda\|\theta\|^2 && (7)\\
&= \|X_R\theta_* - y_R\|^2 + \lambda\|\theta_*\|_2^2 + (\theta-\theta_*)^\top \left(X_R^\top X_R + \lambda I\right)(\theta-\theta_*) && (8)\\
&= c + (\theta-\theta_*)^\top \left(X_R^\top X_R + \lambda I\right)(\theta-\theta_*)\,, && (9)
\end{aligned}
$$

where $c = \|X_R\theta_* - y_R\|^2 + \lambda\|\theta_*\|^2$ is a constant independent of θ. Similarly, we can write oracle matching objective as

$$\|X\theta - X\theta_*\|^2 = (\theta-\theta_*)^\top \left(X^\top X\right)(\theta-\theta_*)\,. \tag{10}$$

Note that both the ridge gradient descent and the oracle matching objectives can be written as

$$(\theta-\theta_*)^T \left(Z^T Z\right)(\theta-\theta_*) + c \tag{11}$$

for some PSD matrix Z^TZ and some constant c. Gradient descent update on objective 11 can be written as:

$$\theta_t = \theta_{t-1} - 2\eta\, Z^\top Z(\theta_{t-1} - \theta_*). \tag{12}$$

Subtracting θ_* from both sides,

$$\theta_t - \theta_* = (I - 2\eta Z^\top Z)(\theta_{t-1} - \theta_*). \tag{13}$$

Unrolling the recursion, squaring both sides, and simplifying then yields the desired result. □

D.2 Proof of Theorem 2

Theorem 2 (Proof in Appendix D.2)**.** *Consider the setting of Theorem 1, where θ_t is the iterate after t steps of unlearning initialized at θ_{full}. Further let $\gamma_{min}, \gamma_{max}$ denote the minimum and maximum non-zero eigenvalues of $X^\top X$. Then, as long as the learning rate $\eta \in (0, \frac{2}{5(\gamma_{max}+\lambda)})$ and rank$(X) > 1$,*

$$\frac{\mathbb{E}\left[\|\theta_t - \theta^*\|^2\right]}{\|\theta_{full} - \theta_*\|^2} \in \begin{cases} O\left((1-\gamma_{min}\eta)^t\right) & \textit{for oracle matching (OM).} \\ \Omega\left(\frac{1}{t^2}\right) & \textit{for ridge gradient descent (GD).}\end{cases} \tag{6}$$

Proof. We will begin with a more general setting than the theorem. In particular, consider an arbitrary convex optimization problem of the form

$$\min_\theta f(\theta), \qquad \text{where } f(\theta) = \frac{1}{n}\sum_{i=1}^n f_i(\theta),$$

where the function f is α-strongly convex, and each f_i is β-smooth. In other words, we have that for any θ and θ',

$$
\begin{aligned}
\langle \nabla f(\theta) - \nabla f(\theta'), \theta - \theta'\rangle &\geq \alpha\|\theta - \theta'\| && (f \text{ is } \alpha\text{-strongly convex})\\
\|\nabla f_i(\theta) - \nabla f_i(\theta')\| &\leq \beta\|\theta - \theta'\| \text{ for all } i \in [n]. && (\text{each } f_i \text{ is } \beta\text{-smooth})
\end{aligned}
$$

Note that both GD and OM are both α-strongly convex and β-smooth for $\alpha = \gamma_{min}(X_R^T X_R) + \lambda$ and $\beta = \gamma_{max}(X_R^T X_R) + \lambda$ for GD, and $\alpha = \gamma_{min}^{+}(X^T X)$ and $\beta = \gamma_{max}(X^T X)$ for OM. Note that for OM the strong convexity parameter is the smallest non-zero eigenvalue (denoted by γ_{min}^{+}, because the OM trajectory is guaranteed to keep θ in span(X), and hence in the span of the non-zero eigenvalues. We also know from Weyl's inequalities that $\gamma_{max}(X^T X) \leq \gamma_{max}(X_R^T X_R) + \lambda_{max}(X_F^T X_F)$, and $\gamma_{min}(X^T X) \geq \gamma_{min}(X_R^T X)$.

We further define a quantity σ_f^2 called *gradient disagreement*, measured as

$$\sigma_f^2 := \mathbb{E}_i \left[\|\nabla f_i(\theta_*) - \mathbb{E}[\nabla f_i(\theta_*)]\|^2 \right] = \mathbb{E}_i \left[\|\nabla f_i(\theta_*)\|^2 \right],$$

where θ_* is the optimum of f.

Upper bound for OM. With these quantities defined, let us begin with the OM upper bound. In fact, this follows directly from a standard SGD convergence proof, e.g., Theorem 5.8 of (Garrigos & Gower, 2023), restated below:

Theorem 3 (Theorem 5.8 of (Garrigos & Gower, 2023))**.** *Suppose f is a α-strongly convex sum of β-smooth convex functions. Consider the sequence of iterates $\{\theta_t\}_{t\in\mathbb{N}}$ generated by stochastic gradient descent with a fixed step size $\eta \in (0, \frac{1}{2\beta})$. For $t \geq 0$,*

$$\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] \leq (1-\eta\alpha)^t \|\theta_0 - \theta_*\|^2 + \frac{2\eta}{\alpha} \cdot \sigma_f^2.$$

A few observations conclude the proof. First, any step size $\eta \leq \frac{2}{5(\gamma_{max}+\lambda)}$ also satisfies $\eta \leq \frac{1}{2\gamma_{max}}$ and we can thus apply the Theorem to our case. Second, for oracle matching, we have that

$$\nabla f_i(\theta) = 2\left(\boldsymbol{x}_i^\top \theta - \boldsymbol{x}_i^\top \theta_*\right)\boldsymbol{x}_i,$$

which means that $\nabla f_i(\theta_*) = 0$ and thus $\sigma_f^2 = 0$, concluding the proof.

Lower bound for GD. We now show that for the same set of learning rates, the stochastic version of ridge gradient descent on the retain set cannot converge faster than $1/t^2$. Key to our analysis will be that, for GD, the gradient disagreement σ_f^2 is non-zero, so long as the dataset is non-degenerate. In particular,

$$\begin{aligned}\sigma_f^2 &= \mathbb{E}_i\left[\|\nabla f_i(\theta_*)\|^2\right] \\ &\geq \frac{1}{n}\max_i \|\nabla f_i(\theta_*)\|^2 \\ &= \frac{4}{n}\max_i \left\|(\boldsymbol{x}_i^\top \theta_* - y_i)\boldsymbol{x}_i + \lambda\theta_*\right\|^2.\end{aligned}$$

To see that this is strictly positive, we can proceed by contradiction. Suppose that $\sigma_f^2 = 0$. Observe that if $\boldsymbol{x}_i^\top \theta_* = y_i$ for any $i \in [n]$, then the corresponding $\|\nabla f_i(\theta_*)\|^2 = 4\lambda^2\|\theta_*\|^2$, and so $\sigma_f^2 > 0$. Thus, $\boldsymbol{x}_i^\top \theta_* \neq y_i$ for all i. In this case, however, we must have that

$$(\boldsymbol{x}_i^\top \theta_* - y_i)\boldsymbol{x}_i = -\lambda\theta_*,$$

meaning that θ_* is parallel to $\boldsymbol{x}_i$. If rank$(X) > 1$, this is a contradiction and so $\sigma_f^2 > 0$.

We can now continue with the rest of the proof. We start with some algebraic manipulation of the gradient update. In particular, for a random $i \in [n]$,

$$\begin{aligned}\theta_t - \theta_* &= \theta_{t-1} - \theta_* - \eta \nabla f_i(\theta_{t-1}) \\ \|\theta_t - \theta_*\|^2 &= \|\theta_{t-1} - \theta_*\|^2 - 2\eta \langle \theta_{t-1} - \theta_*, \nabla f_i(\theta_{t-1})\rangle + \eta^2 \|\nabla f_i(\theta_{t-1})\|^2.\end{aligned}$$

Taking an expectation conditioned on θ_{t-1},

$$\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] = \|\theta_{t-1} - \theta_*\|^2 - 2\eta \langle \theta_{t-1} - \theta_*, \nabla f(\theta_{t-1})\rangle + \eta^2 \mathbb{E}\left[\|\nabla f_i(\theta_{t-1})\|^2\right].$$

Now, we treat the second and third terms separately. In particular, for the second term,

$$\begin{aligned}2\eta \langle \theta_{t-1} - \theta_*, \nabla f(\theta_{t-1})\rangle &\le 2\eta \|\theta_{t-1} - \theta_*\| \|\nabla f(\theta_{t-1})\| && \text{(Cauchy-Schwarz)} \\ &= 2\eta \|\theta_{t-1} - \theta_*\| \|\nabla f(\theta_{t-1}) - \nabla f(\theta_*)\| && \text{(Gradient at optimum is zero)} \\ &\le 2\eta\beta \|\theta_{t-1} - \theta_*\|^2. && \text{(Smoothness)}\end{aligned}$$

For the third term, we use the identity $\|u - v\|^2 \le 2\|u\|^2 + 2\|v\|^2$, which we can rearrange to be $\|u\|^2 \ge \frac{1}{2}\|u - v\|^2 - \|v\|^2$. Letting $u = \nabla f_i(\theta_{t-1})$ and $v = \nabla f_i(\theta_{t-1}) - \nabla f_i(\theta_*)$,

$$\begin{aligned}\eta^2 \mathbb{E}\left[\|\nabla f_i(\theta_{t-1})\|^2\right] &\ge \frac{\eta^2}{2} \mathbb{E}\left[\|\nabla f_i(\theta_*)\|^2\right] - \eta^2 \mathbb{E}\left[\|\nabla f_i(\theta_{t-1}) - \nabla f_i(\theta_*)\|^2\right] \\ &\ge \frac{\eta^2}{2} \mathbb{E}\left[\|\nabla f_i(\theta_*)\|^2\right] - \eta^2\beta^2 \|\theta_{t-1} - \theta_*\|^2 \\ &= \frac{\eta^2 \sigma_f^2}{2} - \eta^2\beta^2 \|\theta_{t-1} - \theta_*\|^2.\end{aligned}$$

Combining everything so far,

$$\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] \ge \left(1 - 2\eta\beta - \eta^2\beta^2\right) \|\theta_{t-1} - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2}$$

Taking an expectation with respect to previous iterates yields

$$\begin{aligned}\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] &\ge \left(1 - 2\eta\beta - \eta^2\beta^2\right)^t \|\theta_0 - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2} \cdot \sum_{\tau=0}^{t-1} \left(1 - 2\eta\beta - \eta^2\beta^2\right)^\tau \\ &\ge \left(1 - 2\eta\beta - \eta^2\beta^2\right)^t \|\theta_0 - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2} \\ &\ge \left(1 - \frac{12}{5}\eta\beta\right)^t \|\theta_0 - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2} && \text{since } \eta \le \frac{2}{5\beta}.\end{aligned}$$

For ease of notation, let $C = \frac{12}{5}\beta$. Note that $\log(1-x) \ge \frac{x^2 - 2x}{1-x}$ for $x < 1$, and so

$$\mathbb{E}\left[\|\theta_t - \theta_*\|^2\right] \ge \exp\left(t \cdot (-C\eta) \frac{2 - C\eta}{1 - C\eta}\right) \|\theta_0 - \theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2}. \tag{14}$$

We now derive the learning rate η that minimizes (14), and show that at this optimal learning rate (14) $= \Omega(1/t^2)$. Note we can see this by inspection even without the formal derivation, because for (14) to be $O(\frac{1}{t^2})$ we need $\eta = O(\frac{1}{t})$ so that the right hand term is $O(1/t^2)$, which forces the first term $\exp\left(t \cdot (-C\eta) \frac{2 - C\eta}{1 - C\eta}\right) = O(1)$. Now, to minimize the right hand side above with respect to η, we take the derivative and set to zero (note that at the extreme points $\eta = 0$ and $\eta = \frac{2}{5\beta}$ we are left with a constant amount of error). The result of this calculation is the fixed learning rate that optimizes the error at time t. Again for ease of notation, let $g(\eta) = \frac{2-C\eta}{1-C\eta}$, so that

$$\begin{aligned}
0 &= \frac{d}{d\eta}\left[\exp\left(t\cdot(-C\eta)g(\eta)\right)\|\theta_0-\theta_*\|^2 + \frac{\eta^2\sigma_f^2}{2}\right] \\
&= -Ct\left(g(\eta)+\eta g'(\eta)\right)\exp\left(t\cdot(-C\eta)g(\eta)\right)\|\theta_0-\theta_*\|^2 + \eta\sigma_f^2 \\
\eta\sigma_f^2 &= Ct\left(g(\eta)+\eta g'(\eta)\right)\exp\left(t\cdot(-C\eta)g(\eta)\right)\|\theta_0-\theta_*\|^2 \\
\log(\eta\sigma_f^2) &= \log(t) + \log(C(g(\eta)+\eta g'(\eta))\|\theta_0-\theta_*\|) - Ct\eta\cdot g(\eta) \\
Ct\eta\cdot g(\eta) - \log(t) &= \log(C(g(\eta)+\eta g'(\eta))\|\theta_0-\theta_*\|) - \log(\eta\sigma_f^2) \\
Ct\eta\cdot g(\eta) &\geq \log(C(g(\eta)+\eta g'(\eta))\|\theta_0-\theta_*\|) - \log(\eta\sigma_f^2) \\
&\geq \log\left(\frac{C(g(\eta)+\eta g'(\eta))\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + \log(1/\eta) \\
t &\geq \frac{\log\left(\frac{C(g(\eta)+\eta g'(\eta))\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + \log(1/\eta)}{C\eta\cdot g(\eta)}
\end{aligned}$$

Now, by definition of $g(\eta)$, we have that for $\eta \in (0, \frac{2}{5\beta})$, $g(\eta) \leq 26$ and $g(\eta) + \eta\cdot g'(\eta) \geq 2$. Thus:

$$\begin{aligned}
t &\geq \frac{\log\left(\frac{2C\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + \log(1/\eta)}{26C\eta} \\
26Ct &\geq \frac{\log\left(\frac{2C\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + \log(1/\eta)}{\eta}
\end{aligned}$$

Using the fact that $\log(1/x) \geq 1 - x$ for $x \in (0,1)$ yields:

$$\begin{aligned}
26Ct &\geq \frac{\log\left(\frac{2C\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + 1}{\eta} - 1 \\
\eta &\geq \frac{\log\left(\frac{2C\|\theta_0-\theta_*\|}{\sigma_f^2}\right) + 1}{26Ct+1}.
\end{aligned}$$

Plugging this result into (14) yields the desired $\Omega(1/t^2)$ lower bound. □

D.3 Experiment details

Details of unlearning algorithms

We consider the various iterative algorithms for unlearning starting from θ_{full}. For all of them, we consider their full-batch as well as the stochastic version. For the stochastic versions, we use a mini-batch size of 5. We search for the learning rate from $\{10, \frac{10}{2}, \frac{10}{2^2}, \cdots, \frac{10}{2^{20}}\}$. We describe the algorithms below:

1. **Ridge Gradient Descent (GD)**: This involves minimizing the ridge regression objective with the retain set points $(X_{\text{retain}}, y_{\text{retain}})$ using gradient descent.

2. **Ridge Gradient Descent + Ascent (GD +GA)**: This method aims to incorporate forget set points in the ridge gradient descent updates. Each step involves moving in a direction that is a linear combination of the gradient descent step on the retain set and the gradient ascent step on the forget set. That is, we set

   $$\theta_t = \theta_{t-1} - \eta(\text{grad}_{\text{retain}}(\theta_{t-1}) - \alpha * \text{grad}_{\text{forget}}(\theta_{t-1})).$$

   Here, $\text{grad}_{\text{retain}}(\theta) = 2X_{\text{retain}}^T(X_{\text{retain}}\theta - y_{\text{retain}}) + 2\lambda\theta$ and $\text{grad}_{\text{forget}}(\theta) = 2X_{\text{forget}}^T(X_{\text{forget}}\theta - y_{\text{forget}})$. We do a hyperparameter search for α in $\{0.01, 0.1, 1, 10\}$.

   In the stochastic setting, in each update step, we draw minibatch points uniformly at random from the full dataset, and calculate the ascent step term only if the drawn points include points from the forget set.

3. **Oracle Matching (OM)**: Here, we assume oracle access to predictions made using the optimal model θ_*. This method involves using gradient descent to minimize the the squared error from the oracle predictions on the full dataset: $||X_{\text{full}}\theta - X_{\text{full}}\theta_*||_2^2$. We include a full algorithm of this in Algorithm A.1.

4. **Oracle Matching on retain set (OM retain set)**: This involves using gradient descent to minimize the squared error from the oracle predictions only on the retain set points: $||X_{\text{retain}}\theta - X_{\text{retain}}\theta_*||_2^2$.

Slow convergence with stochastic gradient descent.

In Section 6, we saw that in the stochastic setting, OM converges much faster than GD, even when λ is large. Here, we dig deeper into the large λ experiment considered in Section 6 to understand this. We observe that stochastic GD remains stable only at small learning rates with large λ, which results in slower progress. Specifically, while the optimal learning rate for full-batch GD is similar in both large and small λ regimes, for stochastic GD, it is about 100 times smaller in the large λ regime. Using a higher learning rate for stochastic GD in the large λ regime leads to instability and non-convergence, an issue not seen with stochastic OM.

In each update step, stochastic OM adjusts the model parameters only within the span of the random subset of points used for that update. On the other hand, stochastic GD decays the model parameters in other directions due to the regularization term. Although we want the parameters to decay in the subspace orthogonal to the span of the retain set, stochastic GD also decays the parameters in directions spanned by the retain set points that are not part of the current update set. As a result, using a high learning rate for stochastic GD in the large λ regime disrupts parameters in the span of the retain set. Therefore, while increasing λ improves full-batch GD's convergence speed and could potentially make it faster than OM, this advantage does not apply to stochastic GD, which has to use a much smaller learning rate.

In Figure D.1a, we illustrate how stochastic GD at high learning rates disrupts the model parameters within the span of the retain set. We plot the progression of relative squared distance to the optimal unlearned parameter within the span of retain set points, $\frac{||P_{\text{retain}}(\theta_t - \theta)||_2^2}{||\theta_{\text{full}} - \theta_*||_2^2}$. Here θ_t is the iterate at time t, θ_{full} is the model trained on the full dataset, θ_* is the optimal unlearned model, and P_{retain} is the projection matrix onto the retain set span. We show this for iterates with the optimal learning rate, as well as for learning rates 4 times faster and 4 times slower. As the learning rate increases, the iterates tend to diverge in the span of the retain set.

In Figure D.1b, we plot the progression of relative squared distance to the optimal unlearned parameter orthogonal in the subspace orthogonal to the retain set points, $\frac{||P_{\text{orth-retain}}(\theta_t - \theta_*)||_2^2}{||\theta_{\text{full}} - \theta_*||_2^2}$, where $P_{\text{orth-retain}}$ is the projection matrix for the orthogonal subspace. Here, we observe that increasing the learning rate beyond the optimal rate leads to faster convergence. Thus, while increasing the learning rate beyond the optimal value accelerates convergence in the subspace orthogonal to the retain points, it harms progress in the span of the retain set points. Therefore, in the large λ regime, stochastic GD must operate at small learning rates to avoid disrupting the model parameters in the span of the retain set.

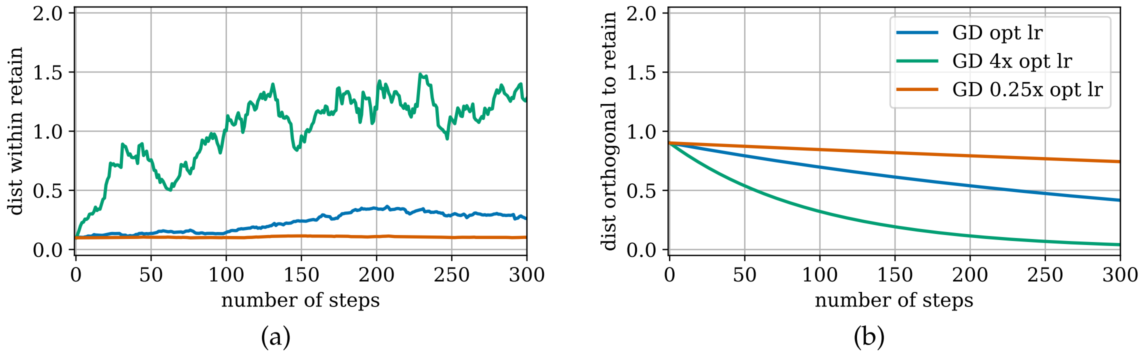
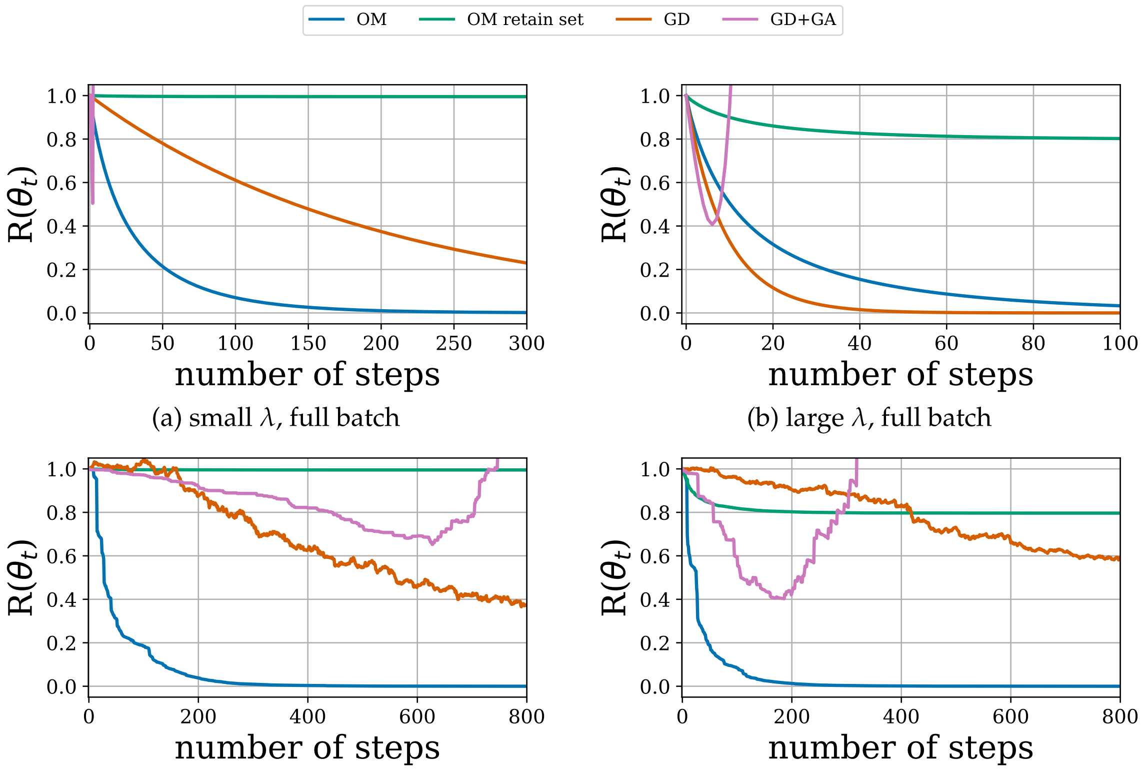


Figure D.1: Stochastic Gradient Descent performance in the large λ regime with varying learning rates. (a) The y-axis shows the relative squared distance to the optimal unlearned parameter within the span of retain set points, $\frac{||P_{\text{retain}}(\theta_t - \theta)||_2^2}{||\theta_{\text{full}} - \theta_*||_2^2}$, where θ_t is the iterate at time t, θ_{full} is the model trained on the full dataset, θ_* is the optimal unlearned model, and P_{retain} is the projection matrix onto the retain set span. Larger learning rates lead to divergence within this span. (b) The y-axis shows the relative squared distance to the optimal unlearned parameter in the subspace orthogonal to the retain set points, $\frac{||P_{\text{orth-retain}}(\theta_t - \theta_*)||_2^2}{||\theta_{\text{full}} - \theta_*||_2^2}$, where $P_{\text{orth-retain}}$ is the projection matrix for the orthogonal subspace. Larger learning rates result in faster convergence in this subspace.

D.4 Additional experiments

In section 6, we discussed an example with linear models that highlighted the qualitative differences between oracle matching and other unlearning methods. Here, we show the comparison for another example with different covariance structure. We draw 100 training points (x_i, y_i) where x_i are drawn i.i.d. from $N(0, \Sigma)$ in 400 dimensions where covariance matrix $\Sigma = diag(1, 1/2, 1/3, \cdots, 1/400)$ ($diag(.)$ represents a diagonal matrix with the specified entries on the diagonal). $y_i = \theta^T x_i + \epsilon_i$, where θ is drawn from $N(0, I)$ and ϵ_i is drawn from $N(0, 1/4)$. These 100 points form $(X_{\text{full}}, y_{\text{full}})$. We fit these points to minimize the ridge regression objective with $\lambda = 1/4$ (for the small λ case) or $\lambda = 5$ (for the large λ case), to obtain the model θ_{full}. Here $\lambda = 1/4$ is the λ value that minimizes the expected squared prediction error. We want to unlearn 5 training points chosen uniformly at random. We show the performance of various methods (in both the stochastic and full-batch setting with small and large λ) in Figure D.2. Even here, we obtain the same qualitative patterns as in Figure D.2.



(a) small λ, full batch

(b) large λ, full batch

(c) small λ, stochastic

(d) large λ, stochastic

Figure D.2: Another example comparing the performance of various unlearning methods with linear models.

E Unlearning evaluation

E.1 KL Divergence of Margins (KLoM)

Below we formally define the KLoM evaluation, which computes the distance between the distribution of outputs for unlearned models and re-trained models. For output, we use the classification margin. We also include a visual representation of our algorithm in Figure E.1.

Algorithm E.1 KLoM

```
1: Input Number of models $N$, dataset $D$, forget set $F \subseteq D$, retain set $R \subseteq D$ (such that $D = F \cup R$), and a validation dataset $V$, training algorithm $A$, unlearning algorithm $U$, margin function $\phi$, and histogram function $H(S)$ .
2: Train $N$ models (Oracles) on the entire dataset, excluding the forget set. $\Theta^o = \{A(D \setminus F)\}$, $|\Theta^o| = N$.
3: Train $N$ models (Unlearned-models) on the entire dataset, then for each model unlearn forget set $F$, $\Theta^f = \{U(A(D), F)\}$, $|\Theta^f| = N$.
4: Initialize a vector of results with all zeros $\vec{r}$.
5: for each point $x$ in {Forget, Retain, Validation} set do
6:     Compute the margins for each oracle $M_o = \{\phi(\theta_i^o(x)) | \theta_i^o \in \Theta^o\}$.
7:     Compute the margins for each unlearned-model $M_f = \{\phi(\theta_i^f(x)) | \theta_i^f \in \Theta^f\}$
8:     Assign $\vec{r}[x] = KL(Hist(M_o), Hist(M_f))$
9: end for
10: return $\vec{r}$
```

Note that in order to approximate the KL divergence, we compute a histogram $H(S)$ that takes a set of real numbers and returns an empirical probability distribution by truncating and binning samples from S.

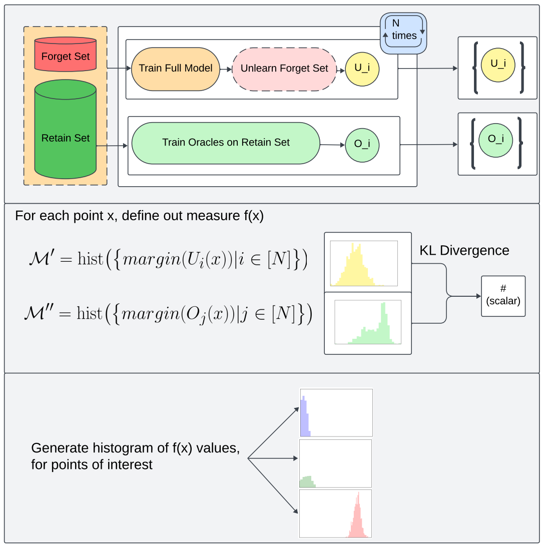

Forget Set
Retain Set
N times
Train Full Model
Unlearn Forget Set
U_i
Train Oracles on Retain Set
O_i
U_i
O_i
For each point x, define out measure f(x)
$\mathcal{M}' = \text{hist}(\{margin(U_i(x)) | i \in [N]\})$
$\mathcal{M}'' = \text{hist}(\{margin(O_j(x)) | j \in [N]\})$
KL Divergence
# (scalar)
Generate histogram of f(x) values, for points of interest

Figure E.1: Visual diagram of KLoM

In practice, we compute KLoM using $N = 100$ oracles compared to $N = 100$ unlearned models. For each point we evaluate we clip the N margins to a range of $[-100, 100]$ to exclude outliers (some methods, like SCRUB, result in extremely large margins) into 20 bins. Then the KL-divergence is computed between the binned histogram of the oracles and the binned histogram of the unlearned models. For any region of the support where one histogram has no support, and the other does, the bin with no support is set to a non-zero probability of $\epsilon = 10^{-5}$. We note, that for this reason, `KLoM` scores are artificially rescaled and capped at value around ≈ 12. This cap can be changed by changing the number of bins or the minimum value ϵ.

E.2 `U-LiRA`

At a high-level `U-LiRA` measures the distinguishability of predictions of an unlearned model from that of retrained models based on adapting membership inference attacks. Implementing the Algorithm E.2, as written, would be computationally infeasible in most cases, as it involves unlearning and retraining T times for each point (x, y). In practice, Hayes et al. (2024) computes `U-LiRA` using the method Algorithm E.4.

Omitting a few details, the main idea is as follows: consider an unlearning algorithm $\mathcal{U}$ and a training algorithm $\mathcal{A}$. Start from a model θ_0 trained on a random subset $S \sim \mathcal{P}^n$, and unlearn one specific (random) forget set, to produce θ_F. Next, construct a collection of shadow models by that producing many unlearned models from random training sets and random forget sets. Lastly for a collection of points in the retain set and in the forget set ($\{x|x \in S_F \cup S\}$) compare how θ_F compares to the subset of shadow models that never-saw-x distribution of margins for models that unlearned-x. Now if $\mathcal{U}$ perfectly unlearns S_F, then θ_0 no longer depends on S_F, and so even *conditioned* on θ_0 the marginal distribution of $x \in S_F|\theta_0$ is still $\mathcal{P}$; the same distribution as any $x \in S_V$. Operationalizing this intuition, `U-LiRA` with probability $\frac{1}{2}$ draws either $x \in S_F$ or $x \in S_V$, and measures the output $y = f_x(\theta_0)$. An (optimal) adversary observes y, and tries to guess whether the corresponding x was an unlearned point or a validation point, e.g. whether $x \in S_F$ or S_V. More generally, if $\mathcal{U}$ is an (ϵ, δ)-unlearning algorithm, the two distributions $y|x \in S_F, \theta_0$ and $y|x \in S_V, \theta_0$ would be (ϵ, δ)-indistinguishable by post-processing, and so even the optimal adversary couldn't have accuracy greater than $\frac{1}{2}e^{\epsilon} + \delta$. The optimal adversary can be implemented by training models with/without x, unlearning S_F, and then measuring the output y. For a more detailed description of `U-LiRA` and overview of similar MIA-based approaches, we refer the reader to (Hayes et al., 2024) (Section 4.2), and we include the pseudocode for the computationally efficient version of this evaluation Efficient-ULIRA in Appendix E.2.

In the original `U-LiRA` paper (Hayes et al., 2024), they report results for Efficient `U-LiRA` for N= 256, forgettable points F all points of class 5, $n_f = 40$ random forget sets per base model, and each forget set $m = 20$. The N base models that they are train are trained on ResNet-18 for 100 epochs, and so are highly overparameterized.

For our evaluation, `U-LiRA` paper (Hayes et al., 2024), we run Efficient `U-LiRA` for N= 50, $n_f = 40$ random forget sets per base model, and then we vary the training setting, the forget size n_f, and total set of forgettable points F. Specifically, we try 3 settings

1. ResNet-18 trained for 100 epochs, evaluated on forget sets of size 200, with the forgettable points F being 1000 points in class 5.

2. ResNet-9 trained for 25 epochs, evaluated on forget sets of size 200, with the forgettable points F being 1000 points in class 5.

3. ResNet-9 trained for 25 epochs, evaluated on forget sets of size 50, with the forgettable points F being 500 random points in the dataset.

Algorithm E.2 U-LiRA (LiRA adapted for machine unlearning) (Hayes et al., 2024)

```
Args: model parameters to evaluate $\theta^*$, learning algorithm $A$, unlearning algorithm $U$, number of shadow models $T$, example $(x,y)$, logit function $\phi$, function that returns probabilities $f(\cdot,\theta)$ given model parameters $\theta$.
Observations: $O \leftarrow \{\}, \hat{O} \leftarrow \{\}$
while $t \leq T$ do
    • $D \leftarrow$ sample a dataset that includes $(x,y)$
    • $\theta^0 \leftarrow A(D)$ train a model
    • $\theta' \leftarrow U(\theta^0, (x,y))$ unlearn $(x,y)$
    • $\theta'' \leftarrow A(D \setminus (x,y))$ retrain without $(x,y)$
    • $O[t] \leftarrow \phi(f(x;\theta'))$
    • $\hat{O}[t] \leftarrow \phi(f(x;\theta''))$
end while
$\mu, \sigma \leftarrow$ fit Gaussian$(O)$
$\hat{\mu}, \hat{\sigma} \leftarrow$ fit Gaussian$(\hat{O})$
$o \leftarrow \phi(f(x,\theta^*))$
$p_{\text{member}} \leftarrow \frac{N(o;\mu,\sigma^2)}{N(o;\mu,\sigma^2)+N(o;\hat{\mu},\hat{\sigma}^2)}$
if $p_{\text{member}} \geq \frac{1}{2}$ then
    • return Predict $(x,y)$ is a member of training
else
    • return Predict $(x,y)$ is not a member of training
```

Algorithm E.3 Sub algorithm: Membership Prediction

```
1: Input: Sets $A, B$ of real-valued numbers, and point $x \in \mathbb{R}$.
2: $(\mu, \sigma \leftarrow$ fit Gaussian$(A)$
3: $(\hat{\mu}, \hat{\sigma} \leftarrow$ fit Gaussian$(B)$
4: $p_{\text{member}} \leftarrow \frac{\mathcal{N}(x;\mu,\sigma^2)}{\mathcal{N}(x;\mu,\sigma^2)+\mathcal{N}(x;\hat{\mu},\hat{\sigma}^2)}$
5: if $p_{\text{member}} > \frac{1}{2}$ then
6:     return Predict $(x,y)$ is a member of set $A$
7: else
8:     return Predict $(x,y)$ is not a member of set $A$
9: end if
```

Algorithm E.4 Efficient-ULIRA

```
1: **Input** Number of base models N, set of forgettable points $F$ (default: all points of class 5),
   Number of random forget sets per base model $n_f$, size of each forget set $m$ (default 200)
2: Train $N$ base models on random 50% subsets of the dataset
3: **for** each base model $\subseteq_i$ **do**
4:     construct $n_f$ random forget sets of size $m$, denoted $F_{i,j}$.
5:     **for** each random forget set $F_{ij}$ **do**
6:         Unlearn forget set $F_{ij}$
7:     **end for**
8: **end for**
9: Split the $n_f \cdot N$ unlearned models into two sets, Shadow models $S$ and Target models $T$
10: initialize accuracy vector $\vec{a} \in \mathbb{R}^{|T|}$, with all 0's.
11: **for** each target model $\subseteq_\sqcup \in T$ **do**
12:     Construct $D_f$ from $m$ from the $m$ points that $\theta_t$ unlearned
13:     Construct $D_v$ from $m$ from the $m$ points that $\theta_t$ was not trained on.
14:     let $D = D_f \cup D_v$
15:     Let $c = 0$
16:     **for** each point $x \in D$ **do**
17:         Let $S_A$ be the set of shadow models that were trained on $x$.
18:         Let $S_B$ be the set of shadow models that unlearned $x$
19:         Construct sets $A = \{\theta'(x)|\theta' \in S_A\}$, $B = \{\theta'(x)|\theta' \in S_B\}$
20:         Run sub-algorithm Membership Prediction E.3 with inputs $(A, B, x)$, returning $l \in \{1, 0\}$
21:         **if** ( **then**$l = 1$ and $x \in D_f$) OR ($l = 0$ and $x \in D_r$)
22:             $c+ = 1$
23:         **end if**
24:
25:     **end for**
26:     Average the model accuracy across all predictions in $D$, $a_t = c/(2m)$
27: **end for**
28: Average the accuracy-per-model over all the target models Return mean($\vec{a}$)
```

E.3 Comparing U-LiRA to KLoM

In this paper we propose a new method for evaluating unlearning, KLoM. Here, we argue that this is a superior measure to existing measures, in particular compared to U-LiRA. The advantages of KLoM are :

1. KLoM requires fewer unlearning trials (on the order of 100) than U-LiRA (which is generally on the order of 2000).
2. KLoM returns a distribution of differences, rather than a binary assignment of if one particular model was more like an unlearned model or a retrained model. This is valuable because it tells you how a method unlearns individual points (e.g. is bad on average, or just bag on specific points)
3. KLoM does not assume the margins can be fit well by a Gaussian. Anecdotally, for U-LiRA, we find this is a decent but not great assumption, and it's currently unclear how much error this really introduces.
4. KLoM has the capacity to look at an unlearning algorithm's ability to handle coherent sets of points, not just random subsets of some set of forgettable points.
5. U-LiRA does not capture closeness to unlearned model, and thus one can force U-LiRA score to go down (implying better unlearning) by having the unlearning method destroy the original model, thus U-LiRA scores must be traded-off against an accuracy drop. KLoM measures distance directly, and thus unlearning can be evaluated with a single measure. We expand on this point below. And is illustrated by figure E.2.

A Toy Example Where U-LiRA Fails:

Consider your unlearning method returns a constant function f (e.g. such that that the margin is always some constant, e.g. 7). In such a situation, the distribution of unlearned models will look radically different from the distribution of models that were fully retrained; in theory, a good unlearning measure should return that this is a bad unlearning method. However, U-LiRA will actually return a nearly perfect unlearning score ($\approx$ 50%). In U-LiRA, one does 2 sets of likelihood ratio tests, first on points the unlearned model unlearned, then on points the unlearned model never saw. The first set of likelihood ratio tests, U-LiRA will get 100% (or nearly), because at margin is constant (7), and so it will be at the unlearned-models peak; thus we get 100% accuracy on these points. Now, on set 2, U-LiRA will get 0% (or nearly), because U-LiRA compute the likelihood ratio at the margin of the unlearned model's prediction, which in this case is a constant (7), thus it will still be at the unlearned-models peak. Thus, U-LiRA will not predict that anything is "not-in-the-training-set," achieving 0% on this set.

However, the problem is that 100 +0 /2 = 50%, which appears to be a perfect unlearning score, but in reality is just a classifier that thinks everything is from an unlearned model.

Advantages of U-LiRA:

1. It is worth noting that our measure requires binning, and some assumptions there. U-LiRA does not (because it makes the Gaussian approximation assumption)
2. U-LiRA reports an accuracy for 1 particular model, and then averages that across multiple unlearning models. this correctly captures individual model performance, rather than aggregate unlearning algorithm performance.

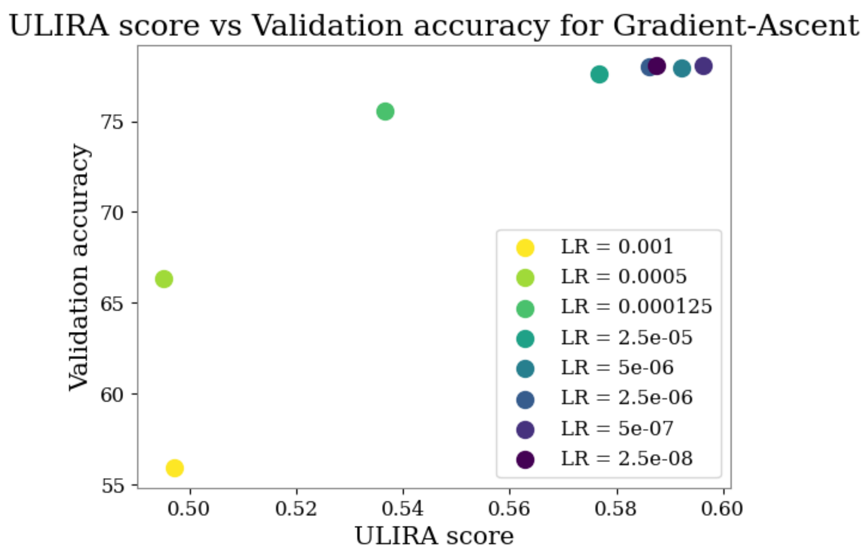


Figure E.2: `U-LiRA` accuracy vs Validation accuracy of the unlearned model, for Gradient Ascent method, for a model trained on CIFAR-10. Unlearning occurs on a random forget sets of size 200 in class 5. Observe that it is possible to achieve nearly perfect `U-LiRA` accuracy (50%) by simply dropping the validation accuracy of the model by 20%. With Gradient Ascent specifically, this is achieved by increasing the learning rate beyond the optimal learning rate, causing the gradient ascent updates to be too significant.

E.4 Sensitivity of unlearning to models and forget sets

See Figure E.3.

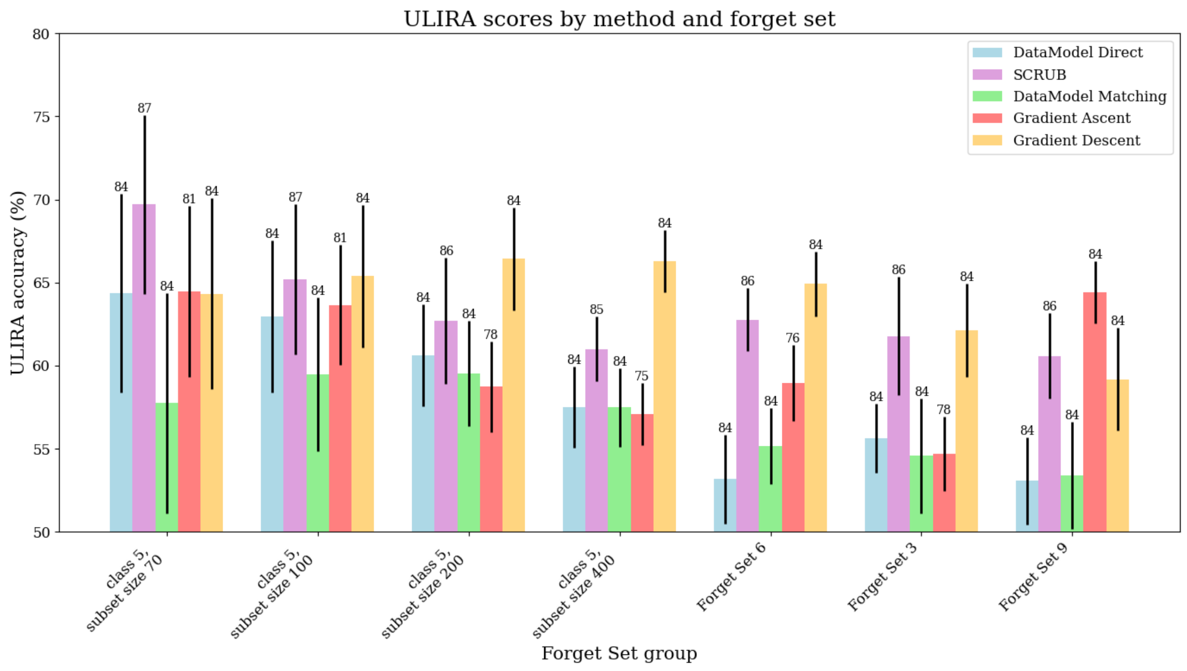


Figure E.3: Results for Efficient-ULIRA run with parameters specified in E.2. The validation accuracies are presented above the bar for the each method-forget set pairing. Observe that Gradient Ascent method is able to achieve an excellent `U-LiRA` score, but does so at the cost of the validation accuracy of the unlearned model.

F Additional results

F.1 Hyperparameter sensitivity of unlearning algorithms

Methods that involve something akin to gradient ascent, like SCRUB, where unlearning for longer or with a larger learning rate can reduce the utility of the model; as opposed to gradient descent based methods, like Oracle Matching, where if the learning rate is sufficiently small, unlearning for longer does not hurt the model. We illustrate this by showing how much the same unlearning algorithm is affected by changing the learning rate, in Figure F.1, where we plot the 85th percentile of `KLoM` score as a function of learning rate[6].

[6]We choose the 85th percentile for KLOM scores, because SCRUB generally performs significantly worse than Oracle Matching, and we found that for higher percentiles, the effect was weaker because SCRUB never achieved particularly good KLOM scores.

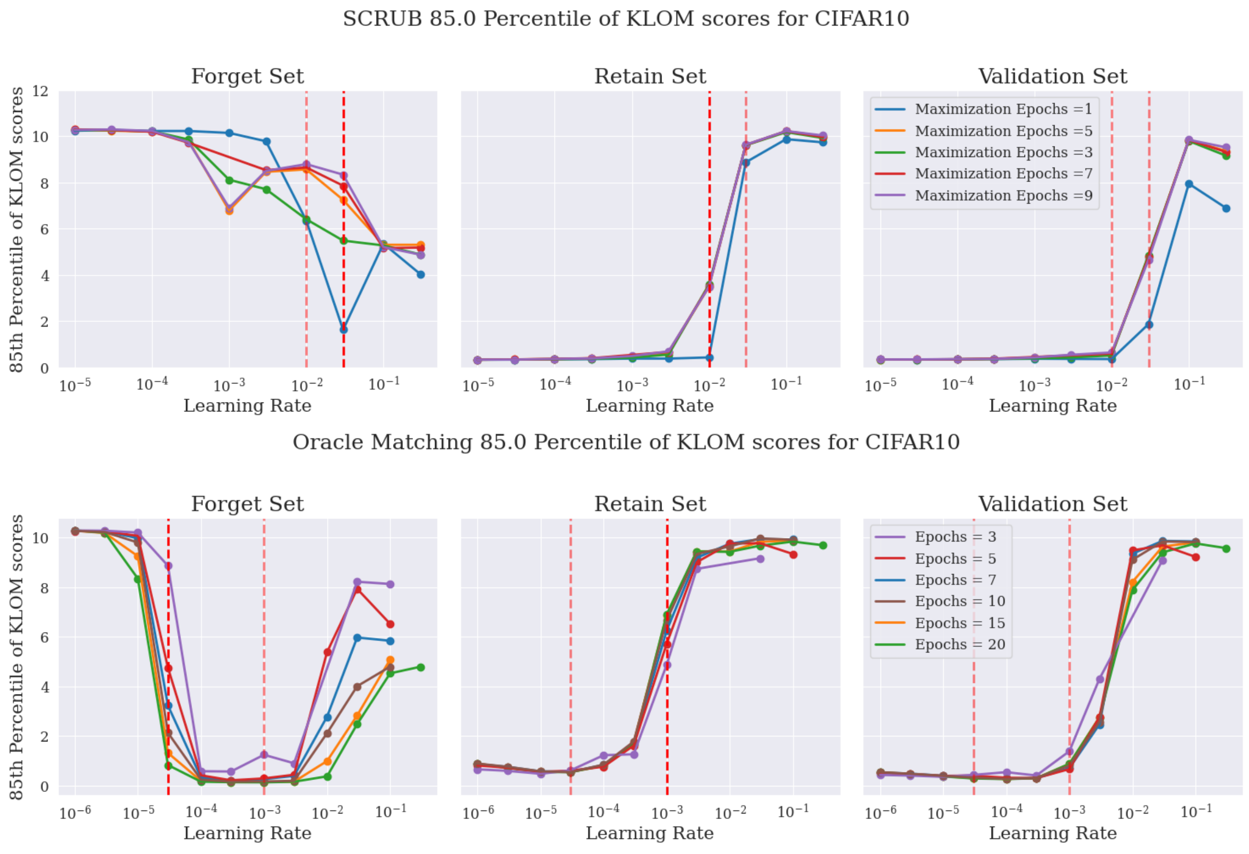


Figure F.1: KLOM sensitivity to learning rate. SCRUB is trained for 10 epochs, with the first "Maximization Epochs" used for Gradient Ascent; Oracle Matching is trained for 'Epochs', both demarcated in the legend. Both algorithms are evaluated on Forget Set 5, which is a non-random forget set of size 100. The red lines denotes the learning rate at which the KLOM score drops below 6 (an arbitrarily chosen threshold which we deemed as the largest value approaching a "reasonable" unlearning). Left red line indicates the largest forget that achieves a reasonable unlearning for Forget points, and the right red line indicates the smallest learning rate that achieves a reasonable unlearning for Validation and Retain points. The line is darker if that subset is the limiting factor for the thresholding.

The key takeaway is that the range of learning rates for SCRUB is much smaller (less than an order of magnitude), while Oracle Matching not only performs significantly better, the range of learning rates that attain good KLOM scores is much larger dynamic range. We do note that because this is a log-scale, the range for the SCRUB learning rates is larger in magnitude; however, the critical observation is how flat the KLOM scores are as a function of learning rate and the relative ranges of learning rates. Flatter and wider ranges (even on a log-scale) make hyperparameter tuning dramatically simpler and significantly more likely to choose a learning rate in a good range.

F.2 Per-sample unlearning over time

We present the rate of unlearning individual data points as a function of epochs in the unlearning algorithm for both Oracle Matching and Data Models matching, which we contrast with SCRUB unlearning, presented in Figure 2.

See Figure F.2.

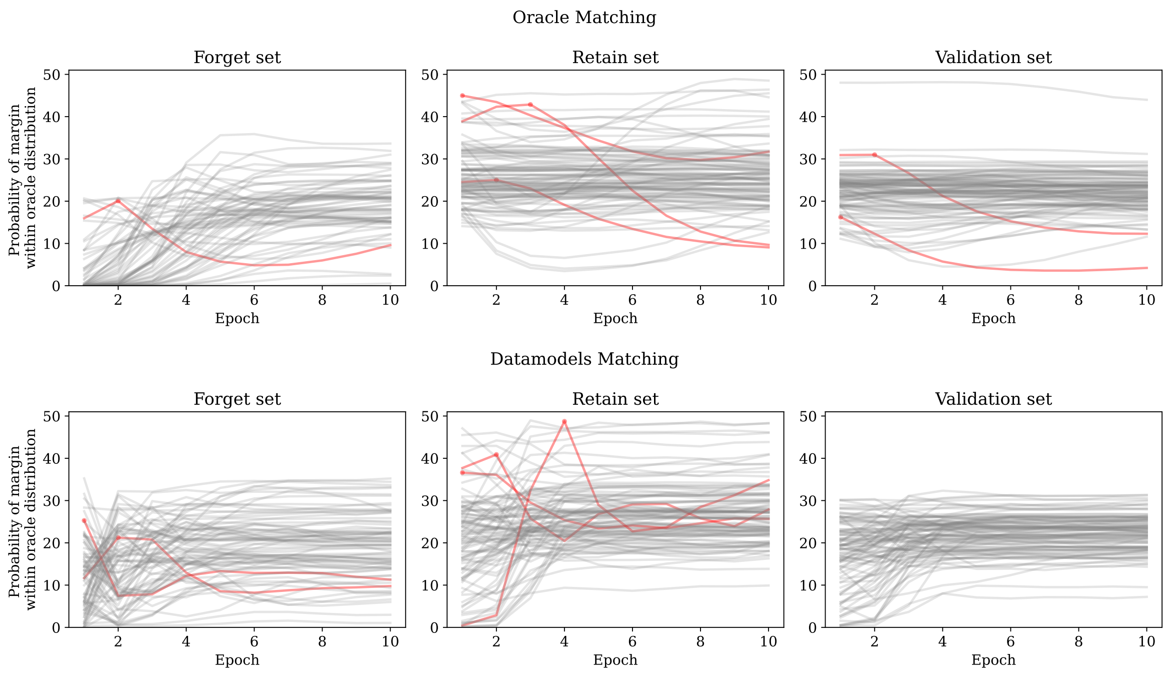


Figure F.2: Plotting unlearning quality over time for Oracle Matching and Datamodels matching. We observe that different points unlearn at different rates. However, unlike for gradient ascent based methods like SCRUB, for gradient descent based methods like Datamodels matching and oracle matching, one a data point is unlearned, it tends to remain unlearned. The red lines highlight examples whose final quality of unlearning is more than 10% worse than their maximum unlearning quality.

F.3 Full `KLoM` evaluation

Building on this point that the effectiveness of different methods is forget set specific, we now share all the `KLoM` scores for the different methods, below in Figures F.3 and F.4.

Observe that Gradient Descent tends to perform very well for the retain and validation sets, but leaves the forget set relatively unchanged. This is parametrizable through hyperparameters like weight-decay, but not explored in this plot.

F.4 Model accuracy after unlearning

For some measures of unlearning it is possible to do well on the machine unlearning measure-of-success, while at the same time doing completely terribly on the target task, resulting in terrible generalization and large validation error. We note that our measure of success, `KLoM`, does not suffer from this problem, as a machine unlearning algorithm is measured on its ability to match the margins of a retrained model, which is assumed to do well in general. However, for completeness, we include a table of the model accuracies for each unlearning methods, evaluated for the method that achieves the lowest `KLoM` 99th-percentile score on the forget set.

Index	GA	GD	Scrub	Retrain an Oracle	DM-Direct	DM-Matching
1	88.34	88.44	88.29	88.48	89.97	89.90
2	88.48	88.49	88.36	88.51	89.98	89.70
3	88.25	88.48	87.55	88.25	89.96	89.61
4	87.85	88.44	88.26	88.53	89.95	89.46
5	88.07	88.48	81.99	88.51	89.93	89.86
6	84.20	88.47	85.65	88.42	89.86	89.47
7	87.98	88.44	88.26	88.49	89.98	89.88
8	88.30	88.49	81.62	88.43	90.03	89.93
9	10.92	88.48	85.36	88.46	89.99	89.89

Table F.1: Model Accuracy Results

We note that the pareto optimality curve is computed separately for each forget set. This is because because we observe that unlearning is quite model-specific and forget-specific; as such, it's reasonable that a practitioner should have a different set of hyperparameters for each unlearning depending on the forget set. Thus, having the pareto-frontier be specified per-forget-set gives each unlearning method the best opportunity to perform in that setting.

F.5 Renyi Divergence Of Margins (`RDoM`)

Throughout this work we largely rely on a statistical notion, we refer to as `KLoM`, formally described in section E.1. Contrary to a measure like ULIRA, which averages a likelihood ratio test over many models and samples, in the computation of `KLoM`, we measure the statistical distance between the margins of models that unlearned a point and models that never saw the point. In E.3 we argue that this difference is quite important to producing a good measure for unlearning. However, the specific choice of distance function used is not as critical. For this paper we rely on `KLoM` because it has natural parallels in formulation to the likelihood ratio test; however, this can trivially be replaced with another measure such as Renyi Divergence. Renyi Divergence with a different choice of α

Definition 4. *(Rényi divergence). For two probability distributions P and Q defined over $\mathcal{R}$, such that $P(x)$ is the density of P at x. The Renyi divergence of order $\alpha > 1$ is*

$$D_\alpha(P\|Q) := \frac{1}{\alpha - 1} \log \mathbb{E}_{x\sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha .$$

It is worth noting that the limit of Renyi divergence for $\lim_{\alpha\to 1}$ is the KL divergence. And on an intuitive level, for two unimodal distributions, the choice of α denotes how much one cares about the difference in the tails of a distribution, rather than the bodies of the distributions.

As a demonstration, to build intuition we now show how KLoM/RDoM values change for a controlled synthetic dataset. For a value of δ, We construct we generate two sets of 1-dimensional datasets, each of size $N = 1000$, one sampled from $\mathcal{N}(\mu = 0, \sigma = 3)$ and the other $\mathcal{N}(\mu = 0, \sigma = 3)$. We sweep over 100 values of δ from 0 to 10. We compute KLoM and RDoM for each of these pairs of datasets and plot a violin plot for KLoM, RDoM for $\alpha = 2$, RDoM for $\alpha = 5$, RDoM for $\alpha = 10$. We plot this below in Figure F.5.

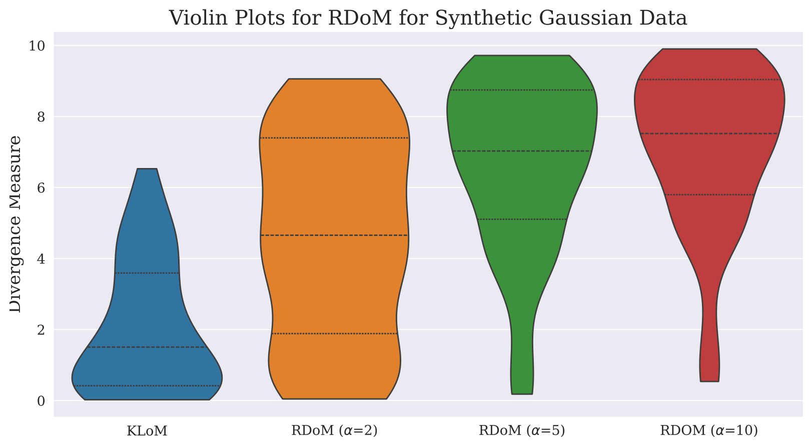


Figure F.5: RDoM and KLoM plot for synthetic Gaussian data.

For completeness we include similar plots, broken down across Forget, Retain, and Validation sets for actual unlearning methods (SCRUB, Datamodels Matching, and Datamodels Direct).

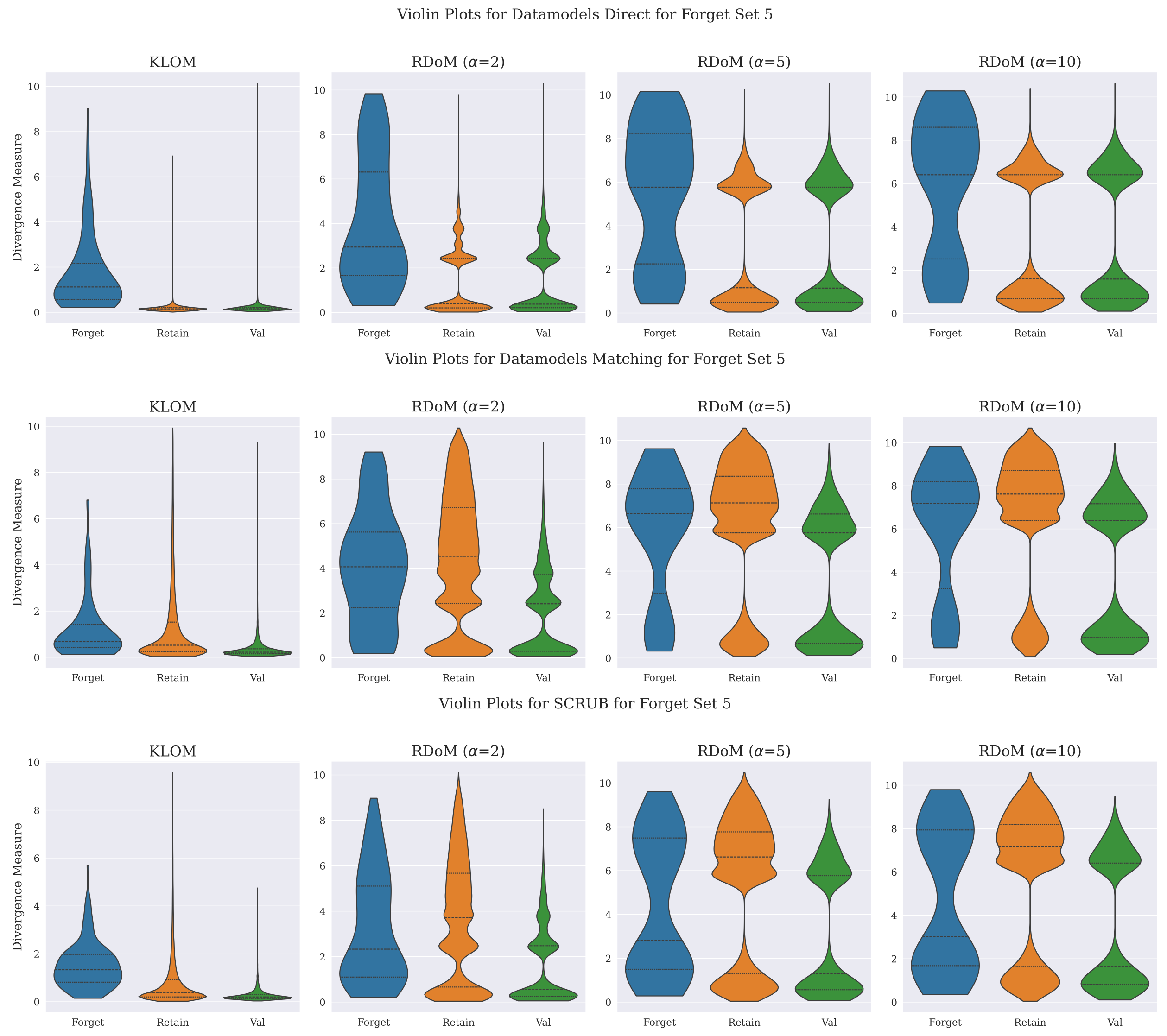


Figure F.6: Three examples of `RDoM` plots for different methods on forget set 5 (a non-random forget set of size 100).

Observe that as the value of α increases we observe a bifurcation in the `RDoM` scores produced. This emerges from the definition of Renyi divergence. Observe that for a specific value z, z^α grows monotonically as a function of α for $z > 1$ and decreases monotonically for $0 < z < 1$. Thus, for the same distributions P and Q the value of $\left(\frac{P(x)}{Q(x)}\right)^\alpha$ grows larger for all x, bifurcates as a function of α depending on if the Renyi divergence integral is dominated by values less than 1 or values larger than 1.

In practice, this has little effect on the conclusions drawn from our methods, as the non-linear effects of changing the choice of α for different values of `RDoM` (and `KLoM`) seem to affect all unlearning methods approximately equally.

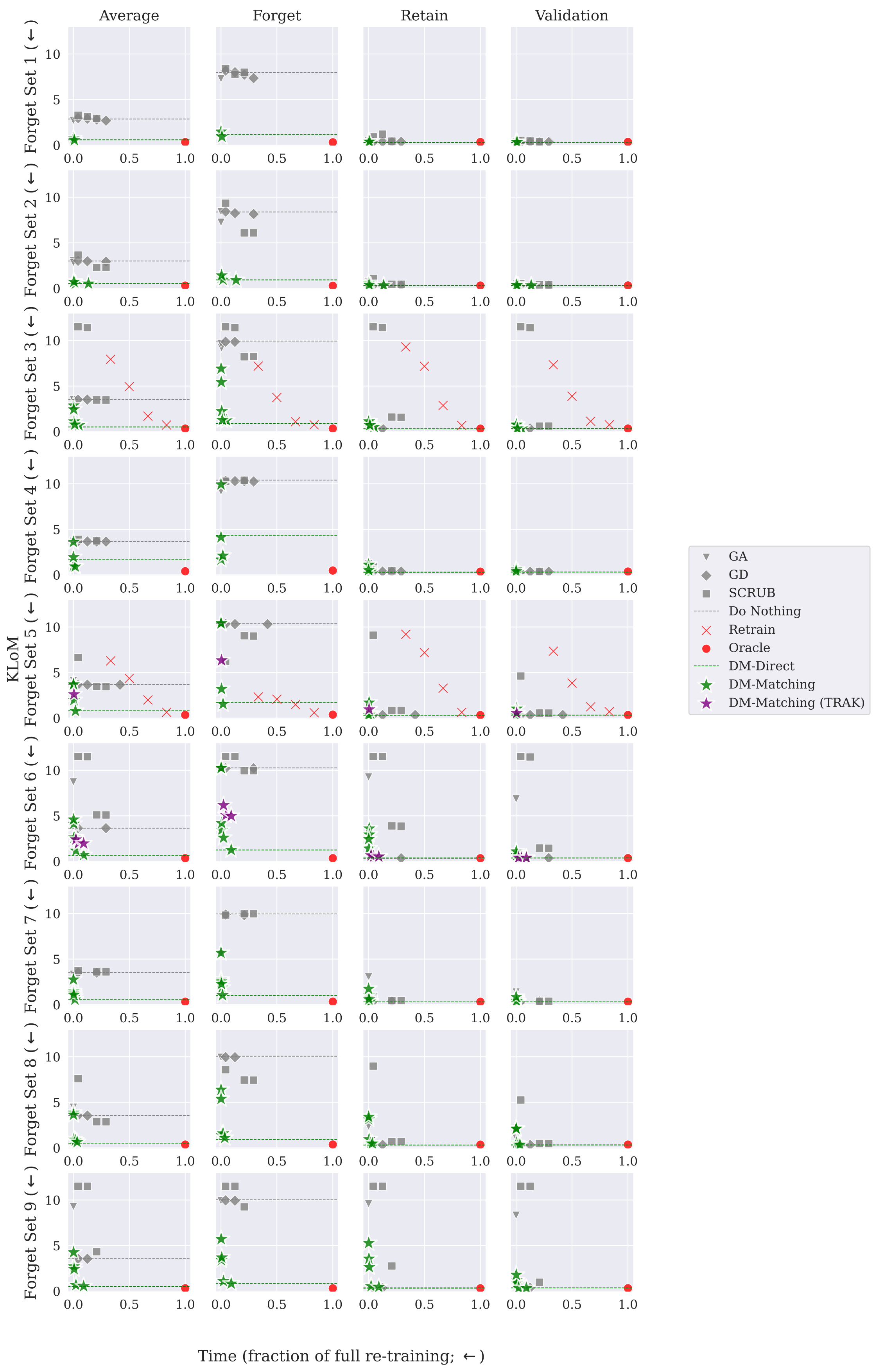


Figure F.3: KLoM results for all forget sets 1-9 on CIFAR-10. The pareto frontier for each method is in a line plot, but each KLoM data point for each method is plotted.

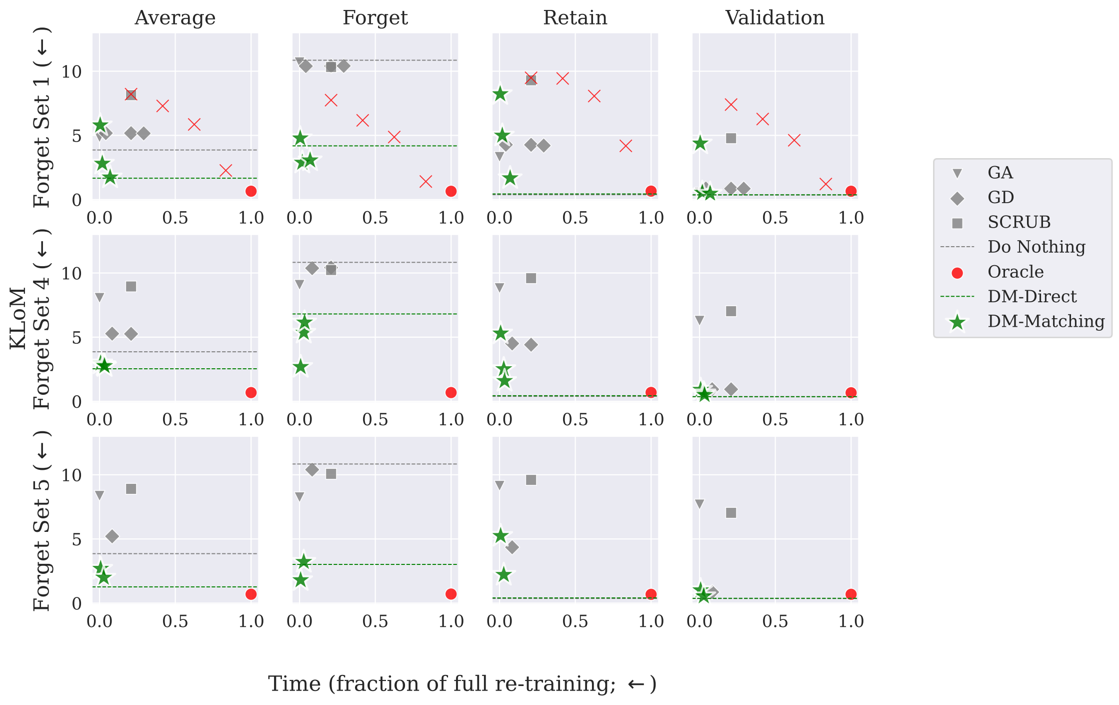


Figure F.4: `KLoM` results for all forget sets 1-3 on Living-17. The pareto frontier for each method is in a line plot, but each `KLoM` data point for each method is plotted.