Weekly meetings

Epsilon dynamic

Note that we can rewrite $\theta(t,s)$ from **??** 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 **??** 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:

$$\theta(t,1) = e^{-Ht}\theta_0 + (\mathrm{Id}_d - e^{-Ht})H^{-1}X^\top y. \tag{0.1}$$

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 $\mathrm{ran}H \oplus \mathrm{ker}H$ and study the dynamic in these two sets.

- In $\mathrm{ker}H$, applying the formula 0.1 we get $\theta(t,1) = \pi_{\mathrm{ker}H}\theta_0$;
- 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 $\mathrm{ker}H = \mathrm{ker}X$ and $\mathrm{ran}H = \mathrm{ran}X^\top$. Therefore, $\pi_{\mathrm{ker}H} = \pi_{(\mathrm{ran}X^\top)^\perp} = I - X^\top(XX^\top)^\dagger X$ (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 $\varepsilon \in (0,1)$ [1], consider $y^j_\varepsilon = (y_1, \dots, \varepsilon y_j, \dots, y_n)$ and similarly X^j_ε and $H^j_\varepsilon = (X^j_\varepsilon)^\top X^j_\varepsilon$. Solving (LRGF) with the above quantities defines:

$$\theta(t,\varepsilon) = e^{-H^j_\varepsilon t}\theta_0 + (\mathrm{Id}_d - e^{-H^j_\varepsilon t})\hat{\theta}_{-j}(\varepsilon). \tag{0.2}$$

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:

$$\min_{\varepsilon \in E_s} \mathcal{E}(\varepsilon) \quad \text{where} \quad \mathcal{E}(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \theta^\star_{-j}\|^2_\Theta \tag{0.3}$$

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

$$\begin{cases} \dot{\theta}(t,\varepsilon(t)) = -\sum_{i \ne j} x_i(x_i^\top \theta - y_i) \\ \theta(0) = e^{-H^j_1 t}\theta_0 + (\mathrm{Id}_d - e^{-H^j_1 t})\hat{\theta}_{-j}(1) \end{cases}.$$

Recall that $H^j_1 = H$ and $\hat{\theta}_{-j}(1)$. At this point, using again 0.2 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:

$$\begin{cases} \dot{\theta}(t,\varepsilon(t)) = -\sum_{i \ne j} x_i(x_i^\top \theta - y_i) - \varepsilon(t)x_j(x_j^\top \theta - y_j) \\ \theta(0) = \theta_0 \end{cases}, \tag{0.4}$$

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

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

and use the method of the Wronskian with the variation of constants, that does not yield a closed-form solution in the general case[2].
Let's see what happens instead for simpler cases doing computations by hand. In the following, consider $s > 0$ fixed.

[1]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
[2]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 $g'(\tau)$:

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

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

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

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:

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

Since both H_{-j} and $e^{-(T-\tau)H_{-j}}$ are positive semidefinite, the second term is always positive. In fact, we can express it as a Mahalanobis distance between $\theta^\star$ and $\theta^\star_{-j}$, where the metric is given by $X_{-j} e^{-(T-\tau)H_{-j}}$.

The first term is more delicate. If H_{-j} and H commute, also the first term is a Mahalanobis distance with the associated matrix being $x_j^\top e^{-TH_{-j}+\tau(H_{-j}-H)}$, but with a negative sign, thus it is always negative.

In general, however, H and H_{-j} do not commute. Indeed, 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 $x_i \parallel x_j$ for each $i \neq j$.

We would still have a chance to get a simple formula if $[[A, B], A] = [[A, B], B] = 0$. But this is not the case:

$$[[x_i x_i^\top, x_j x_j^\top], x_i x_i^\top] = 2(x_i^\top x_j)(\|x_i\|^2 (x_i x_j^\top + x_j x_i^\top) - (x_j^\top x_i) x_i x_i^\top) \neq 0.$$

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 $\varepsilon_{\text{lin},t_1}(t)$ such that:

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

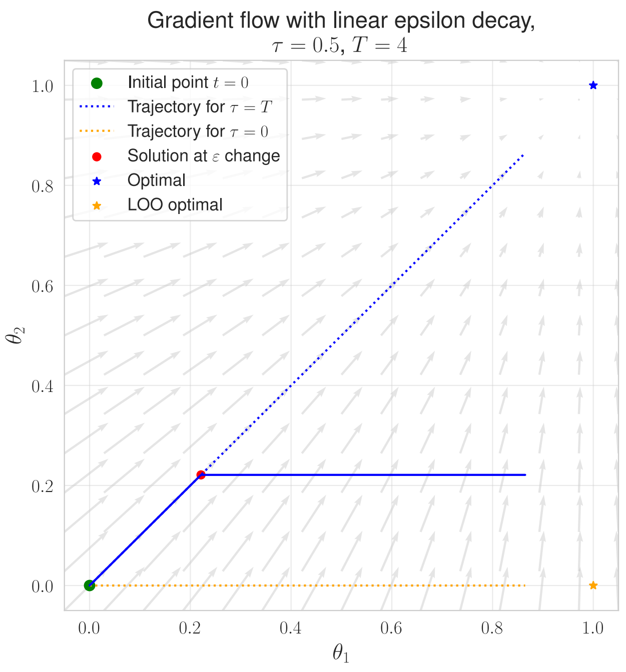


Figure 1: Plot representing the training trajectories for $X = \mathrm{Id}_2$, $y = (1,1)$, $\theta_0 = (0,0)$ and $j = 2$.

The main problem is that now in 0.4 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:

$$\begin{cases} \dot\theta(t) = \left(-\sum_{i\neq j} x_i x_i^\top\right)\theta(t) + \frac{x_j x_j^\top}{s-t_1}\theta(t)\psi(t) + \left(\sum_{i\neq j} x_i y_i + (1 - \frac{s}{(s-t_1)})x_j y_j\right) \\ \dot\psi(t) = 1 \end{cases} . \quad (0.5)$$

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:

$$\dot{\theta}(t) \cdot \frac{\theta^{\star}_{-j} - \theta(t)}{\|\theta^{\star}_{-j} - \theta(t)\|}. \tag{0.6}$$

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 0.6 and computing explicitly the derivative of $\theta(t, \varepsilon)$ from 0.2 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:

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

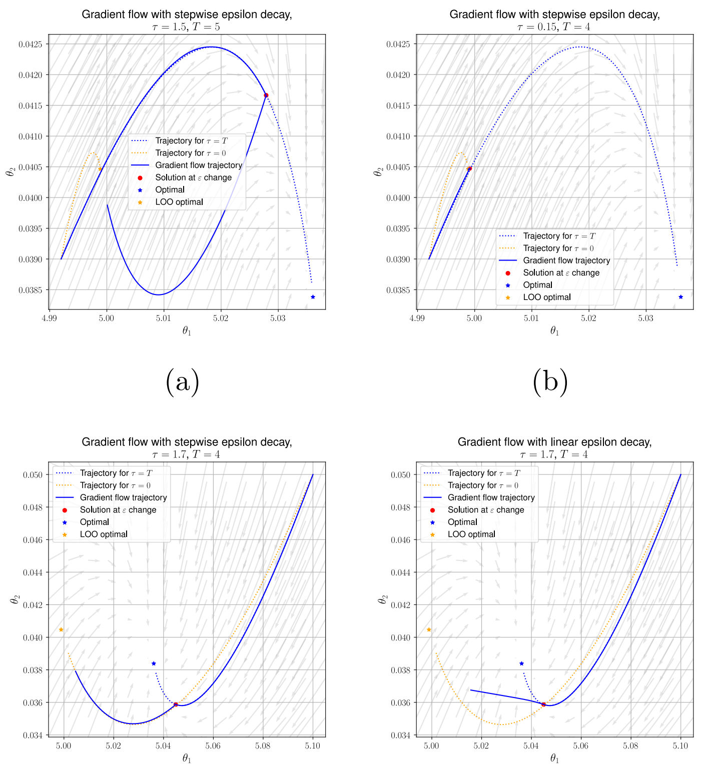
However, this may not be the maximum we can get! Indeed, if $\theta^{\star}_{-j} - \theta(0)$ is for example a line parallel to $\theta^{\star}_{-j} - \theta(0)$, choosing a bigger ε may be better.

Example: consider $X = (1,1)^\top$, $y = (y_1, y_2)^\top$, $j = 1$. In this case, the system 0.4 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 2, the function $\mathbb{1}_{[0,s]}$ appears not to be the



(a) (b) (c) (d)

Figure 2: Phase portrait of the numerical integration of the IVP 0.4 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.

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 0.3, 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.

- $d(\varepsilon) = \frac{1}{2}\|\xi(0,T,\varepsilon;\theta(0)) - \xi(0,+\infty,0;\theta(0))\|^2_\Theta$: 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".

- $d(\varepsilon) = \frac{1}{2}\|\xi(0, T, \varepsilon; \theta(0)) - \xi(0, T, 0; \theta(0))\|^2_\Theta$: 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".

- $d(\varepsilon) = \frac{1}{2}\min_{t>0}\|\xi(0, T, \varepsilon; \theta(0)) - \xi(0, t, 0; \theta(0))\|^2_\Theta$: 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? Measure selection

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 0.4 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* [6]: 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 [3]: 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 1. 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:
$$\xi(0, T, \varepsilon(t); \theta_0) \in B_\beta(\xi(0, t, \mathbf{0}; \theta)).$$

This definition makes sense because by the "continuous dependency from initial conditions" theorem[3], the flux of those two differential equations will stay arbitrarily close

[3]see: `https://poisson.phc.dm.unipi.it/~sgubin/study_res/year2/anal2.pdf` for the proof

from T on.
Additionally, this theorem gives us an explicit bound on the distance between the two trajectories after a time t_1 from the meeting point, if they have minimum distance β:

$$\|\xi(0,\cdot,\mathbf{0};\xi(0,T,\varepsilon(t);\theta_0)) - \xi(0,\cdot,\mathbf{0};\xi(0,t,\mathbf{0};\theta))\|_{\infty,[0,t_1]} \leq e^{Lt_1}\beta,$$

where L is the Lipschitz constant of the $-\nabla_\theta R_j$.

Lemma 0.0.1. *Call $\tilde{\theta}(s) = \xi(0,s,\mathbf{0};\xi(0,T,\varepsilon(t);\theta_0))$ and $\theta(s) = \xi(0,s,\mathbf{0};\xi(0,t,\mathbf{0};\theta))$. Assuming that the data points are normalized, i.e. $\|x_i\|_2 = 1$ for $i = 1,\dots,n$, it holds true that for any $s \in [0,t_1]$:*

$$\|\nabla L(\tilde{\theta}(s)) - \nabla L(\theta(s))\|_2 \leq ne^{Lt_1}\beta.$$

Proof. Assume $\Theta \subseteq \mathbb{R}^d$ with the Euclidean norm. Let's compute the difference between the two gradients:

$$\begin{aligned}\|\nabla L(\tilde{\theta}(s)) - \nabla L(\theta(s))\|_2 &= \|\sum_{i=1}^n x_i(x_i^\top\tilde{\theta}(s) - y_i) - \sum_{i=1}^n x_i(x_i^\top\theta(s) - y_i)\|_2 = \\ &= \|\sum_{i=1}^n x_i x_i^\top(\tilde{\theta}(s) - \theta(s))\|_2 \leq \sum_{i=1}^n \|x_i\|_2^2\|\tilde{\theta}(s) - \theta(s)\|_2 \leq ne^{Lt_1}\beta.\end{aligned}$$

■

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).

Critique and new proposal

The previous definition I proposed doesn't actually take into consideration the probability of our end point being close to a feasible trajectory. Therefore, if the set of initial conditions is the whole space, the former definition is empty.
To address this problem, I would like to modify the definition, taking into account the following family of probablity measures.

Let $(\mathbb{P}_t)_{t\in[0,+\infty]}$ be a family of probabilty measures on Θ and define $\text{supp}\mathbb{P}_0 = \Theta_0 \subseteq \Theta$ as the set of possible initial conditions. Each probability $\mathbb{P}_t$ is then defined as:

$$\mathbb{P}_t[B] = \mathbb{P}_0[\xi_t^{-1}(B)] \quad \forall B \in \mathcal{B}(\Theta),$$

where $\xi_t(\theta_0) = \xi(0, t, \mathbf{0}; \theta_0)$ is the flux of the gradient flow at time t and $\mathcal{B}(\Theta)$ is the Borel sigma algebra over Θ. Note that, since $\xi_t : \Theta \to \Theta$ is continuous, $\xi_t^{-1}(B) \in \mathcal{B}(\Theta)$ for any $B \in \mathcal{B}(\Theta)$ and thus $\mathbb{P}_t$ is well defined.

Remark. From the definition of $\mathbb{P}_t$ we can observe that $\text{supp}\mathbb{P}_\infty = \{\theta \in \Theta \mid \nabla_\theta R_j = 0\}$, i.e., $\mathbb{P}_\infty$ is supported only on the set of critical points for the LOO loss function.

Definition 2. We say that $\varepsilon(t)$ induces an (α, δ)-good unlearning algorithm at time s if, for all $\theta_0 \in \Theta_0$:

$$\frac{\mathbb{P}_s[B_\delta(\xi(0, s, \varepsilon; \theta_0))]}{\mathbb{P}_s[B_\delta(\xi(0, s, \mathbf{0}; \theta_0))]} \geq 1 - \alpha.$$

Let's see what this definition implies for extreme values of the parameters:

- if $s = 0$, then every algorithm is a good unlearning algorithm (it makes sense since in both LOO and modification we can start from any point in Θ_0 so we don't need to modify anything);
- if $s = +\infty$, then also every algorithm works, since $\varepsilon \in E_s$ and therefore it is constantly 0 after T.
- for $\delta \to \infty$ then $\alpha \to 0$, and, for $\delta \to 0$, $\alpha \to 1$.
- if $\Theta_0 = \{\theta_0\}$, call $\bar{\delta} = \sup_{t>0} \|\xi(0, t, \varepsilon; \theta_0) - \xi(0, t, \mathbf{0}; \theta_0)\|$. Then we can express $\alpha(\delta)$ explicitly as $\alpha = \mathbb{1}_{[\bar{\delta}, +\infty]}(\delta)$;

To make the definition even less strict, we may take the mean value over θ_0 instead of requiring the condition to hold for each $\theta_0 \in \Theta_0$.

Figure 3 reports a visual representation of the previous definition. The idea here is that we want the attracting regions to be the same for both the learning dynamics. Namely, if we take two gradient flows initialized on the same point relative to two different datasets, the further the training goes, the more distinguishable the distribution of parameters become. In fact, the estimators generated by the neural nets learn the characteristics of the data in a sequential way [12], causing the observational noise to be learnt at the end of training. This means that the longer the training goes, the more specialised the model becomes, leading to its parameters to converge further away from each others.

MIA and overfitting

From the previous discussion, it seems like algorithms that overfit are bad then tested on our measure.
Does it makes sense from a privacy point of view?
The answer is yes. In fact, let's consider the following very naive example of Membership Inference Attack: in a white-box setting (the attacker has full access to the model), if our algorithm completely overfits, it means that it will have 0 loss on the training set.

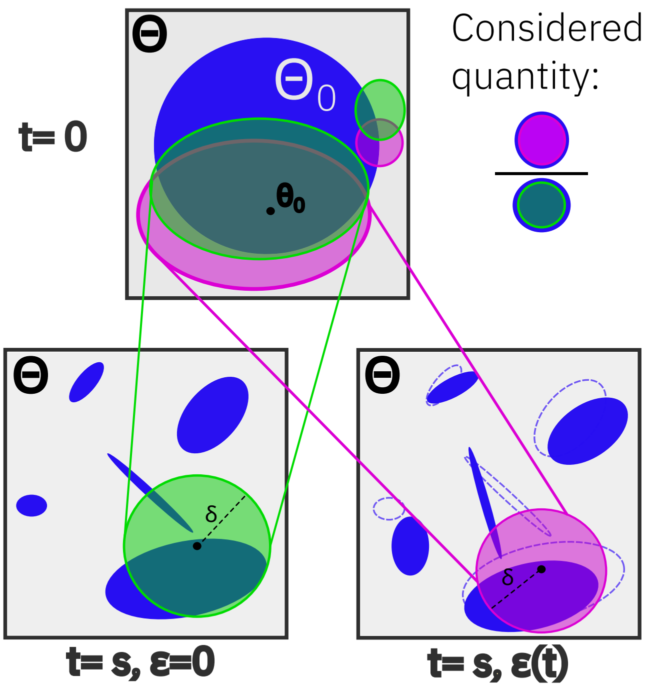


Figure 3: Example representation of the meaning of $\mathbb{P}_s$ when $\Theta_0 \subset \mathbb{R}^2$.

Therefore, the attacker can check if a point z has been used in the training by just checking if the loss of the model on z is 0 or not.

Even in the less extreme setting of [1], the quantity that the authors consider (pp.8,9) increases if we choose an overfitting algorithm. On the other hand, if we could have an oracle procedure that could give us the generating function of the dataset (to which observational noise is added), that algorithm would be independent of initial dataset and therefore be a private algorithm. However, this concept may be more similar to differential privacy rather than unlearning.

Indeed, if $\mathcal{A}$ is such an algorithm, we could write $\mathcal{A}(\mathcal{D}) = \mathcal{A}(\mathcal{D}')$ for any two dataset whose elements are drawn from the same distribution. This means that $\mathcal{A}$ is a 0-DP algorithm. Maybe, to address this problem could be interesting to use the Vicinal Risk Minimization algorithm [4], instead of the ERM one, as it looks to me that may get closer to the optimal distribution rather than the overfitting one.

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 [10], 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 [9], 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 [5] 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 [7], 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 $\varepsilon \neq 0$ [11]: 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 [2]: it can make more sense to consider influence functions in certain epochs of the training, not only at the end [8]. Indeed, the importance of a certain data point may change throughout the training.

Bibliography

[1] Eric Aubinais, Elisabeth Gassiat, and Pablo Piantanida. Fundamental Limits of Membership Inference Attacks on Machine Learning Models, October 2025.

[2] 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.

[3] 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].

[4] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal Risk Minimization. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

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

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

[7] 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].

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

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

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

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

[12] Maria Refinetti, Alessandro Ingrosso, and Sebastian Goldt. Neural networks trained with SGD learn distributions of increasing complexity. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan

Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28843–28863. PMLR, 23–29 Jul 2023.