# PMF-CL: PARETO-MINIMAL-FORGETTING CONTINUAL LEARNER FOR CONFLICTING TASKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In the literature, many *continual learning* (CL) algorithms have been proposed to address the issue of *catastrophic forgetting* in ML models (i.e., learning new tasks leads to the loss of performance on previously learned tasks). Although all CL approaches use some form of memory to retain information about past tasks, a grounded understanding of what information needs to be stored to minimize catastrophic forgetting remains elusive. Recently, it has been recognized that under the strong assumption of the existence of a common global minimizer over all tasks, catastrophic forgetting can be completely avoided. However, in practice, tasks rarely have a common global minimizer, and a certain amount of forgetting is inevitable. In this paper, we propose a foundational reframing of CL as balancing conflicting tasks in hindsight from a *multi-task learning* (MTL) perspective. The approach is based on finding *Pareto-optimal* solutions, i.e., the solutions which, by definition, minimally forget the previous tasks in the Pareto sense. We derive a Pareto-minimal-forgetting CL (PMF-CL) algorithm for linear and basis-function regression. Our algorithm naturally extends to loss functions with quadratic upper-bounds around their minimizers, despite consuming only a *static* memory footprint of $\mathcal{O}\left(d^2\right)$ for $d$ model parameters, *independent* of the number of sequentially occurring tasks $T$. In the quadratic-loss setting, we obtain exact hindsight Pareto-optimal guarantees on forgetting. In the quadratic-upper-bound setting, forgetting is provably bounded, with the tightness of the forgetting bound being directly determined by the tightness of the quadratic bound on the loss function. Our numerical results validate our theoretical claims of convergence to the Pareto-optimal solution, and demonstrate that we consistently outperform prior empirical methods while occupying lesser memory.

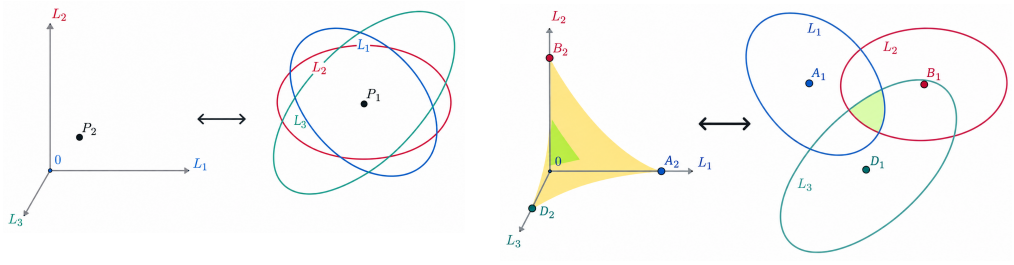
## 1 INTRODUCTION

The success of large-scale models for language (Radford et al., 2019; Touvron et al., 2023), vision (Radford et al., 2021; Ramesh et al., 2022) and multimodal tasks (OpenAI et al., 2024; Yin et al., 2024) has caused a monumental shift in the way ML models are trained and deployed. The clear trend of scaling laws (Kaplan et al., 2020) has led to the AI industry's push towards larger models, larger datasets and more compute for training. This has caused the field to frequently hit roadblocks in terms of hardware and energy requirements (Ahmad et al., 2021; O'Donnell & Crownhart, 2025) to train and fine-tune such models, pushing active research towards efficiency and adaptation methods. Likewise, to keep up with the pace of data evolution, efficiently updating models without completely retraining them or compromising on their performance on past tasks is equally important. Thus, there is a pressing need for *continual learning* algorithms, which guarantee high performing models that do not only adapt to new tasks, but also do so without degrading their performance on past tasks.

Continual Learning (CL) has been studied extensively (Wang et al., 2024a) in the empirical context for at least two decades, where the main motivation is to avoid the so-called *catastrophic forgetting* (French, 1999; Mermillod et al., 2013) of past tasks. Existing CL approaches can be categorized into three subclasses: 1) replay-based, 2) regularization-based, and 3) projection-based methods (see further discussions in Section 2). In terms of geometric intuition, two of these existing methods are relevant: **a)** Gradient-projection methods (Saha et al., 2021; Lin et al., 2022a) make use of geometric structure of gradient updates on MLPs to constrain them to separate subspaces for the different tasks; **b)** regularization-based methods (e.g., EWC (Kirkpatrick et al., 2017)) use the approximate Hessian



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.




(a) ICL setting (Peng et al., 2023)

(b) Our PMF-CL setting

Figure 1: Task settings visualized in the task-loss domain *(Left)* and parameter space *(Right)*. In (a) ICL assumes a common global minimizer making the PF collapse to a point in loss domain. In (b) the yellow region illustrates a Pareto front (PF). The green region depicts a portion of it, and the corresponding set of parameters is the intersection of sublevel sets of the loss surfaces.

matrices of past tasks to nudge the new task updates towards slowly-increasing or non-increasing directions of past task loss functions. While these works use constraints to reduce or minimize catastrophic forgetting, they use heuristic methods that either take advantage of null spaces in the parameter space or use approximate regularization. However, an understanding of *optimality* and *forgetting* in the CL context from first-principles remains limited.

Recently, Peng et al. (2023) introduced an interesting approach to develop an intuitive understanding of forgetting and how representative information from past tasks impacts forgetting via their proposed *Ideal continual learner* (ICL). However, their argument is developed under a strong assumption that the loss functions of different tasks share at least one common global minimizer. In most CL scenarios, this assumption would not hold as the tasks could be in conflict with one another, and the algorithm would inevitably need to trade-off some tasks. Relaxing the common-minimizer assumption completely changes the nature of the problem, since now the CL algorithm needs to find the optimal trade-off point between various loss functions. To illustrate this from the Multi-Task Learning (MTL) perspective (Caruana, 1997; Désidéri, 2012; Hu et al., 2022), the common-minimizer assumption collapses the *Pareto front* in the task-loss domain (defined formally in Definition 3.4) to a single point as illustrated in Fig. 1a. On the other hand, a more realistic scenario is when the tasks have different minima, and the goal is to find a solution on the Pareto-front as illustrated in Fig. 1b.

The previous illustration implies that zero-forgetting in general is impossible, and that an ideal CL algorithm should find the *optimal trade-off* which can minimize all tasks as much as possible. The current state-of-the-art in CL still lacks a fundamental understanding on how to achieve these optimal trade-offs in a sequential learning setup. In this work, we develop a first-principle understanding of CL through the lens of multi-task (or multi-objective) optimization. To this end, we make the following contributions in this paper:

- We propose a general meta-algorithmic framework for a *Pareto-minimal-forgetting* continual learner (PMF-CL) that can handle conflicting tasks with differing minimizers. Since our method is developed from the MTL perspective, it is readily amenable to geometric intuition in the parameter and task-loss function domains. Our formulation sheds new insights into exactly what information needs to be memorized to enable Pareto-minimal-forgetting continual learning.
- We explicitly link CL and MTL for general loss functions and reframe CL as finding an MTL solution sequentially. We note that Peng et al. (2023) made this connection previously. However, they do so under a strong assumption that the tasks share atleast one common global minimizer.
- For the case of *quadratic* and *Quadratic Upper Bounded* (QUB) loss functions, we also investigate the performances of PMF-CL. For the quadratic case, PMF-CL converges to a Pareto-optimal solution of all tasks in hindsight. For QUB loss, PMF-CL finds a Pareto-optimal solution of the surrogate upper-bound functions, and we obtain provable upper-bounds on forgetting. The tightness of the forgetting bounds are directly determined by the tightness of the corresponding QUBs.
- Additionally, we explicitly characterize the memory required to achieve provably minimal forgetting. Interestingly, the memory requirement for quadratic or QUB loss functions is static and bounded as $\mathcal{O}\left(d^2\right)$, where $d$ is the number of parameters. Notably, this scaling does not depend on the number of tasks $T$. Compared to existing replay-buffer methods, this constant memory footage with respect to number of tasks is a significant improvement.

- We perform numerical experiments for continual linear and logistic regression. The experiments on linear regression validate our theoretical results, by demonstrating that, not only does PMF-CL consistently outperform empirical baselines, but it also converges to the hindsight-optimal solution, i.e., the solution obtained by jointly optimizing over all tasks.

The rest of the paper is organized as follows. In Section 2, we summarize the existing state-of-the-art research in CL. In Section 3, we connect CL to the traditional MTL problem, and present our PMF-CL meta-algorithm for Pareto-minimal forgetting in CL. In Sections 4 and 5, we design iterative PMF-CL algorithms for regression and QUB problems. Finally, we conclude in Section 6.

## 2 Related Work

Continual Learning (CL) aims to learn from a sequence of tasks while balancing the trade-off between *plasticity* (learning new data) and *stability* (retaining old patterns) (Thrun & Mitchell, 1995; Mermillod et al., 2013). This paradigm is essential for foundation models in sensitive domains like healthcare to remain current without suffering *catastrophic forgetting* (French, 1999; Ray, 2024).

Existing empirical strategies generally fall into three broad categories: (a) **Experience Replay**, which rehearses samples from previous tasks (Buzzega et al., 2020; Aljundi et al., 2019); (b) **Regularization**, which penalizes parameter deviations in directions critical to past tasks (Kirkpatrick et al., 2017; Zenke et al., 2017); and (c) **Gradient Projection**, which confines updates to orthogonal subspaces to prevent interference (Saha et al., 2021; Lin et al., 2022a). While replay methods are empirically robust, they often lack the geometric clarity of regularization and projection approaches. Recent work has connected CL with Pareto optimization. Lai et al. (2025) learn preference-conditioned stability-plasticity trade-offs, while Wu et al. (2024) use Pareto optimization for buffer-based gradient aggregation across previous tasks. In contrast, we treat tasks as individual objectives and use loss-preserving representations to recover a specified hindsight joint-training solution in structured settings.

Recent theoretical efforts have explored CL through the lens of loss landscape geometry, analyzing how flatness and curvature influence forgetting rates (Liu et al., 2019; Benzing, 2020; Evron et al., 2025). Although avoiding forgetting is fundamentally NP-hard (Knoblauch et al., 2020), our proposed method, PMF-CL, leverages *quadratic upper bounds* to achieve provably Pareto-optimal solutions with a fixed $O(d^2)$ memory footprint. Appendix A contains a more comprehensive literature review.

## 3 Continual Learning as a Multi-Task Optimization Problem

In this section, we prove using first principles that continual learning (CL) can be viewed as a sequential counterpart of Multi-Task Learning (MTL) (Caruana, 1997; Miettinen, 2008). This connection allows us to derive a general CL meta-algorithm that achieves minimal catastrophic forgetting by finding the same solution as the MTL problem in hindsight.

**Notation** We consider a supervised CL setting with $T$ sequential tasks. Each task $t$ has an associated dataset of $n_t$ input-output pairs $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, \boldsymbol{y}_i^t)\}_{i=1}^{n_t}$ We consider a global model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot) : \mathcal{X} \to \mathcal{Y}$ where $\boldsymbol{\theta} \in \mathcal{W}$ is the set of model parameters of $\boldsymbol{f}$. The loss function for task $t$ is defined as:

$$L_t(\boldsymbol{\theta}) := \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t) = \frac{1}{n_t} \sum_{i=1}^{n_t} \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^t), \boldsymbol{y}_i^t) \tag{1}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a sample-wise loss function and remains fixed across tasks. Notably, the model parameters $\boldsymbol{\theta}$ and the loss functions $\mathcal{L}(\cdot;\cdot)$ and $\ell(\cdot,\cdot)$ are shared across all tasks, with the task-specific dataset $\mathcal{D}_t$ influencing the shape and properties of each task's loss function.

### 3.1 Multi-Task Learning

We begin by defining a few key concepts related to MTL and CL.

**Definition 3.1** (Multi-task Learning)**.** The unconstrained Multi-Task Learning (MTL) problem for $T$ tasks with corresponding loss functions $L_t(\theta)$ is defined as: $\min_{\theta \in \mathcal{W}} \{L_1(\theta), L_2(\theta), \ldots, L_T(\theta)\}$ which is a joint optimization of all potentially conflicting task losses over the shared parameter space.

Since the objectives in MTL can be in conflict with each other, zero-forgetting is impossible in general, and one would seek a *Pareto optimal* solution or even the entire *Pareto frontier* (i.e., the collection of all Pareto optimal solutions) rather than a common global minimum for all objectives, which does *not* exist in general. This is in contrast to gradient-projection-based approaches in the CL literature (Saha et al., 2021; Lin et al., 2022a; Peng et al., 2023), which assume a common global minimizer, or approximate each task's loss function until such an approximation exists.

Since we will heavily rely on the concept of Pareto optimality, we formally define these notions next.

**Definition 3.2** (Pareto Dominance)**.** For the MTL problem defined in Definition 3.1, a solution $\boldsymbol{\theta}_a$ is said to *Pareto dominate* another solution $\boldsymbol{\theta}_b$, denoted as $\boldsymbol{\theta}_a \succ \boldsymbol{\theta}_b$, iff: $L_t(\theta_a) \leq L_t(\theta_b), \quad \forall t = 1, 2, \ldots, T$ and there exists at least one task $t'$ such that: $L_{t'}(\theta_a) < L_{t'}(\theta_b)$.

Based on the notion of Pareto dominance, we can now define Pareto optimality as follows:

**Definition 3.3** (Pareto Optimality)**.** For the MTL problem defined in Definition 3.1, the Pareto optimal set, denoted by $\mathcal{W}^\star$, is the set of all parameters in $\mathcal{W}$ that are not Pareto dominated by any other parameters in $\mathcal{W}$.

**Definition 3.4** (Pareto Front)**.** The set of all non-dominated solutions in the loss space is called the *Pareto Front* (PF): $\mathcal{P} := \{(L_1(\boldsymbol{\theta}), \ldots, L_T(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \mathcal{W}^\star\}$.

The most common approach to solving MTL problems is *linear scalarization* (Hwang & Masud, 2012; Miettinen, 2008). For convex problems, it is known that any Pareto optimal solution on the Pareto front can be found by appropriately choosing a *preference vector* $\boldsymbol{\alpha}^{(T)} \in \Delta^T$ from the standard simplex and optimizing the preference-weighted sum of tasks as follows:

$$\boldsymbol{\theta}_T^\star := \arg\min_{\boldsymbol{\theta} \in \mathcal{W}} \sum_{t=1}^{T} \alpha_t^{(T)} L_t(\boldsymbol{\theta}). \tag{2}$$

Choosing the preference vector $\boldsymbol{\alpha}^{(T)}$ in (2) in terms of the number of training samples as:

$$\alpha_t^{(T)} = \frac{n_t}{\sum_{k=1}^{T} n_k}, \tag{3}$$

we can retrieve the global empirical risk minimization (ERM) solution over all tasks, as stated by the next lemma. The proof is quite straightforward and is provided in Appendix D.2.

**Lemma 3.5** (MTL for Global ERM)**.** *The solution to the linearly-scalarized MTL problem in (2) with preferences chosen as in (3) is equivalent to the solution of the global optimization problem* $\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta} \in \mathcal{W}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$*, where* $\mathcal{D}$ *is a concatenation of all datasets* $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$.

*Remark* 3.6. In the scenario where tasks repeat over time, the preference vector in (3) automatically weights the repeated tasks more, which is arguably desirable, and is analogous to having repeated samples/batches in the global dataset $\mathcal{D}$.

### 3.2 Continual Learning as Sequential MTL

The key difference between MTL and CL is in the availability of the data. In MTL, data from all tasks are available simultaneously, so the goal is to find a Pareto optimal solution. In CL, there are two key differences from MTL: **1)** data from tasks are available *sequentially*, one at a time. **2)** the model must *evolve* after each task, without forgetting previous tasks. Thus, a CL algorithm must maintain some *memory* of older tasks during optimization. Here, we derive a general meta-algorithmic framework called *Pareto-Minimal-Forgetting Continual Learner* (PMF-CL).

First, we address the question: *What is the sufficient amount of memory required to preserve the past task loss functions, enough to retrieve a Pareto-optimal trade-off for all tasks in hindsight?* Toward this end, we first concretely define 'forgetting':

**Definition 3.7** (Forgetting)**.** We define the *degree of forgetting* of the $t^{\text{th}}$ task by a parameter $\boldsymbol{\theta} \in \mathcal{W}$ in terms of the loss function $L_t(\boldsymbol{\theta})$ as $F_t(\boldsymbol{\theta}) := L_t(\boldsymbol{\theta}) - \inf_{\boldsymbol{\theta}' \in \mathcal{W}} L_t(\boldsymbol{\theta}')$, and the *average forgetting* for the first $k$ tasks as $\overline{F}_k(\boldsymbol{\theta}) := \frac{1}{k} \sum_{t=1}^{k} F_t(\boldsymbol{\theta})$.

Since by definition $F_t(\boldsymbol{\theta})$ is an objective-wise constant shift, the Pareto-optimality of $F_t(\boldsymbol{\theta})$ and $L_t(\boldsymbol{\theta})$ are equivalent. The task loss function defined in (1) is a function of $\boldsymbol{\theta}$ and fully characterized

**Algorithm 1** Generic PMF-CL Meta-algorithm

**Require:** Loss functions $\mathcal{L}(\cdot;\cdot)$ and $\widetilde{\mathcal{L}}(\cdot;\cdot)$, Tasks $\{\mathcal{D}_t\}_{t=1}^T$

**for** $t \leftarrow 1, \dots, T$ **do**
  **Receive** task dataset $\mathcal{D}_t$
  **Solve** the MTL problem:

$$\min_{\boldsymbol{\theta}\in\mathcal{W}} \{\widetilde{\mathcal{L}}(\boldsymbol{\theta};\mathcal{I}_1), ..., \widetilde{\mathcal{L}}(\boldsymbol{\theta};\mathcal{I}_{t-1}), \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t)\}$$

  in an incremental fashion.
  **Store** minimal information for task $t$: $\mathcal{I}_t$
**end for**

**Algorithm 2** PMF-CL for QUB loss (Section 5)

**Require:** QUB Loss $\mathcal{L}(\cdot;\cdot)$, Tasks $\{\mathcal{D}_t\}_{t=1}^T$

$\boldsymbol{\theta}_1^\star = \boldsymbol{\theta}_1^{\min} = \arg\min_{\boldsymbol{\theta}\in\mathcal{W}} \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_1)$
Initialize $\boldsymbol{A}_1 = \boldsymbol{H}_1$
**for all** $t \leftarrow 2, \dots, T$ **do**
  Compute $\boldsymbol{\theta}_t^{\min} = \arg\min_{\boldsymbol{\theta}\in\mathcal{W}} \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t)$.
  Determine $\boldsymbol{H}_t$, the Hessian upper bound.
  $\boldsymbol{A}_t = \frac{N_{t-1}}{N_t}\boldsymbol{A}_{t-1} + \alpha_t^{(t)}\boldsymbol{H}_t$
  Solve for $\Delta\boldsymbol{\theta}$: $\boldsymbol{A}_t\Delta\boldsymbol{\theta} = \alpha_t^{(t)}\boldsymbol{H}_t(\boldsymbol{\theta}_t^{\min} - \boldsymbol{\theta}_{t-1}^\star)$
  Update $\boldsymbol{\theta}_t^\star = \boldsymbol{\theta}_{t-1}^\star + \Delta\boldsymbol{\theta}$
**end for**

**Algorithm 3** PMF-CL for Linear (*Option I*) and Basis (*Option II*) Regression.

**Require:** Datasets $(\boldsymbol{X}_t, \boldsymbol{y}_t)_{t=1}^T$, basis function $\phi(\cdot)$.

Init $t = 1$
**while** $t < T$ **and** $\sum_{i=1}^t r_i \leq d$ **do**
  *Option I:* $\boldsymbol{U}_t^e, \boldsymbol{\Sigma}_t^e, \boldsymbol{V}_t^e = \text{eSVD}(\boldsymbol{X}_t)$
  *Option II:* $\boldsymbol{U}_t^e, \boldsymbol{\Sigma}_t^e, \boldsymbol{V}_t^e = \text{eSVD}(\phi(\boldsymbol{X}_t))$
  $\widetilde{\boldsymbol{y}}_t = (\boldsymbol{\Sigma}_t^e)^{-1}\boldsymbol{U}_t^{e\top}\boldsymbol{y}_t$
  Store $(\boldsymbol{\Sigma}_t^e, \boldsymbol{V}_t^e, \widetilde{\boldsymbol{y}}_t)$ {Mem: $(2+d)\sum_{i=1}^t r_i$}
  Compute preferences $\boldsymbol{\alpha}^{(k)}$ as in (3)
  Solve for $\boldsymbol{\theta}_t^\star$ using (6)
  Increment $t \leftarrow t+1$
**end while**
Init $\boldsymbol{A}_{t-1} = \sum_{i=1}^{t-1}\alpha_i^{(t-1)}\boldsymbol{V}_i^e\boldsymbol{\Sigma}_i^{e2}\boldsymbol{V}_i^{e\top}$
**for all** $k \leftarrow t, \dots, T$ **do**
  *Option I:* $\boldsymbol{U}_k^e, \boldsymbol{\Sigma}_k^e, \boldsymbol{V}_k^e = \text{eSVD}(\boldsymbol{X}_t)$
  *Option II:* $\boldsymbol{U}_k^e, \boldsymbol{\Sigma}_k^e, \boldsymbol{V}_k^e = \text{eSVD}(\phi(\boldsymbol{X}_t))$
  $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k}\boldsymbol{A}_{k-1} + \alpha_k^{(k)}\boldsymbol{V}_k^e\boldsymbol{\Sigma}_k^{e2}\boldsymbol{V}_k^{e\top}$
  $\widetilde{\boldsymbol{y}}_k = (\boldsymbol{\Sigma}_k^e)^{-1}\boldsymbol{U}_k^{e\top}\boldsymbol{y}_k$
  Solve for $\Delta\boldsymbol{\theta}_k$ as in (7)
  Update $\boldsymbol{\theta}_k^\star = \boldsymbol{\theta}_{k-1}^\star + \Delta\boldsymbol{\theta}_k$
**end for**
**Ensure:** $\{\boldsymbol{\theta}_k^\star\}_{k=1}^T$

by the dataset $\mathcal{D}_t$. Consider an equivalent notation of the task loss in terms of the *Minimal Sufficient Information* (MSI), $\mathcal{I}_t$, required to fully characterize $\mathcal{L}(\cdot,\cdot)$. We formally define the MSI as:

**Definition 3.8** (Minimal Sufficient Information; MSI)**.** For a given model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ and sample loss $\ell(\cdot,\cdot)$, the *MSI* for task $t$, $\mathcal{I}_t$ is defined as the smallest set that can fully characterize the loss function of the task. Formally, $\mathcal{I}_t$ satisfies $0 < |\mathcal{I}_t| \leq |\mathcal{D}_t|$ and $\widetilde{\mathcal{L}}(\boldsymbol{\theta};\mathcal{I}_t) = \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t) + C_t, \forall\, \boldsymbol{\theta} \in \mathcal{W}$ and some constant $C_t$; and there exists no set $\mathcal{I}_t'$ such that $|\mathcal{I}_t'| < |\mathcal{I}_t|$ and $\widetilde{\mathcal{L}}(\boldsymbol{\theta};\mathcal{I}_t') = \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t),\ \forall\, \boldsymbol{\theta} \in \mathcal{W}$.

The minimal task information $\mathcal{I}_t$ is fully determined by the shape and properties of the composition of the model and the loss function, $\ell(f_{\boldsymbol{\theta}}(\cdot),\cdot)$. In the following section, we show that for loss functions with a quadratic structure, $\mathcal{I}_t$ can be compact (such as the *right singular matrix* and the *singular values* of the dataset). In the worst case where the loss surfaces have a complex dependency on the data, the minimal information might require the entire dataset $\mathcal{D}_t$. Thus, how to automatically identify MSI for arbitrary loss functions remains an open challenge. In fact, Knoblauch et al. (2020) already established that this problem is NP-Hard in general. However, for certain structured cases, such as the quadratic case discussed later in this paper, it is indeed possible to exploit their special structure to find the loss-preserving minimal sufficient information.

Based on the notion of MSI, we can propose the PMF-CL meta-algorithm described in Algorithm 1, which would *Pareto-minimally forget*, or optimally trade-off, previous tasks by finding the same solution as MTL in a sequential manner. Algorithm 1 is meant to serve as a general conceptual link between continual learning and sequential MTL. In general, it is not computationally tractable without access to the structure of the loss function. Importantly, note that the ordering of tasks has no impact on the final solution learned, since the Pareto-optimal solution given a fixed preference vector $\alpha^{(k)}$ is not affected by the task order. This is an intentional choice, since we target to arrive at the same hindsight-optimal solution regardless of the order in which the tasks may be presented. In Sections 4 and 5 we will show how the general idea presented in Algorithm 1 can be specialized for loss functions with quadratic structure.

# 4 PMF-CL FOR REGRESSION

In this section, we will study the specialization of the meta-algorithm in Algorithm 1 to the problem of linear and basis-function regression. Specifically, we will derive the MSI, and then design a memory-efficient and provably Pareto-minimal-forgetting, continual learning algorithm for linear regression in Section 4.1 and its extension to basis functions in Appendix D.1.

## 4.1 LINEAR REGRESSION

In linear regression, the model is given by $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ where $\boldsymbol{\theta} \in \mathbb{R}^{d\times 1}$ and the output is a scalar. Under the mean squared error (MSE) loss function, task loss $L_t$ can be written as:

$$L_t(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t) = \frac{1}{2n_t} \left\| \boldsymbol{X}_t \boldsymbol{\theta} - \boldsymbol{y}_t \right\|_2^2, \tag{4}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{n_t \times d}$ (resp. $\boldsymbol{Y}_t \in \mathbb{R}^{n_t \times 1}$) is a matrix with the inputs $\boldsymbol{x}_{t,i}$ as rows (resp. vector of outputs $y_{t,i}$). In order to obtain the MSI of the quadratic loss function, we characterize the task data matrix $\boldsymbol{X}_t$ via the *singular value decomposition* (SVD).

**SVD and economy SVD.** For the task data matrix $\boldsymbol{X}_t$, we define the SVD in two ways: $\boldsymbol{X}_t = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top = \boldsymbol{U}_t^e \boldsymbol{\Sigma}_t^e \boldsymbol{V}_t^{e\top}$, where $\boldsymbol{U}_t \in \mathbb{R}^{n_t \times n_t}$ and $\boldsymbol{V}_t \in \mathbb{R}^{d\times d}$ are orthonormal matrices, i.e., $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{I}_d$, and $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}_{n_t}$, and the singular values are the diagonal elements of $\boldsymbol{\Sigma}_t \in \mathbb{R}^{n_t \times d}$. The *economy SVD* (eSVD) stores only the singular vectors corresponding to the non-zero singular values in $\Sigma_t$. Suppose that the rank of $\boldsymbol{X}_t$ is $r_t$. Then, the matrices $\boldsymbol{U}_t^e \in \mathbb{R}^{n_t \times r_t}$ and $\boldsymbol{V}_t^e \in \mathbb{R}^{d\times r_t}$ can be truncated to contain the first $r_t$ columns of $\boldsymbol{U}_t$ and $\boldsymbol{V}_t$, respectively, and $\boldsymbol{\Sigma}_t^e \in \mathbb{R}^{r_t \times r_t}$ is a diagonal matrix of positive reals.

**Theorem 4.1** (MSI for Linear Regression)**.** *We can write the loss function for linear regression as:*

$$L_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^e}^2 + \min_{\boldsymbol{\theta}' \in \mathcal{W}} L_t(\boldsymbol{\theta}'), \tag{5}$$

*where we use the notation* $\|\boldsymbol{z}\|_{\boldsymbol{A}}^2 := \|\boldsymbol{A}\boldsymbol{z}\|_2^2$ *and* $\widetilde{\boldsymbol{y}} = \boldsymbol{\Sigma}_t^{e-1} \boldsymbol{U}_t^e \boldsymbol{y}_t$*. Then, an MSI representing the linear regression loss is* $\mathcal{I}_t^{\text{lin}} := (\mathcal{S}(\boldsymbol{V}_t^e), \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$*, where* $\mathcal{S}(\cdot)$ *is a compact storage model for the orthonormal singular-vector matrix* $\boldsymbol{V}_t^e$ *using Givens rotations (Shepard et al., 2015). The memory size for this representation is* $\text{size}(\mathcal{I}_t^{\text{lin}}) = dr_t - \frac{r_t(r_t-3)}{2}$ *which is provably* minimal.

A complete proof of Theorem 4.1 is deferred to Appendix B.1.2. Since the last term in the RHS does not depend on $\boldsymbol{\theta}$, PMF-CL does not need to remember it. Note that any representative set smaller than $\mathcal{I}_t^{\text{lin}}$ would inevitably forget the loss function shape, leading to forgetting in a CL setting. Thus, $\mathcal{I}_t^{\text{lin}}$ is indeed the *minimal* information required to remember the $t^{\text{th}}$ task. Considering the CL problem up to task $k$, a Pareto optimal solution $\boldsymbol{\theta}_k^\star$ to the multi-task problem in (2) under linear scalarization with the preference vector $\boldsymbol{\alpha}^{(k)} := [\alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_k^{(k)}]^\top$ can be written as: $\boldsymbol{\theta}_k^\star = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^k \frac{\alpha_t^{(k)}}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^e}^2$, which can be solved by the linear system:

$$\left( \sum_{t=1}^k \frac{\alpha_t^{(k)}}{n_t} \boldsymbol{V}_t^e \boldsymbol{\Sigma}_t^{e2} \boldsymbol{V}_t^{e\top} \right) \boldsymbol{\theta}_k^\star = \left( \sum_{t=1}^k \frac{\alpha_t^{(k)}}{n_t} \boldsymbol{V}_t^e \boldsymbol{\Sigma}_t^{e2} \widetilde{\boldsymbol{y}}_t \right). \tag{6}$$

Thus, one would require a memory of $(d+2)\sum_{t=1}^T r_t$ floating-points per task to store $(\boldsymbol{V}_t^e, \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$.

**Memory-efficient Continual Linear Regression:** Armed with the MSI of Theorem 4.1, one can choose to directly apply the PMF-CL meta-algorithm in Algorithm 1 using (6). However, due to the quadratic nature of task loss functions, we can optimize memory requirements further through iterative updates on the parameters. As we show next, this helps us to limit memory requirements to $\mathcal{O}\left(d^2\right)$ *irrespective of* the number of tasks. We choose preferences $\boldsymbol{\alpha}^{(k)}$ as in (3). Then, the following result, proven in Appendix B.1.1, holds for the Pareto optimal solution up to task $k$:

**Theorem 4.2** (Iterative PMF-CL for Linear Regression)**.** *Using the MSI of the task loss as* $\mathcal{I}_t^{\text{lin}} := (\boldsymbol{V}_t^e, \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$ *from Theorem 4.1, the PMF-CL update at the* $k^{th}$ *task can be written as the solution to:*

$$\boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \alpha_k^{(k)} \boldsymbol{V}_k^e \boldsymbol{\Sigma}_k^{e2} (\widetilde{\boldsymbol{y}}_k - \boldsymbol{V}_k^{e\top} \boldsymbol{\theta}_{k-1}^\star), \qquad k \in [T] \tag{7}$$

*Here,* $\boldsymbol{A}_0 = \boldsymbol{0}$, $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k} \boldsymbol{A}_{k-1} + \alpha_k^{(k)} \boldsymbol{V}_k^e \boldsymbol{\Sigma}_k^{e2} \boldsymbol{V}_k^{e\top}$, $N_k := \sum_{i=1}^k n_i$*, and* $\Delta\boldsymbol{\theta}_k = \boldsymbol{\theta}_k^\star - \boldsymbol{\theta}_{k-1}^\star$.

In (7), we have a linear system of equations to solve for the *incremental update* $\Delta\boldsymbol{\theta}_k$, where we need to store the matrices $\boldsymbol{A}_k$ of size $d^2$, $(\boldsymbol{\nu}_k - M_k\boldsymbol{\theta}^\star_{k-1})$ of size $d$, and $\boldsymbol{\theta}^\star_{k-1}$ of size $d$. From the memory analysis of both procedures (6) and (7), we can initially start by performing the direct computation in (6) until the memory required to store $\mathcal{I}^{\text{lin}}_t$ for all $t$, exceeds that required by (7), and then switch to iterative update. This switching threshold is given by $(d+2)\sum_{t=1}^k r_t \geq (d+2)d$ or $\sum_{t=1}^k r_t \geq d$. The PMF-CL algorithm for linear regression is summarized in Algorithm 3, where preferences $\boldsymbol{\alpha}$ are calculated according to (3). The total memory complexity of Algorithm 3 is $\mathcal{O}\left(d^2\right)$, and remains *constant* with respect to the number of tasks $T$ once the above rank-threshold is reached.

With the sample-proportional preference vector in (3), the recursive update in Algorithm 3 becomes the block, information-form recursive least-squares computation for ERM (Plackett, 1950; Jiang & Zhang, 2004). Our contribution is the Pareto formulation and its CL interpretation via the loss-preserving MSI, which recovers the Plackett recursive update as special cases. PMF-CL for linear regression naturally extends to models that are linear in their parameters, including basis functions and two-layer MLPs under the NTK regime (see Appendix D.1 for full derivations).

### 4.2 Forgetting in Regression

From Algorithm 3, we can derive an exact expression of forgetting as defined in Definition 3.7. Before we state the forgetting result, we give an intuitive understanding on the geometry of the Pareto front (PF) for the loss functions $L_t(\cdot)$ and the corresponding forgetting functions $F_t(\cdot)$. Note that the PFs of quadratic loss functions form a convex surface in the task-loss domain, as shown in Fig. 1b. After the $k^{\text{th}}$ task, the degree of forgetting, as defined in Definition 3.7 for all previous tasks including $k$, would be a identically shaped PF but shifted so that it touches all the coordinate hyperplanes, i.e., hyperplanes $F_1 = 0$, $F_2 = 0, \ldots, F_k = 0$ of dimension $k-1$ atleast once.

The next result (proof in Appendix C.1) characterizes forgetting in linear and basis-function regression:

**Theorem 4.3.** *The forgetting of the $t^{th}$ task after learning $k$ tasks using Algorithm 3 is given by:*

$$F_t(\boldsymbol{\theta}) = \frac{1}{2n_t}\left\|\boldsymbol{V}_t^{e\top}\boldsymbol{A}_k^\dagger\boldsymbol{\nu}_k - \widetilde{\boldsymbol{y}}_t\right\|^2, \tag{8}$$

*where $\boldsymbol{A}_k$ and $\boldsymbol{\nu}_k$ are defined as $\boldsymbol{A}_k := \sum_{t=1}^k \alpha_t^{(k)}\boldsymbol{V}_t^e\boldsymbol{\Sigma}_t^{e2}\boldsymbol{V}_t^{e\top}$, $\boldsymbol{\nu}_k := \sum_{t=1}^k \alpha_t^{(k)}\boldsymbol{V}_t^e\boldsymbol{\Sigma}_t^{e2}\widetilde{\boldsymbol{y}}_t$, and $\dagger$ represents the Moore-Penrose pseudo-inverse.*

From Theorem 4.3, the forgetting $\{F_t(\cdot)\}_{t=1}^k$ of all tasks is Pareto optimal, because the PMF-CL algorithm finds a solution on the PF of the corresponding loss functions up to task $k$, by design. In order to empirically validate our theoretical claims, we perform experiments on synthetic linear regression datasets in Appendix G. Our experiments simulate different optimal values for each task, thus adhering to more general settings than just tasks with common global minimizers.

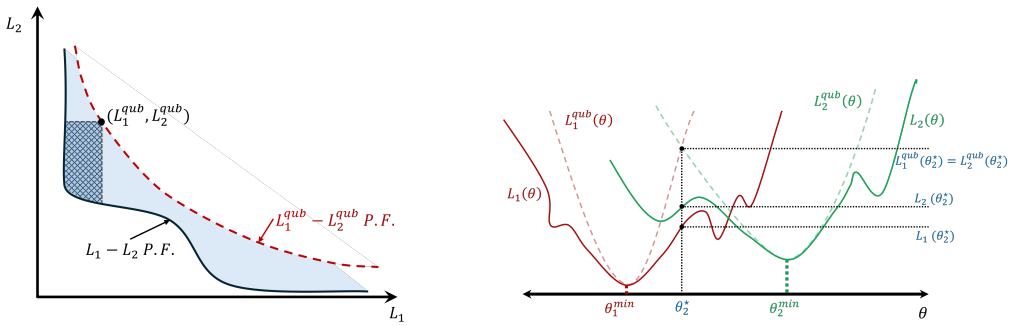
## 5 Quadratic Upper Bounded Loss Functions

In general, it may not be easy to derive a specialized PMF-CL algorithm for any given loss functions, since the MSI $\mathcal{I}_t$ could be hard to determine. However, we can still obtain an upper-bound on the degree of forgetting for competing tasks, if they satisfy the *quadratic upper bound* (QUB) property, which we formally state as follows.

**Assumption 5.1** (Quadratic Upper Bound)**.** We assume that the loss function function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t)$ has a quadratic upper bound (QUB), i.e., for some p.s.d. Hessian matrix $\boldsymbol{H}_t$ and task minimizer $\boldsymbol{\theta}_t^{\min} := \arg\min_{\boldsymbol{\theta}\in\mathcal{W}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t)$, it holds that $L_t(\boldsymbol{\theta}) \leq L_t^{\text{qub}}(\boldsymbol{\theta})$ with $L_t^{\text{qub}}$ given by:

$$L_t^{\text{qub}}(\boldsymbol{\theta}) := L_t(\boldsymbol{\theta}_t^{\min}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min})^T\boldsymbol{H}_t(\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min}). \tag{9}$$

To put Assumption 5.1 into perspective, we note that this assumption can be satisfied by all smooth (gradient-Lipschitz) functions if $\boldsymbol{H}_t$ is positive definite, i.e., for some $h > 0$ and $\forall\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{W}$, functions satisfying: $L_t(\boldsymbol{\theta}) \leq L_t(\boldsymbol{\theta}') + \nabla L_t(\boldsymbol{\theta}')^T(\boldsymbol{\theta} - \boldsymbol{\theta}') + \frac{h}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$. However, Assumption 5.1 allows the local curvature of the loss function around its minimum to vary by using a p.s.d. matrix $\boldsymbol{H}_t$, rather than a single positive scalar $h > 0$.




(a) **Task-loss domain:** A Pareto solution marked at $(L_1^{\text{qub}}, L_2^{\text{qub}})$; the hatch depicts the region in which the respective true loss functions might lie.

(b) **Parameter domain:** $\boldsymbol{\theta}_2^\star$ depicts a PMF-CL solution, with its corresponding QUB and true loss values.

Figure 2: Illustration of the Pareto Fronts (PF) and a 1-D visualization in parameter domain for a pair of generic task-loss functions and their QUB's.

For loss functions satisfying Assumption 5.1, we can derive a PMF-CL algorithm by computing the Pareto solution of the QUB functions. The tightness of the upper-bound (9) directly determines the degree of forgetting of the CL algorithm. Note that Eq. (9) is very similar in form to the quadratic loss from the linear regression case. Then, we compute the Pareto solution to the MTL problem, $\min_{\boldsymbol{\theta}\in\Theta}\{L_1^{\text{qub}}(\boldsymbol{\theta}), L_2^{\text{qub}}(\boldsymbol{\theta}), \ldots, L_k^{\text{qub}}(\boldsymbol{\theta})\}$, using linear scalarization. Using a preference $\boldsymbol{\alpha}^{(k)}$ as in (2), we can compute the Pareto optimal solution $\boldsymbol{\theta}_k^\star$ up to task $k$ by solving the linear equation:

$$\left(\sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{H}_t\right) \boldsymbol{\theta}_k^\star = \sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{H}_t \boldsymbol{\theta}_t^{\min}. \tag{10}$$

As in Section 4.1, the quadratic nature of the problem allows us to derive a memory-efficient iterative update as in Theorem 5.2 (the detailed proof is relegated to Appendix B.2). We present the PMF-CL pseudocode specialized for tasks satisfying QUB in Algorithm 2.

**Theorem 5.2** (Iterative PMF-CL for QUB Loss)**.** *Denoting the QUB-surrogate-preserving information* [1] *of the task loss as* $\mathcal{I}_t = \{\boldsymbol{H}_t, \boldsymbol{\theta}_t^{\min}\}$ *and using* $\boldsymbol{\alpha}^{(k)}$ *from (3), the incremental PMF-CL update specialized for tasks with QUB losses can be written as:*

$$\boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \alpha_k^{(k)} \boldsymbol{H}_k (\boldsymbol{\theta}_k^{\min} - \boldsymbol{\theta}_{k-1}^\star), \tag{11}$$

*where* $\boldsymbol{A}_0 = 0$, $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k}\boldsymbol{A}_{k-1} + \alpha_k^{(k)}\boldsymbol{H}_k$, $\boldsymbol{\theta}_k^{\min} := \min_{\boldsymbol{\theta}} L_k(\boldsymbol{\theta})$, *and* $N_k = \sum_{i=1}^k n_i$.

We note that, in deep learning, the Hessian matrix $\boldsymbol{H}_t \in \mathbb{R}^{d\times d}$ could be large, and the $\mathcal{O}\left(d^2\right)$ scaling can be expensive. However, similar to regularization methods (Kirkpatrick et al., 2017) for log-likelihood loss, one can use the Monte-Carlo approximation of the Fisher information matrix (FIM) as a Hessian estimate. The FIM requires only storing first-order derivatives, thus greatly reducing memory usage from $\mathcal{O}\left(d^2\right)$ to $\mathcal{O}\left(d\right)$.

### 5.1 Forgetting Analysis for QUB Loss

For the QUB loss functions, we can also derive an upper-bound on the degree of forgetting. This is formally stated in Theorem 5.3 below and proven in Appendix C.2.

**Theorem 5.3** (Forgetting for QUB)**.** *Suppose* $\boldsymbol{A}_k$ *is defined as in Theorem 5.2,* $\boldsymbol{\nu}_k := \sum_{i=1}^k \alpha_i^{(k)} \boldsymbol{H}_i \boldsymbol{\theta}_t^{\min}$*, and the PMF-CL solution generated by Algorithm 2 up to task* $k$ *is* $\boldsymbol{\theta}_k^\star = \boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k$*. Then, under Assumption 5.1, the forgetting for task* $t < k$ *after learning task* $k$ *is bounded as:*

$$F_t^{(k)} \leq \frac{1}{2}(\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{H}_t (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min}). \tag{12}$$

[1] The term *minimal sufficient information* is not used here, since technically $(\boldsymbol{H}_t, \boldsymbol{\theta}_{\min})$ is an MSI of the QUB surrogate function and not the actual loss function. The actual loss function would require a potentially larger (or smaller depending on the tasks considered) memory to represent the full task loss faithfully.

Note that, unlike the linear and basis function regression, for QUB loss functions, forgetting is not always Pareto optimal. However, Theorem 5.3 shows that the PMF-CL solution has a worst-case value for forgetting, as visualized in Fig. 2a. We depict the PF of the true loss functions and the PF of the respective QUB's. An ideal algorithm would converge to the true PF, while PMF-CL can converge to any point within the hatched area. In the parameter domain, Fig. 2b illustrates a 1-D case of two arbitrary loss functions and their QUB's, along with an example PMF-CL solution.

The core idea and derivations of PMF-CL can also be extended to the *Federated Continual Learning* (FCL) setting in a straightforward manner as we prove in Appendix F.

### 5.2 Multi-class classification as a QUB Problem

In this section, we show that linear multi-class classification is a QUB problem, and hence admits a static-memory CL algorithm with bounded forgetting. The task *cross-entropy* loss for this problem is:

$$L_t(\boldsymbol{\Theta}) = -\frac{1}{n_t}\sum_{i=1}^{n_t}\sum_{k=1}^{K} y_{i,k}^t \log\left(\frac{\exp(\boldsymbol{\theta}_k^\top \boldsymbol{x}_i^t)}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \boldsymbol{x}_i^t)}\right) \tag{13}$$

where $(y_{i,k}^t)_k$ is a one-hot encoded vector indicating the class of the $i^{\text{th}}$ training sample. We denote the set of learnable parameters as $\boldsymbol{\Theta} := [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K] \in \mathbb{R}^{d\times K}$, where $\boldsymbol{\theta}_k$ are the weights for class $k$. Using Lemma E.1 due to Böhning (Böhning, 1992), enables us to upper bound the loss function as (cf. Appendix E.2 for the full derivation):

$$L_t(\boldsymbol{\Theta}) \leq L_t(\boldsymbol{\Theta}_t^{\min}) + \frac{1}{2n_t}\mathrm{Tr}((\boldsymbol{\Theta}-\boldsymbol{\Theta}_t^{\min})^\top(\boldsymbol{X}_t^\top \boldsymbol{X}_t)(\boldsymbol{\Theta}-\boldsymbol{\Theta}_t^{\min})\overline{\boldsymbol{V}}),$$

where $\overline{\boldsymbol{V}} = \frac{1}{2}\left[\boldsymbol{I}_K - \frac{1}{K}\mathbf{1}\mathbf{1}^\top\right]$, and $\mathbf{1} \in \mathbb{R}^K$ is an all-1 vector. The bound in (14) is a uniform global curvature bound and can be conservative for the task-specific softmax loss. Consequently, the resulting PMF-CL update is Pareto-optimal for these quadratic surrogates, while the associated true-loss forgetting guarantee may be correspondingly loose. Then, the iterative update of PMF-CL for multi-class classification is summarized in the following corollary of Theorem 5.2.

**Corollary 5.4** (Iterative PMF-CL for logistic loss)**.** *Using the QUB-surrogate-preserving information of the task $t$ as $\mathcal{I}_t = \{\mathcal{S}(\boldsymbol{V}_t^e), \boldsymbol{\Sigma}_t^e, \boldsymbol{\Theta}_t^{\min}\}$, under the Givens rotation storage model $\mathcal{S}(\cdot)$, and $\boldsymbol{\alpha}^{(k)}$ as defined in (3), the incremental PMF-CL update for multi-class logistic regression is:*

$$\boldsymbol{A}_k \Delta\boldsymbol{\Theta}_k = \frac{\alpha_k^{(k)}}{n_k}\boldsymbol{V}_t^e \boldsymbol{\Sigma}_t^{e2} \boldsymbol{V}_t^{e\top}(\boldsymbol{\Theta}_t^{\min} - \boldsymbol{\Theta}_{k-1}^\star), \tag{14}$$

*with $\boldsymbol{A}_0 = \mathbf{0}$, $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k}\boldsymbol{A}_{k-1} + \frac{\alpha_k^{(k)}}{n_k}\boldsymbol{V}_k^e \boldsymbol{\Sigma}_k^{e\,2} \boldsymbol{V}_k^{e\top}$; and $\Delta\boldsymbol{\Theta}_k = \boldsymbol{\Theta}_k^\star - \boldsymbol{\Theta}_{k-1}^\star$.*

Memory for the multi-class PMF-CL is static and bounded as $\mathcal{O}\left(Kd + d + d(d+3)/2\right)$ to store $\boldsymbol{V}_t^e$, $\boldsymbol{\Sigma}_t^e$, and the task minimizer $\boldsymbol{\Theta}_t^{\min}$. The resulting forgetting bound is deferred to Appendix E.2. We validate our findings numerically on the Sequential-MNIST dataset in Appendix G.

## 6 Conclusion

In this paper, we proposed a solution to the problem of continual learning with conflicting tasks through the lens of multi-task learning, whereby the learner remembers loss surface representations of older tasks, and jointly optimizes them when learning new tasks. Importantly, we addressed the empirical optimization problem of continual learning. For settings described in this work, PMF-CL retrieves the same solution as the corresponding offline joint-training benchmarks on either the true loss (for exact quadratic) or the loss surrogate (for QUB), and the generalization results follow from that benchmark. Our approach is generally applicable to loss functions admitting a loss-preserving finite representation and tractable incremental updates. We also derived explicit iterative algorithms for PMF-CL linear and basis-function regression, and for general loss functions with quadratic upper bounds (QUB), which subsumes linear classification problems. PMF-CL enjoys constant memory costs with respect to the number of tasks. Future directions of this research include deriving PMF-CL algorithms for more complex models (e.g., standard deep neural networks *without* the QUB property) and population-level generation bounds.

## AI USE STATEMENT

In this work, we used generative AI tools to: generate synthetic data sets, provide critical ingredients for proving mathematical claims, and implement methods. We have not used generative AI tools to: help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist in the writing of proofs, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results; and: assisting with translation is not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, and create or edit software code. We have reviewed all AI-assisted work.

We used Codex with GPT-5.6-Sol for writing code on controlled experiments on linear regression. The synthetic dataset for linear regression was created via Python code written by Codex. We also used ChatGPT for brainstorming the formal proof of minimality of MSI in the linear regression setup (Appendix B.1.2). We have verified all AI-generated results and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

Since our primary contribution is at a theoretical level, the complete proofs for all the theoretical results are presented in the Appendices B-E. Our source code is made available as anonymized supplementary material associated with this submission. All relevant parameters and experiments are described in the numerical experiments for linear and logistic regression in Appendix G.

## REFERENCES


P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, 2008. ISBN 9781400830244. doi: doi: 10.1515/9781400830244. URL https://doi.org/10.1515/9781400830244.

Tanveer Ahmad, Dongdong Zhang, Chao Huang, Hongcai Zhang, Ningyi Dai, Yonghua Song, and Huanxin Chen. Artificial intelligence in sustainable energy industry: Status quo, challenges and opportunities. *Journal of cleaner production*, 289:125834, 2021.

AImageLab. Mammoth: An extendible (general) continual learning framework based on pytorch. https://github.com/aimagelab/mammoth, 2020.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International conference on machine learning*, pp. 322–332. PMLR, 2019.

Jack Bell, Luigi Quarantiello, Eric Nuertey Coleman, Lanpei Li, Malio Li, Mauro Madeddu, Elia Piccoli, and Vincenzo Lomonaco. The future of continual learning in the era of foundation models: Three key directions. *arXiv preprint arXiv:2506.03320*, 2025.

Frederik Benzing. Unifying regularisation methods for continual learning. *arXiv preprint arXiv:2006.06357*, 2020.

Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the institute of Statistical Mathematics*, 44(1):197–200, 1992.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Marcos F. Criado, Fernando E. Casado, Roberto Iglesias, Carlos V. Regueiro, and Senén Barro. Non-iid data and continual learning processes in federated learning: A long road ahead. *Information Fusion*, 88:263–280, 2022. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2022.07.024. URL https://www.sciencedirect.com/science/article/pii/S1566253522000884.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10164–10173, 2022.

Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. doi: 10.1137/S0895479895290954. URL https://doi.org/10.1137/S0895479895290954.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *Conference on Learning Theory*, pp. 4028–4079. PMLR, 2022.

Itay Evron, Ran Levinstein, Matan Schliserman, Uri Sherman, Tomer Koren, Daniel Soudry, and Nathan Srebro. From continual learning to sgd and back: Better rates for continual linear models. *arXiv preprint arXiv:2504.04579*, 2025.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

P.G. Guest and P.G. Guest. *Numerical Methods of Curve Fitting*. Cambridge University Press, 2012. ISBN 9781107646957. URL https://books.google.com/books?id=UjnB0FIWv_AC.

Seungyub Han, Yeongmo Kim, Taehyun Cho, and Jungwoo Lee. On the convergence of continual learning with adaptive methods. In *Uncertainty in Artificial Intelligence*, pp. 809–818. PMLR, 2023.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. Federated learning meets multi-objective optimization. *IEEE Transactions on Network Science and Engineering*, 9(4):2039–2051, 2022.

C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media, 2012.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jin Jiang and Youmin Zhang. A revisit to block and recursive least squares for parameter estimation. *Computers & Electrical Engineering*, 30(5):403–416, 2004.

Peizhong Ju, Xiaojun Lin, and Ness Shroff. On the generalization power of overfitted two-layer neural tangent kernel models. In *International Conference on Machine Learning*, pp. 5137–5147. PMLR, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Satish Kumar Keshri, Nazreen Shah, and Ranjitha Prasad. On the convergence of continual federated learning using incrementally aggregated gradients. *arXiv preprint arXiv:2411.07959*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. In *International Conference on Machine Learning*, pp. 5327–5337. PMLR, 2020.

W.M. Kolb. *Curve Fitting for Programmable Calculators*. SYNTEC, 1984. ISBN 9780943494029. URL https://books.google.com/books?id=ZiLYAAAAMAAJ.

Song Lai, Zhe Zhao, Fei Zhu, Xi Lin, Qingfu Zhang, and Gaofeng Meng. Pareto continual learning: Preference-conditioned learning and adaption for dynamic stability-plasticity trade-off. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18008–18016, 2025.

Ran Levinstein, Amit Attia, Matan Schliserman, Uri Sherman, Tomer Koren, Daniel Soudry, and Itay Evron. Optimal rates in continual linear regression via increasing regularization. *arXiv preprint arXiv:2506.06501*, 2025.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=iEvAf8i6JjO.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Beyond not-forgetting: Continual learning with backward knowledge transfer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=diV1PpaP33.

Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pp. 21078–21100. PMLR, 2023.

Hong Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Towards understanding the transferability of deep representations. *arXiv preprint arXiv:1909.12031*, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Estelle Massart and P.-A. Absil. Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices. *SIAM Journal on Matrix Analysis and Applications*, 41(1):171–198, 2020. doi: 10.1137/18M1231389. URL https://doi.org/10.1137/18M1231389.

H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL http://arxiv.org/abs/1602.05629.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Gilles Meyer, Silvère Bonnabel, and Rodolphe Sepulchre. Regression on fixed-rank positive semidefinite matrices: A riemannian approach. *Journal of Machine Learning Research*, 12(18):593–625, 2011. URL http://jmlr.org/papers/v12/meyer11a.html.

Kaisa Miettinen. Introduction to multiobjective optimization: Noninteractive approaches. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowiński (eds.), *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pp. 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3. doi: 10.1007/978-3-540-88908-3_1. URL https://doi.org/10.1007/978-3-540-88908-3_1.

H. Motulsky and A. Christopoulos. *Fitting Models to Biological Data Using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting*. Oxford University Press, 2004. ISBN 9780198038344. URL https://books.google.com/books?id=g1FO9pquF3kC.

OpenAI, Josh Achiam, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

James O'Donnell and Casey Crownhart. We did the math on ai's energy footprint. here's the story you haven't heard. *MIT Technology Review*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Liangzu Peng, Paris Giampouras, and René Vidal. The ideal continual learner: An agent that never forgets. In *International Conference on Machine Learning*, pp. 27585–27610. PMLR, 2023.

R. L. Plackett. Some theorems in least squares. *Biometrika*, 37(1-2):149–157, 06 1950. ISSN 0006-3444. doi: 10.1093/biomet/37.1-2.149. URL https://doi.org/10.1093/biomet/37.1-2.149.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Partha Pratim Ray. Timely need for navigating the potential and downsides of llms in healthcare and biomedicine. *Briefings in bioinformatics*, 25(3):bbae214, 2024.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3AOj0RCNC2.

Donald Shenaj, Marco Toldo, Alberto Rigon, and Pietro Zanuttigh. Asynchronous federated continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5055–5063, 2023.

Ron Shepard, Scott R Brozell, and Gergely Gidofalvi. The representation and parametrization of orthogonal matrices. *The Journal of Physical Chemistry A*, 119(28):7924–7939, 2015.

Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 90–101. Springer, 2002.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 8, 1995.

Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1):25–46, 1995. ISSN 0921-8890. doi: https://doi.org/10.1016/0921-8890(95)00004-Y. URL https://www.sciencedirect.com/science/article/pii/092188909500004Y. The Biology and Technology of Intelligent Autonomous Agents.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Anastasiia Usmanova, François Portet, Philippe Lalanda, and German Vega. A distillation-based approach integrating continual learning and federated learning for pervasive services. *arXiv preprint arXiv:2109.04197*, 2021.

Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9385–9394, 2021.

Hongyi Wan, Shiyuan Ren, Wei Huang, Miao Zhang, Xiang Deng, Yixin Bao, and Liqiang Nie. Understanding the forgetting of (replay-based) continual learning via feature learning: Angle matters. In *Forty-second International Conference on Machine Learning*, 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024a.

Qiang Wang, Bingyan Liu, and Yawen Li. Traceable federated continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12872–12881, 2024b.

Zi Wang, Fei Wu, Feng Yu, Yurui Zhou, Jia Hu, and Geyong Min. Federated continual learning for edge-ai: A comprehensive survey. *arXiv preprint arXiv:2411.13740*, 2024c.

Yichen Wu, Hong Wang, Peilin Zhao, Yefeng Zheng, Ying Wei, and Long-Kai Huang. Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization. In *Forty-first international conference on machine learning*, 2024.

Xin Yang, Hao Yu, Xin Gao, Hao Wang, Junbo Zhang, and Tianrui Li. Federated continual learning via knowledge fusion: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(8): 3832–3850, 2024.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4782–4793, 2023.

# A Extended Literature Review

**Foundations of Continual Learning.** The field of CL, or lifelong learning, emerged from early research in robotic agents and task rehearsal (Thrun & Mitchell, 1995; Thrun, 1995; Silver & Mercer, 2002). It is deeply rooted in cognitive science, as the ability to consolidate new information without overwriting existing knowledge is a hallmark of human intelligence (French, 1999). In modern applications, maintaining this stability is critical for large-scale models in production, particularly in the biomedical sector (Ray, 2024; Bell et al., 2025).

**Empirical Methodologies.** The literature distinguishes three primary algorithmic families to mitigate forgetting (Wang et al., 2024a):

- **Replay-buffer Methods:** Lopez-Paz & Ranzato (2017); Chaudhry et al. (2019a); Aljundi et al. (2019); Buzzega et al. (2020) utilize a fixed memory buffer to store representative samples or generative features. Despite their success, these methods face challenges regarding memory sufficiency and lack a formal geometric intuition (Verwimp et al., 2021).
- **Regularization Methods:** Kirkpatrick et al. (2017); Zenke et al. (2017); Aljundi et al. (2018) introduce penalty terms that constrain model updates. By identifying parameters critical to previous tasks—often via the Fisher Information Matrix—they ensure the model stays within low-loss regions for past objectives.
- **Gradient-Projection Methods:** Works such as Saha et al. (2021); Lin et al. (2022a;b) focus on the geometry of the gradient space, projecting updates onto the Null space of past feature correlations to ensure zero interference between tasks.

**Theoretical Perspectives.** Understanding CL through loss geometry has gained significant traction. Liu et al. (2019) argue that flatter minima promote better task transferability, while Benzing (2020) provide a unifying framework for regularization. Recent work by Evron et al. (2022) and Levinstein et al. (2025) has established formal forgetting rates, with Levinstein et al. (2025) achieving optimal $O(1/k)$ rates via specific regularization schedules. Furthermore, Wan et al. (2025) and Lin et al. (2023) have analyzed how overparameterization and task signal alignment influence whether forgetting is benign.

**Theoretical Limits and PMF-CL.** While Knoblauch et al. (2020) established that perfect avoidance of forgetting is NP-hard, PMF-CL navigates this by reformulating CL as a multi-task learning (MTL) problem. Unlike existing methods that do not guarantee Pareto-optimality, PMF-CL uses a quadratic upper-bound framework to compute solutions that are consistent with joint optimization in hindsight, operating within a polynomial time complexity.

# B Analysis of MSI and Iterative Updates

## B.1 Linear Regression

### B.1.1 Iterative PMF-CL for Linear Regression (Theorem 4.2)

**Theorem B.1** (Iterative PMF-CL for Linear Regression)**.** *Storing the MSI of the task loss as* $\mathcal{I}_t^{\text{lin}} := (\boldsymbol{V}_t^e, \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$ *from Theorem 4.1, the PMF-CL update at the* $k^{th}$ *task can be written as the solution to:*

$$\forall\, k \in \{2, \dots, T\} \qquad \boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \frac{\alpha_k^{(k)}}{n_k} \boldsymbol{V}_k^e \boldsymbol{\Sigma}_k^{e\,2} (\widetilde{\boldsymbol{y}}_k - \boldsymbol{V}_k^{e\top} \boldsymbol{\theta}_{k-1}^\star) \tag{15}$$

*where* $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k} \boldsymbol{A}_{k-1} + \frac{\alpha_k^{(k)}}{n_k} \boldsymbol{V}_k^e \boldsymbol{\Sigma}_k^{e\,2} \boldsymbol{V}_k^{e\top}$, $N_k := \sum_{i=1}^k n_i$, *and* $\Delta\boldsymbol{\theta}_k = \boldsymbol{\theta}_k^\star - \boldsymbol{\theta}_{k-1}^\star$.

*Proof.* Defining $\boldsymbol{M}_t := \frac{1}{n_t} \boldsymbol{V}_t^e \boldsymbol{\Sigma}_t^{e\,2} \boldsymbol{V}_t^{e\top}$ and $\boldsymbol{\nu}_t := \frac{1}{n_t} \boldsymbol{V}_t^e \boldsymbol{\Sigma}_t^{e\,2} \widetilde{\boldsymbol{y}}_t$, Equation (6) upto the $k^{\text{th}}$ task can be simplified as:

$$\left( \sum_{t=1}^k \alpha_t^{(k)} \boldsymbol{M}_t \right) \boldsymbol{\theta}_k^\star = \sum_{t=1}^k \alpha_t^{(k)} \boldsymbol{\nu}_t \tag{16}$$

Then, denoting the $k^{\text{th}}$ solution in terms of the $(k-1)^{\text{th}}$ solution as an iterative update, i.e., $\boldsymbol{\theta}_k^\star = \boldsymbol{\theta}_{k-1}^\star + \Delta\boldsymbol{\theta}_k$, we can write:

$$\left(\sum_{t=1}^{k-1} \alpha_t^{(k)} \boldsymbol{M}_t + \alpha_k^{(k)} \boldsymbol{M}_k\right) (\boldsymbol{\theta}_{k-1}^\star + \Delta\boldsymbol{\theta}_k) = \sum_{t=1}^{k-1} \alpha_t^{(k)} \boldsymbol{\nu}_t + \alpha_k^{(k)} \boldsymbol{\nu}_k \tag{17}$$

Note that from the definition of $\alpha_t^{(k)}$ in (3), we have the property $\alpha_t^{(k)} = (N_{k-1}/N_k)\alpha_t^{(k-1)}$ for all $t \in [k]$. Upon rearranging (17) and recognizing that $\boldsymbol{\theta}_{k-1}^\star$ satisfies $\left(\sum_{t=1}^{k-1} \alpha_t^{(k-1)} \boldsymbol{M}_t\right) \boldsymbol{\theta}_{k-1}^\star = \sum_{t=1}^{k-1} \alpha_t^{(k-1)} \boldsymbol{\nu}_t$, we get:

$$\alpha_k^{(k)} \boldsymbol{M}_k \boldsymbol{\theta}_{k-1}^\star + \left(\sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{M}_t\right) \Delta\boldsymbol{\theta}_k = \alpha_k^{(k)} \boldsymbol{\nu}_k.$$

Let us define the accumulated information matrix as $\boldsymbol{A}_k := \sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{M}_t$, and the effective target as $\boldsymbol{z}_k := \alpha_k^{(k)}(\boldsymbol{\nu}_k - \boldsymbol{M}_k \boldsymbol{\theta}_{k-1}^\star)$. Then we can rewrite the above equation as:

$$\boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \boldsymbol{z}_k \tag{18}$$

Finally, we need to find a recursive update for $\boldsymbol{A}_k$, which is obtained as follows:

$$\begin{aligned}
\boldsymbol{A}_k &= \sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{M}_t \\
&= \sum_{t=1}^{k-1} \alpha_t^{(k)} \boldsymbol{M}_t + \alpha_k^{(k)} \boldsymbol{M}_k \\
&= \frac{N_{k-1}}{N_k} \sum_{t=1}^{k-1} \alpha_t^{(k-1)} \boldsymbol{M}_t + \alpha_k^{(k)} \boldsymbol{M}_k \\
&= \frac{N_{k-1}}{N_k} \boldsymbol{A}_{k-1} + \alpha_k^{(k)} \boldsymbol{M}_k
\end{aligned} \tag{19}$$

From (19), (18), and the definiton of $\boldsymbol{\theta}_k^\star$, we get the iterative update in the theorem. The iterative method is formally presented in Algorithm 3. □

### B.1.2 MSI FOR LINEAR REGRESSION (THEOREM 4.1)

**Theorem B.2** (MSI for Linear Regression)**.** *Using the eSVD of $\boldsymbol{X}_t$, we can write the loss function for linear regression as:*

$$L_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^e}^2 + \min_{\boldsymbol{\theta}' \in \mathcal{W}} L_t(\boldsymbol{\theta}') \tag{20}$$

*where we use the notation $\|\boldsymbol{z}\|_{\boldsymbol{A}}^2 := \|\boldsymbol{A}\boldsymbol{z}\|_2^2$, $\widetilde{\boldsymbol{y}} = \boldsymbol{\Sigma}_t^{e\,-1} \boldsymbol{U}_t^e \boldsymbol{y}_t$. Then, an MSI representing the linear regression loss (upto an additive constant) is $\mathcal{I}_t^{\text{lin}} := (\mathcal{S}(\boldsymbol{V}_t^e), \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$, where $\mathcal{S}(\cdot)$ is a compact storage model for the orthonormal singular-vector matrix $\boldsymbol{V}_t^e$ using Givens rotations (Shepard et al., 2015). The memory size for this representation is* $\text{size}(\mathcal{I}_t^{\text{lin}}) = dr_t - \frac{r_t(r_t+1)}{2}$ *which is provably* minimal.

*Proof.* Under the *full* SVD of $\boldsymbol{X}_t$, we can rewrite the task loss in (4) as:

$$\begin{aligned}
L_t(\boldsymbol{\theta}) &= \frac{1}{2n_t} \left\| \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top \boldsymbol{\theta} - \boldsymbol{y}_t \right\|^2 \\
&= \frac{1}{2n_t} \left\| \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top \boldsymbol{\theta} - \boldsymbol{U}_t^\top \boldsymbol{y}_t \right\|^2 \\
&= \frac{1}{2n_t} \left\| \begin{bmatrix} \boldsymbol{\Sigma}_t^e & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{V}_t^\top \boldsymbol{\theta} - \begin{bmatrix} \boldsymbol{U}_t^{e\top} \\ \boldsymbol{U}_t^{\perp\top} \end{bmatrix} \boldsymbol{y}_t \right\|^2
\end{aligned} \tag{21}$$

where we get (21) from the relationship between SVD and eSVD. The matrix $\boldsymbol{U}_t^{\perp} \in \mathbb{R}^{n_t \times (n_t - r_t)}$ corresponds to the last $(n_t - r_t)$ columns of $\boldsymbol{U}_t$. Finally, we can write:

$$L_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{\Sigma}_t^e \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \boldsymbol{U}_t^{e\top} \boldsymbol{y}_t \right\|^2 + \frac{1}{2n_t} \left\| \boldsymbol{U}_t^{\perp\top} \boldsymbol{y}_t \right\|^2 \tag{22}$$

By recognizing that $\boldsymbol{\Sigma}_t^e$ is square and positive definite, and using the notations $\|\boldsymbol{z}\|_{\boldsymbol{A}}^2 := \boldsymbol{z}^T \boldsymbol{A}^2 \boldsymbol{z}$ and $\widetilde{\boldsymbol{y}}_t := (\boldsymbol{\Sigma}_t^e)^{-1} \boldsymbol{U}_t^{e\top} \boldsymbol{y}_t$. Also, note that the last term $(0.5/n_t) \left\| \boldsymbol{U}_t^{\perp\top} \boldsymbol{y}_t \right\|^2$ is the minimum task loss attainable by optimizing $\boldsymbol{\theta}$. Hence, we can rewrite (22) as:

$$L_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^e}^2 + \min_{\boldsymbol{\theta}' \in \mathcal{W}} L_t(\boldsymbol{\theta}') \tag{23}$$

Equation (23) is a compact representation of the task loss $L_t(\cdot)$. In order to *remember* the task (modulo an additive constant) for continual learning, the continual learner would need to store the tuple:

$$\mathcal{I}_t^{\text{lin}} := (\boldsymbol{V}_t^e, \operatorname{diag}(\boldsymbol{\Sigma}_t^e), \widetilde{\boldsymbol{y}}_t). \tag{24}$$

Note that in terms of storage $\mathcal{I}_t$ in (24) is not yet optimized, since the matrix $\boldsymbol{V}_t^e$ is orthonormal, i.e., the $dr_t$ entries of $\boldsymbol{V}_t^e$ are not independent.

- **The Stiefel-manifold dimension** The matrix $\boldsymbol{V}_t^e$ belongs to the real Stiefel manifold

$$\operatorname{St}(d, r) := \{\boldsymbol{V}_t^e \in \mathbb{R}^{d \times r} : \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}_r\}. \tag{25}$$

  The symmetric matrix equation $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}_r$ imposes $\frac{r(r+1)}{2}$ independent scalar constraints: $r$ unit-norm constraints and $r(r-1)/2$ pairwise-orthogonality constraints. Therefore,

$$\dim \operatorname{St}(d, r) = dr - \frac{r(r+1)}{2}. \tag{26}$$

  This standard dimension formula and the geometry behind it are discussed in (Edelman et al., 1998; Absil et al., 2008). Next, we show that *Givens rotations* (Shepard et al., 2015) provides a compact representation that matches the dimensions in (26). Finally, we prove that $\mathcal{I}_t$ in (24) is indeed minimal under the storage model represented by Givens rotations.

- **Compact representation via Givens rotations** The Stiefel dimension is not merely an abstract parameter count. At least locally and generically, an orthonormal matrix $\boldsymbol{V}^e \in \operatorname{St}(d, r)$ (dropping the $t$ subscript for convenience) can be encoded using exactly $\dim \operatorname{St}(d, r)$ real scalars. Let $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{I}_r \\ 0 \end{bmatrix} \in \mathbb{R}^{d \times r}$. A Givens rotation $\boldsymbol{G}_{ij}(\phi) \in O(d)$ is an orthonormal matrix that acts as a planar rotation through angle $\phi$ in the coordinate plane spanned by the $i^{\text{th}}$ and $j^{\text{th}}$ standard basis vectors, and acts as the identity on all other coordinates.

  Next, we can apply a QR-style elimination to $\boldsymbol{V}^e$. For column $i$, use Givens rotations to eliminate the entries in rows $d, (d-1), \ldots, (i+1)$. This requires $(d-i)$ rotations for column $i$. Repeating this operation for $i = 1, \ldots, r$ requires

$$\sum_{i=1}^{r} (d - i) = dr - \frac{r(r+1)}{2} = \dim \operatorname{St}(d, r) \tag{27}$$

  rotation angles.

  After elimination, the leading $r \times r$ block is both upper triangular and orthogonal. It must therefore be a diagonal sign matrix $\boldsymbol{D} = \operatorname{Diag}(s_1, \ldots, s_r)$, for $s_i \in \{-1, +1\}$. Reversing the elimination gives a representation of the form

$$\boldsymbol{V}^e = \boldsymbol{G}_1(\phi_1)^\top \boldsymbol{G}_2(\phi_2)^\top \cdots \boldsymbol{G}_m(\phi_m)^\top \boldsymbol{E} \boldsymbol{D}, \qquad m = dr - \frac{r(r+1)}{2}, \tag{28}$$

  where the order and coordinate planes of the rotations are fixed in advance. Thus, only the $m$ real angles and, if desired, $r$ discrete sign bits need to be stored. The sign bits do not contribute

to the manifold dimension. Moreover, for the loss representation in (23), the simultaneous transformation

$$\boldsymbol{v}_i^e \longmapsto -\boldsymbol{v}_i^e, \qquad \widetilde{\boldsymbol{y}}_i \longmapsto -\widetilde{\boldsymbol{y}}_i$$

leaves the loss unchanged. Consequently, column-sign ambiguity can be removed by a canonical convention when the goal is to store the loss rather than a particular numerical SVD.

**Example 1.** Say $d = 4$ and $r = 2$, the displayed matrix $\boldsymbol{V}^e$ has eight entries, but $\dim \mathrm{St}(4, 2) = 5$. It can therefore be represented generically by five fixed-order Givens angles, together with only discrete sign conventions.

- **Minimality of (24) under (28)** We proceed by proving that an alternative representation via the pair$(\boldsymbol{A}, \boldsymbol{b})$ defined by:

$$\boldsymbol{A} := \nabla^2 L(\boldsymbol{\theta}), \qquad \boldsymbol{b} := \boldsymbol{A}\boldsymbol{\theta} - \nabla L(\boldsymbol{\theta}), \tag{29}$$

fully determines the non-constant part of the loss in (23), i.e.:

$$L_t(\boldsymbol{\theta}) \equiv \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}_t \boldsymbol{\theta} - \boldsymbol{b}_t^\top \boldsymbol{\theta}, \tag{30}$$

and is *minimal*. Then, we show that both the representations, (24) with the compact (28) and (29) have identical memory footprint.

In the following proposition, we show that the canonical representation $(\boldsymbol{A}, \boldsymbol{b})$ *uniquely* determines the loss function in (30) upto a constant.

**Proposition B.3** (Uniqueness of the canonical coefficients)**.** *Let*

$$q_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}^\top \boldsymbol{\theta} + c \quad \textit{and} \quad q_{\boldsymbol{A}',\boldsymbol{b}'}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}'\boldsymbol{\theta} - (\boldsymbol{b}')^\top \boldsymbol{\theta} + c', \tag{31}$$

*where $\boldsymbol{A}$ and $\boldsymbol{A}'$ are symmetric. The two functions differ only by an additive constant for every $\boldsymbol{\theta} \in \mathbb{R}^d$ if and only if $\boldsymbol{A} = \boldsymbol{A}'$ and $\boldsymbol{b} = \boldsymbol{b}'$.*

*Proof.* If the difference is constant, its Hessian is zero, which gives $\boldsymbol{A} - \boldsymbol{A}' = \boldsymbol{0}$. Its gradient is then $-(\boldsymbol{b} - \boldsymbol{b}')$, which must also be zero. The reverse implication is immediate. □

This proves that $(\boldsymbol{A}, \boldsymbol{b})$ is a minimal required representation to remember the loss function for linear regression. Lastly, the memory size of $(\boldsymbol{A}, \boldsymbol{b})$ can be determined as follows. Let $r = \text{rank}\,(\boldsymbol{X}) = \text{rank}\,(\boldsymbol{A})$, after dropping the $t$ subscript for convenience. The set of $d \times d$ symmetric p.s.d matrices of fixed rank $r$ is a smooth manifold of dimension

$$\dim\{\boldsymbol{A} = \boldsymbol{A}^\top \succeq 0 : \text{rank}\,(\boldsymbol{A}) = r\} = dr - \frac{r(r-1)}{2}. \tag{32}$$

One way to see this is to write

$$\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{Y}^\top, \qquad \boldsymbol{Y} \in \mathbb{R}^{d\times r}, \quad \text{rank}\,(\boldsymbol{Y}) = r.$$

The factor $\boldsymbol{Y}$ contributes $dr$ parameters, but $\boldsymbol{Y}$ and $\boldsymbol{Y}\boldsymbol{Q}$ give the same $\boldsymbol{A}$ for every orthonormal $\boldsymbol{Q} \in O(\boldsymbol{r})$. Since

$$\dim O(r) = \frac{r(r-1)}{2},$$

the quotient has the dimension in (32); see, for example, (Massart & Absil, 2020; Meyer et al., 2011).

For a least-squares loss,

$$\boldsymbol{b} = \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{y} \in \text{range}(\boldsymbol{X}^\top) = \text{range}(\boldsymbol{X}^\top \boldsymbol{X}) = \text{range}(\boldsymbol{A}).$$

After $\boldsymbol{A}$ is fixed, $\boldsymbol{b}$ therefore contributes $r$ additional degrees of freedom. Consequently,

$$\dim(A, b) = \left(dr - \frac{r(r-1)}{2}\right) + r = dr - \frac{r(r-3)}{2}. \tag{33}$$

*Remark* B.4 (Realizability). Every pair with $\boldsymbol{A} \succeq \boldsymbol{0}$, $\text{rank}\,(\boldsymbol{A}) = r$, and $\boldsymbol{b} \in \text{range}(\boldsymbol{A})$ can be realized by a least-squares problem with sufficiently many rows. Thus, the dimension count is a property of the actual rank-$r$ least-squares loss family, not merely of a larger artificial class of quadratics.

□

### B.2 Quadratic Upper Bound (Theorem 5.2)

**Theorem B.5** (Iterative PMF-CL for QUB loss)**.** *Storing the MSI of the task loss as $\mathcal{I}_t = \{\boldsymbol{H}_t, \boldsymbol{\theta}_t^{\min}\}$ and using $\boldsymbol{\alpha}^{(k)}$ from (3), the iterative PMF-CL update can be written as:*

$$\boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \alpha_k^{(k)} \boldsymbol{H}_k (\boldsymbol{\theta}_k^{\min} - \boldsymbol{\theta}_{k-1}^{\star}). \tag{34}$$

*where $\boldsymbol{A}_0 = 0$, $\boldsymbol{A}_k = \frac{N_{k-1}}{N_k}\boldsymbol{A}_{k-1} + \alpha_k^{(k)}\boldsymbol{H}_k$ and $N_k = \sum_{i=1}^{k} n_i$.*

*Proof.* The proof mostly mirrors the proof of Theorem 4.2. Consider the linear equation for the Pareto-optimal solution by PMF-CL, i.e.:

$$\left(\sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{H}_t\right) \boldsymbol{\theta}_k^{\star} = \sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{H}_t \boldsymbol{\theta}_t^{\min}. \tag{35}$$

Changing the subscript on $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{\star}$ to $k$ for denoting the $k^{\text{th}}$ iterate, and defining $\Delta\boldsymbol{\theta}_k := \boldsymbol{\theta}_k^{\star} - \boldsymbol{\theta}_{k-1}^{\star}$, we can rewrite:

$$\left(\sum_{t=1}^{k-1} \alpha_t^{(k)} \boldsymbol{H}_t + \alpha_k^{(k)} \boldsymbol{H}_k\right) (\boldsymbol{\theta}_{k-1}^{\star} + \Delta\boldsymbol{\theta}_k) = \sum_{t=1}^{k-1} \alpha_t^{(k)} \boldsymbol{H}_t \boldsymbol{\theta}_t^{\min} + \alpha_k^{(k)} \boldsymbol{H}_k \boldsymbol{\theta}_k^{\min}$$

$$\implies \left(\sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{H}_t\right) \Delta\boldsymbol{\theta}_k + \alpha_k^{(k)} \boldsymbol{H}_k \boldsymbol{\theta}_{k-1}^{\star} = \alpha_k^{(k)} \boldsymbol{H}_k \boldsymbol{\theta}_k^{\min} \tag{36}$$

$$\implies \boldsymbol{A}_k \Delta\boldsymbol{\theta}_k = \alpha_k^{(k)} \boldsymbol{H}_k (\boldsymbol{\theta}_k^{\min} - \boldsymbol{\theta}_{k-1}^{\star}) \tag{37}$$

where $\boldsymbol{A}_k := \sum_{t=1}^{k} \alpha_t \boldsymbol{H}_t$. Again, similar to the linear regression case, we can find a recursive update for $\boldsymbol{A}_k$ as:

$$\boldsymbol{A}_k = \frac{N_{k-1}}{N_k} \boldsymbol{A}_{k-1} + \alpha_k^{(k)} \boldsymbol{H}_k, \tag{38}$$

which completes the proof. □

## C Forgetting Analysis

### C.1 Linear and Basis Function Regression (Theorem 4.3)

**Theorem C.1.** *The forgetting of the $t^{th}$ task after learning $k$ tasks using Algorithm 3 is given by:*

$$F_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{A}_k^{\dagger} \boldsymbol{\nu}_k - \widetilde{\boldsymbol{y}}_t \right\|^2 \tag{39}$$

*where $\boldsymbol{A}_k$ and $\boldsymbol{\nu}_k$ are defined as:*

$$\boldsymbol{A}_k := \sum_{t=1}^{k} \frac{\alpha_t^{(k)}}{n_t} \boldsymbol{V}_t^{e} \boldsymbol{\Sigma}_t^{e2} \boldsymbol{V}_t^{e\top} \tag{40a}$$

$$\boldsymbol{\nu}_k := \sum_{t=1}^{k} \alpha_t^{(k)} \boldsymbol{V}_t^{e} \boldsymbol{\Sigma}_t^{e2} \widetilde{\boldsymbol{y}}_t. \tag{40b}$$

*and $\dagger$ represents the Moore-Penrose pseudo-inverse.*

*Proof.* From the definition of forgetting in Definition 3.7 at parameter $\boldsymbol{\theta}$, we have:

$$F_t(\boldsymbol{\theta}) = L_t(\boldsymbol{\theta}) - \inf_{\boldsymbol{\theta}' \in \mathcal{W}} L_t(\boldsymbol{\theta}') \tag{41}$$

From Theorem 4.1, in the linear regression setting, we get:

$$F_t(\boldsymbol{\theta}) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^{e}}^2 \tag{42}$$

Also, since Algorithm 3 uses a memory efficient implementation for computing $\boldsymbol{\theta}_k^\star$ in (6), the optimal parameters can be written as:

$$\boldsymbol{\theta}_k^\star = \boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k \tag{43}$$

where $\boldsymbol{A}_k$ and $\boldsymbol{\nu}_k$ are defined as in (40) and $\boldsymbol{A}^\dagger$ indicates the Moore-Penrose pseudo-inverse. Then, the degree of forgetting the $t^{\text{th}}$ task after learning $k$ tasks is given by:

$$F_t(\boldsymbol{\theta}_k^\star) = \frac{1}{2n_t} \left\| \boldsymbol{V}_t^{e\top} \boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \widetilde{\boldsymbol{y}}_t \right\|^2 \tag{44}$$

Which completes the proof. □

### C.2 Forgetting for QUB Loss (Theorem 5.3)

**Theorem C.2** (Forgetting for QUB Loss)**.** *Suppose $\boldsymbol{A}_k$ is as defined in Theorem 5.2, $\boldsymbol{\nu}_k := \sum_{i=1}^k \alpha_i^{(k)} \boldsymbol{H}_i \boldsymbol{\theta}_t^{\min}$ and the PMF-CL solution $\boldsymbol{\theta}_k^\star = \boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k$ generated by Algorithm 2 upto task $k$. Then, under Assumption 5.1, the degree of forgetting for task $t < k$ after learning task $k$ can be upper-bounded as:*

$$F_t^{(k)} \leq \frac{1}{2} (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{H}_t (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min}). \tag{45}$$

*Proof.* Again, similar to the proof for the regression case, Theorem C.1, we start from the definition of forgetting $F_t(\boldsymbol{\theta})$ from Definition 3.7, but this time for the QUB function:

$$F_t^{\text{qub}}(\boldsymbol{\theta}) = L_t^{\text{qub}}(\boldsymbol{\theta}) - \inf_{\boldsymbol{\theta}' \in \mathcal{W}} L_t^{\text{qub}}(\boldsymbol{\theta}') \tag{46}$$

where, from the QUB for task $t$ in (9) we get:

$$F_t^{\text{qub}}(\boldsymbol{\theta}) = \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{H}_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min}) \tag{47}$$

Then at the PMF-CL solution $\boldsymbol{\theta}_k^\star = \boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k$ after learning task $k$, we get:

$$F_t^{\text{qub}}(\boldsymbol{\theta}_k^\star) = \frac{1}{2} (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{H}_t (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min}) \tag{48}$$

where $\boldsymbol{A}_k := \sum_{i=1}^k \alpha_i^{(k)} \boldsymbol{H}_i$ and $\boldsymbol{\nu}_k := \sum_{i=1}^k \alpha_i^{(k)} \boldsymbol{H}_i \boldsymbol{\theta}_i^{\min}$.

Since the QUB is an upper bound to the true loss, the forgetting measure is also an upper bound to the true loss, since $\min_{\boldsymbol{\theta} \in \mathcal{W}} L_t^{\text{qub}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathcal{W}} L_t^{\text{qub}}(\boldsymbol{\theta}) = \boldsymbol{\theta}_t^{\min}$ by definiton. Hence, we can conclude that:

$$F_t^{(k)} \leq \frac{1}{2} (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{H}_t (\boldsymbol{A}_k^\dagger \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min}), \tag{49}$$

which completes the proof. □

## D Proofs of Other Lemmas

### D.1 Basis Functions and 2-Layer NTK Models

In this section , we show that the framework developed for linear regression in Section 4.1 can be extended to a larger class of models that are linear in their parameters. Consider models $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ that are non-linear in their inputs but linear in $\boldsymbol{\theta}$, i.e., models operating via a *basis function*:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\theta}, \qquad \boldsymbol{\phi}(\cdot) : \mathbb{R}^d \to \mathbb{R}^m \tag{50}$$

where $\phi(\cdot)$ is a non-linear *basis function* map (Hastie et al., 2001). These models cover curve fitting applications and have a long history (Kolb, 1984; Motulsky & Christopoulos, 2004; Guest & Guest, 2012). For instance, polynomial regression, log-linear regression, exponential regression, etc, are all basis function methods. Under the *neural tangent kernel* (NTK) approximation (Jacot et al., 2018; Arora et al., 2019), we can also represent a two-layer neural network as an approximate linear model. Consider a two-layer multi-layer perceptron (MLP) neural network (NN) with ReLU activations:

$$f_{\boldsymbol{w},\boldsymbol{V}}(\boldsymbol{x}) = \boldsymbol{w}^T \sigma(\boldsymbol{V}\boldsymbol{x}), \tag{51}$$

where $\boldsymbol{V} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{w} \in \mathbb{R}^m$. We follow (Arora et al., 2019; Ju et al., 2021) for obtaining the NTK approximation. Here, we state the result as a lemma, and delegate the proof to Appendix D.

**Lemma D.1** (Two-layer NN as Basis Function under NTK)**.** *Under the NTK regime, for small changes in the weights $\Delta \boldsymbol{v}$, the change in the output of the two-layer MLP in (51) can be approximated as: $\Delta f_{\boldsymbol{w},\boldsymbol{V}}(\boldsymbol{x}) \approx \left[\boldsymbol{x} \otimes (\boldsymbol{w} \odot \mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}})\right]^T \Delta \boldsymbol{v}$, where $\otimes$ represents the Kronecker product of two matrices. Then, under fixed $\boldsymbol{w}$, the two-layer MLP can be represented by (50) for: $\phi_{\boldsymbol{w},\boldsymbol{V}} : \boldsymbol{x} \to \boldsymbol{x} \otimes (\boldsymbol{w} \odot \mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}})$.*

*Proof.* Consider a 2-layer network[2] $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ parametrized by the weights $\boldsymbol{w} \in \mathbb{R}^k$ and $\boldsymbol{V} \in \mathbb{R}^{k\times d}$, given by:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\sigma}(\boldsymbol{V}\boldsymbol{x}) \tag{52}$$

where $\boldsymbol{\sigma}(\cdot) : \mathbb{R}^k \to \mathbb{R}^k$ is an element-wise ReLU function, i.e.:

$$\boldsymbol{\sigma}(\boldsymbol{z}) = \mathbb{I}_{\boldsymbol{z}\geq \boldsymbol{0}} \odot \boldsymbol{z}, \tag{53}$$

$\mathbb{I}_{\boldsymbol{z}\geq \boldsymbol{0}}$ is an element-wise indicator function, and $\odot$ is the element-wise product. Suppose that $\boldsymbol{w}$ and $\boldsymbol{V}$ are initialized randomly and $\boldsymbol{w}$ is frozen, and $\boldsymbol{V}$ is updated. Then, the change in the output of the network for a given input $\boldsymbol{x}$ can be represented as:

$$f(\boldsymbol{x}) + \Delta f(\boldsymbol{x}) = \boldsymbol{w}^T \left[\mathbb{I}_{(\boldsymbol{V}+\Delta \boldsymbol{V})\boldsymbol{x}\geq \boldsymbol{0}} \odot (\boldsymbol{V} + \Delta \boldsymbol{V})\boldsymbol{x}\right] \tag{54}$$

For small changes in input, say $\boldsymbol{z} \to \Delta \boldsymbol{z}$, the indicator function $\mathbb{I}_{(\boldsymbol{z}+\Delta \boldsymbol{z})\geq 0} \approx \mathbb{I}_{\boldsymbol{z}\geq 0}$ is approximately constant, so we can approximate the output as (we use the subscript $\boldsymbol{w}, \boldsymbol{V}$ to make explicit the dependence of the output on the original randomly chosen parameters.):

$$\begin{aligned}
\Delta f_{\boldsymbol{w},\boldsymbol{V}}(\boldsymbol{x}) &\approx \boldsymbol{w}^T \left[\mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}} \odot (\boldsymbol{V} + \Delta \boldsymbol{V})\boldsymbol{x}\right] - f(\boldsymbol{x}) \\
&= \boldsymbol{w}^T \left[\mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}} \odot \Delta \boldsymbol{V}\boldsymbol{x}\right] && (55) \\
&= (\boldsymbol{w} \odot \mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}})^T \Delta \boldsymbol{V}\boldsymbol{x} && (56) \\
&= \mathrm{Tr}(\boldsymbol{x}(\boldsymbol{w} \odot \mathbb{I}_{\boldsymbol{V}\boldsymbol{x}\geq \boldsymbol{0}})^T \Delta \boldsymbol{V}) && (57)
\end{aligned}$$

In (55) we use the definition of $f(\boldsymbol{x})$ to cancel out the first expanded term in the square bracket. Clearly this is a linear function of $\Delta \boldsymbol{V}$. Suppose $\boldsymbol{v}$ is a vectorized version of $\boldsymbol{V}$ obtained by stacking its columns one below the other. □

The mapping in Lemma D.1 can be plugged into Algorithm 3 to obtain an approximate PMF-CL algorithm for two-layer MLPs in the NTK regime. For basis function models, one can simply change the eSVD on $\boldsymbol{X}_t$ in Algorithm 3 to an eSVD on the dataset in feature space, i.e., $\boldsymbol{\phi}(\boldsymbol{X}_t)$. Note that the memory complexity would then scale as $\mathcal{O}\left(m^2\right)$ rather than $\mathcal{O}\left(d^2\right)$, where $m$ is the dimension of the feature space.

### D.2 Proof of Lemma 3.5

**Lemma D.2** (MTL for global ERM)**.** *The solution to the MTL problem in (2) with preferences chosen as in (3) is equivalent to the solution of the global optimization problem:*

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}\in\mathcal{W}}{\arg\min}\, \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \tag{58}$$

*where $\mathcal{D}$ is a concatenation of all datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$.*

*Proof.* We proceed by showing that both the MTL problem with the sample-weighted preferences and the global problem have the same objective function. For the linearly scalarized MTL problem

[2]Here we consider the scalar-output case for simplicity, but our method is also extendable to the vector-output case.

with preferences as in (3), the objective function is given by:

$$\sum_{t=1}^{T} \frac{n_t}{n} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t) = \frac{1}{n} \sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t) \tag{59}$$

$$= \frac{1}{n} \sum_{t=1}^{T} \sum_{i=1}^{n_t} \ell(\boldsymbol{\theta}; (\boldsymbol{x}_t^{(i)}, y_t^{(i)})) \tag{60}$$

$$= \frac{1}{n} \sum_{(\boldsymbol{x},y) \in \mathcal{D}} \ell(\boldsymbol{\theta}; (\boldsymbol{x}, y)) \tag{61}$$

$$= \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \tag{62}$$

Thus, both problems minimize the same objective function, and hence have the same solution. □

# E Example QUB Loss: Continual Logistic Regression

## E.1 Binary logistic regression

The method we derived in Section 5 subsumes the case of continual logistic regression, as we will show in this appendix. First, we will look at the simplified setting of binary logistic regression in this section. The task loss function for logistic regression using a linear model parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$ is given as:

$$L_t(\boldsymbol{\theta}) = -\frac{1}{n_t} \sum_{i=1}^{n_t} \left[ y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right] \tag{63}$$

where $p_i = \frac{1}{1+\exp(-\boldsymbol{x}_i^\top \boldsymbol{\theta})} \in (0, 1)$ is the sigmoid function applied to the linear model[3] $f(\boldsymbol{\theta}, \boldsymbol{x}_i) := \boldsymbol{x}_i^\top \boldsymbol{\theta}$. Computing the first derivative of the loss function with respect to $\boldsymbol{\theta}_t^\star$:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} L_t(\boldsymbol{\theta}) &= -\frac{1}{n_t} \sum_{i=1}^{n_t} \left[ \frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i} \right] \nabla_{\boldsymbol{\theta}} p_i \\
&= -\frac{1}{n_t} \sum_{i=1}^{n_t} \left[ \frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i} \right] p_i(1 - p_i)(-\boldsymbol{x}_i) \\
&= \frac{1}{n_t} \sum_{i=1}^{n_t} (y_i - p_i) \boldsymbol{x}_i. \qquad (64)
\end{aligned}$$

And, the second derivative:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}^2 L_t(\boldsymbol{\theta}) &= \frac{1}{n_t} \sum_{i=1}^{n_t} \boldsymbol{x}_i \left( p_i(1 - p_i) \boldsymbol{x}_i^\top \right) \\
&= \frac{1}{n_t} \boldsymbol{X}_t^\top \mathrm{Diag}(\boldsymbol{q}_t) \boldsymbol{X}_t \qquad (65)
\end{aligned}$$

where $\boldsymbol{q}_t$ is the vector with the $i^{\text{th}}$ element $p_i(1 - p_i)$, and $\mathrm{Diag}(\cdot)$ is the corresponding diagonal matrix.

Next, note that the quantity $p_i(1 - p_i)$ is upper bounded by $0.25$. Thus, the $\mathrm{Diag}(\boldsymbol{q}_t)$ in the Hessian in (65), is always upper bounded by the matrix $\overline{\boldsymbol{Q}}_t := 0.25\boldsymbol{I}$, in the sense that $\left(\overline{\boldsymbol{Q}}_t - \mathrm{Diag}(\boldsymbol{q}_t)\right)$ is always positive semi-definite. Thus, we can construct the upper-bounding surrogate loss function:

$$\begin{aligned}
L_t^{\text{qub}}(\boldsymbol{\theta}) &= L_t(\boldsymbol{\theta}_t^{\min}) + \frac{1}{8n_t} (\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min})^\top \boldsymbol{X}_t^\top \boldsymbol{X}_t (\boldsymbol{\theta} - \boldsymbol{\theta}_t^{\min}) \\
&= L_t(\boldsymbol{\theta}_t^{\min}) + \frac{1}{8n_t} \left\| \boldsymbol{X}_t \boldsymbol{\theta} - \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min} \right\|_2^2 \qquad (66) \\
&= L_t(\boldsymbol{\theta}_t^{\min}) + \frac{1}{8n_t} \left\| \boldsymbol{V}_t^e \boldsymbol{\theta} - \widetilde{\boldsymbol{y}}_t \right\|_{\boldsymbol{\Sigma}_t^e}^2 \qquad (67)
\end{aligned}$$

[3] We omit the superscript of $t$ on $\boldsymbol{x}_i^t$ and $y_i^t$ to avoid notational clutter.

where $\boldsymbol{U}_t^e \boldsymbol{\Sigma}_t^e \boldsymbol{V}_t^{e\top}$ is the SVD of $\boldsymbol{X}_t$, and $\widetilde{\boldsymbol{y}} := (\boldsymbol{\Sigma}_t^e)^{-1} \boldsymbol{U}_t^{e\top} \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min}$. which is identical to the MSE loss in linear regression with the substitution $\boldsymbol{y}_t := \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min}$. As in the linear regression case, the MSI can be reduced significantly by storing the right singular matrix and the non-zero singular values of $\boldsymbol{X}_t$. Hence the MSI for task $t$ is $\mathcal{I}_t = (\boldsymbol{V}_t^e, \boldsymbol{\Sigma}_t^e, \widetilde{\boldsymbol{y}}_t)$, where $\widetilde{\boldsymbol{y}}_t := (\boldsymbol{\Sigma}_t^e)^{-1} \boldsymbol{U}_t^{e\top} \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min}$.

Now, the problem reduces to computing the Pareto-optimal solution of the surrogate task loss functions. For each task, first we compute the minimizer of the task loss $\boldsymbol{\theta}_t^{\min}$, then, we can directly apply the updates in Algorithm 3, with $\boldsymbol{y}_t = \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min}$ to compute the Pareto-optimal solution of the surrogate loss functions. Note that since the surrogate functions are upper bounds to the corresponding true objectives, the solutions computed after each task will satisfy the forgetting upper bound in Theorem 5.3.

**Deriving the forgetting upper-bound.** From Theorem 5.3 the forgetting upper bound of our Continual Logistic Regression algorithm can be derived as follows. Recall that for QUB problems, the learned parameters after task $k$, the forgetting for a previous task $t < k$ is upper bounded by:

$$F_t^{(k)} \leq \frac{1}{2} (\boldsymbol{A}_k^{\dagger} \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min})^{\top} \boldsymbol{H}_t (\boldsymbol{A}_k^{\dagger} \boldsymbol{\nu}_k - \boldsymbol{\theta}_t^{\min}) \tag{68}$$

where $\boldsymbol{A}_k$ is recursively defined as: $\boldsymbol{A}_0 = 0, \boldsymbol{A}_k = \frac{N_{k-1}}{N_k} \boldsymbol{A}_{k-1} + \alpha_k^{(k)} \boldsymbol{H}_k, \quad \forall\ k \in [T]$. and $\boldsymbol{\nu}_k := \sum_{i=1}^{k} \alpha_i^{(k)} \boldsymbol{H}_i \boldsymbol{\theta}_i^{\min}$. For logistic regression, the Hessian upper bound $\boldsymbol{H}_t$ is given by, $\boldsymbol{H}_t = \frac{1}{4n_t} \boldsymbol{X}_t^{\top} \boldsymbol{X}_t$. Substituting this into (68) we obtain:

$$F_t^{(k)} \leq \frac{1}{8n_t} \left\| \boldsymbol{X}_t \boldsymbol{A}_k^{\dagger} \boldsymbol{\nu}_k - \boldsymbol{X}_t \boldsymbol{\theta}_t^{\min} \right\|^2, \tag{69}$$

where $\boldsymbol{\nu}_k := \sum_{i=1}^{k} \frac{\alpha_i^{(k)}}{4n_i} \boldsymbol{X}_i^{\top} \boldsymbol{X}_i \boldsymbol{\theta}_i^{\min}$.

### E.2 Multi-class logistic regression

Note that the loss for multi-class logistic regression with $K$ classes is given by:

$$L_t(\boldsymbol{\theta}) = -\frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{k=1}^{K} y_{i,k}^t \log \left( \frac{\exp(\boldsymbol{\theta}_k^{\top} \boldsymbol{x}_i^t)}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^{\top} \boldsymbol{x}_i^t)} \right). \tag{70}$$

Denoting $p_{i,k} := \frac{\exp(\boldsymbol{\theta}_k^{\top} \boldsymbol{x}_i^t)}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^{\top} \boldsymbol{x}_i^t)}$, and $\boldsymbol{p}_i = [p_{i,1}, \dots p_{i,K}]^{\top}$ as the output of the softmax function, the Hessian for (13) is given by:

$$\nabla_{\boldsymbol{\Theta}}^2 L_t = \frac{1}{n_t} \sum_{i=1}^{n_t} \left( \mathrm{Diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i \boldsymbol{p}_i^{\top} \right) \otimes \left( \boldsymbol{x}_i \boldsymbol{x}_i^{\top} \right). \tag{71}$$

We first state the following lemma due to Böhning (Böhning, 1992) that will be useful to derive the Hessian upper bound for multi-class logistic regression.

**Lemma E.1** (Böhning (1992))**.** *For a probability vector $\boldsymbol{p} \in \Delta^{K-1}$, the matrix $\boldsymbol{V}(\boldsymbol{p}) := \mathrm{Diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^{\top}$ is upper bounded by:*

$$\overline{\boldsymbol{V}} := \frac{1}{2} \left[ \boldsymbol{I}_K - \frac{1}{K} \mathbf{1}\mathbf{1}^{\top} \right], \tag{72}$$

*where $\mathbf{1}$ is the all-ones vector of size $K$. In other words $\overline{\boldsymbol{V}} - \boldsymbol{V}(\boldsymbol{p})$ is positive semi-definite.*

Substituting the bound $\boldsymbol{V}(\boldsymbol{p}) \preceq \overline{\boldsymbol{V}}$ into (71), we obtain the Hessian upper bound as:

$$\boldsymbol{H}_t := \frac{1}{n_t} \overline{\boldsymbol{V}} \otimes \boldsymbol{X}_t^{\top} \boldsymbol{X}_t \tag{73}$$

**Computing the Loss Upper Bound.** From (73), we can compute the iterative PMF-CL update for multi-class classification as follows. First, the upper bound for the $t^{\text{th}}$ task is given by:

$$L_t^{\mathrm{ub}}(\boldsymbol{\Theta}) = L_t(\boldsymbol{\Theta}_t^{\min}) + \frac{1}{2n_t} \mathrm{vec}(\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min})^{\top} \left( \overline{\boldsymbol{V}} \otimes \boldsymbol{X}_t^{\top} \boldsymbol{X}_t \right) \mathrm{vec}(\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min}) \tag{74}$$

$$= L_t(\boldsymbol{\Theta}_t^{\min}) + \frac{1}{2n_t} \mathrm{Tr}((\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min})^{\top} (\boldsymbol{X}_t^{\top} \boldsymbol{X}_t)(\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min}) \overline{\boldsymbol{V}}) \tag{75}$$

$$= L_t(\boldsymbol{\Theta}_t^{\min}) + \frac{1}{2n_t} \mathrm{Tr}((\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min})^{\top} (\boldsymbol{X}_t^{\top} \boldsymbol{X}_t)(\boldsymbol{\Theta} - \boldsymbol{\Theta}_t^{\min}) \overline{\boldsymbol{V}}) \tag{76}$$

where $\mathrm{vec}(\cdot)$ is the vectorization operator. We use the property that $\mathrm{vec}(\boldsymbol{A})^\top(\boldsymbol{B}\otimes\boldsymbol{C})\mathrm{vec}(\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}^\top\boldsymbol{C}\boldsymbol{A}\boldsymbol{B})$ for any matrices $\boldsymbol{A}\in\mathbb{R}^{d\times K}$, symmetric matrix $\boldsymbol{B}\in\mathbb{R}^{K\times K}$, and gram matrix $\boldsymbol{X}^\top\boldsymbol{X} =: \boldsymbol{C}\in\mathbb{R}^{d\times d}$.

**Iterative PMF-CL Update.** Computing the linear scalarization of the upper bounds, and setting its derivative with respect to $\boldsymbol{\Theta}$ to zero, we get the following linear equation:

$$\left(\sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}(\boldsymbol{X}_t^\top\boldsymbol{X}_t)\boldsymbol{\Theta}\right)\overline{\boldsymbol{V}} = \left(\sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}\boldsymbol{X}_t^\top\boldsymbol{X}_t\boldsymbol{\Theta}_t^{\min}\right)\overline{\boldsymbol{V}} \tag{77}$$

The centering matrix $\overline{\boldsymbol{V}}$ has a rank of $K-1$ making it singular. Thus, it cannot be dropped directly. However, note that the classifier is still defined through the softmax function, which has a gauge-invariance property, i.e.: $\mathrm{softmax}(\boldsymbol{x}^T\boldsymbol{\Theta}\boldsymbol{P}) = \mathrm{softmax}(\boldsymbol{x}^T\boldsymbol{\Theta})$ for any $\boldsymbol{\Theta}\in\mathcal{W}$, and $\boldsymbol{P} = 2\overline{\boldsymbol{V}}$ is the centering matrix. Hence, the centered $\boldsymbol{\Theta}$, i.e., $\boldsymbol{\Theta}_c = \boldsymbol{\Theta}\boldsymbol{P}$, is an equivalent solution to $\boldsymbol{\Theta}$ itself:

$$\left(\sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}\boldsymbol{X}_t^\top\boldsymbol{X}_t\right)\boldsymbol{\Theta}_c = \left(\sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}(\boldsymbol{X}_t^\top\boldsymbol{X}_t)\boldsymbol{\Theta}_t^{\min}\right)\boldsymbol{P} \tag{78}$$

Hereafter, we define $\boldsymbol{\Theta}_{c,t}^{\min} = \boldsymbol{\Theta}_t^{\min}\boldsymbol{P}$ as the centered version of the task-loss minimizer. Then, the iterative solution can be determined by defining the accumulation matrix $\boldsymbol{A}_k$ as:

$$\boldsymbol{A}_k := \sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}\boldsymbol{X}_t^\top\boldsymbol{X}_t = \frac{N_{k-1}}{N_k}\boldsymbol{A}_{k-1} + \frac{\alpha_k^{(k)}}{n_k}\boldsymbol{V}_k^e{\boldsymbol{\Sigma}_k^e}^2{\boldsymbol{V}_k^e}^\top \tag{79}$$

where we expand the economy-SVD form of $\boldsymbol{X}_k$. Then, the iterative PMF-CL update follows from (11) with $\boldsymbol{H}_k$ replaced by $\frac{1}{n_k}\boldsymbol{X}_k^\top\boldsymbol{X}_k$, i.e.:

$$\boldsymbol{A}_k\Delta\boldsymbol{\Theta}_{c,k} = \frac{\alpha_k^{(k)}}{n_k}\boldsymbol{V}_k^e{\boldsymbol{\Sigma}_k^e}^2{\boldsymbol{V}_k^e}^\top(\boldsymbol{\Theta}_{c,k}^{\min} - \boldsymbol{\Theta}_{c,k-1}^{\star}) \tag{80}$$

which is identical to the iterative update of linear regression with $\widetilde{y}_t$ replaced with ${\boldsymbol{V}_t^e}^\top\boldsymbol{\Theta}_{c,k}^{\min}$. This iterative update enables us to compute the PMF-CL solution with a worst-case static memory footprint of $\mathcal{O}\left(d^2\right)$ to store the singular values and the right singular matrix. The full solution would then lie in the set:

$$\boldsymbol{\Theta}_k^{\star}\in\{\boldsymbol{\Theta}_{c,k}^{\star} + \boldsymbol{u}\boldsymbol{1}^\top : \boldsymbol{u}\in\mathbb{R}^d\} \tag{81}$$

**Forgetting bound.** From the loss upper bound obtained in (76), we can get the forgetting bound of task $t$ after training upto task $k$ as (using centered parameters):

$$F_t^{(k)} \le \frac{1}{2n_t}\mathrm{Tr}((\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})^\top(\boldsymbol{X}_t^\top\boldsymbol{X}_t)(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})\overline{\boldsymbol{V}}). \tag{82}$$

Note that $\overline{\boldsymbol{V}}$ is a symmetric p.s.d. matrix, hence we can write $\overline{\boldsymbol{V}} = \overline{\boldsymbol{V}}^{1/2}\overline{\boldsymbol{V}}^{1/2}$. Hence, we get:

$$F_t^{(k)} \le \frac{1}{2n_t}\mathrm{Tr}((\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})^\top(\boldsymbol{X}_t^\top\boldsymbol{X}_t)(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})\overline{\boldsymbol{V}}^{1/2}\overline{\boldsymbol{V}}^{1/2}) \tag{83}$$

$$= \frac{1}{2n_t}\mathrm{Tr}(\overline{\boldsymbol{V}}^{1/2}(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})^\top(\boldsymbol{X}_t^\top\boldsymbol{X}_t)(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})\overline{\boldsymbol{V}}^{1/2}) \tag{84}$$

$$= \frac{1}{2n_t}\left\|\boldsymbol{X}_t(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}_{c,t}^{\min})\overline{\boldsymbol{V}}^{1/2}\right\|_F^2 \tag{85}$$

where the last equality is because $\overline{\boldsymbol{V}}$ is symmetric. The final bound is obtained by replacing $\boldsymbol{\Theta}_c$ with the value computed by PMF-CL, which is $\boldsymbol{\Theta}_c^{\star} = \boldsymbol{A}_k^{\dagger}\boldsymbol{\nu}_k$, where $\boldsymbol{\nu}_k = \sum_{t=1}^{k}\frac{\alpha_t^{(k)}}{n_t}(\boldsymbol{X}_t^\top\boldsymbol{X}_t)\boldsymbol{\Theta}_{c,t}^{\min}$. Then, we obtain the final forgetting bound as:

$$F_t^{(k)} \le \frac{1}{2n_t}\left\|\boldsymbol{X}_t(\boldsymbol{A}_k^{\dagger}\boldsymbol{\nu}_k - \boldsymbol{\Theta}_{c,t}^{\min})\overline{\boldsymbol{V}}^{1/2}\right\|_F^2 \tag{86}$$

**Algorithm 4** Federated PMF-CL for QUB loss functions

**Input:** Number of clients $m$, number of time-steps $T$.
**Initialize:** $\boldsymbol{\theta}_0^\star$, $\alpha_{0,i}^{(0)} = 1$ and $\beta_i^{(0)} = 1$ for all $i \in [m]$.
**Initialize:** For all clients $i \in [m]$, $\boldsymbol{B}_{0,i} = \mathbf{0}$ and $N_{0,i} = 0$.
**for** $k = 1$ to $T$ **do**
  **for** each client $i = 1$ to $m$ **in parallel do**
    **Compute** $\boldsymbol{\theta}_{k,i}^{\min}$ and $\boldsymbol{H}_{k,i}$ based on $\mathcal{D}_{k,i}$.
    **Update** $N_{k,i} = N_{k-1,i} + n_{k,i}$.
    **Update** $\boldsymbol{B}_{k,i} = \frac{N_{k-1,i}}{N_{k,i}} \boldsymbol{B}_{k-1,i} + \alpha_{k,i}^{(k)} \boldsymbol{H}_{t,i}$.
    **Compute** $\boldsymbol{\gamma}_{k,i} = \alpha_{k,i}^{(k)} \boldsymbol{H}_{k,i} \left( \boldsymbol{\theta}_{k,i}^{\min} - \boldsymbol{\theta}_{k-1}^\star \right)$.
    **Send** $N_{k,i}$, $\boldsymbol{B}_{k,i}$ and $\boldsymbol{\gamma}_{k,i}$ to the central server.
  **end for**
  At server, **compute** preferences $\beta_i^{(k)}$ for all $i \in [m]$ based on $\{N_{k,i}\}_{i=1}^m$.
  **Aggregate** $\boldsymbol{B}_k = \sum_{i=1}^m \beta_i^{(k)} \boldsymbol{B}_{k,i}$ and $\boldsymbol{\gamma}_k = \sum_{i=1}^m \beta_i^{(k)} \boldsymbol{\gamma}_{k,i}$.
  Solve for $\Delta\boldsymbol{\theta}_k$ in $\boldsymbol{B}_k \Delta\boldsymbol{\theta}_k = \boldsymbol{\gamma}_k$.
  **Update** $\boldsymbol{\theta}_k^\star = \boldsymbol{\theta}_{k-1}^\star + \Delta\boldsymbol{\theta}_k$.
  **Send** $\boldsymbol{\theta}_k^\star$ to all clients.
**end for**
**Output:** $\{\boldsymbol{\theta}_k^\star\}_{k=1}^T$.

# F Extension to a Federated CL setting

Our results in Sections 3-5 can be extended to a Federated Continual Learning (FCL) setting as well. While various works have studied FCL in different settings (Yoon et al., 2021; Shenaj et al., 2023; Wang et al., 2024b; Zhang et al., 2023; Usmanova et al., 2021; Dong et al., 2022; Keshri et al., 2024; Han et al., 2023), note that these works are predominantly empirically motivated, and it is challenging to guarantee their convergence in a Pareto-optimal sense. A more comprehensive list of related works on FCL can be found in the surveys (Yang et al., 2024; Wang et al., 2024c; Criado et al., 2022). In the following, we first explain our FCL setting and then derive a federated iterative variant of the PMF-CL algorithm.

## F.1 Federated Continual Learning: Setting

In our federated learning setting, we assume that there are $m$ participating clients with evolving datasets. The dataset $\mathcal{D}_{t,i}$ arrives at time-step $t \in [T]$, at client $i$; and for simplicity, this process is synchronous across clients. Orchestrated by a central server, the $m$ clients aim to collaboratively learn a common model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ parametrized by $\boldsymbol{\theta}$ without sharing their private datasets. As in Section 3 the goal is to compute a Pareto-optimal solution to $\boldsymbol{\theta}_k^\star$ at each time-step $k \in [T]$ by saving task-loss-aware MSI for each task.

**Preference Vectors.** Previously, in (3) we defined the preference vector for each task. We appropriately modify it to also include a client index $i$ as follows. Denoting the number of samples in the dataset $\mathcal{D}_{i,t}$ as $n_{t,i}$, and the net total samples seen at client $i$ upto time-step $k$ as $N_{k,i} := \sum_{t=1}^k n_{t,i}$:

$$\alpha_{t,i}^{(k)} := \frac{n_{t,i}}{\sum_{t'=1}^k n_{t',i}}. \tag{87}$$

For the federated setting, we also introduce a preference vector for each client, based on the relative number of samples seen by each client upto time-step $k$ as:

$$\beta_i^{(k)} = \frac{N_{k,i}}{\sum_{i'=1}^m N_{k,i'}}. \tag{88}$$

## F.2 Federated PMF-CL

As in Section 5, we will consider problems with a quadratic upper bound (QUB) structure. The Federated PMF-CL algorithm for linear or basis function regression would follow a very similar

derivation, and is omitted here for brevity. At a time-step $k$ during the continual learning process, the Pareto-optimal solution to all the tasks seen by all clients so far[4], would be given by:

$$\left(\sum_{t=1}^{k}\sum_{i=1}^{m}\frac{n_{t,i}}{\sum_{t'=1}^{k}\sum_{i'=1}^{m}n_{t',i'}}\boldsymbol{H}_{t,i}\right)\boldsymbol{\theta}_k^\star=\sum_{t=1}^{k}\sum_{i=1}^{m}\frac{n_{t,i}}{\sum_{t'=1}^{k}\sum_{i'=1}^{m}n_{t',i'}}\boldsymbol{H}_{t,i}\boldsymbol{\theta}_{t,i}^{\min} \tag{89}$$

where $\boldsymbol{H}_{t,i}$ is the Hessian upper bound of the loss function $L_{t,i}(\boldsymbol{\theta})$, and $\boldsymbol{\theta}_{t,i}^{\min}$ is the optimal solution corresponding to $\mathcal{D}_{t,i}$. In terms of the preferences denoted in (87) and (88), the Pareto-optimal solution can be written as:

$$\left(\sum_{i=1}^{m}\beta_i^{(k)}\sum_{t=1}^{k}\alpha_{t,i}^{(k)}\boldsymbol{H}_{t,i}\right)\boldsymbol{\theta}_k^\star=\sum_{i=1}^{m}\beta_i^{(k)}\sum_{t=1}^{k}\alpha_{t,i}^{(k)}\boldsymbol{H}_{t,i}\boldsymbol{\theta}_{t,i}^{\min} \tag{90}$$

Next, we proceed to derive an iterative algorithm to compute $\boldsymbol{\theta}_k^\star$ at each time-step $k$ based on the previous optimal model $\boldsymbol{\theta}_k^\star$. We can rewrite the above equation as:

$$\left[\sum_{i=1}^{m}\beta_i^{(k)}\sum_{t=1}^{k-1}\alpha_{t,i}^{(k)}\boldsymbol{H}_{t,i}+\sum_{i=1}^{m}\beta_i^{(k)}\alpha_{k,i}^{(k)}\boldsymbol{H}_{k,i}\right]\boldsymbol{\theta}_k^\star=\sum_{i=1}^{m}\beta_i^{(k)}\sum_{t=1}^{k-1}\alpha_{t,i}^{(k)}\boldsymbol{H}_{t,i}\boldsymbol{\theta}_{t,i}^{\min}+\sum_{i=1}^{m}\beta_i^{(k)}\alpha_{k,i}^{(k)}\boldsymbol{H}_{k,i}\boldsymbol{\theta}_{k,i}^{\min} \tag{91}$$

Next, we introduce the model update $\Delta\boldsymbol{\theta}_k=\boldsymbol{\theta}_k^\star-\boldsymbol{\theta}_{k-1}^\star$, and use the recursive rules for $\alpha_{t,i}^{(k)}$ and $\beta_i^{(k)}$:

$$\alpha_{t,i}^{(k)}=\alpha_{t,i}^{(k-1)}\frac{N_{k-1,i}}{N_{k,i}} \tag{92}$$

$$\beta_i^{(k)}=\beta_i^{(k-1)}\frac{N_{k,i}}{N_{k-1,i}}\cdot\frac{\sum_{j=1}^{m}N_{k-1,j}}{\sum_{j=1}^{m}N_{k,j}}. \tag{93}$$

Also, recognizing that (90) holds for $k-1$ as well, we can simplify the above as:

$$\left[\sum_{i=1}^{m}\beta_i^{(k)}\sum_{t=1}^{k}\alpha_{t,i}^{(k)}\boldsymbol{H}_{t,i}\right]\Delta\boldsymbol{\theta}_k=\sum_{i=1}^{m}\beta_i^{(k)}\alpha_{k,i}^{(k)}\boldsymbol{H}_{k,i}\left(\boldsymbol{\theta}_{k,i}^{\min}-\boldsymbol{\theta}_{k-1}^\star\right). \tag{94}$$

The iterative update in (94) is straightforward to implement in a federated learning setup. In Algorithm 4, we present an implementation of a federated PMF-CL algorithm that solves (94) in a distributed fashion.

**A note on memory requirements.** The federated PMF-CL Algorithm 4 would require the clients to upload the triple $(N_{k,i}, \boldsymbol{B}_{k,i}, \boldsymbol{\gamma}_{k,i})$ at each iteration, which would be a total of $\mathcal{O}\left(d^2\right)$ floating-point values. This value is much larger compared to conventional FedAvg (McMahan et al., 2016), but is perhaps the price paid to arrive at Pareto-optimal solutions. As in Section 5, one can replace the Hessian matrices in $\boldsymbol{B}_{k,i}$ with their diagonal approximations. This would bring down the memory requirement to $\mathcal{O}\left(d\right)$, at the cost of approximation. The download from the server would require $\mathcal{O}\left(d\right)$ memory, which is identical to conventional FedAvg. A systematic study to optimize the communication and memory requirements of Algorithm 4 is an interesting direction for future work.

## G Experimental Simulation Results

To empirically validate the efficacy of our proposed PMF-CL framework, we evaluate its performance against several established continual learning baselines. Our experiments aim to answer the following questions: (1) How well does PMF-CL mitigate catastrophic forgetting compared to rehearsal and regularization-based methods? (2) Can PMF-CL successfully converge to the global Pareto-optimal parameter set across multiple tasks? (3) What are the computational trade-offs regarding memory footprint and execution time?

[4]This is simply an extension of (10) to the case of datasets distributed across clients (the space dimension), along with the time dimension.

### G.1 Experimental Setup and Implementation

**Task Design:** We evaluate the proposed methods on a synthetic continual learning benchmark consisting of 10 sequential tasks. For each task $k$, the input features $X_k$ are sampled from a Gaussian distribution, while regression targets $y_k$ are generated using ground truth parameters $\theta_k^*$ that undergo a controlled stochastic shift between tasks. Specifically, for task $k = 1$, $\theta_1^*$ is initialized randomly, and for subsequent tasks $k > 1$, the parameters are updated via $\theta_k^* = \theta_{k-1}^* + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ simulates drifting local minimizers.

**PMF-CL Implementation:** Our framework utilizes a Pareto-minimal forgetting approach implemented within the *Mammoth* continual learning framework (AImageLab, 2020). The model employs Singular Value Decomposition (SVD) on task-specific data to facilitate efficient parameter updates by storing only the decomposed components $(V_t, \Sigma_t)$ rather than full covariance matrices. These components are used to incrementally update an aggregate curvature matrix $A_t = (N_{t-1}/N_t)A_{t-1} + \alpha^{(t)} V_t \Sigma_t^2 V_t^T$, where $\alpha^{(t)}$ serves as a preference weight to modulate task importance. To ensure numerical stability, we apply Tikhonov regularization by adding $\lambda I$ (with $\lambda = 10^{-6}$) to the curvature matrix before solving for the Pareto-optimal parameter shift $\Delta\theta$. The final update $\theta_t = \theta_{t-1} + \Delta\theta$ allows the model to shift toward the global optimum of all observed tasks without requiring access to previous raw datasets.

**Evaluation Protocol:** All reported metrics and plotted learning curves represent the average performance across 10 independent trials using randomly generated synthetic datasets. Shaded regions in the plots denote $\pm 1$ standard deviation from the mean.

**Baselines:** We evaluate the performance of PMF-CL against three standard CL approaches, and a naive SGD baseline which completely forgets previous tasks:

- **SGD (Naive Fine-Tuning):** Sequential training on new tasks using Stochastic Gradient Descent without any memory or regularization. This serves as an empirical lower bound, illustrating the maximum extent of catastrophic forgetting.
- **Experience Replay (ER)** (Chaudhry et al., 2019b)**:** A rehearsal strategy that maintains a fixed-size memory buffer of past episodic data, randomly interleaving it with the current task's data to anchor historical knowledge.
- **Dark Experience Replay (DER)** (Buzzega et al., 2020)**:** An advanced rehearsal baseline that stores past data alongside their pre-softmax logits, enforcing consistency by penalizing deviations from historical outputs.
- **Synaptic Intelligence (SI)** (Zenke et al., 2017)**:** A regularization-based approach that computes the importance of individual parameters online, applying a surrogate loss penalty to restrict updates on weights deemed critical for preceding tasks.

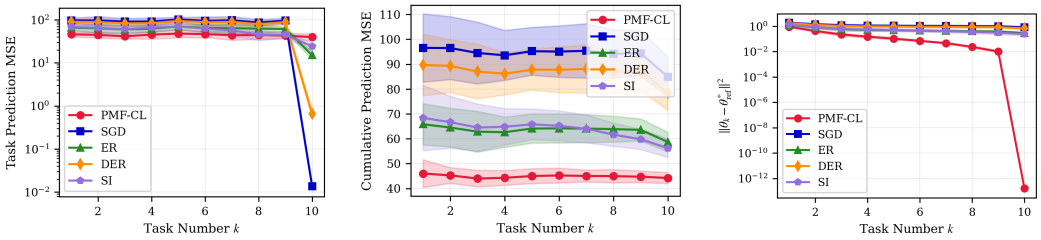
**Compute resources.** All experiments were conducted using a single workstation equipped with an NVIDIA H100 GPU. Numerical operations were implemented using the PyTorch framework (Paszke et al., 2019) in Python.

### G.2 Prediction Accuracy and Parameter Convergence

We first assess the predictive performance and theoretical convergence properties of the models. Figure 3 illustrates the Mean Squared Error (MSE) across tasks and the parameter distance to the global optimum.

**Current Task Accuracy:** Figure 3a reports the independent prediction error (MSE) evaluated sequentially after the conclusion of each task $k$. PMF-CL consistently achieves the lowest MSE. Crucially, the PMF-CL policy demonstrates no recency bias; it maintains uniform performance across the task sequence rather than skewing optimization strictly toward the most recently observed distribution.

**Cumulative Forward Transfer:** Figure 3b measures the accumulated data prediction accuracy, calculated as the MSE over all datasets up to and including the current task $k$. PMF-CL establishes a clear performance advantage by Task 3 and remains the dominant method through Task 10. This trajectory highlights PMF-CL's strong resistance to catastrophic forgetting and its capacity for effective forward transfer.



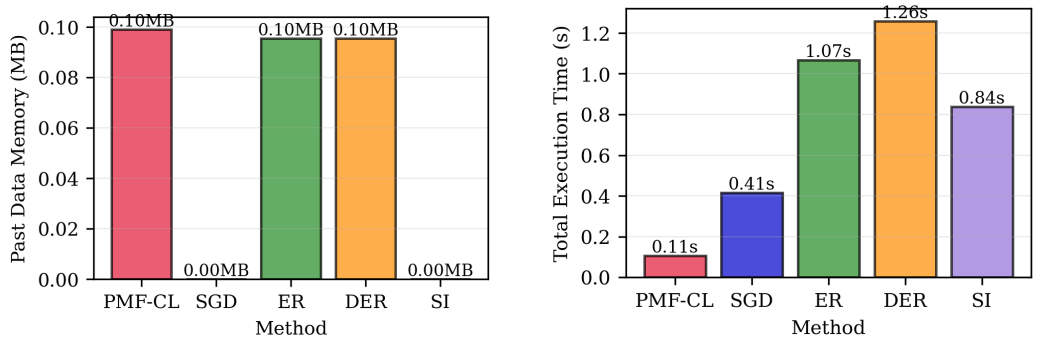

(a) Current Task Accuracy (b) Cumulative Accuracy (c) Parameter Learning Accuracy

Figure 3: Performance metrics across 10 sequential tasks. (Red = PMF-CL, Blue = SGD, Green = ER, Orange = DER, Purple = SI).

**Convergence to the Pareto Optimum:** Figure 3c evaluates the MSE between the parameters learned after task $k$ and the global Pareto-optimal parameters (the theoretical optimum derived by jointly training on all data simultaneously). Because PMF-CL explicitly optimizes for this combined dataset optimum, it successfully converges to the true Pareto-optimal solution upon the completion of all 10 tasks. In contrast, all baseline methods fail to locate this global reference point, plateauing at suboptimal parameter configurations.

### G.3 Computational Complexity Analysis

Continual learning algorithms must carefully balance predictive performance with resource constraints. We analyze the memory and time complexities of the evaluated methods in Figure 4.




(a) Rehearsal Memory Complexity (b) Time Complexity Comparison

Figure 4: Computational trade-offs showing (a) total memory footprint and (b) execution time per method.

**Memory Footprint:** As shown in Figure 4a, the memory complexity of PMF-CL is highly competitive, performing on par with standard rehearsal techniques (ER, DER) in this benchmark. While the replay buffer size for ER/DER aligns with the data retained by our method in this specific setting, PMF-CL offers a distinct structural advantage: its memory footprint is deterministic and fixed entirely by the problem's dimensionality, whereas the memory requirements for rehearsal methods are heavily setting- and buffer-dependent.

**Execution Time:** Figure 4b demonstrates that PMF-CL executes highly efficiently on this benchmark (averaging 0.11s). However, we note that the theoretical time complexity of PMF-CL is bounded by the exact Singular Value Decomposition (SVD) operations required for parameter updates. Consequently, for significantly higher-dimensional problems, PMF-CL's execution time may scale poorly compared to SGD-based methods. Future iterations of PMF-CL could integrate approximate SVD methods to drastically reduce this overhead. Furthermore, it should be noted that the baseline execu-

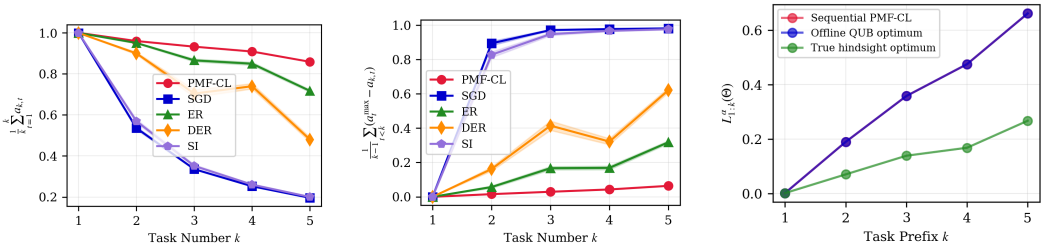
tion times are heavily parameterized by the chosen number of training iterations per task, making relative speed comparisons partially dependent on hyperparameter configurations.

### G.4 Continual Logistic Regression on Sequential MNIST

We next evaluate the QUB extension of PMF-CL for multiclass logistic regression on Sequential MNIST. This experiment complements the exact quadratic study above: it tests both the continual-learning behavior of the method and whether its recursive updates recover the offline optimum of the quadratic surrogate derived in Appendix E.

**Experimental setup:** We use the class-incremental Sequential MNIST benchmark with five tasks and two new classes per task, in the order $(0, 1), (2, 3), \ldots, (8, 9)$. The model is a linear 784-dimensional input layer with ten output logits. Results are averaged over ten runs (seeds 42–51), and shaded regions show $\pm 1$ standard deviation. We compare PMF-CL with SGD, ER, DER, and SI, using one training epoch per task for these standard baselines. ER and DER use buffers of 500 examples. PMF-CL uses the Böhning quadratic upper bound and an $\ell_2$ coefficient of $10^{-4}$; its task-local minimizers and the offline reference solutions are fitted to numerical convergence. Here, the baseline comparison is meant to be a controlled validation under a fixed one-epoch budget, and we do not claim to have exhaustively fine-tuned all the methods.

#### G.4.1 Accuracy, Forgetting, and Surrogate Validation




(a) Cumulative accuracy (b) Average forgetting (c) Scalarized true logistic loss

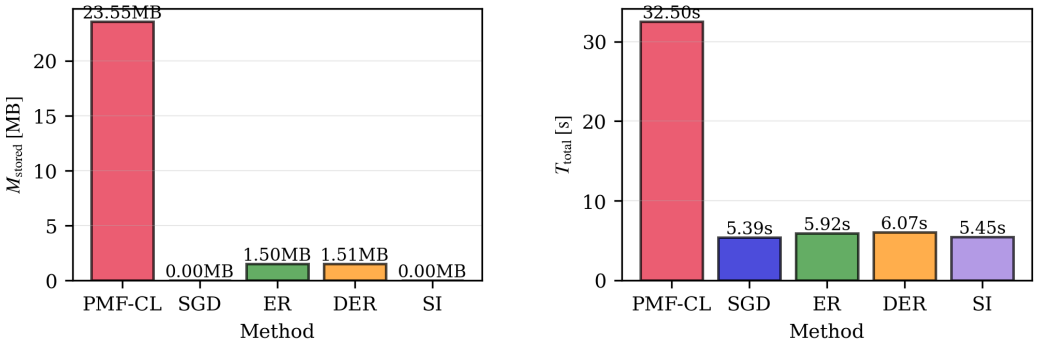
Figure 5: Sequential MNIST results over five two-class tasks. Panels (a) and (b) compare PMF-CL with the continual-learning baselines. Panel (c) evaluates the true sample-weighted logistic objective at the sequential PMF-CL iterate, the offline QUB optimum, and the true hindsight optimum for each task prefix.

**Cumulative accuracy and forgetting:** Figure 5a reports the mean accuracy over all tasks observed through prefix $k$. PMF-CL retains $85.85\%$ cumulative accuracy after the fifth task, compared with $71.69\%$ for ER, $48.00\%$ for DER, $20.04\%$ for SI, and $19.61\%$ for naive SGD. The corresponding forgetting curves in Figure 5b show the same stability advantage: final average forgetting is $0.0639$ for PMF-CL, versus $0.3180$ for ER, $0.6209$ for DER, $0.9754$ for SI, and $0.9806$ for SGD. This gain in stability comes with reduced plasticity on the newest task: PMF-CL's final current-task accuracy is $78.52\%$, while the four baselines obtain between $95.34\%$ and $96.37\%$.

**Recovery of the offline surrogate solution:** In Figure 5c, the sequential PMF-CL and offline-QUB curves visually coincide at every prefix. Direct comparison of their parameters confirms that the recursion error remains below the prescribed $10^{-8}$ tolerance, with final mean-squared parameter error $2.47 \times 10^{-11} \pm 4.50 \times 10^{-11}$. Hence, the continual sufficient- information update faithfully recovers the solution that would be obtained by optimizing the accumulated quadratic surrogate offline.

The green curve separates this implementation check from the approximation introduced by the QUB. At prefix five, the sequential/offline-QUB solution has true sample-weighted logistic loss $0.6612$, whereas the true hindsight logistic optimum attains $0.2675$; their parameter MSE is $0.160$. All measured QUB-minus-true-loss gaps were nonnegative. Therefore, the remaining discrepancy is attributable to looseness of the quadratic surrogate, rather than error in the recursive PMF-CL update.

### G.4.2 Computational Trade-offs




(a) Stored memory

(b) Total runtime

Figure 6: Resource usage for the Sequential MNIST experiment. PMF-CL stores fixed-dimensional sufficient information, while ER and DER store their configured replay buffers; SGD and SI have no replay memory.

Figure 6 quantifies the cost of the exact surrogate updates in this implementation. PMF-CL stores $23.55$ MB, compared with $1.50$ MB for ER and $1.51$ MB for DER, and takes $32.50$ s in total, versus approximately $5.39$–$6.07$ s for the baselines. The larger cost is consistent with storing and manipulating the quadratic sufficient statistics for a $784 \times 10$ linear classifier and with fitting the task-local surrogates to convergence. Importantly, this state is determined by model dimension and does not grow with the number of tasks, whereas replay memory is controlled by a user-selected buffer size. The wall-clock comparison also depends on the unequal optimization budgets described above and is included to expose this computational trade-off, not to claim a speed advantage.