Abstract

Machine-learning pipelines deployed in practice must learn continuously, serve predictions, and honor "right-to-erasure" requests-—yet state-of-the-art unlearning methods remain train-then-forget scripts that freeze the model once deletions begin. We introduce memory pairs: a stream-native, two-algorithm construct $(A, \bar{A})$ whose single API supports simultaneous learning, inference, and certified unlearning on unbounded data streams. The memory pair achieves sub-linear cumulative regret $R_T = O(\sqrt{T})$ and provides online (ε, δ)-unlearning for any sequence of at most $m^\star(\gamma) = \Theta(n_\gamma)$ deletions.

These results establish memory pairs as the first practical framework that is provably private, capacity-aware, and relentlessly adaptive– bridging the gap between regulatory compliance and real-time model improvement.

1 Introduction

Industrial machine-learning (ML) pipelines have crystalized around a *train-then-serve* recipe:

- *Bulk ingestion:* Raw events are accumulated in a data lake, periodically snapshot-cleaned, and exported as an IID matrix.
- *Isolated Training:* a compute cluster solves an empirical risk minimization problem once per refresh cycle
- *Static Deployment:* the frozen model is pushed behind an inference service where it lies, dormant and unlearning, until the next refresh

This workflow has powered recommender systems, fraud detectors, and language models at scale, yet two structural deficiencies have become apparent.

1.1 Staleness of Inference

Real-world data distributions drift continuously–think of evolving slang on social media or shifting fraud patterns during a holiday season. Quarterly retrains let accuracy degrade for weeks, and hot-fixes require ad-hoc learning patches that violate the original training guarantees.

1.2 Regulatory Pressure to Forget

Global statutes such as the EU GDPR and the California CCPA enshrine a *right to be forgotten*. A user may demand that all personal data–and any model behavior derived from it–be removed. Traditional pipelines can do little more than log the request and schedule a full retrain, incurring days of downtime, terabytes of re-processing, and substantial carbon cost.

Certified-unlearning research has begun to address the privacy side. The frameworks of Sekhari et al. and Qiao et al. provide provable (ε, δ) guarantees that a post-hoc deletion yields (in distribution) the same parameters as a fresh retrain. However, both assume an offline model that never learns again after deletion. Production models don't have the luxury of an offline training loop, and must continue to learn as the context requires.

2 Stream-Native Learning

The next generation of ML systems must treat *learning*, *forgetting*, and *predicting* as first-class, intertwined operations performed on a live data stream. To this end we introduce the **Memory Pair** pipeline:

- *Stream-native learning:* A learner A_{on} ingests each example (x_t, y_t) and performs an $O(d)$ update in micro-seconds, maintaining only lightweight sketches instead of per-sample gradients.
- *Deferred inference gate:* Predictions are withheld until the running sample count exceeds a theoretically derived sample-complexity threshold, ensuring the first answer is already PAC-competitive.
- *Symmetric unlearning:* A paired algorithm $\bar{A}_{\text{on}}$ accepts deletion requests via the same API and issues a one-step *negative* update that preserves the learner's regret bound. The model is guaranteed to be within an arbitrary precision of the ideal cold-start model trained from scratch.
- *Live deletion-capacity odometer:* Each unlearning step depletes the model's deletion capacity budget. When a model unlearns its capacity, the learner is flagged for retraining to preserve accuracy. We ensure tight accounting on model capacity to not only take models down on time, but to stretch the duration of a model's reliability.

2.1 Contributions

The batch unlearning framework from Sekhari et al. provides the theoretical inspiration for true online unlearning. By focusing on population risk, they establish that unlearning with generalization guarantees is possible. However, their algorithm is designed for a static, i.i.d. world.

Our Memory Pair framework adapts these core ideas for a dynamic, online setting. We replace the assumption of i.i.d. data with a non-stationary stream, substitute the excess risk metric with the more robust notion of cumulative regret, and swap the expensive, offline Hessian-based update with a lightweight, stream-native L-BFGS approximation.

The Memory Pair is not just a generalization of the batch algorithm but a *necessary evolution* designed to handle the continuous learning, forgetting, and inference demands of a live data stream.

3 Preliminaries

We start by framing our learners within the online learning paradigm. Batch learning relies on minimizing an empirical risk, or population risk as used in Sekhari et al. We instead use the seminal notion of regret, which appropriately compares the online learner to the best possible model in hindsight.

This demonstrates the key factor of the Sekhari paper. By requiring the model to satisfy population risk guarantees, the model is expected to perform well against some true but unknown parameter distribution. This is very similar to the case of online learning because the model's evaluation is continuous, and so it makes sense to use online learning tools to evaluate our learner, such as the notion of regret.

Definition 1 (Online learner with vanishing average regret)**.** *Let $\mathcal{W} \subseteq \mathbb{R}^d$ be a convex hypothesis set and $\{\ell_t\}_{t=1}^{\infty}$ an arbitrary sequence of (not necessarily stationary) loss functions $\ell_t : \mathcal{W} \to \mathbb{R}_{\geq 0}$. An algorithm $\mathcal{A}$ that, after observing an outcome (x_t, y_t), outputs a weight vector $w_t = \mathcal{A}(x_{1:t}, y_{1:t}) \in \mathcal{W}$ is called an* online learner *iff for every outcome sequence $\{(x_t, y_t)\}_{t=1}^{\infty}$ the* average regret

$$\frac{1}{n} R_n(\mathcal{A}) \;:=\; \frac{1}{n}\Big(\sum_{t=1}^{n} \ell_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{n} \ell_t(w)\Big)$$

converges to zero, i.e.

$$\frac{1}{n} R_n(\mathcal{A}) \;\xrightarrow[n\to\infty]{}\; 0.$$

In the online setting, a model is expected not only to learn incrementally, but also process data deletion requests with some measure of guarantee. The unlearning algorithm must yield a model that is statistically indistinguishable from the model's ideal state: a model trained from scratch without the offending data. This tight, state-dependent relationship requires a unified framework where learning and unlearning procedures are tightly coupled.

We formalize the notion of a memory pair as a coupled learning-unlearning algorithm designed to operate on a stream of insertions and deletions. A valid memory pair must simultaneously guarantee learning accuracy constraints, unlearning fidelity, and the capacity to handle a certain number of deletions.

Definition 2 (Memory Pair[1]). *Let $\{E\}_{t=1}^{N}$ be an event stream where $E_t \in \{\mathit{insert}(x_t, y_t), \mathit{delete}(u_t)\}$. A pair of algorithms $(A, \bar{A})$ with shared state θ_{t-1} acts as follows:*

$$\textit{learn step: } \theta_t = A(\theta_{t-1}, E_t)$$
$$\textit{unlearn step: } \bar{\theta}_t = \bar{A}(\theta_{t-1}, E_t)$$

Denote by $\tilde{\theta}_t$ the ideal replay model: the model state achieved by processing all INSERT *events up to time t except those whose index has appeared in a prior* DELETE *event. Fix an accuracy target γ, confidence δ, privacy budget $(\varepsilon^\star, \delta^\star)$, and deletion capacity m. Then $(A, \bar{A})$ is a $(\gamma, m, \varepsilon^\star, \delta^\star)$-*memory pair *if the following hold for every event stream and for all horizons N:*

1. ***Online accuracy.*** *The learner's average regret over the realized stream satisfies*

$$\Pr\big[\tfrac{1}{N} R_N(\theta) \le \gamma\big] \;\ge\; 1 - \delta.$$

2. (ε, δ)***-Online Unlearning.*** *For every measurable set $S \subseteq \mathcal{W}$ and every time $t \le N$, the unlearned model $\bar{\theta}_t$ is $(\varepsilon^\star, \delta^\star)$-indistinguishable from the ideal replay model $\tilde{\theta}_t$:*

$$\Pr[\tilde{\theta}_t \in S] \le e^{\varepsilon^\star} \Pr[\bar{\theta}_t \in S] + \delta^\star \quad \textit{and} \quad \Pr[\bar{\theta}_t \in S] \le e^{\varepsilon^\star} \Pr[\tilde{\theta}_t \in S] + \delta^\star.$$

3. ***m-deletion capacity.*** *Let D_N be the number of* DELETE *events up to N. If $D_N \le m$, then Conditions (I)–(II) hold.*

[1]Randomness is over the internal coins of (A, $\mathit{bar}A$); guarantees are *uniform* over the realized insert–delete stream.

The memory pair definition allows for interleaved insertion, deletion, and inference operations. It combines the accuracy guarantees of learning algorithms with the certifiable unlearning guarantees that are ubiquitous in machine unlearning. Bundling the learning and unlearning algorithms is a natural extension to a body of work that has increasingly found them inseparable. Indeed, recent Hessian-free methods of machine unlearning explicitly use the learning process to gain gradient information that is later used to unlearn.

3.1 Related Work

This algorithm is inspired by the Newton Step algorithm for certifiable machine unlearning from Sekhari et al. The (now seminal) paper demonstrates the inherent difficulty of the unlearning problem when minimizing population risk. They instead show the nontrivial bound of deletion capacity if the learner is to perform well in production. Specifically, the notion that performance should generalize for points not yet seen by the model.

There is a very natural relationship between performing well on an unseen test dataset and performing well on a sequence of incrementally-revealed stream events in the online setting. We build on Sekhari et al.'s initial work by pairing the learning and unlearning processes into a single framework, expanding performance guarantees to the online case, and providing online generalizations of the notions of sample complexity and deletion capacity. We additionally replace the expensive Hessian inversion with an online variant of L-BFGS that removes the linearly-scaled memory footprint.

4 Memory Pair

4.1 Newton-Step Approach to Machine Unlearning

Machine unlearning has a rich tool set of algorithms to approximate the ideal retrained model. One of the strongest is Sekhari et al.'s application of the Newton Step optimization method to remove the influence of a set of unlearned points. With careful noise calibration, the unlearned method is certifiably indistinguishable from the ideal cold-start retrained model.

The unlearning approximation draws its strength from the second-order information of the model's training curve stored in the Hessian. Scaling our average loss by the the inverse Hessian allows us to approximate the impact of those unlearned points on the model's parameters in the form of a model update. A small amount of calibrated noise is then added to those weights

to ensure certifiable indistinguishability from the retrained model. But with an $O(d^2)$ runtime complexity, the option is not computationally efficient.

There has also been recent work into Hessian-Free unlearning, which stores gradient approximations of the training data to create a suitable estimate of the inverse Hessian using Hessian Vector Products. But with a memory footprint that scales linearly, it is not feasible for the online use case where $n >> d$.

4.2 Quasi-Newton Approaches to Online Learning

We specifically target the Hessian inversion operation of the Newton-Step update for our unlearning algorithm. We replace the Hessian inversion with an online variant of the L-BFGS optimization algorithm. This eliminates the need for the precomputation stage present in Hessian-Free and standard Newton Step unlearning. In fact, it enables a broader prediction space entirely. With no precompute required for deletion, interleaved learning and unlearning operations can be processed sequentially as the events are read from a data stream. This is especially attractive for the prospect of federated learning with on-device learning. More on this in Future Work.

The insert method implements online learning without the expensive Hessian inversion that's typically done for the model update. It approximates the same using an online variant of BFGS optimization specifically formulated for constrained memory environments.

The method efficiently updates the gradient near-instantaneously by computing the current gradient with respect to the point. The curvature points from the L-BFGS are then used to approximate something like a Hessian inversion. The key is to use the second-order curvature information from the saved points to approximate the curvature of the whole function.

When the curvature points are used to approximate the second-order information for a whole function, the choice of points holds some weight. The online variant of L-BFGS is used for precisely this reason, easily ingesting and disposing of curvature points as additional information is fed through the insertion operations.

The deletion method works by computing the unlearned point's gradient using the model's learned parameters. The influence of the point is approximated using the L-BFGS approximation and used in place of the explicitly calculated inverse Hessian in typical Newton methods.

Before the weight parameter is returned, there is a calibrated amount of noise added to the estimate. This amount of noise serves two purposes: (1) differential privacy with respect to the unlearned data and (2) blurring the remaining difference between the unlearned model and the ideal cold-

Algorithm 1 Memory Pair Insertion

Require: New data point (x, y), Model state (θ, lbfgs)

$g_{\text{old}} \leftarrow \nabla_\theta \mathcal{L}(\theta; x, y)$ ▷ Compute gradient at current parameters

$d \leftarrow \text{lbfgs.direction}(g_{\text{old}})$ ▷ Compute quasi-Newton direction

$\theta_{\text{new}} \leftarrow \theta + \alpha d$ ▷ Update parameters with learning rate α

$s \leftarrow \theta_{\text{new}} - \theta$ ▷ Compute change in parameters

$\theta \leftarrow \theta_{\text{new}}$ ▷ Commit the parameter update

$g_{\text{new}} \leftarrow \nabla_\theta \mathcal{L}(\theta; x, y)$ ▷ Compute gradient at new parameters

$y_{\text{vec}} \leftarrow g_{\text{new}} - g_{\text{old}}$ ▷ Compute change in gradient

$\text{lbfgs.add\_pair}(s, y_{\text{vec}})$ ▷ Update L-BFGS curvature memory

start. Too little noise will violate the indistinguishability requirement for the incrementally unlearned model from the ideal retrain. Too much noise will violate our regret convergence guarantees.

When chosen carefully, the noise acts as a shock absorber for the model. It dampens the unlearning effect that would leak the deleted user's information. Our variance is specifically chosen to outpace both the (1) Hessian-sketch error of L-BFGS and (2) the cumulative impact of all future deletions. We can estimate the second because we explicitly limit the number of deletions with deletion capacity and ensure its compliance with our capacity odometer.

The unlearning algorithm is defined analogously. It takes the point to be deleted and computes the gradient of the loss at the point of prediction. The L-BFGS method is used to estimate the influence of that point on the historical trajectory of the algorithm (approximated using the curvature points) to estimate the same Hessian inversion performed by the Newton Step.

4.3 Privacy Accounting via a Deletion Odometer

A critical component of the Memory Pair framework is its ability to manage the trade-off between unlearning fidelity and model accuracy. This is accomplished through a strict accounting mechanism that we term a **privacy odometer**. Unlike a dynamic process where capacity might change, our framework treats the privacy budget as a finite resource that is determined at initialization and consumed with each deletion request.

The model is initialized with a fixed **m-deletion capacity**, corresponding to the `max_deletions` parameter, which represents the total number of unlearning operations the model can perform before its guarantees are void. The total privacy budget, $(\epsilon_{\text{total}}, \delta_{\text{total}})$, is apportioned uniformly across this

Algorithm 2 Memory Pair Deletion

Require: Data point to delete (x, y), Model state (θ, lbfgs), Budgets $(\epsilon_{\text{step}}, \delta_{\text{step}})$

if deletions_so_far $\geq K$ **or** len(lbfgs) == 0 **then**
  raise RuntimeError
end if

$g \leftarrow \nabla_\theta \mathcal{L}(\theta; x, y)$ ▷ Compute gradient of the point to unlearn
$d \leftarrow \text{lbfgs.direction}(g)$
$\theta \leftarrow \theta - d$ ▷ Remove the approximate influence

$\text{sensitivity} \leftarrow \|d\|_2$
$\sigma \leftarrow \frac{\text{sensitivity} \cdot \sqrt{2 \ln(1.25/\delta_{\text{step}})}}{\epsilon_{\text{step}}}$ ▷ Calibrate noise
$\eta \sim \mathcal{N}(0, \sigma^2 I_d)$
$\theta \leftarrow \theta + \eta$ ▷ Inject noise to ensure unlearning guarantee

$\epsilon_{\text{spent}} \leftarrow \epsilon_{\text{spent}} + \epsilon_{\text{step}}$ ▷ Update privacy odometer
deletions_so_far $\leftarrow$ deletions_so_far $+ 1$

capacity. The per-step budget for each deletion is therefore:

$$\epsilon_{\text{step}} = \frac{\epsilon_{\text{total}}}{2 \cdot m} \quad \text{and} \quad \delta_{\text{step}} = \frac{\delta_{\text{total}}}{2 \cdot m}$$

This apportionment ensures that the cumulative privacy loss remains bounded after the model has serviced all m deletion requests.

The framework acts as accountant by tracking two metrics: `deletions_so_far` and `eps_spent`. When a `delete` operation is called, the accountant first checks if the capacity has been exceeded. If `deletions_so_far` has reached m, a `RuntimeError` is raised, halting further unlearning and signaling that the model must be retrained from scratch to preserve its integrity.

This finite capacity is not an arbitrary limitation but a necessary consequence of the inherent tension between privacy and accuracy. Each deletion operation injects calibrated noise into the model's parameters to satisfy the (ϵ, δ)-unlearning guarantee. While a single injection has a negligible effect, the cumulative impact of this noise degrades model accuracy. The m-deletion capacity thus represents the maximum number of deletions the model can sustain while ensuring that the cumulative noise does not overwhelm the learning signal and violate the learner's average regret bound, γ.

5 Theoretical Evaluation

The move from batch to stochastic setting is made easier by Sekhari et al.'s proof of an unlearning algorithm with guaranteed performance on unseen data. Such a tool wonderfully translates the stochastic notion of population risk to the online notion of regret. While population risk measures the model's performance against a static underlying dataset, cumulative regret evaluates our learner against the *best static model in hindsight for the specific, realized sequence of data.*

Our goal becomes more complex because of the optimizations used to approximate the Hessian. We first establish the sensitivity of a deletion performed using online L-BFGS before using that information to ensure the unlearned model is private.

5.1 Analyzing the Impact of a Single Deletion

We first analyze the impact of a single DELETE operation on the model's state. We use the convexity assumptions that are common in linear optimization to derive a bound in terms of the strength of the convexity assumption, λ, and the number of parameters, L.

Lemma 1 (Bounded Influence of a Single Deletion)**.** *For a deletion operation on a data point $z = (x, y)$, let the influence direction be d. Under Assumptions A and B, the L2-norm of the influence direction is bounded such that:*

$$||d||_2 \leq \frac{L}{\lambda}$$

Assumption 1 (L-Bounded Gradients)**.** *The loss function $\mathcal{L}(\theta; z) = \frac{1}{2}(\theta^T x - y)^2$ has L-bounded gradients for any data point z. For $g = \nabla_\theta \mathcal{L}(\theta; z)$, we assume $||g||_2 \leq L$.*

Assumption 2 (Stable L-BFGS Approximation)**.** *The online L-BFGS procedure maintains an inverse Hessian approximation, B^{-1}, whose spectral norm is bounded. This reflects the λ-strong convexity of the learning problem, such that $||B^{-1}||_2 \leq 1/\lambda$.*

Proof. The unlearning algorithm computes the influence direction d by approximating the Newton step $d \approx -B^{-1}g$, where g is the gradient for the deleted point. Taking the L2-norm and applying the property of matrix norms, we have:

$$||d||_2 = || - B^{-1}g||_2 \leq ||B^{-1}||_2 ||g||_2$$

Substituting the bounds from Assumption 2 ($||B^{-1}||_2 \leq 1/\lambda$) and Assumption 1 ($||g||_2 \leq L$), we arrive at the bound:

$$||d||_2 \leq \frac{L}{\lambda}$$

□

The key is in the calibration. We use the upper bound of the norm of the step direction to indicate the "worst-case" of a single deletion. We're then able to carefully calibrate the noise in our Gaussian Mechanism to "blur" the effect of the deletion itself. This ensures that the deletion of one point is statistically indistinguishable from that of another, and prevents adversarial information leakage.

Theorem 1. *Let the influence direction d be bounded as in Lemma 1. If noise $\eta \sim \mathcal{N}(0, \sigma^2 I)$ is drawn with variance σ^2 calibrated such that:*

$$\sigma \geq \frac{(L/\lambda)\sqrt{2\ln(1.25/\delta)}}{\epsilon}$$

The updated model $\bar{\theta} = \theta - d + \eta$ satisfies (ϵ, δ)-unlearning with respect to the deleted point.

Proof. The proof follows from the application of the Gaussian mechanism for differential privacy. The mechanism guarantees (ϵ, δ)-differential privacy for a function if Gaussian noise is added with a standard deviation σ calibrated to the function's L2 sensitivity, $\Delta_2 f$.

1. From Lemma 1, we established that the L2 sensitivity of our unlearning operation is bounded by $\Delta_2 f = ||d||_2 \leq L/\lambda$.
2. The standard calibration for the Gaussian mechanism requires the noise scale σ to be at least $\frac{\Delta_2 f \sqrt{2\ln(1.25/\delta)}}{\epsilon}$.
3. By substituting our sensitivity bound, we set the noise level as required by the theorem.

Adding this calibrated noise to the parameters ensures that the output model $\bar{\theta}$ is (ϵ, δ)-indistinguishable from a model trained without the deleted point, thus satisfying the definition of unlearning. □

5.2 Analyzing Cumulative Performance Over a Stream

We then show that the performance guarantee from Theorem 1 holds over any valid sequence of at most m deletions. We do this by proving Condition 1 from the memory pair definition.

We adapt the proof from Mokhtari et al. that demonstrates convergence of the stochastic L-BFGS method with probability $p = 1$.

Lemma 2 (Positive Curvature Condition)**.** *Let the variable variation be* $v_t = w_{t+1} - w_t$ *and the stochastic gradient variation be* $\hat{r}_t = \hat{s}(w_{t+1}, \tilde{\theta}_t) - \hat{s}(w_t, \tilde{\theta}_t)$*. If Assumption 3 holds, the inner product of these variations is strictly positive and bounded below:*

$$\hat{r}_t^T v_t \geq \tilde{m}||v_t||^2$$

where $\tilde{m} > 0$ *is the lower bound on the eigenvalues of the instantaneous Hessian.*

Assumption 3 (Bounded Instantaneous Hessian)**.** *The per-step loss function* $l_t(w)$ *is twice differentiable, and the eigenvalues of its Hessian,* $\nabla^2 l_t(w)$*, are bounded between constants* $0 < \tilde{m}$ *and* $\tilde{M} < \infty$*.*

Proof. The proof relies on the Mean Value Theorem. The gradient variation $\hat{r}_t$ can be expressed as $\hat{r}_t = \bar{H}_t v_t$, where $\bar{H}_t$ is the average Hessian of the loss function along the segment from w_t to w_{t+1}.

1. We can write the inner product as $\hat{r}_t^T v_t = (\bar{H}_t v_t)^T v_t = v_t^T \bar{H}_t v_t$.
2. Since the eigenvalues of the instantaneous Hessian are lower-bounded by $\tilde{m}$ (Assumption 3), the eigenvalues of the average Hessian $\bar{H}_t$ are also lower-bounded by $\tilde{m}$.
3. Therefore, the quadratic form is bounded: $v_t^T \bar{H}_t v_t \geq \tilde{m}||v_t||^2$.

Since $\tilde{m} > 0$ and $||v_t||^2 \geq 0$, the inner product is strictly positive whenever $v_t \neq 0$. This ensures the L-BFGS updates are well-defined. □

Lemma 3 (Bounded Trace and Determinant of the Hessian Approximation)**.** *Consider the Hessian approximation* B_t *generated by the online L-BFGS updates. If Assumption 3 holds, then the trace of* B_t *is uniformly upper-bounded and its determinant is uniformly lower-bounded for all* $t \geq 1$*:*

1. $tr(B_t) \leq (n + \tau)\tilde{M}$
2. $det(B_t) \geq \frac{\tilde{m}^{n+\tau}}{[(n+\tau)\tilde{M}]^\tau}$

Proof. The proof follows the method of Mokhtari and Ribeiro (2015).

1. The trace bound is established by recursively applying the trace operator to the BFGS update rule. At each step, the trace increases by at most $\tilde{M}$, the upper bound on the eigenvalues of the instantaneous Hessian. Summing these increases over the τ steps in the L-BFGS memory window and adding the trace of the initial matrix (which is also bounded by $n\tilde{M}$) yields the result

2. The determinant bound is found by recursively applying the determinant operator. The recursive formula for the determinant of the updated matrix is shown to be lower-bounded at each step by a factor related to $\tilde{m}$ and the trace bound. Compounding these factors over the τ memory steps results in the stated lower bound.

□

Lemma 4 (Uniformly Bounded Eigenvalues of the Hessian Approximation)**.** *If Assumption 3 holds, the eigenvalues of the Hessian approximation matrix B_t are uniformly bounded for all $t \geq 1$. Specifically, there exist constants $c > 0$ and $C < \infty$ such that:*

$$cI \leq B_t \leq CI$$

where $C = (n+\tau)\tilde{M}$ and $c = \frac{\tilde{m}^{n+\tau}}{[(n+\tau)\tilde{M}]^{n+\tau-1}}$.

Proof. The bounds are a direct consequence of Lemma 3.

1. **Upper Bound (C):** The trace of a positive definite matrix is the sum of its positive eigenvalues. Therefore, any single eigenvalue must be less than or equal to the trace. From Lemma 3, we have $\lambda_{max}(B_t) \leq tr(B_t) \leq (n+\tau)\tilde{M} = C$.

2. **Lower Bound (c):** The determinant of a matrix is the product of its eigenvalues. For any eigenvalue λ_j, we have $\lambda_j = det(B_t)/\prod_{k \neq j} \lambda_k$. Using the lower bound for the determinant from Lemma 3 and the upper bound for each of the other $n-1$ eigenvalues, we establish the uniform lower bound c.

□

5.3 Composable Unlearning and the Privacy Odometer

We show that the regret bounds established for a single event hold across the entire stream.

Theorem 2 (Stream-wide privacy & regret guarantee)**.** *Fix a privacy target* $(\varepsilon^\star, \delta^\star) \in (0,1]^2$ *and a maximum deletion capacity* $m \in \mathbb{N}$*. Let the memory pair* $(A, \bar{A})$ *operate on a stream of* T *events that contains at most* m *delete requests. During the* j^{th} *delete* $(1 \le j \le m)$ *the unlearning routine*

1. *computes the influence vector* $d_j = -B_{t_j}^{-1} \nabla \ell_{t_j}(w_{t_j})$*, which is bounded by Lemma 1*
2. *adds Gaussian noise* $\eta_j \sim \mathcal{N}\big(0, \sigma_{step}^2 \mathbf{I}_d\big)$ *with*

$$\sigma_{step}^2 \;=\; \Big(\tfrac{L}{\lambda}\Big)^2 \frac{2\ln\big(1.25/\delta_{step}\big)}{\varepsilon_{step}^2}, \qquad \varepsilon_{step} := \frac{\varepsilon^\star}{m}, \quad \delta_{step} := \frac{\delta^\star}{m},$$

and sets $w_{t_j}^{new} = w_{t_j} - d_j + \eta_j$

Then, for any *sequence of at most* m *deletions and* any *adversarially chosen stream of* T *loss functions* $\{\ell_t\}_{t=1}^T$ *that satisfy Assumption* **??**,

1. ***Privacy / unlearning.*** *The entire weight sequence* $\{w_t^{new}\}_{t=1}^T$ *is* $(\varepsilon^\star, \delta^\star)$*-indistinguishable from the ideal replay sequence* $\{\tilde{w}_t\}_{t=1}^T$ *in the sense of Definition 2 (II).*
2. ***Utility / regret.*** *With probability at least* $1 - \delta^\star - \delta_{\mathrm{B}}$ *(for an arbitrary* $\delta_{\mathrm{B}} \in (0,1)$*), the cumulative regret against the best fixed comparator* $w^\star \in \arg\min_{w \in \mathcal{W}} \sum_{t=1}^T \ell_t(w)$ *obeys*

$$R_T \;=\; \sum_{t=1}^{T} \big[\ell_t(w_t^{new}) - \ell_t(w^\star)\big] \;\le\; GD\sqrt{cCT} + \frac{mL}{\lambda}\sqrt{\frac{2\ln\big(1.25m/\delta^\star\big)}{\varepsilon^\star}}\,\sqrt{2\ln(1/\delta_{\mathrm{B}})},$$

where the first term is the deterministic $\tilde{O}\big(\sqrt{T}\big)$ *bound from Theorem* **??** *and the second term captures the additional loss incurred by the injected noise. Consequently,* $R_T = O\big(\sqrt{T}\big)$ *as* $T \to \infty$*.*

Proof. **(i) Privacy.** Each deletion step is a Gaussian mechanism whose $(\varepsilon_{\text{step}}, \delta_{\text{step}})$ parameters are calibrated exactly as in Theorem **??**. By basic sequential composition for differential privacy, the m deletions together are $(m\varepsilon_{\text{step}}, m\delta_{\text{step}})$-DP, i.e. $(\varepsilon^\star, \delta^\star)$-DP, which coincides with the online-unlearning requirement of Def. 2(II).

(ii) Regret. Let R_T^0 denote the regret incurred by the noise-free iterates $\{\hat{w}_t\}$ obtained from Algorithm 1 (insertions) and noiseless Algorithm 2 (deletions). Theorem **??** gives $R_T^0 \le GD\sqrt{cCT}$.

For the j^{th} deletion, loss Lipschitzness implies $\left|\ell_{t_j}(w_{t_j}^{\text{new}}) - \ell_{t_j}(\hat{w}_{t_j})\right| \leq G\|\eta_j\|_2$. Because $\|\eta_j\|_2$ is sub-Gaussian with parameter σ_{step}, a union bound plus $\|\eta_j\|_2 \leq (L/\lambda)\sqrt{2\ln(1.25m/\delta^\star)}\,/\varepsilon^\star \cdot \sqrt{2\ln(1/\delta_{\text{B}})}$ holds simultaneously for all m deletions with probability $1 - \delta^\star - \delta_{\text{B}}$. Adding these m increments to R_T^0 yields the stated bound.

Since the noise term is $O(m)$ while the first term is $O(\sqrt{T})$, the overall regret remains sub-linear in T. □

5.4 Deriving Capacity and Complexity Bounds

We're able to then describe the γ-deletion capacity bound or the γ-sample complexity bound as follows.

Theorem 3 (γ-Deletion Capacity Bound)**.** *The memory pair processes valid sequences of up to m deletions without sacrificing γ regret bound guarantees.*

$$\frac{1}{N}R_N(\theta, m) \leq \gamma$$

Theorem 4 (γ-Sample Complexity Bound)**.** *At least N training examples are required to achieve γ-regret guarantees.*

We take the inequality from Theorem 4 and plug in the maximum value for N, which provides the number of insert events that are needed before inference may begin. This ensure the first answer is PAC-competitive.

Theorem 5 (Bounded Cumulative Regret for Online L-BFGS)**.** *Let the online learner operate under Assumption 4. Let the inverse curvature matrix B_t^{-1} be updated by the L-BFGS algorithm and satisfy the eigenvalue bounds from Lemma 4. With a non-increasing stepsize sequence, the cumulative regret R_T against the best fixed comparator $w^\star$ is bounded by:*

$$R_T \leq GD\sqrt{cCT} = O(\sqrt{T})$$

Assumption 4 (Online Convex Learning Setting)**.** *Let $\{\ell_t\}_{t=1}^T$ be a sequence of convex loss functions where:*

- *Each $\ell_t : \mathcal{W} \to \mathbb{R}$ is G-Lipschitz, such that $||\nabla \ell_t(w)|| \leq G$ for all $w \in \mathcal{W}$.*
- *The learner is constrained to a closed, convex set $\mathcal{W} \subseteq \mathbb{R}^d$ of diameter $D < \infty$.*

Proof. The proof proceeds by first bounding the per-round regret and then summing these bounds over all T rounds. The learner produces weights via the update $w_{t+1} = \Pi_{\mathcal{W}}(w_t + \eta_t d_t)$, where $d_t = -B_t^{-1} g_t$ and $g_t = \nabla \ell_t(w_t)$.

Step 1: Bounding the per-round regret.

By the convexity of ℓ_t, the per-round regret is bounded by the inner product with the gradient:

$$\ell_t(w_t) - \ell_t(w^\star) \leq \langle g_t, w_t - w^\star \rangle$$

We analyze the distance between the next iterate and the comparator $w^\star$. By the non-expansiveness of the projection $\Pi_{\mathcal{W}}$, we have:

$$\begin{aligned} \|w_{t+1} - w^\star\|^2 &\leq \|w_t + \eta_t d_t - w^\star\|^2 \\ &= \|w_t - w^\star\|^2 + \eta_t^2 \|d_t\|^2 + 2\eta_t \langle d_t, w_t - w^\star \rangle \end{aligned}$$

Rearranging this inequality and substituting $d_t = -B_t^{-1} g_t$ gives an expression for the inner product we wish to bound:

$$\langle g_t, w_t - w^\star \rangle = \langle -B_t d_t, w_t - w^\star \rangle \leq \frac{\|w_t - w^\star\|^2 - \|w_{t+1} - w^\star\|^2}{2\eta_t} + \frac{\eta_t \|d_t\|^2}{2}$$

From Lemma 4, we know $||B_t^{-1}||_2 \leq 1/c$, so $||d_t|| = ||B_t^{-1} g_t|| \leq \frac{1}{c}||g_t||$. Using the G-Lipschitz assumption from Assumption 4, we get $||d_t||^2 \leq \frac{G^2}{c^2}$. Substituting this yields the per-round regret bound:

$$\ell_t(w_t) - \ell_t(w^\star) \leq \frac{\|w_t - w^\star\|^2 - \|w_{t+1} - w^\star\|^2}{2\eta_t} + \frac{\eta_t G^2}{2c} \tag{1}$$

Step 2: Summing over rounds.

We sum the bound in (1) from $t = 1$ to T. The first term on the right-hand side telescopes:

$$R_T = \sum_{t=1}^{T} (\ell_t(w_t) - \ell_t(w^\star)) \leq \sum_{t=1}^{T} \frac{\|w_t - w^\star\|^2 - \|w_{t+1} - w^\star\|^2}{2\eta_t} + \frac{G^2}{2c} \sum_{t=1}^{T} \eta_t$$

Since $\{\eta_t\}$ is a non-increasing sequence, we can bound the telescoping sum. For simplicity and a standard result, we set the step size at $\eta_t = \frac{D}{G\sqrt{T}} \sqrt{\frac{c}{C}}$. The sum becomes:

$$R_T \leq \frac{\|w_1 - w^\star\|^2}{2\eta_T} + \frac{G^2}{2c} \sum_{t=1}^{T} \eta_t$$

Using $\|w_1 - w^\star\|^2 \leq D^2$ and substituting the value for η_t:

$$R_T \leq \frac{D^2 G\sqrt{T}}{2D}\sqrt{\frac{C}{c}} + \frac{G^2}{2c}\frac{TD}{G\sqrt{T}}\sqrt{\frac{c}{C}} = \frac{DG\sqrt{T}}{2}\sqrt{\frac{C}{c}} + \frac{GD\sqrt{T}}{2\sqrt{cC}}$$

While a more careful analysis of the telescoping sum with a time-varying step size would yield a tighter bound, this demonstrates that the cumulative regret scales with $\sqrt{T}$. A standard choice of $\eta_t \propto 1/\sqrt{t}$ with the integral bound $\sum_{t=1}^{T} t^{-1/2} \leq 2\sqrt{T}$ leads to the final bound:

$$R_T \leq GD\sqrt{cCT} = O(\sqrt{T})$$

Since c and C are constants independent of T, the cumulative regret is sublinear, completing the proof. □

6 Experimental Analysis

7 Related Work

8 Conclusion

9 Appendix

9.1 Additional Definitions

Definition 3 (Sample Complexity of Online Learning)**.** *The γ-sample complexity of an online algorithm A is defined as the minimum number of learned points such that average regret is bounded to γ.*

$$n_\gamma(A) = \min\{n \mid \exists\ A \text{ s.t. } \mathbb{E}[R_N] \leq \gamma \text{ and } c_{A(N)} \geq m\}$$

where $c_{A(N)}$ is the remaining odometer budget after A has processed at least N samples.

Definition 4 (γ-Deletion Capacity)**.** *The γ-deletion capacity of an unlearning algorithm $\bar{A}$ is the largest integer m such that for every outcome stream of length N, and deletion schedule $\mid \mathcal{D} \mid \leq m$ the learner satisfies*

$$\Pr[R_N^{\bar{A}} \leq \gamma] \geq 1 - \delta$$

even after having fulfilled any sequence of up to m deletion requests.

References