Weekly meetings

If Influence Functions are the Answer, Then What is the Question?

Setting

Consider a prediction task (regression problem) with:

- Input space $\mathcal{X}$;
- Output space $\mathcal{Y}$;
- Training set $\mathcal{D}^n = \{z_i\}_{i=1}^n$ where $z_i = (x_i, y_i)$ for all $i = 1, \ldots, n$; $X = (x_1, ..., x_n)$, $Y = (y_1, ..., y_n)$;
- Parameter $\theta \in \Theta := \mathbb{R}^d$;
- $f(\theta; x)$ estimator of $\mathcal{Y} \mid \mathcal{X}$;
- Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ (e.g., $\ell(y', y) \mapsto \|y' - y\|^2$).

We aim to minimize the training error (empirical risk):

$$L(\theta; \mathcal{D}^n) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(f(\theta; x_i),\, y_i\big).$$

Call

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} L(\theta; \mathcal{D}^n).$$

How different is it from

$$\hat{\theta}_{\varepsilon, -z} = \arg\min_{\theta \in \Theta} \big(L(\theta; \mathcal{D}^n) - \varepsilon\, \ell(f(\theta; x), y)\big)$$

when $\epsilon = 1/n$ and $z = z_{\bar{i}}$ for some $i \in [n]$?
To answer this question, we can re-train the whole model on $\mathcal{D}^n \setminus \{z\}$ (Leave-One-Out method), or use influence functions.
In the context where the influence functions are well defined, they are a powerful tool. However, when applied to multi-layer perceptrons, for example, their capability of approximating the effect of the LOO decreases drastically. This paper presents a new point of view: IFs do not approximate the LOO retraining, but instead the effect of another method they present, called PBRF.

Influence functions

Definition 1. Given $(\bar{x}, \bar{y}) = \bar{z} \in \mathcal{D}^n$, the *influence loss difference* relative to $\bar{z}$ is

$$\mathcal{Q}(\bar{z}; \hat{\theta}) = \frac{d}{d\varepsilon} \left[\ell(f(\hat{\theta}_{\varepsilon,-z}, \bar{x}), \bar{y}) \right] \Big|_{\varepsilon=1/n}.$$

Interpretation: It measures how much the training error changes when the data point $\bar{z}$ is removed.

Definition 2. Given $(\bar{x}, \bar{y}) = \bar{z} \in \mathcal{D}^n$, the *influence function* relative to $\bar{z}$ is

$$\mathcal{I}(\bar{z}; \hat{\theta}) = \lim_{\varepsilon \to 0} \frac{\hat{\theta}_{\varepsilon+1/n,-\bar{z}} - \hat{\theta}_{1/n,-\bar{z}}}{\varepsilon}.$$

Interpretation: It represents the direction in which the optimal parameter moves when the training objective is perturbed by removing $\bar{z}$.
Assuming that L is strongly convex in θ, we can rewrite the previous quantities in the closed forms:

$$\mathcal{I}(z; \hat{\theta}) = H_{\hat{\theta}}^{-1} \nabla \ell(f(\hat{\theta}; x), y), \qquad \mathcal{Q}(z; \hat{\theta}) = \nabla \ell(f(\hat{\theta}; x), y)^\top H_{\hat{\theta}}^{-1} \nabla \ell(f(\hat{\theta}; x), y),$$

where $H_{\hat{\theta}}$ is the Hessian of L at $\hat{\theta}$.

Remark. This part is not really clear: referring to the original reference [3], the derivation should be as follow, with our notation.
Let $L_i(\theta) = (L(\hat{\theta}; \mathcal{D}^n) - 1/n\, \ell(f(\hat{\theta}; x_i), y_i))$. Assuming that L is twice differentiable, we can use Taylor near $\hat{\theta}$:

$$L_i(\theta) = L_i(\hat{\theta}) + (\theta - \hat{\theta})^\top L_i'(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^\top L_i''(\hat{\theta})(\theta - \hat{\theta}) + \mathrm{O}(\|\theta - \hat{\theta}\|^3)$$

Then, by minimizing the difference $L_i(\theta) - L_i(\hat{\theta})$ for $\theta \neq \hat{\theta}$, we get:

$$\theta = \hat{\theta} - (HL_i(\hat{\theta}))^{-1} JL_i(\hat{\theta}).$$

From this expression is also clear that the Influence Function estimator represents the parameter after one step of the Newton algorithm starting from $\hat{\theta}$ trying to get to $\hat{\theta}_{1/n,-z_i}$.

Unfortunately, there is some problems with these formulae:

1. the strong convexity is essential. If at a minimum point H has any 0-eigenvalue, we cannot invert it.

2. Even when L is strongly convex, the problem is not trivial because computing the inverse of the Hessian and the matrix-vector product are heavy computations.

Solution 1. For iHVP, there exist efficient approximations requiring $O(nd)$ flops instead of $O(n^3)$.

Solution 2. Change point of view: Influence functions are not approximators of LOO retraining, but instead of the proximal Bregman response function (PBRF).

Response functions

We define the response function as:

$$\hat{r}_z(\varepsilon) = \arg\min_{\theta\in\Theta} \left(L(\theta;\mathcal{D}^n) - \varepsilon\,\ell(f(\theta;x),y)\right).$$

Observe that $\hat{r}_z(\varepsilon) = \hat{\theta}_{\varepsilon,-z}$ and $\hat{r}_z(0) = \hat{\theta}$.

Since $\hat{r}$ is differentiable at 0, we can expand with Taylor at first order and we get:

$$\hat{r}_{z,\mathrm{lin}}(1/n) \approx \hat{\theta} + \frac{1}{n} H_{\hat{\theta}}^{-1}\,\nabla\ell(f(\hat{\theta};x),y).$$

We require $H_{\hat{\theta}}$ positive definite to invert it; thus $\hat{\theta}$ must be a minimizer.
To compute influence functions for MLPs, we can approximate $H_{\hat{\theta}}$ using the Gauss–Newton Hessian (GNH) and add damping:

$$\mathcal{I}^{\dagger}(z;\hat{\theta}) = \left(J_{y,\hat{\theta}}^{\top} H_{\ell,\hat{\theta}} J_{y,\hat{\theta}} + \lambda I\right)^{-1} \nabla\ell(f(\hat{\theta};x),y),$$

where $J_{y,\hat{\theta}}$ is the Jacobian of $F(\theta) = (f(\theta;x_1),\dots,f(\theta;x_n))$ and $H_{\ell,\hat{\theta}}$ is the hessian of $\ell(f(\theta;x),y)$ in $\theta = \hat{\theta}$.
Note that the damped GNH is always positive definite, as long as $H_{\hat{\theta}}$ is SPD.
We can get the previous formula by linearizing the response function of the regularized loss:

$$\hat{r}_{z,\mathrm{damp}}(\varepsilon) = \arg\min_{\theta\in\Theta} \left(L(\theta;\mathcal{D}^n) - \varepsilon\,\ell(f(\theta;x),y) + \tfrac{\lambda}{2}\|\theta-\hat{\theta}\|^2\right),$$

$$\hat{r}_{z,\mathrm{damp,lin}}(1/n) \approx \hat{\theta} + \frac{1}{n}\mathcal{I}^{\dagger}(z;\hat{\theta}).$$

Another issue is that, in practice, the parameter we utilise is not a minimizer of L. Thus, we want to consider a risk for which the early-stopped point θ^s is optimal:

$$\mathcal{L}(\theta;\theta^s,\mathcal{D}^n) = \frac{1}{n}\sum_{i=1}^{n} D_{\ell^{(i)}}\left(f(\theta;x_i), f(\theta^s;x_i)\right),$$

with $D_{\ell^{(i)}}$ being the Bregman difference:

$$D_{\ell^{(i)}}(y,y') = \ell(y,y_i) - \ell(y',y_i) - \nabla_1\ell(y',y_i)^{\top}(y-y').$$

Intuitively, this quantity measures the difference between the evaluation of ℓ on y and the first order Taylor expansion of ℓ around y' computed on y.

Observation. This quantity is always non-negative as long as ℓ is convex. To exemplify, choose $\ell(y,y') = \|y-y'\|^2/2$. This yields:

$$D_{\ell^{(i)}}(y,y') = \|y-y_i\|^2/2 - \|y'-y_i\|^2/2 - \langle\nabla\|y'-y_i\|^2/2, y-y_i\rangle = \|y-y'\|^2/2.$$

Consequently, we can define the *Proximal Bregman Response Function* (PBRF) as:

$$r^b_{z,\mathrm{damp}}(\varepsilon) = \arg\min_{\theta\in\Theta} \Big(\mathcal{L}(\theta;\theta^s,\mathcal{D}^n) - \varepsilon\, \ell(f(\theta;x),y) + \tfrac{\lambda}{2}\|\theta-\theta^s\|^2 \Big).$$

The interesting property of this object is that the linearized PBRF satisfies

$$r^b_{z,\mathrm{damp,lin}}(1/n) = \theta^s + \frac{1}{n}\mathcal{I}^\dagger(z;\theta^s).$$

As a consequence, influence functions are *not* approximating LOO retraining under the original loss L. Instead, they approximate the effect of training from θ^s under the modified objective

$$\mathcal{L}(\theta;\theta^s,\mathcal{D}^n) - \tfrac{1}{n}\ell(f(\theta;x),y) + \tfrac{\lambda}{2}\|\theta-\theta^s\|^2.$$

This fact makes PBRF a more suitable benchmark when testing the performances of IFs.
Concrete examples show that actually PBRF achieves good results in tasks such as mislabeled example detection, making it a viable alternative to LOO retraining.

Error decomposition

The approximation error of influence functions decomposes into:

- **Warm-start gap:** LOO starts from a random parameter (cold start), while IF are related to θ^s; we can then converge to another "optimal" point;
- **Proximity gap:** the factor $||theta - theta^s||$ induces the warm start not to move far away from $theta^s$;
- **Non-convergence gap:** in practice we almost never start from a fully trained network;
- **Linearization error:** produced by approximating the Taylor expansion at first order;
- **Solver error:** inexact iHVP computation.

Remark. The PBRF formulation eliminates the first three gaps.

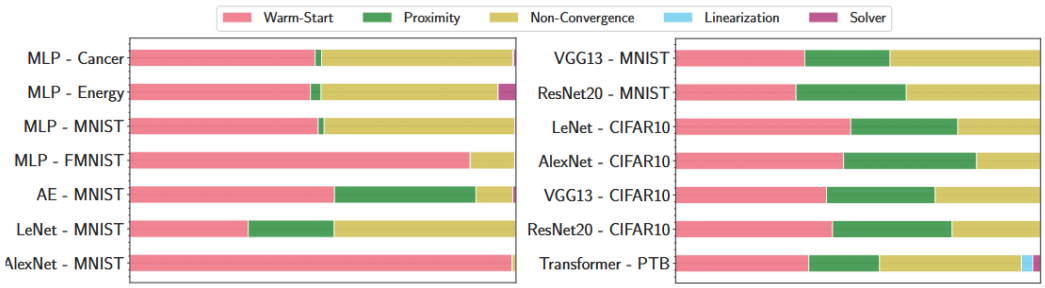

Figure 1: Visual representation of the error defomposition on different datasets and models. The main focus is that the largest components are the first three.

Influence Functions vs Leverage

For simplicity, let's consider the Ordinary Least Squares problem.
Let $X = (x_1|\dots|x_n)^\top \in \mathbb{R}^{(n\times d)}$, $y = (y_1,\dots,y_n) \in \mathbb{R}^n$, $e = (e_1,\dots,e_n) \in \mathbb{R}^n$, and $\theta \in \mathbb{R}^d$. In this setting, assume the dataset is generated by $y = f(\theta^\star; x) + e = X\,\theta^\star + e$ where e is the observational error. We can estimate $\theta^\star$ with $\hat{\theta} = (X^\top X)^{-1}X^\top y$. Consequently, the predicted data with our model will be $\hat{y} = X(X^\top X)^{-1}X^\top y$ and we call $P = X(X^\top X)^{-1}X^\top$, as it is an orthogonal projection on $\mathrm{ran}(X)$.

Definition 3. The *leverage score* of the i-th sample data is $P_{ii} = x_i(X^\top X)^{-1}x_i^\top$.

This quantity describes how much the i-th sample data affects the i-th prediction of our model. The bigger it is, the more probable it is that the i-th sample point is an outlier.
Indeed, if we consider $X = JF(\theta)$ as the X in our case, then the influence loss difference is approximately [1] the same as the leverage. However, the interpretation of the problem would become different.
In any case, an expression of interest that involves both leverage and influence functions is the following (cf. [6]):

$$\hat{\theta} - \hat{\theta}_{1/n,-z} = \frac{(X^\top X)^{-1}x_i\hat{e}_i}{1 - P_{ii}},$$

where $\hat{e}_i = y_i - x_i^\top\hat{\theta}$.
Observe that the $1 - P_{ii}$ at denominator means that outliers also have high influence in the training.

Example: IFs for OLS

Let's compute all the quantities we defined so far in the case of OLS with the euclidean loss.
We have:

- Loss function: $\ell(f(\theta; x_i), y_i) = \frac{1}{2}(y_i - x_i^\top\theta)^2$;
- Gradient: $\nabla\ell(f(\theta; x_i), y_i) = -x_i(y_i - x_i^\top\theta)$;
- Hessian: $H_\theta = \sum_{i=1}^n x_i x_i^\top = X^\top X$;
- Influence function: $\mathcal{I}(z_i; \hat{\theta}) = (X^\top X)^{-1}x_i(y_i - x_i^\top\hat{\theta})$;
- PBRF: $\frac{1}{2}\arg\min_{\theta\in\Theta}\left(\frac{1}{n}\sum_{i=1}^n(x_i^\top(\theta - \hat{\theta}))^2 - \frac{1}{n}\|y_i - x_i^\top\theta\|^2 + \lambda||\theta - \hat{\theta}||^2\right)$; note that in this case $\theta^s = \hat{\theta}$ since we can compute it explicitly.

[1] we can write $HF(\theta) = JF(\theta)^\top JF(\theta) + \sum\dots$. If we omit the second term, we have the sought approximation. This is acceptable when the parameter is close to optimal, since the sum annihilates for optimal parameters.

Reformulation of influence functions

Remark. The object similar to what we are going to discuss, which is present in the paper, is $\hat{r}_z(\varepsilon)$.

Let $\mathcal{X}$, $\mathcal{Y}$ be two measurable spaces and fix $D_n \subset (\mathcal{X} \times \mathcal{Y})^n$ such that $D_n = (z_i)_{i=1}^n$ with $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.
Given two random variables X and Y respectively on $\mathcal{X}$ and $\mathcal{Y}$, we are interested in studying the distribution of $Y|X$. To do so, we choose a Banach space Θ and a parametric function $f : \Theta \times \mathcal{X} \mapsto \mathcal{Y}$ as an estimator of such distribution.
In order to evaulate the estimators, we give the following definitions:

Definition 4. Given $\ell \in C^2(\mathcal{Y} \times \mathcal{Y}; \mathbb{R}_+)$ (called *loss function*), we define the *empirical risk* for the estimator $f(\theta; x)$ on the dataset D_n as:

$$R(\theta) := \sum_{i=1}^{n} \ell(f(\theta; x_i); y_i). \tag{0.1}$$

Given $\varepsilon \in [0, 1]$ and $j \in [n]$, we are interested in solving the minimization problem:

$$\min_{\theta \in \Theta} \sum_{i=1}^{n} \ell(f(\theta; x_i); y_i) + (\varepsilon - 1)\ell(f(\theta; x_j); y_j).$$

For simplicity, assume that for any ε the objective function has a unique minimum. We call $\theta(\varepsilon)$ such minimum. Let us introduce the notation:

$$\ell_j(\theta) = \ell(f(\theta; x_j); y_j),$$
$$R_j(\theta) = \sum_{\substack{i \in [n] \\ i \neq j}}^{n} \ell_i(\theta).$$

Thus, we can rewrite:

$$\theta(\varepsilon) = \arg\min_{\theta \in \Theta} R_j(\theta) + \varepsilon \ell_j(\theta).$$

Notice that $\theta(\varepsilon)$ symbolizes the parameter for which our model best fits the data while we change the weight of one data point in the training. It is particularly of interest to understand how

$$\theta(1) = \theta^\star = \arg\min R(\theta)$$

and

$$\theta(0) = \theta^\star_{-j} = \arg\min R_j(\theta)$$

are different, since they represent respectively the optimal parameter obtained while training on the whole data and on the dataset minus one particular point.
One way to analyze this, is by the means of influence functions.

Definition 5. The *influence function* of z_j is:

$$I(j) := \frac{d}{d\varepsilon}\theta(\varepsilon)\Big|_{\varepsilon=0} = \dot{\theta}(0). \tag{0.2}$$

Remark. The previous quantity is well defined because $\theta(\varepsilon)$ is C^1 thanks to the Implicit function theorem (applied to $\nabla\mathscr{L}(\theta,\varepsilon) = L(\theta) + \varepsilon\ell_j(\theta)$ we get that there exists $\theta(\varepsilon)$ differentiable such that $\nabla\mathscr{L}(\theta(\varepsilon),\varepsilon) = 0$ in a neighbourhood of 1 at least, since by definition $\nabla\mathscr{L}(\theta^\star,1) = \nabla R(\theta^\star) = 0$).
It is not clear yet if we are more interested in $\dot{\theta}(0)$ or $\dot{\theta}(1)$.

Proposition 0.0.1. *We can write $I(j)$ explicitly as:*

$$I(j) = -H_{R_j}^{-1}\nabla_\theta\ell_j(\theta^\star_{-j}) \tag{0.3}$$

where H_{R_j} is the Hessian of R_j in $\theta^\star_{-j}$.

Proof. By definition, $\theta(\varepsilon)$ satisfies:

$$\nabla_\theta(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon))) = 0.$$

Taking another derivative in ε yields:

$$\begin{aligned}
\nabla^2_{\theta,\varepsilon}(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon))) &= 0 \iff &(0.4)\\
\nabla^2_{\theta\theta}R_j(\theta(\varepsilon))\dot{\theta}(\varepsilon) + \nabla_\theta\ell_j(\theta(\varepsilon)) + \varepsilon\nabla^2_{\theta\theta}\ell_j(\theta(\varepsilon))\dot{\theta}(\varepsilon) &= 0 \iff &(0.5)\\
\dot{\theta}(\varepsilon) = -(\nabla^2_{\theta\theta}(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon))))^{-1}\nabla_\theta\ell_j(\theta(\varepsilon)). &&(0.6)
\end{aligned}$$

Evaluating in $\varepsilon = 0$ concludes the proof:

$$\dot{\theta}(0) = -(\nabla^2_{\theta\theta}R_j(\theta(0)))^{-1}\nabla_\theta\ell_j(\theta(0)).$$

■

Remark. If we wanted to compute $\dot{\theta}(1)$, it would also have a nice form:

$$\dot{\theta}(1) = -H_R^{-1}\nabla_\theta\ell_j(\theta^\star).$$

where H_R is the Hessian of R in $\theta^\star$. Indeed, if our goal is to *unlearn* a data point, this formulation does not require us to retrain the model, as we have already access to $\theta^\star$ (but not to $\theta^\star_{-j}$).

It could also be of interest to consider:

$$Q(z) := \frac{d}{d\varepsilon}(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon)))\Big|_{\varepsilon=0}. \tag{0.7}$$

Corollary 0.0.1. *The following formulae hold:*

$$\begin{aligned}
\frac{d}{d\varepsilon}(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon)))\Big|_{\varepsilon=0} &= \ell_j(\theta^\star_{-j}),\\
\frac{d}{d\varepsilon}(\ell_j(\theta(\varepsilon)))\Big|_{\varepsilon=0} &= -\nabla_\theta\ell_j(\theta^\star_{-j})^\top H_{R_j}^{-1}\nabla_\theta\ell_j(\theta^\star_{-j}),
\end{aligned} \tag{0.8}$$

Proof. For the first equality, taking the derivative yields:

$$\nabla_\theta R_j(\theta(\varepsilon))\dot{\theta}(\varepsilon) + \ell_j(\theta(\varepsilon)) + \varepsilon\nabla_\theta \ell_j(\theta(\varepsilon))\dot{\theta}(\varepsilon)$$

and recalling that on $\theta(\varepsilon)$ it holds $\nabla(R_j(\theta(\varepsilon)) + \varepsilon\ell_j(\theta(\varepsilon))) = 0$ we can conclude.
For the second one it suffices to substitute the definition of $\dot{\theta}$ from the previous equation after taking the derivative.

■

Remark. As discussed in [7], it is also possible to consider $\boldsymbol{\epsilon} = (\varepsilon_1, \dots, \varepsilon_n) \in [0,1]^n$ and $R_{\boldsymbol{\epsilon}}(\theta) = \sum_{i=1}^n \varepsilon_i \ell_i(\theta)$. Note also that in this case, their notation figures "$\theta(0)$", but indeed in our notation it would be $\theta(1)$ (which makes sense since we are considering a group of points).
In such work, they provide formal statements about when the influence function approximations are accurate, taking into account also the difference between the influence function estimation and the *Newton estimation*.

Definition 6. We call Newton estimation the quantity:

$$I_{Nt}(j) = -(HR_j(\theta(1)))^{-1}\nabla R_j(\theta(1))$$

.

The name for this stems from the fact that this formula estimates the parameter after one step of Newton method while minimizing ∇R_j (equivalently, we are considering the second order Taylor expansion of R_j in $\theta(1)$):

$$\theta_{Nt} = \theta(1) - (HR_j(\theta(1)))^{-1}\nabla R_j(\theta(1)).$$

Unfortunately, $I(j)$ and $I_{Nt}(j)$ are close when the smallest eigenvalue of the hessian is big, which is not always the case. One way to ensure this difference is small, is to add the regularization term $\lambda\|\theta\|$ in the risk. This way, we can decrease the aforementioned error by choosing a large λ.

When working with more than one index, it makes sense to define the following quantity:

$$\theta^j(\varepsilon) = \theta\underbrace{(1, \dots, \varepsilon, \dots, 1)}_{j\text{-th place}}.$$

Question: Is there a link between $\dot{\theta^j}$ and $J\theta$?

Gradient Descent case

Let us specialize the analysis to the instance where we are using the gradient descent algorithm to pursue the minimization task. Furthermore, consider the gradient flow dynamics:

$$\begin{cases} \frac{d}{dt}\theta(t,\varepsilon) = -\nabla R_j(\theta(t,\varepsilon)) - \varepsilon\nabla\ell_j(\theta(t,\varepsilon)) \\ \theta(0,\varepsilon) = \theta_0 \qquad \forall\varepsilon \in [0,1] \end{cases}, \tag{GF}$$

where we consider fixed $j \in [n]$ and $\theta_0 \in \Theta$.

Let us call $\xi(t_0, t_1, \varepsilon; \theta_0)$ the flux of (GF),i.e., $\theta(t, \varepsilon) = \xi(0, t, \varepsilon; \theta_0)$. By the differential equation theory, since we are assuming $\ell \in C^2$, we know that at least $\xi \in C^2_t \times C^0_\varepsilon$ (even Lipschitz in ε). Therefore, $\theta : [0, +\infty] \times [0, 1] \to \Theta$ can be seen as a homotopy between the learning trajectories in the parameters' space (taking as definition $\theta(+\infty, \varepsilon) := \theta(\varepsilon)$).

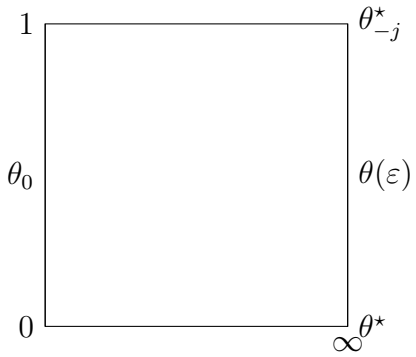

Figure 2: Visual representation of the homotopy $\theta(t, \varepsilon)$ where on the horizontal axis we have evolution in time and on vertical axis the change of parameter ε.

Question: How different really is $\dot{\theta}(1)$ from $\dot{\theta}(0)$?

Assuming $\theta(\varepsilon)$ is C^2, we can give the bound:

$$\dot{\theta}(1) - \dot{\theta}(0) = \int_0^1 \ddot{\theta}(\varepsilon) d\,\varepsilon \leq \|\ddot{\theta}\|_{\infty,[0,1]},$$

but its computation requires a third derivative of R_j...

I would like to do some experiments in a simple setting for visualizing $\theta(\varepsilon)$. This may give insights.

Some experiments are summarized in Figure 3. What I found interesting about them is that the distance from the target function does not affect the linearity of the trajectory. In fact, in the figure you can see that the first considered point is close to the real value, but the trajectory of its parameter is not a rect, unlike the second point (which is more distant).

Question: Consider now the map $\theta(t, s) : [0, +\infty] \times [0, T]$, with $T \in (0, +\infty]$ fixed a priori, such that:

$$\begin{cases} \theta(t, s) = \xi(0, t, 1; \theta_0) & t \in [0, s] \\ \theta(t, s) = \xi(s, t, 0; \xi(0, s, 1; \theta_0)) & t \in [s, T] \end{cases}. \tag{0.9}$$

What can we say about $\theta(T, s)$?

If $T = \infty$, then for any $s < \infty$ it holds that $\theta(\infty, s) = \theta^\star_{-j}$, but for finite T the answer is not clear.

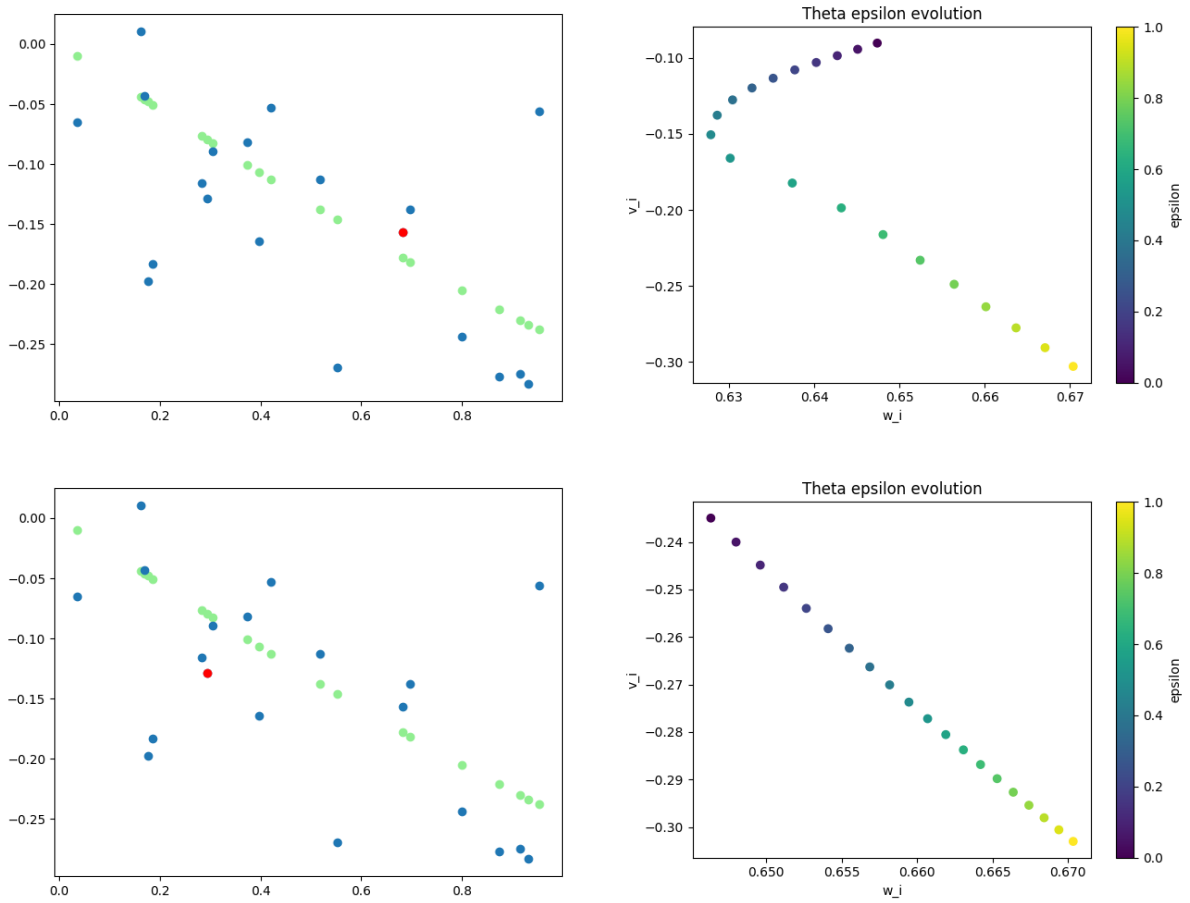

Figure 3: I used a 3-neuron 1-hidden layer MLP trained on 20 points generated by $y_i = f_*(x_i) + x_i$ where f_* is another MLP of the same type and $\xi_i \sim \mathcal{N}(0, 0.1)$. On the left column we can observe the conisdered data point we are deleting in red, in blue the other data points and in green the target function. On the right side, I represented the trajectory of the first neuron as a function of ε (I trained the model for 1000 iterations, it should be enough)

Epsilon dynamic

Note that we can rewrite $\theta(t,s)$ from 0.9 by transforming ε itself in a variable of time:

$$\theta(t,s) = \xi(0,t,\varepsilon_s;\theta(0)) \quad \text{where} \quad \varepsilon_s(t) = \mathbb{1}_{[0,s]}(t) \ .$$

From this formula, a natural question arises: is a step function really the best choice for $\varepsilon(t)$?

To reply, we first need to analyse the dynamics in the very simple case of linear regression with squared loss to see if we can get any insight.
With this goal in mind, let's rewrite explicitly GF in this specific case (I will use the same notation as the IF vs Leverage part):

$$\begin{cases} \frac{d}{dt}\theta(t,1) = -\nabla\frac{1}{2}(X\theta - y)^2 = -X^\top X\theta + X^\top y \\ \theta(0) = \theta_0 \end{cases} . \tag{LRGF}$$

This is a linear ODE, so, when H is invertible, we can find an explicit solution, which reads:

$$\theta(t,1) = e^{-Ht}\theta_0 + (\mathrm{Id}_d - e^{-Ht})H^{-1}X^\top y. \tag{0.10}$$

Lemma 0.0.1. *The system LRGF has a unique asymptotically stable equilibrium point:*

$$\theta^\star = (X^\top X)^\dagger X^\top y + (\mathrm{Id}_d - X^\top (XX^\top)^\dagger X)\theta_0,$$

where $A^\dagger$ denotes the Moore-Penrose pseudo-inverse. In particular, when $X^\top X$ is invertible, we get:

$$\theta^\star = \lim_{t\to\infty} \theta(t,1) = (X^\top X)^{-1}X^\top y = \hat{\theta}_{\mathrm{LS}}.$$

Proof. The case of H invertible can be proven by looking at the explicit solution stated above: for $t \to \infty$, it yelds: $\theta^\star = H^{-1}X^\top y =: \hat{\theta}_{\mathrm{LS}}$. When $\mathrm{rnk}H = r < d$, let us begin by observing that since $H = X^\top X$, H is diagonalizable, i.e., there exists Q orthogonal matrix such that $H = Q^\top DQ$. Therefore, up to permutation matrices, we can rewrite D as $\mathrm{diag}(d_1,\dots,d_r,0,\dots,0)$ where $d_1,\dots,d_r \neq 0$ and we have $d-r$ zero entries. By the definition of the matrix exponential, it holds true that $e^{-Ht} = Q^\top e^{-Dt}Q$ and $e^{Dt} = \mathrm{diag}(e^{d_1 t},\dots,e^{d_r t},1,\dots,1)$. For this reason, we can consider the two invariant subspaces $\mathrm{ran}H \oplus \ker H$ and study the dynamic in these two sets.

- In $\ker H$, applying the formula 0.10 we get $\theta(t,1) = \pi_{\ker H}\theta_0$;
- In $\mathrm{ran}H$, the same formula, if we call $\tilde{H} = H|_{\mathrm{ran}H}$, yields $\theta(t,1) = e^{-\tilde{H}t}\pi_{\mathrm{ran}H}\theta_0 + (\mathrm{Id}_d - e^{-\tilde{H}t})\tilde{H}^{-1}\tilde{X}^\top\pi_{\mathrm{ran}H}y$.

To conclude, observe that $\ker H = \ker X$ and $\mathrm{ran}H = \mathrm{ran}X^\top$. Therefore, $\pi_{\ker H} = \pi_{(\mathrm{ran}X^\top)^\perp} = I - X^\top(XX^\top)^\dagger X$ (note that here the M-P pseudoinverse pops up naturally from the request of $\pi^2 = \pi$) and the projections in the second case are not needed. The thesis is achieved by glueing together the results in the two subspaces. ■

In order to extend this to the case of every other $\varepsilon \in (0,1)$ [2], consider $y_\varepsilon^j = (y_1, \dots, \varepsilon y_j, \dots, y_n)$ and similarly X_ε^j and $H_\varepsilon^j = (X_\varepsilon^j)^\top X_\varepsilon^j$. Solving (LRGF) with the above quantities defines:

$$\theta(t,\varepsilon) = e^{-H_\varepsilon^j t}\theta_0 + (\mathrm{Id}_d - e^{-H_\varepsilon^j t})\hat{\theta}_{-j}(\varepsilon). \tag{0.11}$$

With s, T fixed, what we want to study is what is the "fastest route" to $\theta^\star_{-j}$ when we change $\varepsilon_s(t)$. In other words, we are trying to solve:

$$\min_{\varepsilon \in E_s} \mathcal{E}(\varepsilon) \quad \text{where} \quad \mathcal{E}(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \theta^\star_{-j}\|^2_\Theta \tag{0.12}$$

for $E_s = \{f \in [0,1]^{[0,+\infty]} \mid f(t) = 1 \ \forall t \le s \wedge f(t) = 0 \ \forall t \ge T\}$.
Question: Shall we consider instead $\mathcal{E}(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \xi(0,T,0;\theta(0))\|^2_\Theta$?

Computed on our baseline (i.e., $\varepsilon_s(t) = \mathbb{1}_{[0,s]}(t)$), this quantity is:

$$\mathcal{E}(\varepsilon_s) = \frac{1}{2}\|e^{-H_{-j}(T-s)}(e^{-Hs}\theta_0 + (\mathrm{Id}_d - e^{-Hs})\theta^\star - \theta^\star_{-j})\|^2.$$

In fact, applying 0.11 for $\varepsilon = 1$ and $T = s$ yields the initial condition for the IVP with $\varepsilon = 0$, which reads:

$$\begin{cases} \dot{\theta}(t,\varepsilon(t)) = -\sum_{i\neq j} x_i(x_i^\top\theta - y_i) \\ \theta(0) = e^{-H_1^j t}\theta_0 + (\mathrm{Id}_d - e^{-H_1^j t})\hat{\theta}_{-j}(1) \end{cases}.$$

Recall that $H_1^j = H$ and $\hat{\theta}_{-j}(1)$. At this point, using again 0.11 for $\varepsilon = 0$ and substituting in the definition of $\mathcal{E}(\varepsilon)$ gives the equality.

Can we do any better?

Unfortunately, as soon as we add the time dependancy of ε in the equation:

$$\begin{cases} \dot{\theta}(t,\varepsilon(t)) = -\sum_{i\neq j} x_i(x_i^\top\theta - y_i) - \varepsilon(t)x_j(x_j^\top\theta - y_j) \\ \theta(0) = \theta_0 \end{cases}, \tag{0.13}$$

we cannot obtain a closed form solution anymore. In fact, even if we rewrite the first line as:

$$\begin{aligned} \dot{\theta}(t,\varepsilon(t)) &= a(t)\theta(t) + b(t), \ \text{where} \\ a(t) &= -\sum_{i\neq j} x_i x_i^\top - \varepsilon(t)x_j x_j^\top \\ b(t) &= \sum_{i\neq j} x_i y_i + \varepsilon(t)x_j y_j \end{aligned}$$

and use the method of the Wronskian with the variation of constants, that does not yield a closed-form solution in the general case[3].
Let's see what happens instead for simpler cases doing computations by hand. In the following, consider $s > 0$ fixed.

[2] for $\varepsilon = 0$ consider for simplicity X', y' obtained by removing the target datapoint, otherwise we can integrate that case in the general one using the Moore-Penrose pseudoinverse

[3] This method requires $\varepsilon \in C^1$ because to apply the Wronskian we need to take another derivative in the homogeneous equation and consider $\ddot{\theta} - \dot{\theta}a(t) - \theta\dot{a}(t)$.

Step functions Is $\mathbb{1}_{[0,s]}$ the best step function we can choose in E_s?

Lemma 0.0.2. *Assume $H = \mathrm{Id_d}$. Given $0 \leq s < T$, the τ that minimizes $\mathcal{E}(\varepsilon_\tau)$ in $\{\varepsilon_\tau = \mathbb{1}_{[0,\tau]} \mid \tau \in [s,T]\}$ is ε_s.*

Proof. Starting from the assumption that $H = \mathrm{Id}_d$, we can prove that both H_{-j} and $(H_{-j} - \mathrm{Id}_d)$ are projections, i.e., they have the property that $P^2 = P$. In fact, $H = \mathrm{Id}_d$ iff X is orthogonal. Thus (call $X_{-j} = (I - \hat{e}_j\hat{e}_j^\top)X$):

$$H_{-j} = X_{-j}^\top X_{-j} = X^\top (I - \hat{e}_j\hat{e}_j^\top)^\top (I - \hat{e}_j\hat{e}_j^\top)X,$$

and, since $XX^\top = \mathrm{Id}_d$ and $(I - \hat{e}_j\hat{e}_j^\top)$ is a projection:

$$H_{-j}^2 = X^\top (I - \hat{e}_j\hat{e}_j^\top)^\top (I - \hat{e}_j\hat{e}_j^\top) XX^\top (I - \hat{e}_j\hat{e}_j^\top)^\top (I - \hat{e}_j\hat{e}_j^\top)X = X^\top (I - \hat{e}_j\hat{e}_j^\top)X = H_{-j}.$$

Moreover, it holds that $H_{-j} = \mathrm{Id}_d - vv^\top$ where $v = X^\top e_j = x_j$. Using the fact that for a projection P holds $e^{aP} = \mathrm{Id_d} + (e^a - 1)P$, we can then rewrite $\mathcal{E}(\varepsilon_\tau)$ as:

$$\begin{aligned}
\mathcal{E}(\varepsilon_\tau) &= \frac{1}{2}\|e^{-(T-\tau)H_{-j}}(e^{-\tau}\theta_0 + (\mathrm{Id}_d - e^{-\tau})\theta^\star - \theta^\star_{-j})\|^2 \\
&= \frac{1}{2}\| \left(\mathrm{Id}_d + (e^{-(T-\tau)} - 1)H_{-j}\right)(e^{-\tau}\theta_0 + (\mathrm{Id}_d - e^{-\tau})\theta^\star - \theta^\star_{-j})\|^2.
\end{aligned}$$

To find the optimal $\tau \in [s,T)$, let's compute the derivative in τ of the previous quantity and study when it is equal to 0.
Recall that $\|x\|^2 = \langle x, x\rangle$. Thus, if we call

$$g(\tau) = \left(\mathrm{Id}_d + (e^{-(T-\tau)} - 1)H_{-j}\right)(e^{-\tau}\theta_0 + (\mathrm{Id}_d - e^{-\tau})\theta^\star - \theta^\star_{-j}),$$

we get that $\frac{d}{dt}\mathcal{E}(\varepsilon_\tau) = \langle g(\tau), g'(\tau)\rangle$. First, let's compute $g'(\tau)$:

$$\begin{aligned}
\frac{d}{d\tau}g(\tau) &= e^{-(T-\tau)}H_{-j}(e^{-\tau}\theta_0 + (\mathrm{Id}_d - e^{-\tau})\theta^\star - \theta^\star_{-j}) + \\
&\quad + \left(\mathrm{Id}_d + (e^{-(T-\tau)} - 1)H_{-j}\right)(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star) = \\
&= e^{-(T-\tau)}H_{-j}(\theta^\star - \theta^\star_{-j}) + (\mathrm{Id}_d - H_{-j})(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star).
\end{aligned}$$

Consequently, differentiating $\mathcal{E}$, yields:

$$\begin{aligned}
\frac{d}{d\tau}\mathcal{E}(\varepsilon_\tau) &= \langle g'(\tau), g(\tau)\rangle \\
&= \left\langle e^{-(T-\tau)}H_{-j}(\theta^\star - \theta^\star_{-j}) + (I_d - H_{-j})(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star),\, g(\tau)\right\rangle \\
&= e^{-(T-\tau)}\left\langle H_{-j}(\theta^\star - \theta^\star_{-j}), g(\tau)\right\rangle + \left\langle (I_d - H_{-j})(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star), g(\tau)\right\rangle \\
&= e^{-(T-\tau)}\left\langle H_{-j}(\theta^\star - \theta^\star_{-j}), H_{-j}g(\tau)\right\rangle + \left\langle (I_d - H_{-j})(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star), (I_d - H_{-j})g(\tau)\right\rangle \\
&= e^{-(T-\tau)}\left\langle H_{-j}(\theta^\star - \theta^\star_{-j}), e^{-(T-\tau)}H_{-j}(\theta^\star - \theta^\star_{-j})\right\rangle \\
&\quad + \left\langle (I_d - H_{-j})(-e^{-\tau}\theta_0 + e^{-\tau}\theta^\star), e^{-\tau}(I_d - H_{-j})(\theta^\star - \theta_0)\right\rangle \\
&= e^{-2(T-\tau)}\|H_{-j}(\theta^\star - \theta^\star_{-j})\|^2 + e^{-2\tau}\|(I_d - H_{-j})(\theta^\star - \theta_0)\|^2.
\end{aligned}$$

The above derivative is always positive unless both norms annihilate, which is a probability zero event if we randomly initialise the starting point. In such case, $\mathcal{E}(\tau)$ is constant because removing the j-th datapoint does not change the learning trajectory. We can thus conclude that the function $\mathcal{E}(\tau)$ attains its minimum on the interval $[s, T]$ on s, making ε_s the best candidate among the step functions. ■

Remark. If we do not assume $H = \mathrm{I}$, the derivative of $g(\tau)$ has the following form:

$$\frac{d}{dt}g(\tau) = e^{-(T-\tau)H_{-j}}\left([H_{-j} - H]e^{-\tau H}(\theta_0 - \theta^\star) + H_{-j}(\theta^\star - \theta^\star_{-j})\right),$$

and, when taking the scalar product, it yields:

$$\begin{aligned}\frac{d}{dt}\mathcal{E}(\varepsilon_\tau) &= \langle g'(\tau), g(\tau)\rangle \\ &= \left\langle g'(\tau), e^{-(T-\tau)H_{-j}}(e^{-\tau H}\theta_0 + (\mathrm{Id}_d - e^{-\tau H})\theta^\star - \theta^\star_{-j})\right\rangle \\ &= \left\langle [H_{-j} - H]e^{-\tau H}(\theta_0 - \theta^\star), e^{-\tau H}(\theta_0 - \theta^\star)\right\rangle_{e^{-(T-\tau)H_{-j}}} + \\ &\left\langle H_{-j}(\theta^\star - \theta^\star_{-j}), \theta^\star - \theta^\star_{-j}\right\rangle_{e^{-(T-\tau)H_{-j}}}.\end{aligned}$$

In general, we have to be careful also because H and H_{-j} do not commute. In fact, we can rewrite $H = \sum_{i=1}^n x_j x_j^\top$ and $H_{-j} = H - x_j x_j^\top$ and observe that the commutator is:

$$[x_i x_i^\top, x_j x_j^\top] = (x_i^\top x_j)(x_i x_j^\top - x_j x_i^\top).$$

Thus, H and H_{-j} commute only when $x_i \perp x_j$ or $x_i = x_j$ for each $i \neq j$.

Note that usually in practice $n > d$ and both H and H_{-j} are full rank.
It can also happen that $H_{-j} = H$, for example when x_j is a linear combination of the other x_i's, but in that case, $\theta^\star_{-j} = \theta^\star$ and the derivative is 0.

Linear functions Consider the class of functions $\varepsilon_{\mathrm{lin},t_1}(t)$ such that:

$$\varepsilon_{\mathrm{lin},t_1}(t) = \mathbb{1}_{[0,s]}(t) + \mathbb{1}_{[s,t_1]}(t)\left(1 - \frac{(s-t)}{(s-t_1)}\right).$$

The main problem is that now in 0.13 from s to t_1 we have the following non-autonomous equation:

$$\begin{cases}\dot\theta(t, \varepsilon(t)) = -\sum_{i\neq j} x_i(x_i^\top\theta - y_i) - \left(1 - \frac{(s-t)}{(s-t_1)}\right)x_j(x_j^\top\theta - y_j) \\ \theta(0) = \xi(0, s, 1; \theta_0)\end{cases}.$$

To try solve this, let's define $\psi(t) = t$, thus $\dot\psi = 1$. Then, we can the previous equation as a dynamical system:

$$\begin{cases}\dot\theta(t) = \left(-\sum_{i\neq j} x_i x_i^\top\right)\theta(t) + \frac{x_j x_j^\top}{s-t_1}\theta(t)\psi(t) + \left(\sum_{i\neq j} x_i y_i + (1 - \frac{s}{(s-t_1)})x_j y_j\right) \\ \dot\psi(t) = 1\end{cases}. \tag{0.14}$$

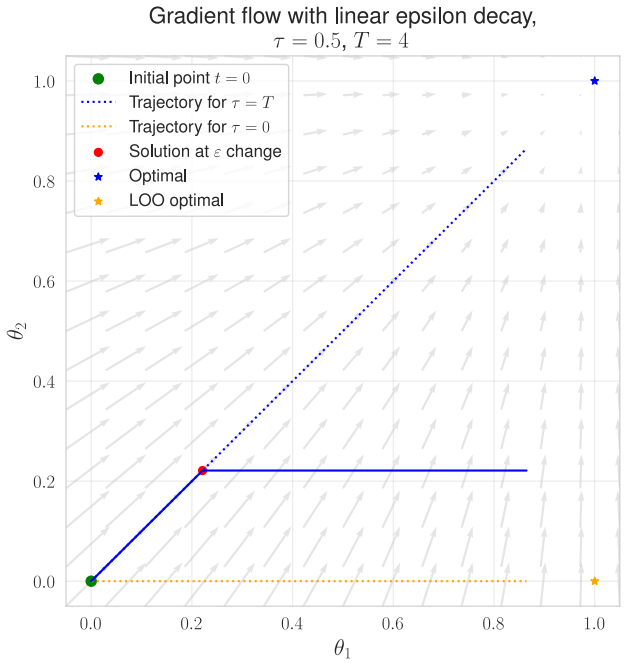

Figure 4: Plot representing the training trajectories for $X = \mathrm{Id}_2$, $y = (1, 1)$, $\theta_0 = (0, 0)$ and $j = 2$.

Note that this system is a special case of an inhomogeneous Lotka-Volterra system and it has no closed form solution. Unfortunately, I think that even though we can describe qulitatively the solutions, for our purpose we would need quantitative results, which I don't know how to obtain.

General case Since the general statement is clear and from my perspective the optimal solution should be a step function, I would like to try and approach this problem from a more general point of view, hoping to exploit some deeper structure that could avoid us some intricate computation. My idea is that we can check optimality of $\varepsilon(t)$ by imposing that at each time, the deplacement is maximum towards θ^*_{-j}. In this sense, we could consider the global ODE as an uncountable set of ODEs with fixed ε and, since we know the explicit solution in this case, we can maximize the previous quantity. My hope is that we always get that optimality is achieved by setting $\varepsilon = 0$ for every instance.

Let me give a formalization of the above mentioned procedure (I am not sure about considering an ODE at each dt, but if we use Gradient Descent it totally makes sense to consider T different equations).
In the gradient descent case, I would consider the quantity:

$$(\theta(t+1) - \theta(t)) \cdot \frac{\theta^\star_{-j} - \theta(t)}{\|\theta^\star_{-j} - \theta(t)\|}.$$

If we see $(\theta(t+1) - \theta(t))$ as $(\theta(t+\delta) - \theta(t))/\delta$ for $\delta = 1$, then taking the limit for

$\delta \to 0$ yields the derivative. Therefore, I would like to maximize the quantity:

$$\dot{\theta}(t) \cdot \frac{\theta^\star_{-j} - \theta(t)}{\|\theta^\star_{-j} - \theta(t)\|}. \tag{0.15}$$

In addition, since we want to change ε at every time, we should also update the initial condition $\theta(0)$ accordingly and then consider $\dot{\theta}(0)$.
Substituting $t = 0$ in 0.15 and computing explicitly the derivative of $\theta(t, \varepsilon)$ from 0.11 yields:

$$\frac{\langle -H^j_\varepsilon \theta_0 + (\mathrm{Id}_d + H^j_\varepsilon)\hat{\theta}_{-j}(\varepsilon), \theta^\star_{-j} - \theta(0)\rangle}{\|\theta^\star_{-j} - \theta(0)\|} = \frac{\langle (\mathrm{Id}_d + H^j_\varepsilon)(\hat{\theta}_{-j}(\varepsilon) - \theta(0)), \theta^\star_{-j} - \theta(0)\rangle}{\|\theta^\star_{-j} - \theta(0)\|} + \frac{\langle \theta(0), \theta^\star_{-j} - \theta(0)\rangle}{\|\theta^\star_{-j} - \theta(0)\|}.$$

Note that the second term does not depend on ε (at this step; it depends however on previous steps, since they lead to θ_0).
If we choose $\varepsilon = 0$, then $\hat{\theta}_{-j}(\varepsilon) = \theta^\star_{-j}$. Thus (let's assume $H = \mathrm{Id}$), the previous quantity equates:

$$2\|\theta^\star_{-j} - \theta(0)\| - \frac{([\theta^\star_{-j} - \theta(0)]_j)^2}{\|\theta^\star_{-j} - \theta(0)\|} + \frac{\langle \theta(0), \theta^\star_{-j} - \theta(0)\rangle}{\|\theta^\star_{-j} - \theta(0)\|}.$$

However, this may not be the maximum we can get! Indeed, if $\theta^\star_{-j} - \theta(0)$ is for example a line parallel to $\theta^\star_{-j} - \theta(0)$, choosing a bigger ε may be better.

Example: consider $X = (1, 1)^\top$, $y = (y_1, y_2)^\top$, $j = 1$. In this case, the system 0.13 becomes:

$$\begin{cases} \dot{\theta} = \varepsilon(y_1 - \theta) + (y_2 - \theta) \\ \theta(0) = \theta_0 \in \mathbb{R} \end{cases}.$$

Here, $\theta^\star_{-1} - \theta(0)$ is aligned with $\theta^\star_{-1} - \theta(0)$ and if we choose $\theta_0 = 0$, $y_1 = 2$, $y_2 = 1$, then the biggest step is actually achieved by $\varepsilon = 1$. Indeed, $\hat{\theta} = y_1 + y_2$ and $\hat{\theta}_{-1} = y_2$, since the general solution is $\theta(t) = e^{-(1+\varepsilon)t} + (1 - e^{-(1+\varepsilon)t})(\varepsilon y_1 + y_2)$.

Numerical experiments Let's see if we can get some more insights with numerical experiments: From the results in Figure 5, the function $\mathbb{1}_{[0,s]}$ appears not to be the minimizer for every instance of the problem, as proven in the stepfunction paragraph. Moreover, it seems to hold that step functions are better than polynomial decay.

Choice of distance measure

In the definition of our minimization problem in 0.12, the choice of $\mathcal{E}(\varepsilon)$ was arbitrary. What is important, is that $\mathcal{E}(\varepsilon) = d(\varepsilon, 0)$ for some $d : E_s \times [0, 1]^{[0,+\infty]} \to \mathbb{R}$ distance in the metric space of functions from $\mathbb{R}_+$ to $[0, 1]$. Let's discuss more in detail some choices for $d(\varepsilon) = d(\varepsilon, 0)$ and what does it mean to implement them.

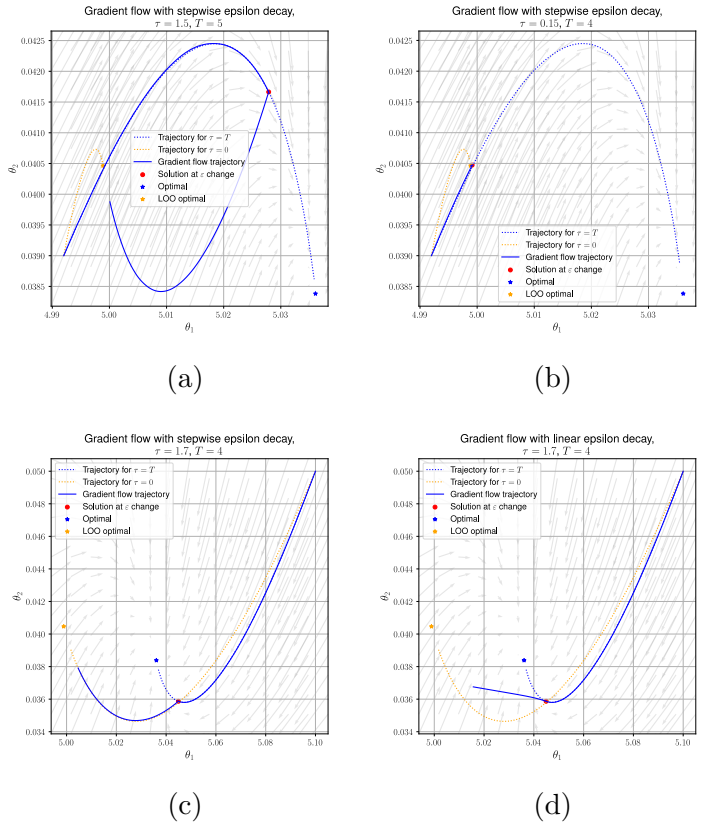

Figure 5: Phase portrait of the numerical integration of the IVP 0.13 for different $\varepsilon(t)$ and initial conditions θ_0 when $X \in \mathbb{R}^{20\times 2}$. In (a) we can see that changing ε too late in the training results in way worse results compared to (b), where, if chosen carefully, $\varepsilon > 0$ can be faster than LOO. However, choosing a different initial condition, as shown in (c), can lead to every $\varepsilon > 0$ to be suboptimal. Figure (d) shows that linear decay is slower than stepwise decay.

- $d(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \xi(0,+\infty,0;\theta(0))\|_\Theta^2$: minimizing this quantity is equivalent to minimize $\|\frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \theta^\star_{-j}\|_\Theta$, which is the distance in the space of parameters between the optimal parameter for the LOO retraining and the parameter obtained at time T of training with the weight switch $\varepsilon \in E_s$. In simpler words: "we take the most out of your data before completely forgetting about it".

- $d(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \xi(0,T,0;\theta(0))\|_\Theta^2$: in this case, we are comparing the trajectory for the LOO with the one obtained changing weights with $\varepsilon(t)$ at the same final time T. Simply said: "at the moment of removing your data, we learnt the same as if we hadn't been using your data from the start".

- $d(\varepsilon) = \frac{1}{2}\min_{t>0}\|\xi(0,T,\varepsilon;\theta(0)) - \xi(0,t,0;\theta(0))\|_\Theta^2$: this choice is more difficult to implement, as it is a double minimization problem. In fact, we are trying to find the final point of trajectory obtained by training with $\varepsilon(t)$ which is closest to

the orbit of parameters trained with LOO. The advantage compared to the first option is that we are more likely to be close the LOO trajectory; in relation with the second distance, this one ensures a better usage of the informations, as we can end up farther. In other words: "We used your data to speed up the training we would have achieved without your information".

As far as proofs are concerned, in the stepwise case, Lemma 0.0.2 works for all 3 of the above cases, since we never utilize any property of $\theta^\star_{-j}$.

What is the problem?

We are given a training set $D = \{(x_i, y_i)\}_{i=1}^n$ and an index $j \in [n]$ of a data point in the training set. We are also given a learning algorithm $\mathcal{A}(D)$, which, for simplicity, we will assume to be gradient flow on the empirical risk. Our goal is to find a modification function $\mathcal{M}$ for the algorithm $\mathcal{A}$ that will approximate the effect of retraining the model from scratch on $D \setminus (x_j, y_j)$. To do so, we want to choose $\varepsilon(t)$ from 0.13 among a class of functions E_s that will allow us to forget the j-th data point as much as possible before time T, while achieving a good performance as a predictor.

For this problem to be well posed, we first need a proper definition for "forgetting a data point". In the literature, I could find two different definitions for this concept, which are the following:

- *Certified Removal* [5]: given $\alpha > 0$, we say that $\mathcal{M}$ is an α-certified removal algorithm if:

$$e^{-\alpha} \leq \frac{\mathbb{P}(\mathcal{M}(\mathcal{A}(D), j) \in S)}{\mathbb{P}(\mathcal{A}(D \setminus \{(x_j, y_j)\}) \in S)} \leq e^{\alpha} \quad \forall S \subseteq \Theta.$$

- Same distribution [2]: let $\mathbb{D}_{\mathcal{M}}$ denote the distribution of models learned using mechanism $\mathcal{M}$ on D trying to unlearn the j-th data point. Let $\mathbb{D}_{\text{real}}$ be the distribution of models learned using $\mathcal{A}$ on $D \setminus \{(x_j, y_j)\}$. $\mathcal{M}$ is a good unlearning algorithm if $\mathbb{D}_{\mathcal{M}} = \mathbb{D}_{\text{real}}$. This definition is equivalent to 0-CR.

However, I don't think either of these definitions is what I am looking for. In fact, I am convinced that as long as $\mathcal{M}(\mathcal{A}(D), j)$ ends in a point that can be reached by $\mathcal{A}(D \setminus \{(x_j, y_j)\})$, then we can say that $\mathcal{M}$ is a good unlearning algorithm. The reason for this is that the gradients only depend on the spatial position of θ and not on time. Thus, I'm expecting that there is no way to extract information about the forgotten data point from a gradient that can be computed without using such data.
The main advantage of this definition is that if $\mathcal{M}$ was the oracle algorithm, it would be considered a good unlearning algorithm, whereas for the previous definitions it is not.

Written more formally, I would like to say that $\mathcal{M}$ is a good unlearning algorithm if $\mathbb{P}(\mathcal{M}(\mathcal{A}(D), j) \in B_\beta(\mathcal{A}(D \setminus \{(x_j, y_j)\}))) = 1$ for any $\beta > 0$, where $B_\beta(\mathcal{A}(D \setminus \{(x_j, y_j)\}))$ is the ball of radius β around $\mathcal{A}(D \setminus \{(x_j, y_j)\})$. Note that if the algorithm $\mathcal{A}$ starts from a random initial condition, then $\mathcal{A}(D)$ is a random variable.

For the sake of clarity, I will rewrite this condition in the notation we have been adopting so far.

Definition 7. We say that $\varepsilon(t)$ induces a good unlearning algorithm $\mathcal{M}$ if, for each $\theta_0 \in \Theta_0 \subseteq \Theta$, for each $\beta > 0$ there exist (at least) a feasible initial condition $\theta \in \Theta_0$ and $t < T$ such that:

$$\mathbb{P}[\xi(0, T, \varepsilon(t); \theta_0) \in B_\beta(\xi(0, t, 0; \theta))] = 1.$$

This definition makes sense because by the "continuous dependency from initial conditions" theorem, the flux of those two differential equations will stay arbitrarily close from T on.

Remark. If we can approximate exactly $\theta^\star_{-j}$, then all the definitions are satisfied. Namely, in the case of Linear Regression with the squared loss, the Newton method (IF estimation) is exact and therefore it's a good unlearning algorithm.

Some applications of the influence functions I would eventually like to investigate include:

- recognition of informative images in a set;
- variation of weights for mislabeled samples (in this case, it is useful to study what is the $\varepsilon(t)$ that minimizes the distance from optimum at the end of the training).

Algorithm implementations: D2D vs R2D

An application for which we may want to use influence functions is unlearning a data point (for example for privacy reasons). However, as we have seen, IFs may not work well in the MLP and Deep Learning setting. Therefore, in the literature some algorithms have been developed to unlearn training points without using the highly inefficient LOO retraining. For example, in [11], the authors take inspiration from the *Descend-to-Delete* algorithm (D2D) and create a more efficient version called *Rewind-to-Delete* (R2D).

The former algorithm is very intuitive and consists on simply continuing the training of the model but on the updated dataset where we deleted the target data point. By continuing for enough iteration, there have been proven some theoretical guarantees that the final parameter distribution is in some sense undistinguishable from the one of LOO method.
The problem with this algorithm is that such guarantees only hold in the convex case and cannot be extended to the more general setting (e.g. ReLU and tanh functions are not convex, so this result does not hold).

On the other hand, the R2D algorithm saves a check-point parameter during the training (let's say after the K-th iteration) and then keeps going until iteration T. The idea is that when we receive the request of unlearning the target point, we will continue the training on the updated dataset starting from $\theta(K)$, instead of $\theta(T)$.
The authors of the paper show that also this method yields indistinguishalbility, but in this case the result holds also for non-convex functions and the iterations required are much less than the total training time T (required by LOO).

In [10], the same researchers extend the previous results also to the (projected) SGD case. In particular, in addition to all previous observations, they highlight that D2D provides tighter bounds for the undistinguishability due to its reliance on the convergence to a unique global minimum, while R2D has more loose estimates as it only counts on the underlying contractivity of gradient systems.

Other interesting things we might want to explore

- If our goal is to gain a better understanding of what data points are the most informative during the training, instead of trying to unlearn certain data, the results in [4] might be interesting. In their paper, the authors prove that their Shapley values-based method performs better on this task rather than influence function methods.

- It may be of interest to study Bayesian Influence Functions (BIFs) as well as frequentistic ones. In [8], the researchers present an unlearning method that uses BIFs instead of IFs. The reason for this is we don't need to compute the iHVP to evaluate BIFs, therefore this method works better with more singular loss landscapes. As another consequence, we don't need to evaluate these quantities on local minima (we dont need the hessian to be invertible), which is one of the less realistic hypotheses for IFs.

On the other hand, computations are not always faster than IFs (here, the leading cost is estimating the covariance between two elements) and achieving good results requires more hyperparameter tuning than classical methods.

- Paper about machine unlearning with $\varepsilon \neq 0$ [12]: they introduce a soft-weigthed framework for unlearning data with the aim of improving robustness or fairness. The main point is that if we are not in a privacy-related field, there is no need to completely forget a datapoint; instead, it would be more beneficial to consider it with a lower weight. Moreover, instead of training from scratch the model with the updated weights, the authors propose to correct the result of the usual training with influence functions.

- Something I had not seen yet in machine unlearning, but I have seen for example in plateau explanation [1]: it can make more sense to consider influence functions in certain epochs of the training, not only at the end [9]. Indeed, the importance of a certain data point may change throughout the training.

Bibliography

[1] Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *Foundations of Computational Mathematics*, 25(5):1627–1710, August 2024.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning, December 2020. arXiv:1912.03817 [cs].

[3] R. D. Cook and S. Weisberg. *Residuals and Influence in Regression.* Springer, 1983.

[4] Amirata Ghorbani and James Zou. Data Shapley: Equitable Valuation of Data for Machine Learning, June 2019. arXiv:1904.02868 [stat].

[5] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified Data Removal from Machine Learning Models, November 2023.

[6] Fumio Hayashi. *Econometrics.* Princeton University Press, 2000.

[7] Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. On the Accuracy of Influence Functions for Measuring Group Effects, November 2019. arXiv:1905.13289 [cs].

[8] Philipp Alexander Kreer, Wilson Wu, Maxwell Adam, Zach Furman, and Jesse Hoogland. Bayesian Influence Functions for Hessian-Free Data Attribution, September 2025. arXiv:2509.26544 [cs].

[9] Jin Hwa Lee, Matthew Smith, Maxwell Adam, and Jesse Hoogland. Influence Dynamics and Stagewise Data Attribution, October 2025. arXiv:2510.12071 [cs].

[10] Siqiao Mu and Diego Klabjan. Descend or Rewind? Stochastic Gradient Descent Unlearning, November 2025. arXiv:2511.15983 [cs].

[11] Siqiao Mu and Diego Klabjan. Rewind-to-Delete: Certified Machine Unlearning for Nonconvex Functions, October 2025. arXiv:2409.09778 [cs].

[12] Xinbao Qiao, Ningning Ding, Yushi Cheng, and Meng Zhang. Soft Weighted Machine Unlearning, May 2025. arXiv:2505.18783 [cs].