Soft Weighted Machine Unlearning

Xinbao Qiao, Ningning Ding, Yushi Cheng, Meng Zhang

https://arxiv.org/abs/2505.18783

Setting

Consider a prediction task (regression problem) with:

- Input space $\mathcal{X}$;
- Output space $\mathcal{Y}$;
- Training set $\mathcal{D}^n = \{z_i = (\boldsymbol{x_i}, y_i)\}_{\{i=1\}}^n \subset \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$;
- Parameter $\theta \in \Theta := \mathbb{R}^d$;
- $f(\theta; x)$ estimator of $y|\boldsymbol{x}$;
- $\ell : \mathcal{Z} \times \Theta \to \mathbb{R}$ loss function (e.g., $\ell(z, \theta) \mapsto \|y - f(\theta; \boldsymbol{x})\|^2$).

We seek the empirical risk minimizer:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(z_i, \theta).$$

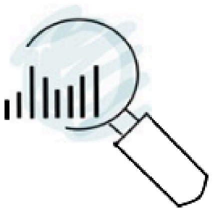
Goal

For some problems (data poisoning, fairness, robustness, etc.), it is desirable to unlearn the contribution of certain training data.

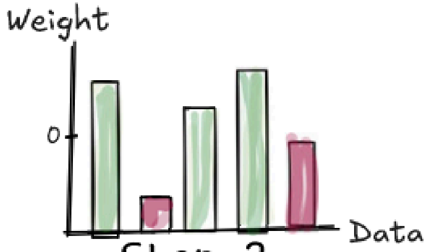
Challenges:

- Retraining the model from scratch is computationally expensive;
- The removal may decrement the performance of the model on the remaining data.

Solution: Use influence functions with soft weights.



Step1:
Influence Evaluation



Step 2:
Data Reweighting

	Step1: Influence Evaluation	Step 2: Data Reweighting
Hard	1. Compute each sample's influence on the target task	Sample z: ✗ ✗ ✓ ••• ✓ ✓ Weight ϵ: -1 -1 0 0 0 Select the top-k detrimental samples on target task, and set weights to -1 (unlearn).
Soft	1. Compute each sample's influence on the target task 2. Compute each sample's influence on the utility.	Sample z: ✗ ✗ ✓ ••• ✓ ✓ Weight ϵ: -1.2 -0.3 0.1 0.8 1.1 Optimize a set of weights based on the influence on the utility and the target task.

Influence Functions

Given $z_j \in \mathcal{D}^n$ and $\varepsilon_j \in \mathbb{R}$, we can define:

$$\hat{\theta}(z_j, \varepsilon_j) = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(z_i, \theta) + \varepsilon_j \ell(z_j, \theta).$$

The *influence function* relative to z_j is:

$$\mathcal{I}(z_j) = \frac{d}{d\varepsilon_j} \hat{\theta}(z_j, \varepsilon_j) \Big|_{\varepsilon_j = -1}$$

Interpretation: It indicates how the training error would change if we removed the training data z_j.

1. If ℓ is at least C^2, we can write the influence function in a closed form:

$$\mathcal{I}\big(z_j, -1\big) = -\frac{1}{n} H_{\hat{\theta}}^{-1} \nabla_\theta \ell\big(z_j, \hat{\theta}\big)$$

where H is the Hessian matrix of the empirical risk at $\hat{\theta}$.

2. The influence function is a *first-order approximation* of the change in the estimator when we remove z_j from the training set.

In fact, if we do the *Taylor expansion* of $\hat{\theta}$ at $\varepsilon_j = -1$, we have:

$$\hat{\theta} = \hat{\theta}\big(z_j, -1\big) + \mathcal{I}\big(z_j\big) + O\big(\varepsilon_j^2\big).$$

We will consider the influence on these 3 metrics (negative influence is better):

- **Utility:** $\mathcal{I}_{\text{util}}(z_j, -1) = \sum_{z \in \mathcal{T}} \nabla_\theta \ell\left(z, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell\left(z, \hat{\theta}\right)$, where $\mathcal{T}$ is the validation set;
- **Fairness:** $\mathcal{I}_{\text{fair}}(z_j, -1) = \nabla_\theta f_{\text{fair}}\left(\mathcal{T}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell\left(z, \hat{\theta}\right)$, with f_{fair} a fairness metric (e.g., demographic parity);
- **Robustness:** $\mathcal{I}_{\text{robust}}(z_j, -1) = \sum_{\tilde{z} \in \tilde{\mathcal{T}}} \nabla_\theta \ell\left(\tilde{z}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell\left(z, \hat{\theta}\right)$, for a perturbed dataset $\tilde{\mathcal{T}}$.

Algorithm 1: Soft-Weighted Unlearning Framework

Input: Model $\hat{\theta}$, Training Dataset $\mathcal{D}$, Validationa and Testing Dataset $\mathcal{T}$, Adversarial Samples $\tilde{z} \in \tilde{\mathcal{T}}$

 # Step 1: Influence Evaluation.

for *each sample* $z_i \in \mathcal{D}$ **do**

  Evaluate influence of z_i on validation set;

  Utility: $\mathcal{I}_{\text{util}}(z_i; -1) \leftarrow$ Equation (3).

  Fairness: $\mathcal{I}_{\text{fair}}(z_i; -1) \leftarrow$ Equation (4).

  Robustness: $\mathcal{I}_{\text{robust}}(z_i; -1) \leftarrow$ Equation (5).

end

 # Step 2: Weights Optimization.

Weights $\{\epsilon_i^*\}_{i=1}^n \leftarrow$ Equation (7)

 # Step 3: Model Correction.

if $f \leftarrow f_{fair}(\mathcal{T}; \theta)$ *or* $\sum_{\tilde{z} \in \tilde{\mathcal{T}}} \nabla_\theta \ell(\tilde{z}; \theta) \leq \delta$ **then**

  $\theta \leftarrow$ Equation (8) or Other Unlearning Algorithms

end

Output: θ

Soft weights

Instead of a binary decision (keep or remove), we want to assign a customized weight to each training data.
Namely, we want to find $\varepsilon^\star = (\varepsilon_1, ..., \varepsilon_n)$ and use it for the one-shot updating rule:

$$\hat{\theta}(\mathcal{D}, \varepsilon^\star) \approx \hat{\theta} + \varepsilon^\star H_\theta^{-1} \left(\nabla_\theta \ell\left(z_1, \hat{\theta}\right), ...\nabla_\theta \ell\left(z_n, \hat{\theta}\right)\right)^\top.$$

I.e., we update each training data with a re-scaled loss function

$$\varepsilon_i \mathcal{J}(z_i, -1) = \mathcal{J}(z_i, \varepsilon_i).$$

Weight computations

We need weights to be easy to retrieve, since they must be computed for each training data.

They are defined as the solution of the following optimization problem:

$$\min_{\boldsymbol{\varepsilon} \in \mathbb{R}^n} \sum_{i=1}^{n} \mathcal{I}_{\text{fair}}(z_i, \varepsilon_i) + \lambda \, \|\boldsymbol{\varepsilon}\|_2^2$$

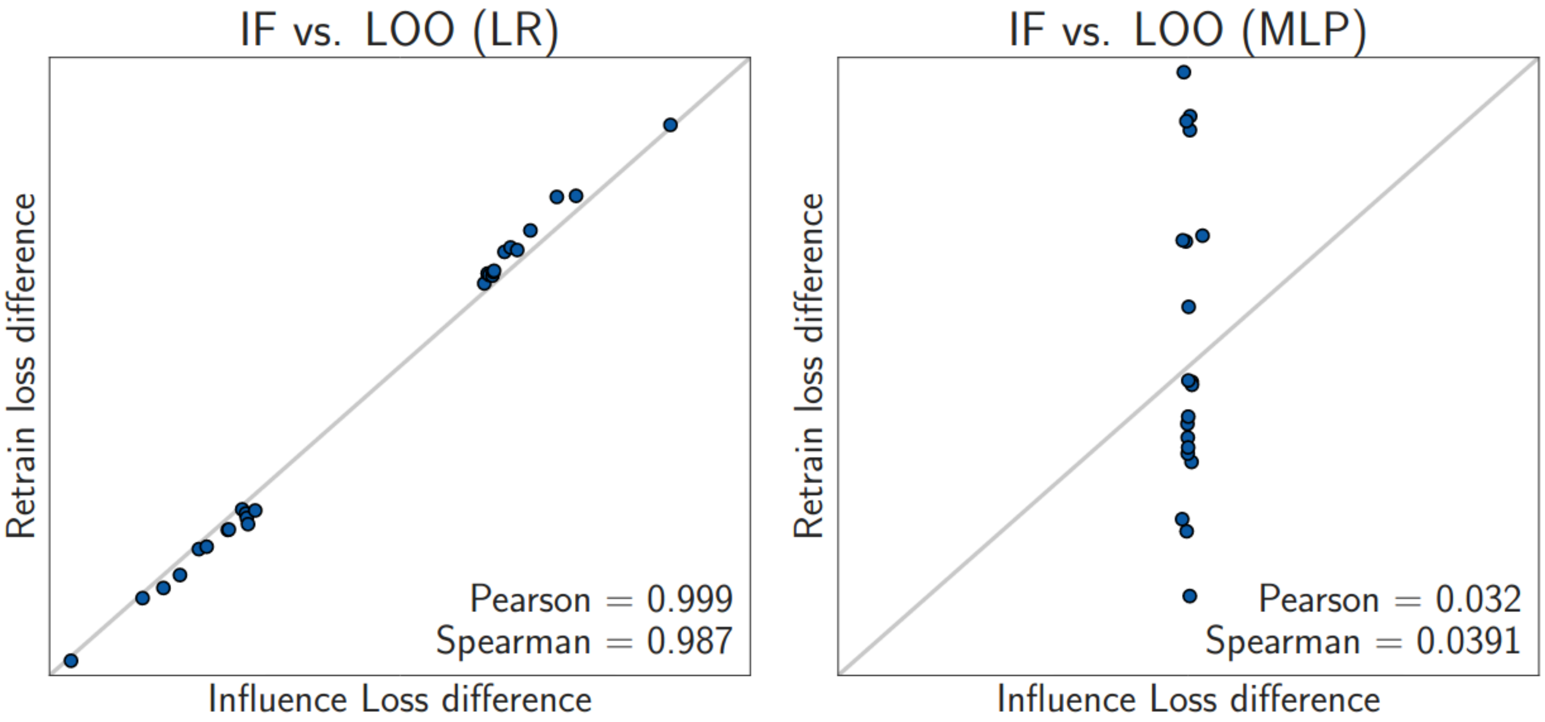
(quadratic problem) subject to:

$$\sum_{i=1}^{n} \mathcal{I}_{\text{fair}}(z_i, \varepsilon_i) \geq -f_{\text{fair}}\left(\mathcal{T}; \hat{\theta}\right), \quad \sum_{i=1}^{n} \mathcal{I}_{\text{util}}(z_i, \varepsilon_i) \leq 0$$

Main issues

- We assume to know $\hat{\theta}$ so that $H_{\hat{\theta}}$ is PD, thus invertible. In practice we only have an approximation;
- Computing the inverse of the hessian is expensive. We must use approximations;
- Such approximations don't work well for neural networks.

Bad approximation



[“If Influence Functions are the Answer, Then What is the Question?”, J. Bae, N. Ng, A. Lo, M. Ghassemi, R. Grosse, 2022]

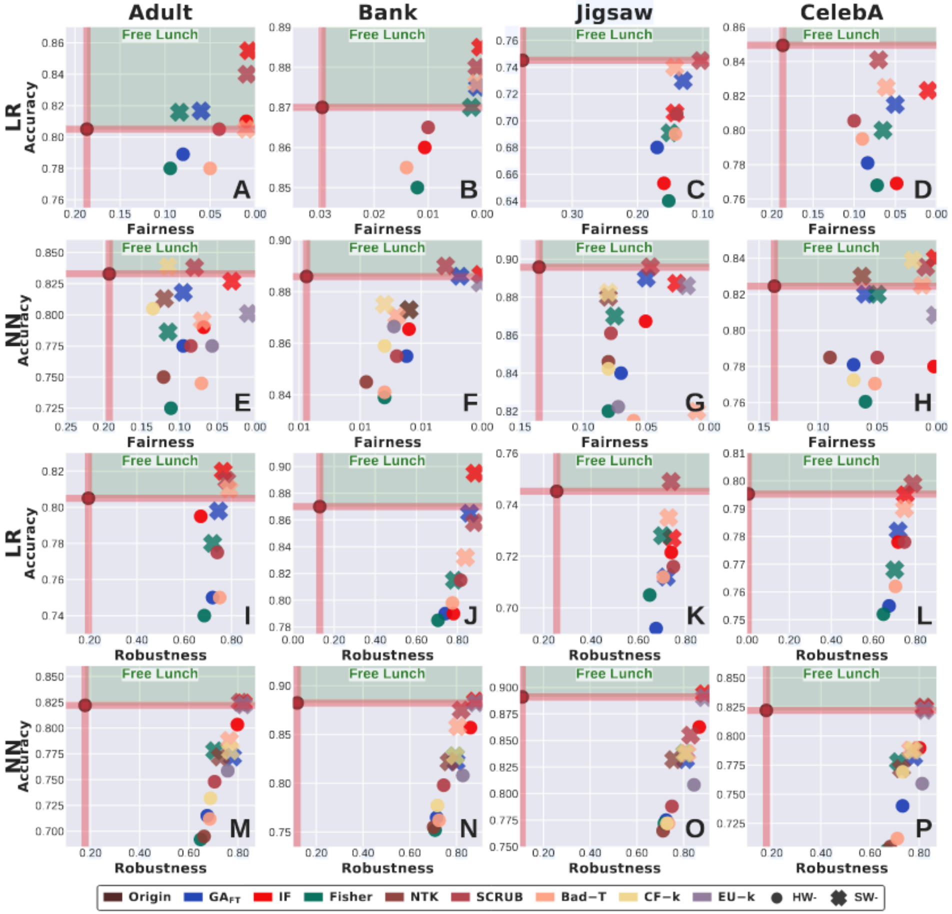
A solution

1. Add a regularization term to the loss function. This way, $H_{\hat{\theta}}$ is guaranteed to be PD.
2. Approximate the hessian with σI. This way, we have:

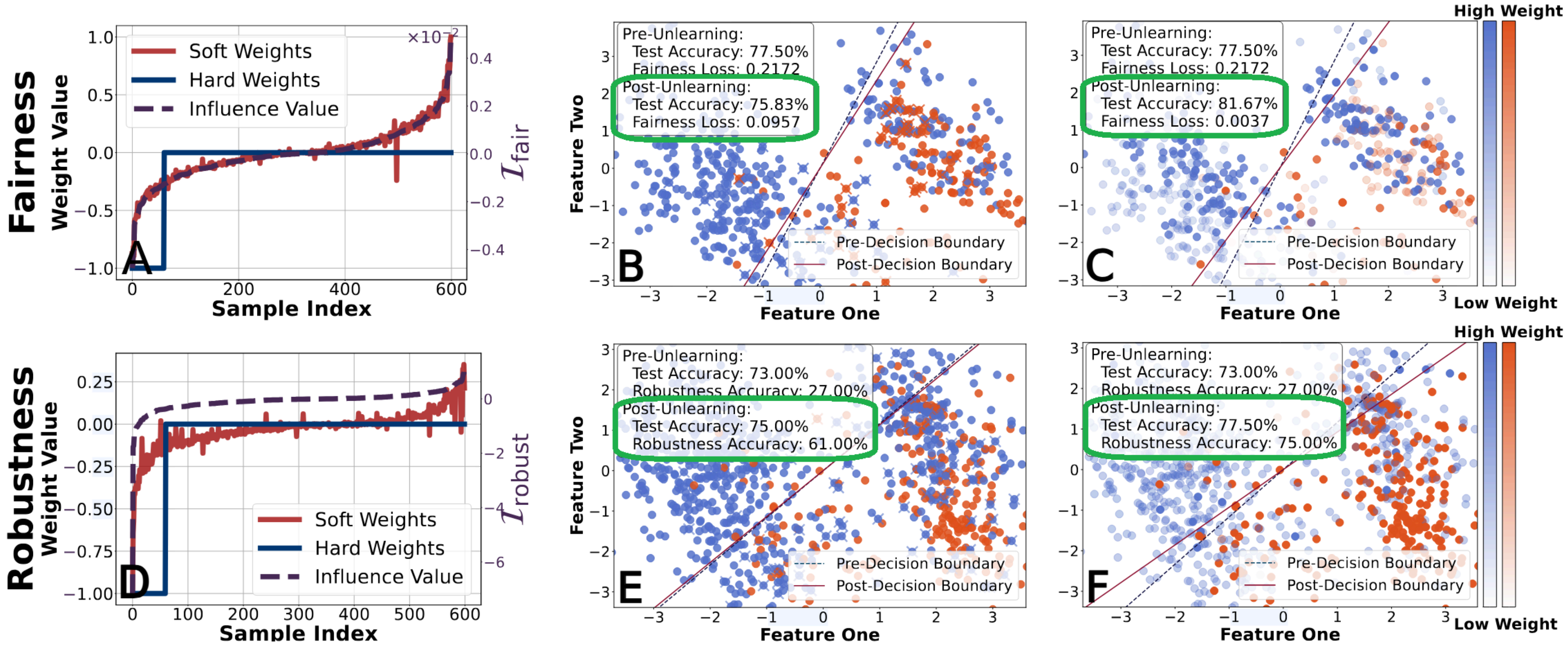
$$\hat{\theta}(z_j, \varepsilon_j^\star) - \hat{\theta} \approx \varepsilon_j^\star \frac{\sigma}{n} \nabla_\theta \ell\left(z_j, \hat{\theta}\right).$$

If we see $\frac{\sigma}{n}$ as a learning rate, we can index it by time and use also others unlearning algorithms instead of IFs to do the update in an iterative fashion(?).

Numerical evidence



Numerical evidence



Conclusion

Take-away

Using some smart approximations, we can adapt all the existing unlearning algorithms to the soft weighted setting.
This family of methods performs better than the HW version for non-privacy applications and can also improve the performance of the model on the remaining data.

[“Soft Weighted Machine Unlearning”, X. Qiao, N. Ding, Y. Cheng, M. Zhang, 2025]