Attribute-to-Delete: Machine Unlearning via Datamodel Matching

Presenter: Ting-Wei Li

11/7/2024

Outline

- **Topics**: Machine Unlearning (MU), Data Attribution (DA)
  - Definition and Evaluation of MU
  - Definition of DA
- **Background**: *Datamodels: Predicting Predictions from Training Data*[1]
- **Paper**: *Attribute-to-Delete: Machine Unlearning via Datamodel Matching*[2]
  - Motivation
  - Datamodels
  - MU algorithm: DMM (Datamodel Matching)
  - Evaluation
  - Future Directions

[1] Ilyas, Andrew, et al. "Datamodels: Understanding predictions with data and data with predictions." International Conference on Machine Learning. PMLR, 2022.
[2] Georgiev, Kristian, et al. "Attribute-to-Delete: Machine Unlearning via Datamodel Matching." arXiv preprint arXiv:2410.23232 (2024).

Notations

- $S \in \mathcal{X}^n$ be a fixed training dataset (with size n) drawn from an example space $\mathcal{X}$ (data universe)
- a *learning algorithm* $\mathcal{A}: \mathcal{X}^* \to \Theta$ as a (possibly random) function mapping from **datasets** to **a ML model** $\theta \in \Theta$ (parameter universe)
- for any example $x \in \mathcal{X}$, we use $f_x: \Theta \to \mathbb{R}^k$ to denote a *model evaluation* on x. (i.e. an **example-specific output function** where model param is a variable)
- Focus on *classification problem* for this paper!
- f_x can be k-dimensional logits or $\boxed{\text{margins}}$ of a classifier.

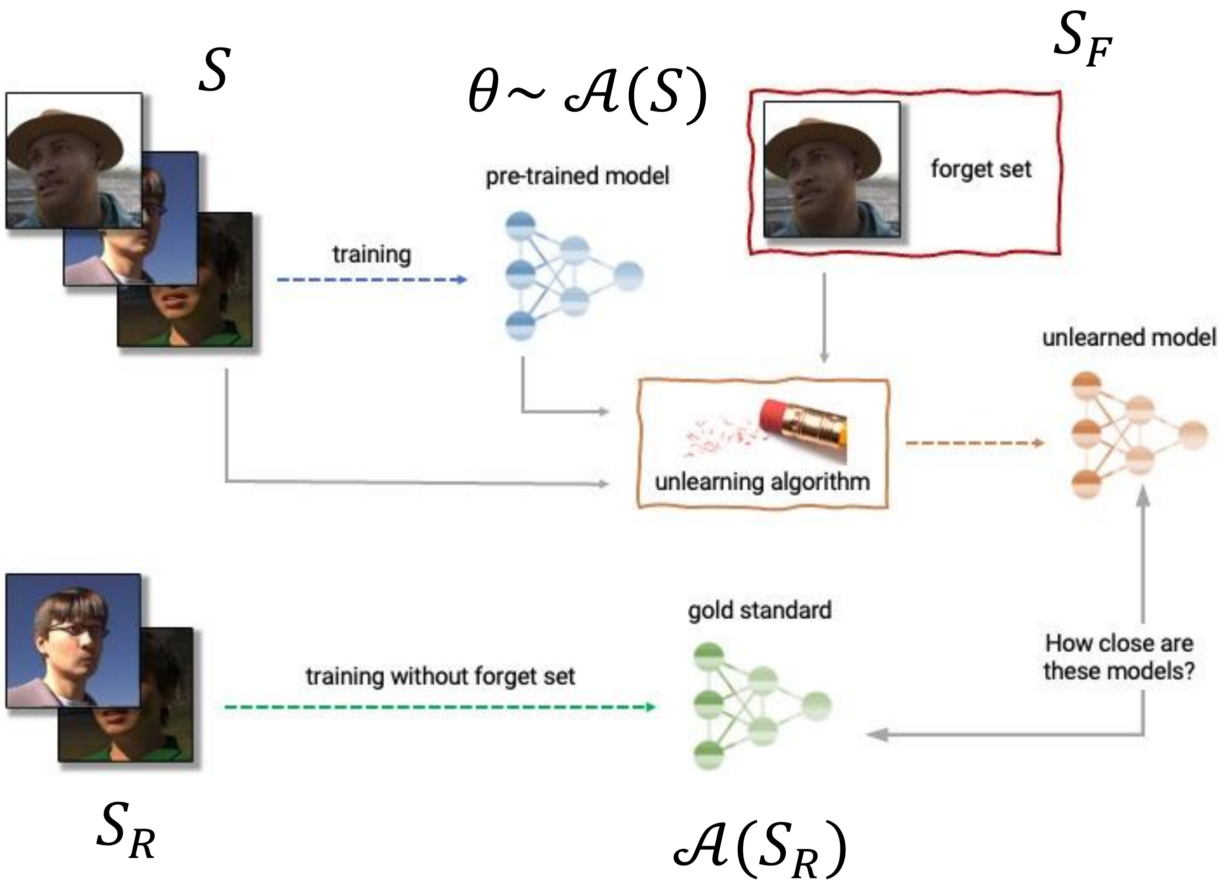
Can be $\log \frac{\mathrm{p}}{1-p}$ or the logit for the correct class minus the second highest class

Machine Unlearning (MU)

- Given (1) a ML model $\theta \sim \mathcal{A}(S)$ trained on dataset S and (2) a *forget set* $S_F \subset S$ (also, a corresponding *retain set* $S_R = S \setminus S_F$).

- **We want**: a model sampled from $\mathcal{A}(S_R)$ starting from the trained model θ.

- A MU algorithm $\mathcal{U}: \Theta \times 2^{|S|} \to \Theta$ takes a learned model θ, training dataset S, and the forget set $S_F \subset S$ as input and output a "unlearned model" that is *indistinguishable from a model trained on S_R*.



Approximate Formulation for MU

Some "memberships"

- Definition 1 ((ϵ,δ)-unlearning.[4]) a MU algorithm $\mathcal{U}$ is an (ϵ,δ)-<u>approximate unlearning algorithm</u> if, for all $\boxed{\mathcal{O}} \subset \Theta, S_F \subset S$, we have that

The unlearned model that forgets S_F from trained model $\theta \sim \mathcal{A}(S)$

$$Pr\left[\boxed{\mathcal{U}(\mathcal{A}(S),S_F)} \in \mathcal{O}\right] \le e^{\epsilon}\,\Pr\left[\boxed{\mathcal{A}(S_R)} \in \mathcal{O}\right] + \delta$$
$$Pr\left[\mathcal{A}(S_R) \in \mathcal{O}\right] \le e^{\epsilon}\,\Pr\left[\mathcal{U}(\mathcal{A}(S),S_F) \in \mathcal{O}\right] + \delta$$

The **golden** model that is trained on S_R from scratch

- Intuition: the distributions of unlearned models are (ϵ,δ) - indistinguishable from retrained models (**in parameter space!)**
- A hint for a *direction evaluation metric* for MU

[4] Neel, Seth, Aaron Roth, and Saeed Sharifi-Malvajerdi. "Descent-to-delete: Gradient-based methods for machine unlearning." Algorithmic Learning Theory. PMLR, 2021.

A Direct Evaluation for MU

Similar to the golden standard model but more broad

- Compute *the divergence of* model outputs between unlearned models and **safe models** (not compare them in the parameter space!)
- Safe models ($safe(S_F)$): some models that have not been trained on S_F.

The unlearned model that forgets S_F from trained model θ

- Formally, we want $\Delta_\delta\left(\boxed{\mathcal{U}(\mathcal{A}(S), S_F)}, safe(S_F)\right) \leq \epsilon$, where Δ_δ is a δ-approximate divergence and $safe(S_F)$ is a distribution of safe models w.r.t. forget set S_F.

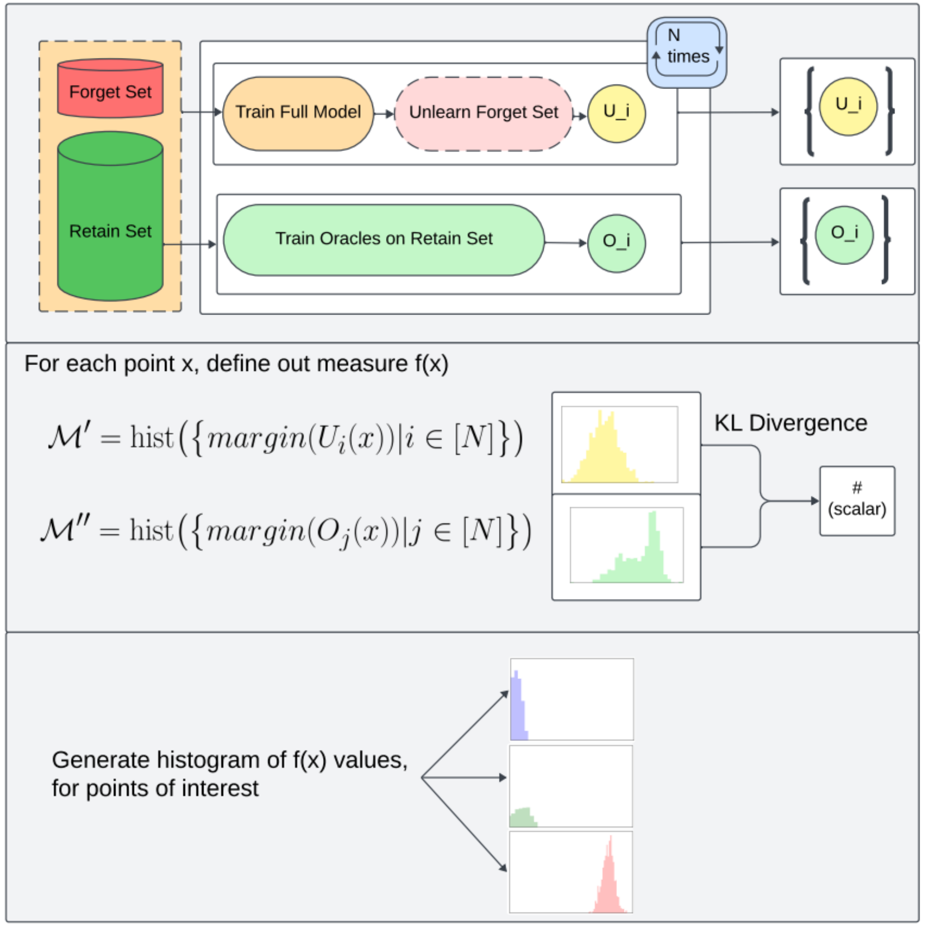
KLoM: A Direct Evaluation

- We thus have the definition of *KL divergence of margins* (KLoM) that (1) uses Δ=KL divergence, (2) allows arbitrary safe(S_F) and (3) studies distributions of model outputs (margins) f_x rather than parameters.
- Definition 2: (*KL divergence of margins (KLoM)*.) For an unlearning algorithm $\mathcal{U}$, reference distribution safe(S_F), and input x, *KLoM* is given by

$$KLoM(\mathcal{U}) \coloneqq D_{KL}\left(safe(S_F), f_x(\mathcal{U}(\mathcal{A}(S), S_F))\right).$$

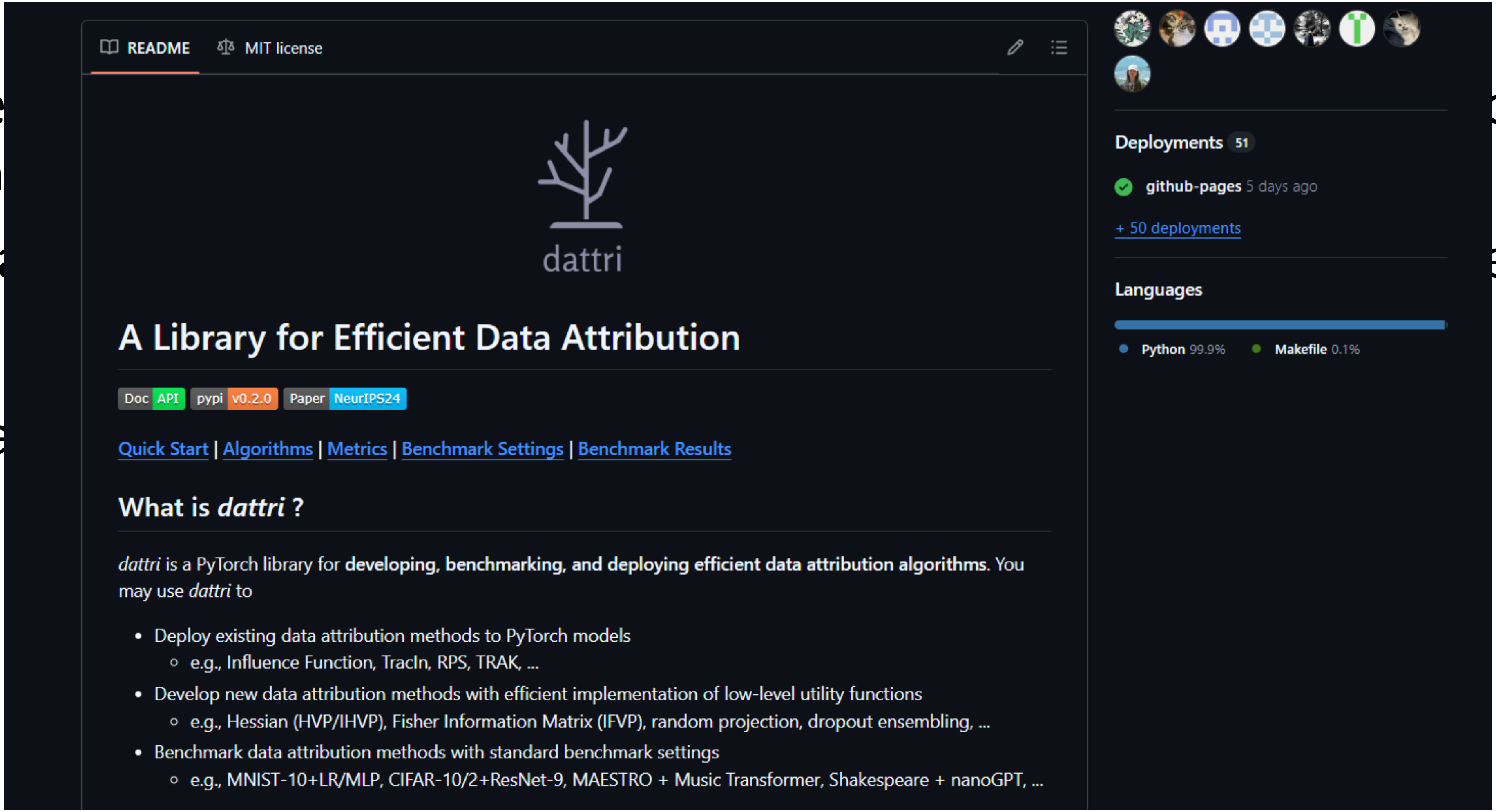
- Mostly, we just take safe$(S_F) \coloneqq \mathcal{A}(S \setminus S_F)$. And we will compute KLoM for data inputs x from S_F, S_R to evaluate MU algorithm.
- Intuitively, KLoM should be small for inputs x from all sets.

KLoM: A Direct Evaluation



Predictive Data Attribution

- The ce of each trainin
- Or wh et?
- Popula
- A rece



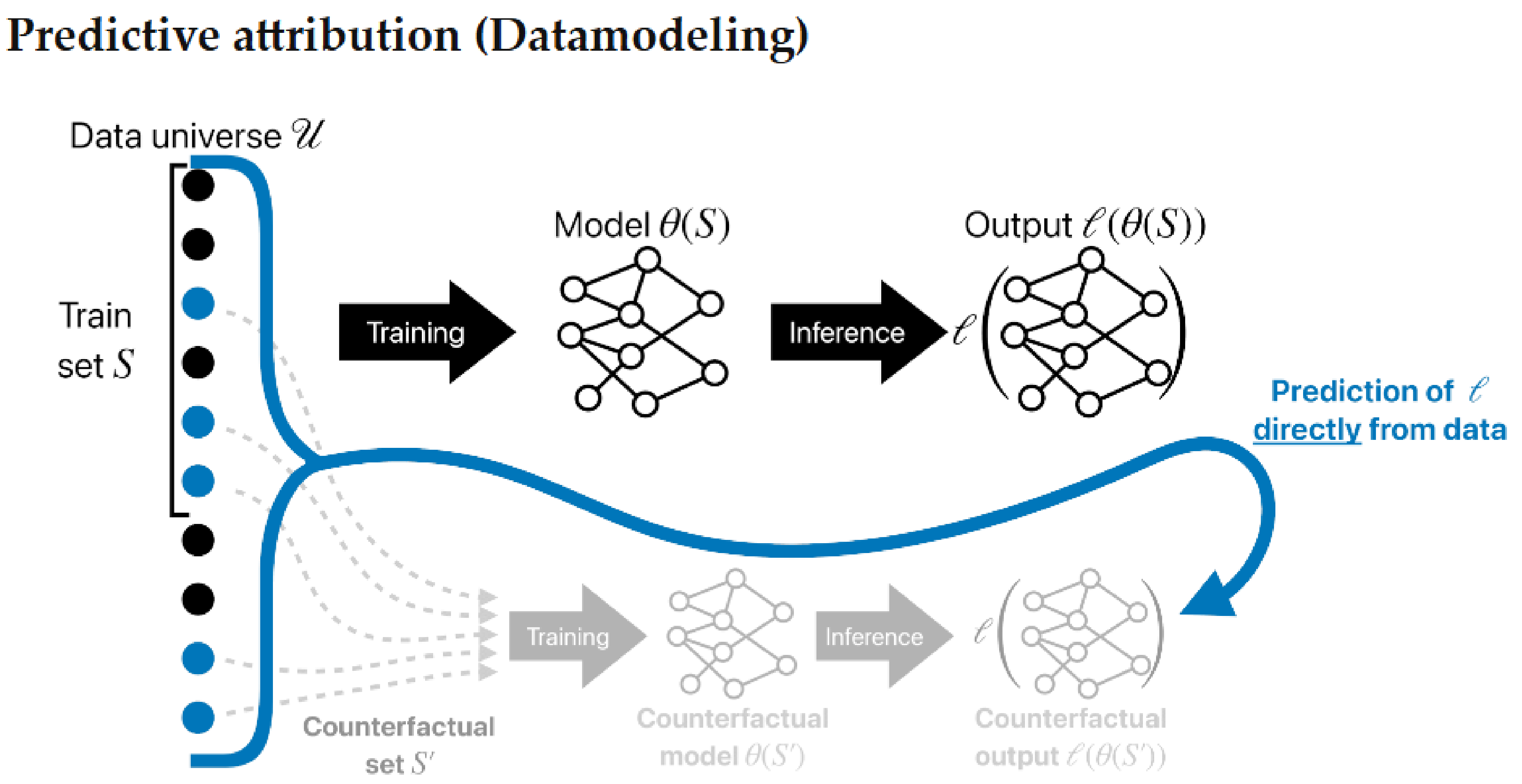
[6] https://ml-data-tutorial.org/chapter-1 (ICML 2024 tutorial "Data Attribution at Scale.)
[dattri] Deng, Junwei, et al. "$\texttt {dattri} $: A Library for Efficient Data Attribution." arXiv preprint arXiv:2410.04555 (2024). https://github.com/TRAIS-Lab/dattri

Predictive Data Attribution: Datamodeling

- Goal: compute an estimator (or *datamodel*[1]) that takes a training set as input and predicts the behavior of a ML model (trained on this training set) as output.
- Formally, 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 $S' \subset S, \hat{f}(S') \approx f_x(\mathcal{A}(S'))$.
- Intuition: *datamodel* takes an indicator vector representing training indexes as input and output *approximate* model evaluation on x , where this model can be trained on arbitrary subset of the training set.

[1] Ilyas, Andrew, et al. "Datamodels: Understanding predictions with data and data with predictions." International Conference on Machine Learning. PMLR, 2022.

Predictive Data Attribution



Delving deeper into Datamodels

- Surprisingly, *linear datamodels* work well to accurately predict modern complex ML/DL model behaviors.
- Formally, for an example x , we want to compute *a coefficient vector* $\beta \in \mathbb{R}^{|S|}$ such that, for subsets $S' \subset S$,

$$\boxed{\hat{f}(S')} := \beta^T \mathbf{1}_{S'} = \sum_{i \in S'} \beta_i \approx \boxed{f_x\big(\mathcal{A}(S')\big).}$$

Datamodel output on S'

output of x evaluated on the model trained on S'

- Intuition: For each example x, each training sample in S has its own "importance". For a specific learning algorithm $\mathcal{A}$, the model evaluation of x can be seen as a **summation over data importance used for training**.

$\mathbf{1}_{S'} \in \mathbb{R}^{|S|}$ is a binary vector indicating the subset S'.

How to learn Datamodels

- We first sample many subsets $S_1, \cdots, S_k$ and use $\mathcal{A}$ to train corresponding model.
- We then compute the model evaluation of x on these models (i.e. $f_x\left(\mathcal{A}(S_i)\right), for\ i \in [k]$). Then we simply solve the following *regularized regression problem:*

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

where $\mathbf{1}_{S_i} \in \mathbb{R}^{|S|}$ is a binary vector indicating the subset S_i.

Motivation: The Missing Targets Problem

For complex models, may simple MU algorithm work?

- *Gradient Descent* on retain set may not alter forget set predictions much
- If choose to *increase* loss on forget set (*Gradient Ascent*)

-> loss for forget set points might overshoot or undershoot since we don't know the expected loss under a perfectly retrained model (i.e. unlearn too much or not enough)

Motivation: Case Study on SCRUB

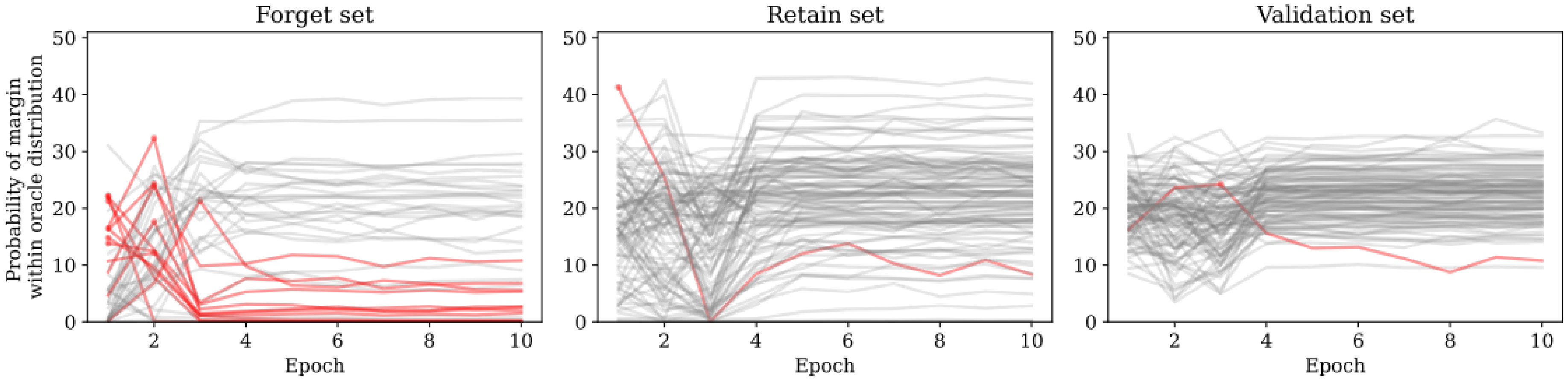


Figure 2: **The missing targets problem.** We apply the SCRUB [KTT24] algorithm to unlearn a forget set of CIFAR-10, and measure how well different (random) points are unlearned over time. To quantify how well a given point x is unlearned, we fit a Gaussian distribution to the outputs of oracle models at x, and compute the likelihood of the average outputs from unlearned models under this distribution. We track this likelihood (y-axis) for random points across the duration of unlearning algorithm (x-axis). 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.

[7] Kurmanji, Meghdad, et al. "Towards unbounded machine unlearning." Advances in neural information processing systems 36 (2024).

Oracle Matching (OM) Algorithm

- Question: Given access to **sample outputs from the oracle model** (re-trained without the forget set, i.e. $f^{oracle}(x) \coloneqq f_x(\mathcal{A}(S_R))$),
- can we efficiently fine-tune an existing model (trained on the full dataset) to **match** the outputs of the sample?
- Main point: *whether gradient-based optimization can match predictions of the oracle*.

Oracle Matching (OM) Algorithm

Algorithm 1 Oracle Matching (OM)

Input: Trained model θ; oracle predictions $f^{\text{oracle}}(x)$; fine-tuning retain set size r
Output: Unlearned model θ_{UL}
Initialize $\theta_0 = \theta$
for $t = \{0, ..., T-1\}$ **do** ▷ T epochs
    $S'_R \leftarrow S \setminus S_F$ ▷ Sub-sample r points from retain set
    $S_{\text{fine-tune}} = S_F \bigcup S'_R$ ▷ Combine with forget set
    for $x \sim S_{\text{fine-tune}}$ **do**
        $L(\theta_t) = \|f_x(\theta_t) - f^{\text{oracle}}(x)\|^2$ ▷ Compute MSE loss
        $\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta L(\theta_t)$ ▷ Perform update with gradient
    end for
end for
Return Model $\theta_{UL} = \theta_T$

Evaluating OM

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).

Implications of OM

- OM can constantly outperform other baselines in terms of KLoM scores and running time.
- Observation 1: in parameter space, there is indeed θ' that is close to the oracle retrained without the forget set.
- Observation 2: fine-tuning θ can quickly converge to that θ' with small number of samples of oracle outputs.
- But OM is not a practical algorithm.

- Predictive DA (our favorite *Datamodel*) comes to the rescue!

Efficient Proxy of Oracles: Datamodels

- Intuition: let us replace the oracle models by *datamodels*!
- Formally, for each input x, we replace $f^{oracle}(x)$ by $\hat{f}_x$, which is the *datamodel* of x.
- Reminder: a *datamodel* maps arbitrary data subset to model output on input x, i.e., $\hat{f}_x(S \setminus S_F) \approx f_x\big(\mathcal{A}(S \setminus S_F)\big) = f_x\big(\mathcal{A}(S_R)\big) \coloneqq f^{oracle}(x)$.
- Note that if we focus on *linear datamodel*, then we have

$$\hat{f}_x(S \setminus S_F) \coloneqq \beta_0 + \sum_{i \in S \setminus S_F} \beta_i(x) = \left(\beta_0 + \sum_{i \in S} \beta_i(x)\right) - \sum_{i \in S_F} \beta_i(x) = f_x(\theta) - \sum_{i \in S_F} \beta_i(x).$$

- We don't need to approximate the first term; we know that in priori.

DM-Direct

- Thus, we now focus on the oracle output function:

$$h(x) := f_x(\theta) - \sum_{i \in S_F} \beta_i(x) .$$

- This can be a working MU method if we only care about prediction but not model weights. But still, we may (1) want to delete data from full model θ and (2) avoid training one datamodel for each new input z.

Algorithm A.2 DM-DIRECT

1: **Input:** Trained model θ; 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**

Datamodel Matching (DMM)

- One step further: let us replace the "true oracle" in OM algorithm by $h(x)$ through estimated datamodels (for all x we are interested in).
- Datamodels are estimated through a subset of retain points and the entire set of forget points.

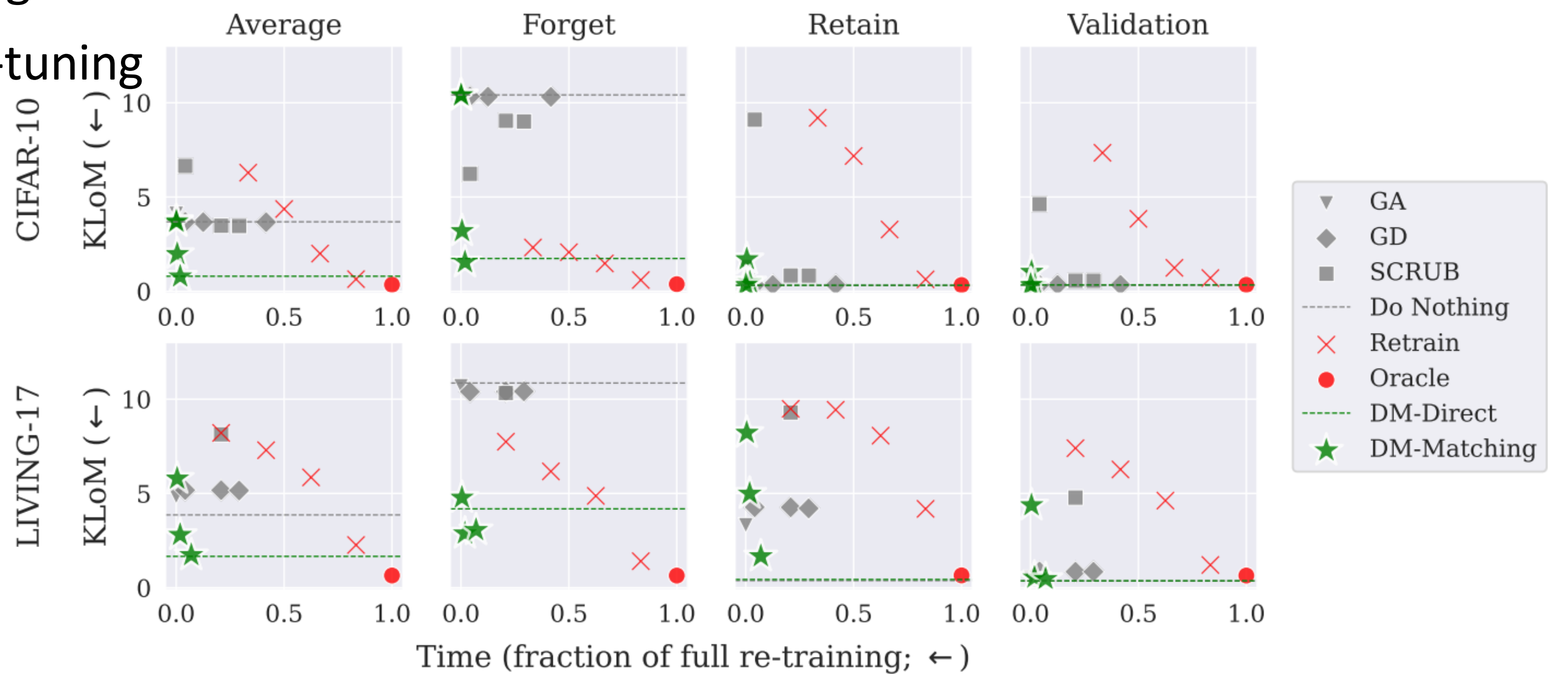
Datamodel Matching (DMM)

Algorithm A.3 Datamodel Matching (DMM)

Input: Trained model θ; datamodels $\beta(\cdot)$; fine-tuning set size r
Output: Unlearned model θ_{UL}
$S'_R \leftarrow S \setminus S_F$; $S_{\text{fine-tune}} = S_F \bigcup S'_R$
$\boxed{h \leftarrow \text{DM-DIRECT}(\theta, \beta, S_f)}$ ▷ Simulate oracles with datamodels
for $t = \{0, ..., T-1\}$ **do** ▷ T epochs
    for $x \sim S_{\text{fine-tune}}$ **do** ▷ mini-batch
        $L(\theta_t) = \|f_x(\theta_t) - \boxed{h(x)}\|^2$ ▷ Compute loss
        $\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta L(\theta_t)$ ▷ Perform update with gradient
    end for
end for
Return Model $\theta_{UL} = \theta_T$

Evaluation through KLoM

- GA: gradient ascent on Forget set starting from θ
- GD: gradient descent on Retain set starting from θ
- Re-train: re-train on Retain set starting from scratch
- DM-Direct: directly use $h(x)$, no fine-tuning
- DMM: use $h(x)$ and fine-tuning
- SCRUB[7]: previous SOTA



[7] Kurmanji, Meghdad, et al. "Towards unbounded machine unlearning." Advances in neural information processing systems 36 (2024).

Ablation of DMM (can be skipped)

- DMM consists of (i) oracle matching (the fine-tuning algorithm) and (ii) estimating datamodels (approximating oracle outputs)
- Let us study two components separately
  - OM: stability across unlearning time and generalization from small sample size
  - Datamodel: scaling with cost and efficient alternative (TRAK[8])

- Intuition: does DMM work as expected?

[8] Park, Sung Min, et al. "TRAK: Attributing Model Behavior at Scale." International Conference on Machine Learning. PMLR, 2023.

Stability: SCRUB vs OM

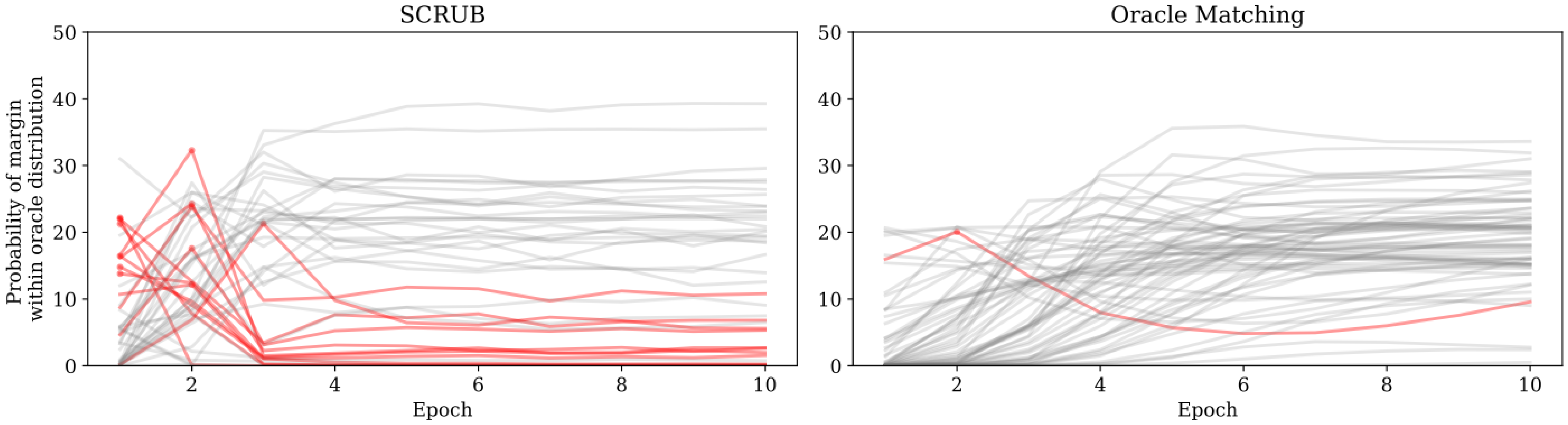


Figure 5: **Oracle matching circumvents the stopping time problem.** We revisit the earlier analysis for SCRUB (left) and apply the same analysis to Oracle Matching (right). The red lines highlight examples in the forget set whose unlearning quality is hurt by training longer. This "overshooting" happens frequently with SCRUB, but only rarely with Oracle Matching.

Generalization of OM

- Focus: the amount of forget/retain data used to derive “oracles”.

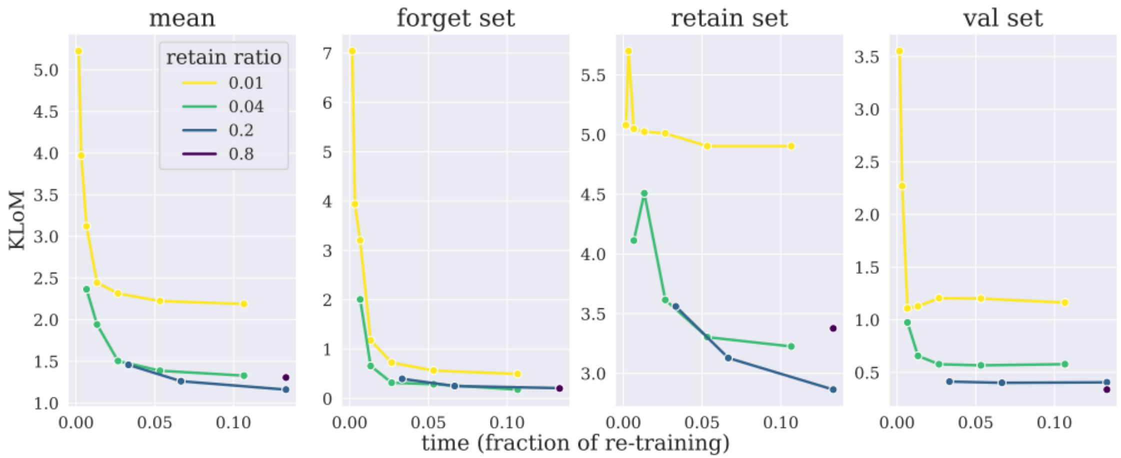


Figure 6: Varying the fraction of retain set (of the full dataset) sampled for oracle matching. A sufficiently large fraction (≥ 0.04) appears to be sufficient in enabling OM to generalize to out-of-sample. In other words, OM can “distill“ the oracle model using a small subset of the training data.

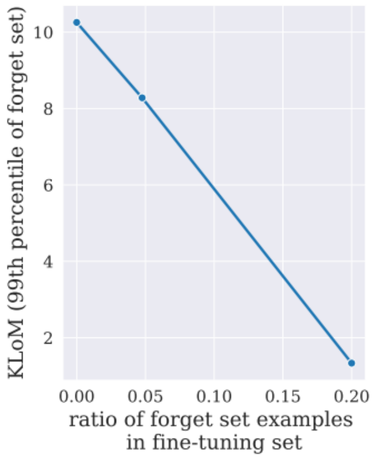


Figure 7: Higher ratio of forget set samples in the fine-tuning set for OM improves performance on the forget set.

Scaling costs for Datamodel

- Natural trade-off: computational cost vs datamodel predictiveness
- Observation: unlearning effectiveness will saturate even the datamodels are still improving
- Implication: efficient alternatives

might exist!

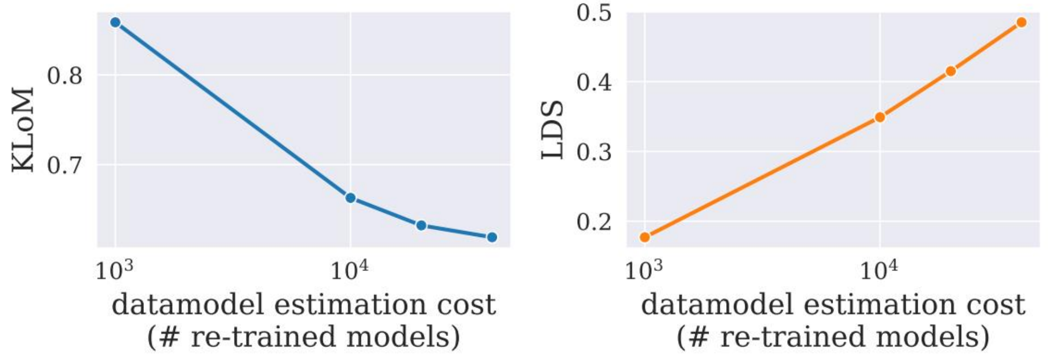


Figure 9: **The effect of estimation cost on unlearning performance and datamodel predictiveness.** On CIFAR-10, we show how unlearning performance of DM-DIRECT (measured by KLoM; lower is better) and datamodel predictiveness (measured by the linear datamodeling score; higher is better) scales with datamodel estimation cost (number of re-trained models, in $\{10^3, 10^4, 2\times10^4, 4\times10^4\}$). KLoM is averaged over different forget sets.

Efficient unlearning with TRAK

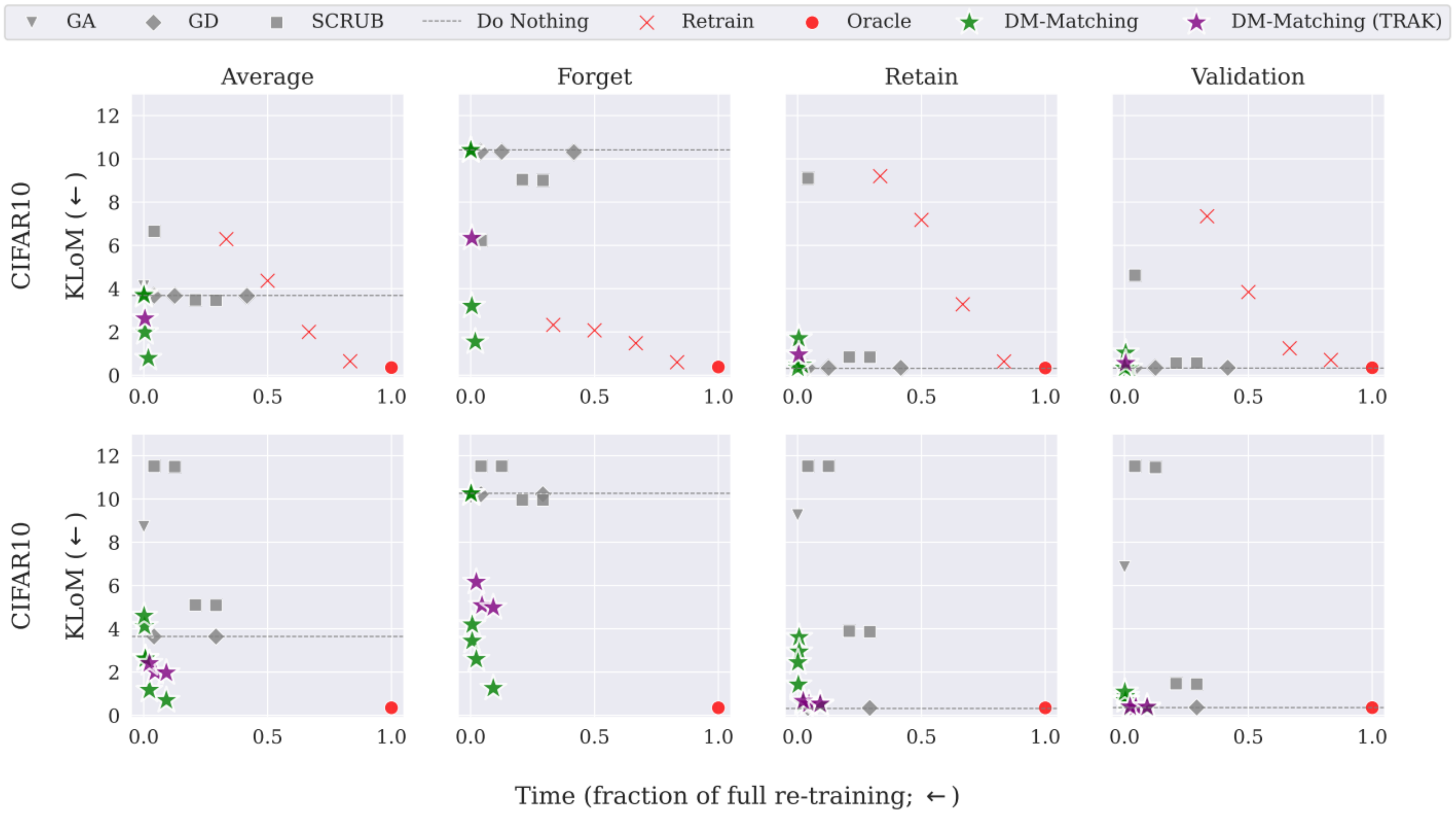
- TRAK[8]: a more efficient predictive DA algorithm compared to datamodels.

(1000x less computation time)

- Observation: natural trade-off again but here with cheaper alternative, DMM still outperform baselines.
- Note: no matter what kind of approximate oracle we use (datamodels or TRAK), this is a one-time cost.

[8] Park, Sung Min, et al. "TRAK: Attributing Model Behavior at Scale." International Conference on Machine Learning. PMLR, 2023.

Efficient unlearning with TRAK



Takeaways

- Challenge in machine unlearning: missing targets problem and efficiency
- Predictive data attribution model: help us connect data and complex model output through interpretable, simple modeling (*linear datamodel*)
- **A general framework for MU (DMM)**: fine-tuning current model to match approximate oracles, which is produced through predictive DA algorithm
- **A more direct evaluation for MU (KLoM)**: measure the difference in distributions of unlearned model's outputs from oracle

Future Directions

- Beyond classification problems
- Better understanding of OM – how to sample from retain set
- Better time efficiency – originated from DA algorithm
- More practical scenarios – multiple unlearning requests

Reference

- [1] Ilyas, Andrew, et al. "Datamodels: Understanding predictions with data and data with predictions." International Conference on Machine Learning. PMLR, 2022.
- [2] Georgiev, Kristian, et al. "Attribute-to-Delete: Machine Unlearning via Datamodel Matching." arXiv preprint arXiv:2410.23232 (2024).
- [3] Ginart, Antonio, et al. "Making ai forget you: Data deletion in machine learning." Advances in neural information processing systems 32 (2019).
- [4] Neel, Seth, Aaron Roth, and Saeed Sharifi-Malvajerdi. "Descent-to-delete: Gradient-based methods for machine unlearning." Algorithmic Learning Theory. PMLR, 2021.
- [5] Hayes, Jamie, et al. "Inexact unlearning needs more careful evaluations to avoid a false sense of privacy." arXiv preprint arXiv:2403.01218 (2024).
- [6] Deng, Junwei, et al. "$\texttt {dattri} $: A Library for Efficient Data Attribution." arXiv preprint arXiv:2410.04555 (2024).
- [7] Kurmanji, Meghdad, et al. "Towards unbounded machine unlearning." Advances in neural information processing systems 36 (2024).

Thank you!