Provable Lifelong Learning of Representations

Xinyuan Cao
Georgia Tech

Weiyang Liu
University of Cambridge & MPI-IS

Santosh S. Vempala
Georgia Tech

Abstract

In lifelong learning, tasks (or classes) to be learned arrive sequentially over time in arbitrary order. During training, knowledge from previous tasks can be captured and transferred to subsequent ones to improve sample efficiency. We consider the setting where all target tasks can be represented in the span of a small number of unknown linear or nonlinear features of the input data. We propose a lifelong learning algorithm that maintains and refines the internal feature representation. We prove that for any desired accuracy on all tasks, the dimension of the representation remains close to that of the underlying representation. The resulting sample complexity improves significantly on existing bounds. In the setting of linear features, our algorithm is provably efficient and the sample complexity for input dimension d, m tasks with k features up to error ϵ is $\tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$. We also prove a matching lower bound for any lifelong learning algorithm that uses a single task learner as a black box. We complement our analysis with an empirical study, including a heuristic lifelong learning algorithm for deep neural networks. Our method performs favorably on challenging realistic image datasets compared to state-of-the-art continual learning methods.

1 Introduction

Recent years have witnessed significant advances in both theory and practice of supervised learning. While a variety of techniques are available for learning individual target functions, much less is known about *continual* or *lifelong* learning, where the learner is adding new target functions to their repertoire. Inspired by how humans learn and transfer knowledge during their lifespan, lifelong learning has many applications in computer vision [Parisi et al., 2019] and robotics [Thrun and Mitchell, 1995].

A central idea for lifelong learning is to learn an *efficient representation* that facilitates the collection of target functions to be learned. For example, if deep feed-forward networks are being used for classification, the goal might be to learn a hidden layer whose outputs are relevant and useful features for the family of tasks. Building a classifier on top of them is relatively easy or less expensive than building one from the original input features. This representation itself is incrementally refined as more target functions are learned.

We consider a very general setting of task/class incremental learning, where new samples from different tasks/classes are presented sequentially over time. The goal of the learner is to maintain hypothesis functions that work for all tasks/classes encountered so far. We assume that all targets are simple functions of a bounded number of unknown *linear or nonlinear* features.

Prior work [Balcan et al., 2015] considered the task-incremental setting where the target functions are linear classifiers of the input that all lie in a common low-dimensional subspace. Under this assumption, a simple algorithm can be shown to learn a good representation of size comparable to the optimal one (*i.e.*, a basis of the common low-dimensional subspace). The algorithm proceeds as follows: maintain a small number of linear features; learn the next function as a linear function of the features; if the error is too high, learn the new function directly on the input, and add it as a new feature. Under mild assumptions on the input distribution (log-concavity), with a suitable choice of error parameters, this algorithm is guaranteed to learn a small set of features that work well for all the target functions. More recent works [Du et al., 2020, Tripuraneni et al., 2020, Chua et al., 2021] focus on the sample complexity of multi-task learning under strong distributional assumptions on both the data and the tasks.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

Our paper is motivated by the following questions:

- Can the theoretical guarantees for linear features be extended to a representation with nonlinear features?
- Does the refinement of the internal representation have provable benefits?
- What is the best possible sample complexity of lifelong learning?

Our work addresses these questions for both task-incremental learning (classification or regression) and class-incremental learning (where we do not have access to the task ID). Our analysis applies to a broad class of lifelong learning algorithms that dynamically change the network architecture. First, we analyze the setting where the underlying common features are nonlinear, which is considerably more general than those previously considered. We prove that this natural lifelong learning algorithm is guaranteed to learn low-error targets by creating only a small number of nonlinear features. Secondly, we propose a new algorithm, with a refinement step, and show that it improves the sample complexity using a new perspective on feature subspaces. The resulting sample complexity improves significantly on known bounds for the setting of linear features, and perhaps surprisingly, we show that it is the best possible in the setting of linear features, assuming that the lifelong learner has black-box access to a single task learner to any desired level of accuracy. This is done by constructing a hard distribution over tasks. Inspired by the theoretical findings, we propose a lifelong learning heuristic for deep neural networks that performs reasonably well.

Finally, we conduct experiments on class-incremental learning using benchmark data sets and find that our proposed algorithm outperforms state-of-the-art continual learning algorithms.

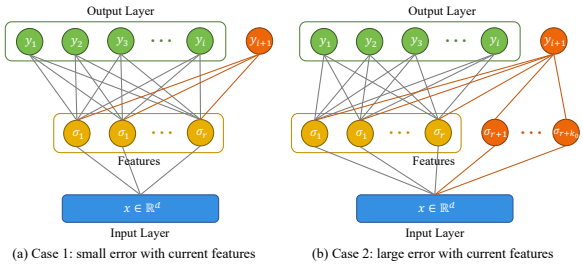
1.1 Problem Settings

We consider m tasks (or m-class classification) where the tasks (classes) arrive sequentially over time. Let $X = \mathbb{R}^d$ be the input space and Y be the label space. We study a discriminative model, where the target function of each task can be learned using a linear combination of at most k linear/nonlinear features. The goal is to learn a hypothesis function with small generalization errors on all tasks.

Formally, the problem is associated with a distribution P over $X \times Y$, D is the marginal of P over X. The label for an input data point $\boldsymbol{x} \in \mathbb{R}^d$ is given by

$$\ell(\boldsymbol{x}) = \phi\left(\langle \boldsymbol{c}^*, \boldsymbol{\sigma}^*(\boldsymbol{x})\rangle\right)$$

where $\boldsymbol{\sigma}^*(\boldsymbol{x}) = (\sigma_1^*(\boldsymbol{x}), \ldots, \sigma_k^*(\boldsymbol{x}))^\top \in \mathbb{R}^k$ is a vector of unknown features, $\boldsymbol{c}^* \in \mathbb{R}^k$, $\phi(\cdot) : \mathbb{R} \to Y$ is the map to the label space. ($k \ll \min(m, d)$). Equivalently, we can view it as a two-layer network with k neurons in the hidden layer (Figure 1.1).



(a) Case 1: small error with current features (b) Case 2: large error with current features

Figure 1.1: An illustration of LLL. Given a new task y_{i+1}, the algorithm tries to learn the class with existing features $\sigma_1, \cdots, \sigma_r$. If the error is small (case1, Figure (a)), then it moves to the next task. Otherwise (case2, figure (b)), it learns a new set of features $\sigma_{r+1}, \cdots, \sigma_{r+k_0}$ and a linear combination of all features; for linear features, it learns a single new feature σ_{r+1}.

Our goal is to learn a good hypothesis function $\hat{\ell}(\cdot)$ parameterized by $(\boldsymbol{c}^*, \boldsymbol{\sigma}^*)$ with a small generalization error $err = \mathbb{P}_{(\boldsymbol{x},y)\sim P} L(l(\boldsymbol{x}), \hat{l}(\boldsymbol{x}))$, where $L(\cdot,\cdot)$ is some loss function for the specific task.

We use a similar model with multi-task learning, where all tasks share the same low-dimensional feature subspace. However, it is different from multi-task learning, which has T_1 source tasks to learn all-at-once and use the features learned to solve the target tasks. The assumption is made there that the features of the target task are covered by all features that have been learned. Instead, lifelong learning algorithms learn all tasks sequentially, with no prior knowledge of the incoming tasks during training.

Here we focus on the task-incremental learning of binary classification tasks. Extensions to task-incremental learning of linear regression and multi-class classification tasks are given in Appendix D.

Let $\boldsymbol{X} = \mathbb{R}^d$ be the input space, $\boldsymbol{Y} = \{\pm 1\}$ be the label space. For any task $i \in [m]$, any sample $(\boldsymbol{x}, y)$ drawn from P satisfies $y = l_i(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{c}_i^*, \boldsymbol{\sigma}^*(\boldsymbol{x})\rangle)$, where the features are $\boldsymbol{\sigma}^*(\boldsymbol{x}) = \boldsymbol{W}^*\boldsymbol{x}$ in the linear case and $\boldsymbol{\sigma}^*(\boldsymbol{x}) = f(\boldsymbol{W}^*\boldsymbol{x})$ in the nonlinear case. Here $f(\cdot)$ is a nonlinear activation function, *e.g.*ReLU. $\boldsymbol{W}^* \in \mathbb{R}^{k\times d}, \boldsymbol{c}_i^* \in \mathbb{R}^k$ etc. Specifically, in the linear case, for each task i, we equivalently have $y = \text{sign}(\langle \boldsymbol{a}_i^*, \boldsymbol{x}\rangle)$, where $\boldsymbol{a}_i^* = \boldsymbol{W}^{*\top}\boldsymbol{c}_i^* \in \mathbb{R}^d$. WLOG we assume that each $\boldsymbol{a}_i^*$ is a unit vector, *i.e.*, $\|\boldsymbol{a}_i^*\|_2 = 1, \forall i \in [m]$. The generalization error is defined as $err = \mathbb{P}_{(\boldsymbol{x},y)\sim P}(l_i(\boldsymbol{x}) \neq \hat{l}_i(\boldsymbol{x}))$.

1.2 Main Results

In all the results and analysis, we only have Assumption 1, which as Lemma 1 from [Balcan et al., 2015]) asserts, is satisfied by all log-concave distributions after

an affine transformation. This class includes many common distributions, such as Gaussian, Uniform and Gamma distributions [Lovász and Vempala, 2007].

Assumption 1. (Data Distribution Assumption) Let $\theta(\cdot,\cdot)$ denote the angle between two vectors. We assume that there exist universal constants $c_2 > c_1 > 0$ s.t., for any unit vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$,

$$c_1\theta(\boldsymbol{u},\boldsymbol{v}) \le \mathbb{P}_{\boldsymbol{x}\sim D}(\mathrm{sign}(\boldsymbol{u}\cdot\boldsymbol{x}) \ne \mathrm{sign}(\boldsymbol{v}\cdot\boldsymbol{x})) \le c_2\theta(\boldsymbol{u},\boldsymbol{v})$$

Our theoretical upper bounds are summarized below. The detailed statements for the results appear as Theorem 3, Theorem 4 and Theorem 5 in Section 3. These results are based on the algorithms described in Section 2, called Basic Lifelong Learning (LLL) and Lifelong Learning with Representation Refinement (LLL-RR).

Theorem 1 (Summary of Upper Bounds)**.** *Consider the lifelong learning setting of input dimension d, m tasks with k common features. The basic lifelong learning algorithm achieves a target error of ϵ on all tasks with sample complexity $\tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$ for linear features and a factor of k higher for nonlinear features. With representation refinement using at most $2k$ features, the sample complexity is $\tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$. In the linear setting refinement runs in polynomial-time.*

This raises the question of whether there exist algorithms with better sample complexity. We show that the answer is NO in a general sense. We assume that we have black-box access to a single-task learner that works as follows: it takes as input labeled examples and a target accuracy ϵ, and outputs some feasible solution with error at most ϵ. Then we show that any lifelong learning algorithm that achieves ϵ error for all tasks needs $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$ samples.

Theorem 2 (Lower Bound)**.** *Suppose that a lifelong learner has black-box access to a single task learner that takes an error parameter ϵ as input and is allowed to return any vector that is within distance ϵ of the true target unit vector, using $\Theta(d/\epsilon)$ samples in $\mathbb{R}^d$. Then, there exists a distribution of m tasks, $m = 2^{\Theta(k)}$ such that for any lifelong learning algorithm, WHP, the total number of samples required to learn all m tasks up to error ϵ is $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$.*

Our contributions can be summarized as follows:

Sample complexity. We bound the sample complexity of lifelong learning for both the linear and nonlinear cases. In the linear case, our bound for the lifelong learning is $\tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$. This improves the dependence on both k and ϵ compared to past work [Balcan et al., 2015], which proved a bound of $\tilde{O}(dk^2/\epsilon^2 + km/\epsilon)$. It also improves existing theoretical results for multi-task learning [Du et al., 2020, Tripuraneni et al., 2020], where the best sample complexity is $\tilde{O}(dk^2/\epsilon + km/\epsilon)$. Moreover, this bound is the best possible up to logarithmic factor for any lifelong learning algorithm.

Representation refinement. We propose and analyze the step of sample-free representation refinement in the lifelong learning setting. Specifically, this step aims to reduce the dimension of the feature subspace while keeping the subspace close to the true one. In the linear setting, we provide an algorithmically efficient approach via an SDP relaxation. To the best of our knowledge, we provide the first provable bound for representation refinement.

Proof techniques. Our analysis is based on geometric insights. The test error translates to the distance between the target and learned vectors. To show that our learned feature subspace is close to the true one, we consider the set of candidate k-dimensional subspaces. We would like to show that the measure of this set decreases rapidly during learning. Instead, we identify the set of well-approximated vectors by our current learned subspace and show that the set grows at a geometric rate until it includes all vectors in the true subspace.

Empirical results. We evaluate our lifelong learning algorithms on standard benchmarks and compare them with state-of-the-art methods, demonstrating their practice efficiency. We also perform simulations for the setting of linear features, and exhibit results that match our theoretical bounds.

1.3 Related work

Lifelong learning [Thrun and Mitchell, 1995] aims to solve different tasks arriving in a stream, where knowledge from current and previous tasks is re-used in subsequent tasks to improve efficiency and sample complexity. Early works found that lifelong learning can encounter Catastrophic Forgetting (CF) [McCloskey and Cohen, 1989], especially when using back-propagation [Ratcliff, 1990]. That is, the performance on old tasks can drop dramatically after learning a new task. There are three main approaches to addressing this problem: *adding a regularization term* [Li and Hoiem, 2017, Kirkpatrick et al., 2017], *freezing the network from previous tasks and adding branches to new tasks* [Xu and Zhu, 2018, Rusu et al., 2016, Liu et al., 2019, Liu et al., 2021, Yoon et al., 2017]

and *replaying previous tasks' exemplars* [Rebuffi et al., 2017]. Our work is closest to the second approach in that we dynamically change the architecture to overcome CF. Although we do not know the number of tasks in advance, we prove that our algorithm has a small model size and efficient sample complexity.

Despite a vast literature on lifelong learning methods, theoretical investigations are relatively few. [Yin et al., 2020] studies the optimization and generalization properties of the regularization-based method by analyzing the loss landscape. [Bennani et al., 2020, Doan et al., 2021] analyze the generalization of the OGD algorithm [Farajtabar et al., 2020] through NTK [Jacot et al., 2018]. [Balcan et al., 2015] gives an upper bound on the architecture size when we grow the network when training binary classifiers. We improve their bounds getting nearly tight sample complexity in the linear case, and generalize the approach to the nonlinear regime.

Two topics closely related to lifelong learning are meta-learning and transfer learning. There is a line of work where all tasks approximately [Finn et al., 2017, Khodak et al., 2019, Balcan et al., 2019, Denevi et al., 2019] or conditionally [Wang et al., 2020, Denevi et al., 2020, Denevi et al., 2021] share a common representation. However, our work focuses on the setting where all tasks share one common low-dimensional representation. [Chua et al., 2021] shows the benefits of task-specific fine-tuning, which is fundamentally different from our refinement step. Our refinement step aims to reduce the representation dimension with slight information loss and help to improve the sample complexity of subsequent tasks. This procedure needs no additional data.

There are other works with similar settings to ours where all tasks share one common representation. [Baxter, 1997] bounds the sample complexity to achieve low average error from a Bayesian/information-theoretic point of view. We compare our results with recent work [Du et al., 2020, Tripuraneni et al., 2020, Balcan et al., 2015] in Table 1 for the linear case. Previous works on multi-task learning [Du et al., 2020, Tripuraneni et al., 2020] need $\tilde{O}(dk^2/\epsilon)$ or more samples from previous tasks to learn the hidden features, while our algorithm needs $\tilde{O}(dk^{1.5}/\epsilon)$ samples. After that, each new task can be learned up to ϵ error with $\tilde{O}(k/\epsilon)$ samples. These results illustrate the efficiency of lifelong learning compared to all-at-once training. Our analysis can also generalize to the setting of nonlinear features, e.g., if labels are generated by a two-layer neural network, with k hidden units. We prove that our lifelong learning algorithm is sample efficient with only Assumption 1, which is relatively weak and clean compared to existing work.

Method	Input Assumptions	Feature Dimension	Total Samples
[Du et al., 2020]†	Sub-gaussian	k	$\tilde{O}(\frac{dk^2}{\epsilon}+\frac{km}{\epsilon})$
[Tripuraneni et al., 2020]†	Sub-gaussian	k	$\tilde{O}(\frac{dk^2}{\epsilon}+\frac{km}{\epsilon})$
[Balcan et al., 2015]⋆	Log-concave	k	$\tilde{O}(\frac{dk^2}{\epsilon^2}+\frac{km}{\epsilon})$
LLL (ours)⋆	Well-spread (Assumption 1)	$2k\log(\frac{\log(k)}{\epsilon})$	$\tilde{O}(\frac{dk^{1.5}}{\epsilon}+\frac{km}{\epsilon})$
LLL-RR (ours)⋆	Well-spread (Assumption 1)	$2k$	$\tilde{O}(\frac{dk^{1.5}}{\epsilon}+\frac{km}{\epsilon})$

Table 1: Comparison of different transfer learning algorithms in the linear setting. †: the method trains all source tasks all at once, and then uses the representation to train the target tasks. ⋆: all source tasks and target tasks are learned sequentially.

Notation. We use bold upper-case letters to refer to matrices (*e.g.*$\boldsymbol{X}$) and bold lower-case letters to refer to vectors (*e.g.*$\boldsymbol{x}$). We use $[m] = \{1, 2, \cdots, m\}$. We use $\tilde{O}$ to hide polylogarithmic factors. O, Ω, Θ are standard notations for order of growth. For any two vectors $\boldsymbol{x}, \boldsymbol{y}$, let $\theta(\boldsymbol{x}, \boldsymbol{y})$ be the angle between them. The angle between a vector $\boldsymbol{x}$ and a subspace $\boldsymbol{U}$ is defined as $\theta(\boldsymbol{x}, \boldsymbol{U}) = \min_{\boldsymbol{u}\in\boldsymbol{U}} \theta(\boldsymbol{x}, \boldsymbol{u})$. For two subspaces $\boldsymbol{U}, \boldsymbol{V}$, define $\theta(\boldsymbol{U}, \boldsymbol{V}) = \max_{\boldsymbol{u}\in\boldsymbol{U}} \theta(\boldsymbol{u}, \boldsymbol{V})$. Thus $\theta(\boldsymbol{U}, \boldsymbol{V}) \leq \alpha$ iff for all $\boldsymbol{u} \in \boldsymbol{U}, \exists \boldsymbol{v} \in \boldsymbol{V}$ s.t. $\theta(\boldsymbol{u}, \boldsymbol{v}) \leq \alpha$. We define the distance from a vector $\boldsymbol{u}$ to a subspace $\boldsymbol{F}$ as the orthogonal distance: $d(\boldsymbol{u}, \boldsymbol{F}) = \min_{\boldsymbol{v}\in\boldsymbol{F}} \|\boldsymbol{u} - \boldsymbol{v}\|_2$. For a distribution D and two vectors $\boldsymbol{u}, \boldsymbol{v}$, we define $d_D(\boldsymbol{u}, \boldsymbol{v}) = \mathbb{P}_{\boldsymbol{x}\sim D}(\text{sign}(\boldsymbol{u}\cdot\boldsymbol{x}) \neq \text{sign}(\boldsymbol{v}\cdot\boldsymbol{x}))$.

2 Algorithms

We study three algorithms: the *basic lifelong learning* algorithm (basic LLL) in Section 2.1, *lifelong learning with representation refinement* algorithm (LLL-RR) in Section 2.2, and *heuristic lifelong learning* algorithm (H-LLL) in Section 2.3. We prove guarantees for basic LLL and LLL-RR in Section 3. We show that H-LLL for deep neural networks (Section 2.3) outperforms state-of-the-art continual learning algorithms in Section 4.2.

2.1 Basic Lifelong Learning

Our algorithm maintains a set of features $\sigma_1(.), \ldots, \sigma_r(.)$ while tasks are presented incrementally. As is shown in Figure 1.1, when the next task, say $(i+1)$-th task arrives, the algorithm first tries to learn a new linear combination y_{i+1} of existing features using examples from the current task. If the best such combination has a low error, it records the linear combination parameters and moves on to the next task. If the error is higher than a threshold ϵ, then it learns a new set of features $\sigma_{r+1}, \cdots, \sigma_{r+k_0}$ and a new linear combination of them with error up to ϵ_{acc}. Denote $\tilde{k}$ as the number of steps that the algorithm learns new features. Let k_0 be the number of features

learned at one time, it is a constant dependent on whether the features are linear or not. We describe the algorithm in Algorithm 1.

Algorithm 1 Basic Lifelong learning Algorithm (Basic LLL)

Input: d, m, k, labeled examples of m tasks, threshold parameters ϵ_{acc}, ϵ.

The algorithm maintains a set of features $\sigma_1(.), \ldots, \sigma_r(.)$ along training. When task $i+1$ arrives,

- Use the data from the $(i+1)$-th task, attempt to learn the linear function $\tilde{\boldsymbol{c}}_{i+1}$ using the current features $\boldsymbol{\sigma}(\cdot) = (\sigma_1(\cdot), \cdots, \sigma_r(\cdot))^\top$.
- Check whether the hypothesis $\boldsymbol{x} \mapsto \text{sign}(\tilde{\boldsymbol{c}}_{i+1}^\top \boldsymbol{\sigma}(\boldsymbol{x}))$ has error less than ϵ.
  1. If yes, record the linear combination parameters $\tilde{\boldsymbol{c}}_{i+1}$.
  2. Otherwise, learn a new set of features $\boldsymbol{\sigma}'(\cdot) = (\sigma_{r+1}(\cdot), \cdots, \sigma_{r+k_0}(\cdot))^\top$ and a linear function $\tilde{\boldsymbol{c}}_{i+1}$ such that the predictor $\boldsymbol{x} \mapsto \text{sign}(\tilde{\boldsymbol{c}}_{i+1}^\top \boldsymbol{\sigma}'(\boldsymbol{x}))$ has error less than ϵ_{acc}. Update the representation $\boldsymbol{\sigma}(\cdot) = (\sigma_1(\cdot), \cdots, \sigma_{r+k_0}(\cdot))^\top$.

return *m predictors:* $\boldsymbol{x} \to \text{sign}(\tilde{\boldsymbol{c}}_i^\top \boldsymbol{\sigma}(\boldsymbol{x}))$, *where* $\boldsymbol{\sigma}(\boldsymbol{x}) = (\sigma_1(\boldsymbol{x}), \cdots, \sigma_{\tilde{k}k_0}(\boldsymbol{x}))^\top, 1 \le i \le m$.

The algorithm works for both linear and nonlinear features. For linear features, if a new target function does not have a good representation as a combination of the features learned so far, the new target is itself a new feature since everything is linear (Algorithm 1, [Balcan et al., 2015]), so k_0 above is 1. For nonlinear features, when the current representation is not good enough, we can learn a set of $k_0 \le k$ nonlinear features with low error since each task corresponds to a target with at most k features. Here we assume that a single such combination can be learned efficiently (*i.e.*, a neural network with a small, single hidden layer) [Bartlett et al., 2019]. Section 3 proves that the number of features to be learned can be upper bounded by $\tilde{O}(kk_0)$. We give the full guarantees for this basic LLL algorithm in Theorem 3.

2.2 Lifelong Learning with Representation Refinement

Similar to the basic LLL algorithm, LLL-RR also expands the feature space gradually. Whenever we learn a new task i, we attempt to learn it using the current representation and check whether a linear combination exists with an error less than ϵ. If yes, we record the classifier for the current task and move to the next one. Otherwise, we learn a new classifier for the current task with error at most ϵ_{acc}, via new features; we then do a step of *representation refinement* on all the features learned so far. The refinement step can also be done when the number of features grows above a threshold rather than every time a new task is learned to high accuracy. The formal description of LLL-RR is given in Appendix C.

Refinement algorithm. Denote $\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_{(\hat{k}+1)k_0}$ as all the features learned so far. The goal of refinement is to find a minimal dimensional feature subspace that is within distance ϵ_{acc} to all learned features. We *minimize the dimension* of feature subspace while keeping it close to the original representation by solving the optimization problem (2.1). This problem is NP-hard, but we provide an efficient approximation algorithm for the linear case (and practical implementation for the general case in Section 2.3).

Algorithm 2 Representation Refinement (RR)

Input: All features learned so far $\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_{(\hat{k}+1)k_0}$, and the desired feature subspace dimension k.

Solve the following optimization problem, and get the solution $\boldsymbol{V}'$.

$$\min_{\boldsymbol{V}} \quad \dim(\boldsymbol{V}) \qquad \text{s.t.} \quad d(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}) \le \epsilon_{acc}, \quad \text{for } 1 \le i \le \left(\hat{k}+1\right) k_0 \tag{2.1}$$

return *Refined representation* $\boldsymbol{V}'$.

The refinement step is provably beneficial to the total sample complexity. Theorem 4 guarantees that lifelong learning algorithm with representation refinement (LLL-RR) can be ended with learning new features in $\tilde{O}(k)$ steps. The analysis is shown in Section 3.

Linear features. We provide an efficient implementation for the linear case by using a Semi-definite Programming (SDP) relaxation (2.2) and then applying Principal Component Analysis (PCA) to round the SDP solution. The relaxation from (2.1) to (2.2) is natural. The positive semi-definite (PSD) matrix $\boldsymbol{X}$ represents the projection matrix to $\boldsymbol{V}^\perp$, the complement of the subspace $\boldsymbol{V}$. It is a relaxation since $\boldsymbol{X}$ might have fractional eigenvalues between 0 and 1. We describe the formal algorithm in Algorithm 3. As is proved in Theorem 5 (Section 3), if the optimal dimension of the feature subspace is k, this linear case implementation will output a $(2k-1)$-dimensional subspace $\boldsymbol{V}'$ with $d(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}') \le \sqrt{2}\epsilon_{acc}, \forall i \in [\hat{k}+1]$. Consequently, LLL-RR terminates with feature dimension $O(k)$.

Algorithm 3 Representation Refinement (RR) Implementation in Linear Case

Input: All features learned so far $\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_{\hat{k}+1}$, and the desired feature subspace dimension k.

1. Solve the following SDP, and get the solution $\boldsymbol{X}^*, t^*$.

$$\begin{aligned} \min_{\boldsymbol{X}, t} \quad & t \\ \text{s.t.} \quad & \tilde{\boldsymbol{w}}_i^\top \boldsymbol{X} \tilde{\boldsymbol{w}}_i \le t, 1 \le i \le \hat{k}+1 \\ & 0 \preceq \boldsymbol{X} \preceq I \\ & \mathrm{Tr}(\boldsymbol{X}) = d - k \end{aligned} \tag{2.2}$$

2. Do the singular value decomposition $\boldsymbol{X}^* = \sum_{i=1}^d \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$, where $0 \le \lambda_1 \le \cdots \le \lambda_d \le 1$.

return $\boldsymbol{V}' = \mathrm{span}(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{2k-1})$ *as the refined representation.*

2.3 A Lifelong Learning Heuristic for Deep Neural Networks

In order to apply our basic LLL algorithm to deep neural networks, we propose a heuristic lifelong learning (H-LLL) algorithm. The intuition of our LLL algorithm is to build an expandable and dynamic representation that can adapt to incoming tasks/classes without sacrificing the quality for previous tasks/classes. Following this intuition, we propose to learn a separate encoder for each task. We observe the training data $\mathcal{D}_i$ for the i-th task and the memory buffer $\mathcal{M}_i$ for the previous tasks. The memory buffer is constructed based on herding selection [Welling, 2009, Rebuffi et al., 2017]. H-LLL works iteratively in two phases. First, H-LLL learns the representation with a separate encoder f_i in the i-th task, while the other encoders $f_j, j < i$ are frozen during the training in the i-th task. Second, H-LLL finetunes the last classifier layer using the memory buffer $\mathcal{M}_i$ and the current task data $\mathcal{D}_i$. These two steps are iterated as the training proceeds. We take the i-th task as an example. Since we train a separate encoder f_i for the i-th task, the representation of a sample $\boldsymbol{x}$ (by the end of the i-th task) is constructed by concatenating all the learned features: $\boldsymbol{v}_i(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_i(\boldsymbol{x})\}$ where $\boldsymbol{v}_i$ denotes the representation after learning the i-th task. The training uses cross-entropy loss on both the memory buffer $\mathcal{M}_i$ and the current dataset $\mathcal{D}_i$:

$$\mathcal{L} = -\frac{1}{|\mathcal{M}_i \cup \mathcal{D}_i|} \sum_{i=1}^{|\mathcal{M}_i \cup \mathcal{D}_i|} \log\left(\mathrm{SoftMax}\left(\boldsymbol{W}_{\mathrm{cls}}^\top \boldsymbol{v}_i(\boldsymbol{x})\right)\right) \tag{2.3}$$

where $\boldsymbol{W}_{\mathrm{cls}}$ is the weight of the last classifier layer. After training of the representation is completed, we follow [Yan et al., 2021] and re-train the classifier layer with a heated-up softmax [Zhang et al., 2018] and a balanced finetuning method [Castro et al., 2018]. Note that, for each encoder $f_j, \forall j$, we can parameterize it with any neural network. In this paper, we use ResNet-18 for all the encoders $f_j, \forall j$.

3 Theoretical Guarantees

Here we state the main theorems for the basic LLL algorithm and LLL-RR algorithm, bounding the representation size and complexity. Here our algorithm and analysis apply for both linear and nonlinear features. For nonlinear features, we consider the kernel induced by them. These features live in a potentially infinite-dimensional space (or exponential in d dimensional space if, *e.g.*, the input is from the Boolean hypercube).

The main theorems are stated as follows. Complete proof for all theoretical guarantees are in Appendix A.

We begin with a bound for the basic LLL algorithm.

Theorem 3 (Basic LLL)**.** *Consider the lifelong learning setting of input dimension d, m tasks with k common features. Let $\epsilon_{acc} = \frac{\epsilon}{c\sqrt{k}}$ for a sufficiently small constant $c > 0$. Under Assumption 1, the basic LLL algorithm, learns new features at most $\tilde{k} = O(k \log(\log(k)/\epsilon))$ times and the dimension of the learned feature space is $O(k \log(\log(k)/\epsilon))$ for linear features and $O(k^2 \log(\log(k)/\epsilon))$ for nonlinear features. The total number of labeled examples to learn all tasks to within error ϵ is $O(\frac{dk^{1.5}}{\epsilon} \log(\frac{\log(k)}{\epsilon}) \log(\frac{k}{\epsilon}) + \frac{km}{\epsilon} \log(\frac{\log(k)}{\epsilon}) \log(\frac{1}{\epsilon})) = \tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$ for linear features and a factor of k higher for nonlinear features.*

Our main result analyzes the lifelong learning algorithm with representation refinement.

Theorem 4 (LLL with Representation Refinement)**.** *Consider the lifelong learning setting of input dimension d, m tasks with k common features. Suppose that the algorithm has access to an oracle that gives a constant-factor approximation of Optimization Problem 2.1. Set $\epsilon_{acc} = \frac{\epsilon}{c\sqrt{k}}$ for a sufficiently small constant $c > 0$. Under Assumption 1, the LLL-RR algorithm learns at most $O(k \log(\log(k)/\epsilon))$ new features, and the dimension of the feature space is $O(k)$. The total number of labeled examples to learn tasks to within error ϵ is $O(\frac{dk^{1.5}}{\epsilon} \log(\frac{\log(k)}{\epsilon}) \log(\frac{k}{\epsilon}) + \frac{km}{\epsilon} \log(\frac{1}{\epsilon})) = \tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$.*

In the linear setting, we provide an efficient implementation of the constant-factor approximation oracle in Algorithm 3 with the following guarantee.

Theorem 5 (Approximation)**.** *In the linear case, for the optimization problem (2.1), if there exists a subspace*

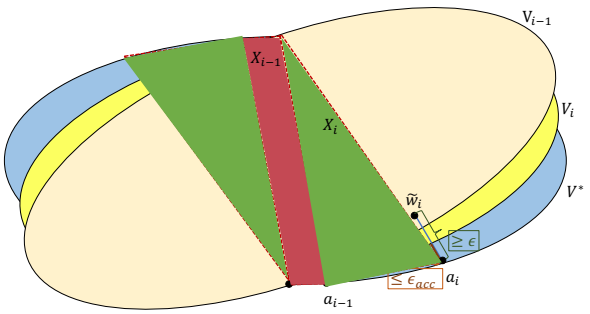


Figure 3.1: Geometric illustration of the proof sketch. For any target $\boldsymbol{a}_i$ that has more than ϵ error based on the previous feature subspace $\boldsymbol{V}_{i-1}$, the algorithm accurately learns a new feature within error ϵ_{acc}, and therefore pushes the new feature subspace $\boldsymbol{V}_i$ towards the the true one $\boldsymbol{V}^*$.

$\boldsymbol{V}^*$ of k dimension with $d(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}^*) \leq \epsilon_{acc}, \forall i \in [\hat{k}+1]$, then for any constant $c > 1$, we can get a $(ck-1)$-dimensional subspace solution with approximation factor $\sqrt{1+\frac{1}{c-1}}$ in maximum distance. Specifically, let $c = 2$, the output of Algorithm 3, $\boldsymbol{V}'$, is a $(2k-1)$-dimensional subspace s.t. $d(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}') \leq \sqrt{2}\epsilon_{acc}, \forall i \in [\hat{k}+1]$.

3.1 Proof idea and plan of Theorem 4

Our proof is based on geometry. Firstly, we show that Assumption 1 guarantees that the distance between hypothesis vectors approximates the test error within a constant factor. As is shown in Figure 3.1, for any target $\boldsymbol{a}_i$ that has a large error based on the previous features, $d(\boldsymbol{a}_i, \boldsymbol{V}_{i-1}) \geq \epsilon$ neglecting constants. Learning $\boldsymbol{a}_i$ accurately will help reduce the angle between the feature subspace of the algorithm $\boldsymbol{V}_i$ and the true feature subspace $\boldsymbol{V}^*$. To quantify the improvement in the angle between the feature subspaces, we construct a convex set whose volume grows at a geometric rate. This leads to the upper bound on the number of new features.

To be more specific, denote $i_1, \cdots, i_{\tilde{k}}$ as the indices of tasks where we learn new features. At step $i_{\hat{k}}$, we construct a set $Y_{i_{\hat{k}}}$ of all possible subspaces that are feasible solutions to the refinement optimization problem (2.1). Let $X_{i_{\hat{k}}}$ be the set of vectors in the unit ball in the true k-dimensional feature subspace that are within distance $O(\epsilon/\sqrt{k})$ to all subspaces in $Y_{i_{\hat{k}}}$. Then we can show that $X_{i_{\hat{k}}}$ is a symmetric convex set. Clearly, the set $Y_{i_{\hat{k}}}$ shrinks during training as we have more and more constraints in the optimization problem. Alongside, the volume of X_i increases exponentially. The learning procedure terminates when $X_{i_{\hat{k}}}$ covers the ball $B_k(0, 1/2\sqrt{k})$, which means that a target function (unit vector) spanned by the true k features will have error $O(\epsilon)$ to the solution learned by LLL-RR. In other words, the feature subspace we learn can solve all future tasks with small errors using only hypothesis vectors from the learned feature subspace.

We will prove this step by step. Lemma 1 bridges the test error to the distance metric. Lemma 2, Lemma 3 and Corollary 1 show that the convex hull of true feature vectors is contained in the set $X_{i_{\hat{k}}}$. Lemma 4 carefully analyzes the maximum volume ellipsoid in $X_{i_{\hat{k}}}$, whose volume grows exponentially. Based on these facts, we can bound the number of new features and prove Theorem 4.

Let $\boldsymbol{X} = \mathbb{R}^d$, for each task i, there exists unit length $\boldsymbol{a}_i$ such that all $(\boldsymbol{x}, y)$ drawn from P satisfies $\text{sign}(\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle) = y$. Let $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, rows of which are $\boldsymbol{a}_i^\top$. Since the parameters $\boldsymbol{a}_i$ lie in some k-dimensional subspace with $k \ll \min(m, n)$, there exists $\boldsymbol{W} \in \mathbb{R}^{k \times d}, \boldsymbol{C} \in \mathbb{R}^{m \times k}$ such that $\boldsymbol{A} = \boldsymbol{C}\boldsymbol{W}$. Rows $\boldsymbol{w}_1^\top, \cdots, \boldsymbol{w}_k^\top$ can be seen as k linear meta-features that are sufficient to learn m tasks. In each step when the current feature subspace cannot achieve low error, we learn new features. Then we take the refinement step to keep a minimal dimensional subspace that is close to all current features $\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_{\hat{k}}$.

Lemma 1. *Given two unit vectors $\boldsymbol{u}, \boldsymbol{v}$ and a distribution D. If D satisfies Assumption 1, then there exist nonzero constants c' and c'' such that $c'\|\boldsymbol{u} - \boldsymbol{v}\|_2 \leq d_D(\boldsymbol{u}, \boldsymbol{v}) \leq c''\|\boldsymbol{u} - \boldsymbol{v}\|_2$.*

Lemma 2. *Let S be a set of subspaces. Let $X = \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x}, \boldsymbol{Y}) \leq r, \forall \boldsymbol{Y} \in S\}$. Then the set X is a symmetric convex set.*

Lemma 3. *For any $\hat{k} \in [\tilde{k}]$, let*

$$Y_{i_{\hat{k}}} := \{\boldsymbol{V} | d(\tilde{\boldsymbol{w}}_{i_j}, \boldsymbol{V}) \leq c_1 \epsilon_{acc}, \forall j \leq \hat{k}\},$$

$$X_{i_{\hat{k}}} := \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x}, \boldsymbol{V}) \leq (c_1 + \frac{1}{c'})\epsilon_{acc}, \forall \boldsymbol{V} \in Y_{i_{\hat{k}}}\},$$

where c_1, c' are small constants. Then for any $j \geq \hat{k}, \pm\boldsymbol{a}_{i_{\hat{k}}} \in X_{i_j}$.

Corollary 1. *For any $\hat{k} \in [\tilde{k}]$,*

$$\text{conv}(\pm\boldsymbol{a}_{i_1}, \cdots, \pm\boldsymbol{a}_{i_{\hat{k}}}) \subseteq X_{i_{\hat{k}}}.$$

Lemma 4 (Max Ellipsoid)**.** *Let $K \subset \mathbb{R}^k$ be a symmetric convex body and $E(K)$ be the maximum volume ellipsoid contained in K. For a vector $\boldsymbol{u}$ on the boundary of $E(K)$, let $K' = \text{conv}(K, 2\sqrt{k}\boldsymbol{u}, -2\sqrt{k}\boldsymbol{u})$. Then,*

$$\frac{\text{vol}(E(K'))}{\text{vol}(E(K))} \geq \frac{13}{10}.$$

Proof idea of Lemma 4. We first consider $E(K) = B_k(0,1)$, a unit ball around the origin and the vector $u = (1, 0, \cdots, 0)^\top \in \text{bd}(E(K))$. By symmetry, we can assume the ellipsoid $E(K'') = \{\boldsymbol{x} | \frac{x_1^2}{a^2} + \sum_{i=2}^k \frac{x_i^2}{b^2} \leq 1\}$ and only consider the two-dimensional slice first, where $\boldsymbol{u} = (1,0)^\top$, $E(K'') = \{(x,y) | \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1\}$. Then

we calculate the volume of the ellipsoid contained in $E(K'')$, which is greater than $\frac{13}{10}$ of the volume of the unit ball. Finally we define an affine bijective transformation to a general ellipsoid $E(K)$. See Figure 3.2 as a sketch of part of the calculation.

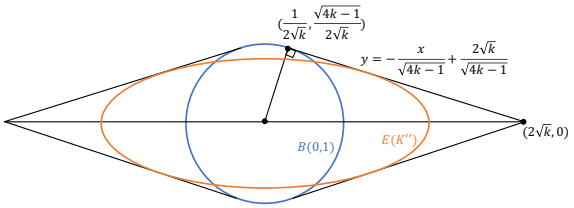


Figure 3.2: Maximum volume ellipsoid

It is noteworthy that the convex set X_i (of feature vectors in $\mathbb{R}^k$) we keep in the proof is defined to be close to any possible subspace that is close to the subspace spanned by new features $\tilde{\boldsymbol{w}}_i$. So our proof is quite general for any lifelong learning algorithm that dynamically expands the architecture, *e.g.*the basic LLL algorithm.

3.2 Proof of Theorem 5

Let $(\boldsymbol{X}^*, t^*)$ be a solution of the SDP (2.2) with singular value decomposition (SVD) $\boldsymbol{X}^* = \sum_{i=1}^d \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$, where $0 \le \lambda_1 \le \cdots \le \lambda_d \le 1$, and $\sum_{i=1}^d \lambda_i = d - k$. Let $\boldsymbol{V}'$ be the span of $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{k'}\}$. Then the squared distance of any vector $\boldsymbol{a}$ to $\boldsymbol{V}'$ is $\sum_{i=k'+1}^d (\boldsymbol{a}^\top \boldsymbol{u}_i)^2$. In the meantime, the SDP assigns a value of $\boldsymbol{a}^\top \boldsymbol{X}^* \boldsymbol{a} = \sum_{i=1}^d \lambda_i (\boldsymbol{a}^\top \boldsymbol{u}_i)^2$. Thus, the multiplicative increase in squared distance is at most

$$\frac{\sum_{i=k'+1}^d (\boldsymbol{a}^\top \boldsymbol{u}_i)^2}{\sum_{i=1}^d \lambda_i (\boldsymbol{a}^\top \boldsymbol{u}_i)^2} \le \frac{1}{\lambda_{k'+1}}$$

Now since the sum of all eigenvalues is $d-k$ and each one is at most 1, for $k' \ge k$, we must have

$$\lambda_{k'+1} \ge \frac{(d-k) - (d-k'-1)}{k'+1} = \frac{k'-k+1}{k'+1}$$

Choose $k' = ck - 1$, and we would have

$$\frac{\sum_{i=ck}^d (\boldsymbol{a}^\top \boldsymbol{u}_i)^2}{\sum_{i=1}^d \lambda_i (\boldsymbol{a}^\top \boldsymbol{u}_i)^2} \le \frac{1}{\lambda_{ck}} \le 1 + \frac{1}{c-1}$$

Since there exists a subspace $\boldsymbol{V}^*$ of k dimension with $d(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}^*) \le \epsilon_{acc}, \forall i \in [\hat{k}+1]$, we have $\sum_{i=1}^d \lambda_i (\tilde{\boldsymbol{w}}_i^\top \boldsymbol{u}_i)^2 \le \epsilon_{acc}^2$. Consequently, $\text{span}(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{ck-1})$ is a $(ck-1)$-dimensional approximation of $\boldsymbol{V}^*$ with the approximation factor $\sqrt{1 + \frac{1}{c-1}}$ in maximum distance.

Specifically, for $c = 2$, we have $d^2(\tilde{\boldsymbol{w}}_i, \boldsymbol{V}') = \sum_{i=2k}^d (\tilde{\boldsymbol{w}}_i^\top \boldsymbol{u}_i)^2 \le 2\epsilon_{acc}^2$.

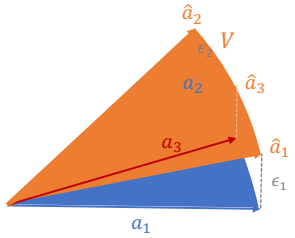


Figure 3.3: Geometric illustration of the lower bound examples when $k = 2, d = 3$. The errors that the algorithm makes concentrate on the third coordinate, and thus lead to the large angle between the learned feature subspace V and the underlying one U. Any new task, *e.g.*, $\boldsymbol{a}_3$ that lies on the span of $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ cannot help improve the representation V.

3.3 Proof idea of Theorem 2 (lower bound)

Consider a sequence of tasks

$$\boldsymbol{a}_i = \begin{cases} \boldsymbol{e}_i, & 1 \le i \le k \\ \sum_{j \in S} x_j \boldsymbol{e}_j, & i > k \end{cases}$$

where $x_j \overset{\text{i.i.d}}{\sim} \text{Bernoulli}(1/2)$, $\boldsymbol{e}_j$ is the standard unit vector, $S \subset [k]$ is a subset of indices. The proof is mainly by constructing an adversarial output of the algorithm where the errors that it makes concentrate on one coordinate, say $k+1$. (See Figure 3.3.) We will show by the following steps: (1) After learning the first k tasks, each with error ϵ_i, the angle between the learned feature subspace and the underlying one is at least $\Omega(\sqrt{\sum_{i \in S} \epsilon_i^2})$ for some subset of at least $k/2$ tasks. (2) For each subsequent task, the angle of the new task to the learned subspace is at least $\Omega(\sqrt{\sum_{i \in S} \epsilon_i^2})$ with high probability. (3) Learning such a new task does not improve the representation. (4) To solve all tasks up to error ϵ, we will need each $\epsilon_i = O(\epsilon/\sqrt{k})$, and it leads to the sample complexity bound. The full proof and related lemmas are in Appendix B.

4 Simulations and Empirical Results

In this section, we describe our experimental studies. In Section 4.1, we run the basic LLL and LLL-RR algorithms in a task-incremental binary classification setting. Then we conduct class-incremental experiments on real dataset using our H-LLL algorithm in Section 4.2. The performance shows the benefits of our algorithm compared to existing continual learning algorithms.

4.1 Linear Features

Here we consider task-incremental lifelong learning in the setting of binary classification where $y = \text{sign}(\langle \boldsymbol{c}_i^*, \boldsymbol{W}^* \boldsymbol{x} \rangle)$. We choose the input dimension, $d = 100$, the number of tasks, $m = 100$, the number of examples per task, $N = 200$, the dimension of feature subspace, $k = 5$. The parameters $c_{ij}^*, W_{ij}^* \sim$

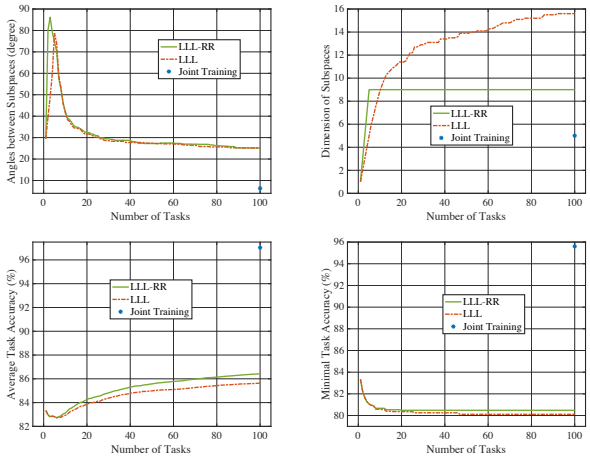


Figure 4.1: Simulation on linear features. $N = 200, k = 5, m = 100, d = 100$, averaged on 10 trials.

$\mathcal{N}(0, 1)$. The input data $X_i \sim \mathcal{N}(0, 1)$. We set the error threshold to be $\epsilon = 0.1$. We compare three methods: LLL (Basic lifelong learning algorithm), LLL-RR (lifelong learning algorithm with representation refinement), and Joint Training (offline training with all data jointly).

The average task accuracy and minimal task accuracy are computed for tasks encountered so far based on the current model. The angle between feature subspaces is calculated as their maximal principal angle. Formally, for two subspaces F and G, let $\boldsymbol{P}, \boldsymbol{Q}$ to be the orthogonal matrices whose columns form an orthonormal basis of F and G. For the singular value decomposition $\boldsymbol{P}^\top \boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, we define the principal angles between F and G as $\theta_i = \arccos(\boldsymbol{\Sigma}_{ii})$, $\frac{\pi}{2} \geq \theta_1 \geq \cdots, \geq \theta_k \geq 0$. We calculate the angle between two subspaces F and G as the maximal principal angle, *i.e.*, $\arccos(\boldsymbol{\Sigma}_{11})$.

As we can see in Figure 4.1, lifelong learning can continually learn better features while learning more tasks. Moreover, lifelong learning with refinement improves average accuracy, min accuracy, model size and convergence to the underlying feature subspace.

4.2 Image Classification

Experimental settings. We generally follow the experimental settings and evaluation protocol in [Rebuffi et al., 2017]. In our experiments, we evaluate our H-LLL algorithm on CIFAR-100. We train all 100 classes in 10 splits and each split contains 10 classes. There is no class overlap between different splits. Each training data split can be viewed as a task and is fed to the neural network incrementally. Similar to [Rebuffi et al., 2017], we use a fixed memory size of 2,000 exemplars. The final result are curves of the classification accuracies after each batch of classes. We

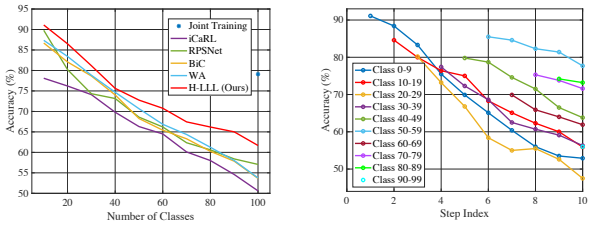


Figure 4.2: Incremental Classification accuracy on CIFAR-100.

use ResNet-18 [He et al., 2016] for all the encoders $f_j, \forall j$ and SGD with weight decay 0.0005. All the ResNet encoders are trained from scratch. For different methods, we use the same class split on CIFAR-100 to ensure fair comparison.

Accuracy vs. number of classes. In Fig. 4.2, we first show the comparison of incremental accuracies to some of the state-of-the-art methods including iCaRL [Rebuffi et al., 2017], RPSNet [Rajasegaran et al., 2019], BiC [Hou et al., 2019] and WA [Zhao et al., 2020]. One can observe that our H-LLL algorithm significantly outperforms the other methods and yields an average incremental accuracy [Rebuffi et al., 2017] of 73.8%, while the second best approach (WA) only achieves 69.8% accuracy.

Accuracy for different classes. In order to gain deeper understanding of the H-LLL algorithm, we examine the accuracy of different class splits in each step. From Fig. 4.2, we can see that the incremental accuracy for different class groups decreases in a slow and smooth way. This indicates that H-LLL is able to preserve knowledge of class concepts and effectively avoid catastrophic forgetting.

5 Discussion

We study, theoretically and empirically, the efficiency of lifelong learning when tasks share a low-dimensional feature representation. We introduce a refinement algorithm and bound its representation and sample complexity, and prove a matching lower bound for the sample complexity (for any lifelong learning algorithm). Our results show that: (1) lifelong learning provably converges for nonlinear feature representations, (2) refinement has provable benefits, and (3) lifelong learning is an efficient approach to multi-class/multi-task learning. Our work also indicates that (a) refinement can be practical and can dynamically keep the dimension of the representation bounded and (b) remembering only a small subset of previous examples suffices for efficient lifelong learning.

These results raise several questions; we mention a few. In the general setting of nonlinear features, how can we guarantee that the refinement is efficient in terms of time complexity?

Acknowledgements. This work was supported in part by NSF awards CCF-1909756, CCF-2007443 and CCF-2134105. Weiyang Liu is supported by a Cambridge-Tübingen Fellowship, an NVIDIA GPU grant, DeepMind and the Leverhulme Trust via CFI. We thank Le Song for helpful discussions.

References

[Balcan et al., 2015] Balcan, M.-F., Blum, A., and Vempala, S. (2015). Efficient representations for lifelong learning and autoencoding. In *Conference on Learning Theory*, pages 191–210. PMLR. 1, 2, 3, 4, 5, 21

[Balcan et al., 2019] Balcan, M.-F., Khodak, M., and Talwalkar, A. (2019). Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433. PMLR. 4

[Balcan and Long, 2013] Balcan, M.-F. and Long, P. (2013). Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316. PMLR. 16

[Bartlett et al., 2019] Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.*, 20(63):1–17. 5

[Baxter, 1997] Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39. 4

[Bennani et al., 2020] Bennani, M. A., Doan, T., and Sugiyama, M. (2020). Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*. 4

[Castro et al., 2018] Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *ECCV*. 6

[Chua et al., 2021] Chua, K., Lei, Q., and Lee, J. D. (2021). How fine-tuning allows for effective meta-learning. *arXiv preprint arXiv:2105.02221*. 1, 4

[Denevi et al., 2020] Denevi, G., Pontil, M., and Ciliberto, C. (2020). The advantage of conditional meta-learning for biased regularization and fine-tuning. *arXiv preprint arXiv:2008.10857*. 4

[Denevi et al., 2021] Denevi, G., Pontil, M., and Ciliberto, C. (2021). Conditional meta-learning of linear representations. *arXiv preprint arXiv:2103.16277*. 4

[Denevi et al., 2019] Denevi, G., Stamos, D., Ciliberto, C., and Pontil, M. (2019). Online-within-online meta-learning. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 32 (NIPS 2019)*, volume 32, pages 1–11. Neural Information Processing Systems (NeurIPS 2019). 4

[Doan et al., 2021] Doan, T., Bennani, M. A., Mazoure, B., Rabusseau, G., and Alquier, P. (2021). A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR. 4

[Du et al., 2020] Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. (2020). Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*. 1, 3, 4

[Farajtabar et al., 2020] Farajtabar, M., Azizan, N., Mott, A., and Li, A. (2020). Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR. 4

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR. 4

[Finner, 1992] Finner, H. (1992). A generalization of holder's inequality and some probability inequalities. *The Annals of probability*, pages 1893–1901. 20

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 9

[Hou et al., 2019] Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *CVPR*. 9

[Jacot et al., 2018] Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*. 4

[Khodak et al., 2019] Khodak, M., Balcan, M.-F., and Talwalkar, A. (2019). Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*. 4

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural

networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526. 3

[Li and Hoiem, 2017] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947. 3

[Liu et al., 2021] Liu, W., Lin, R., Liu, Z., Rehg, J. M., Paull, L., Xiong, L., Song, L., and Weller, A. (2021). Orthogonal over-parameterized training. In *CVPR*. 3

[Liu et al., 2019] Liu, W., Liu, Z., Rehg, J., and Song, L. (2019). Neural similarity learning. In *NeurIPS*. 3

[Lovász and Vempala, 2007] Lovász, L. and Vempala, S. (2007). The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358. 3, 15

[McCloskey and Cohen, 1989] McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier. 3

[Parisi et al., 2019] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71. 1

[Rajasegaran et al., 2019] Rajasegaran, J., Hayat, M., Khan, S. H., Khan, F. S., and Shao, L. (2019). Random path selection for continual learning. In *NeurIPS*. 9

[Ratcliff, 1990] Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285. 3

[Rebuffi et al., 2017] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010. 4, 6, 9

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*. 3

[Thrun and Mitchell, 1995] Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46. 1, 3

[Tripuraneni et al., 2020] Tripuraneni, N., Jin, C., and Jordan, M. I. (2020). Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*. 1, 3, 4

[Wang et al., 2020] Wang, R., Demiris, Y., and Ciliberto, C. (2020). A structured prediction approach for conditional meta-learning. *Advances in Neural Information Processing Systems*. 4

[Welling, 2009] Welling, M. (2009). Herding dynamical weights to learn. In *ICML*. 6

[Xu and Zhu, 2018] Xu, J. and Zhu, Z. (2018). Reinforced continual learning. *arXiv preprint arXiv:1805.12369*. 3

[Yan et al., 2021] Yan, S., Xie, J., and He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In *CVPR*. 6

[Yin et al., 2020] Yin, D., Farajtabar, M., Li, A., Levine, N., and Mott, A. (2020). Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*. 4

[Yoon et al., 2017] Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2017). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*. 3

[Zhang et al., 2018] Zhang, X., Yu, F. X., Karaman, S., Zhang, W., and Chang, S.-F. (2018). Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*. 6

[Zhao et al., 2020] Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S.-T. (2020). Maintaining discrimination and fairness in class incremental learning. In *CVPR*. 9

Supplementary Material: Provable Lifelong Learning of Representations

A Proof of Theorem 3 and Theorem 4

Here we restate and prove all the lemmas before giving the full proof of the two main theorems.

Lemma 1. *Given two unit vectors $\boldsymbol{u}, \boldsymbol{v}$ and a distribution D. If D satisfies Assumption 1, then there exist nonzero constants c' and c'' such that $c'\|\boldsymbol{u}-\boldsymbol{v}\|_2 \le d_D(\boldsymbol{u},\boldsymbol{v}) \le c''\|\boldsymbol{u}-\boldsymbol{v}\|_2$.*

Proof. By Assumption 1, $\exists c_1, c_2$ s.t. $c_1\theta(\boldsymbol{u},\boldsymbol{v}) \le d_D(\boldsymbol{u},\boldsymbol{v}) \le c_2\theta(\boldsymbol{u},\boldsymbol{v})$. Using the Taylor expansion of cosine function, we know $1-x^2/2 \le \cos(x) \le 1-x^2/2!+x^4/4! \le 1-11x^2/24$. Since $\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 = 2-2\cos(\theta(\boldsymbol{u},\boldsymbol{v}))$, we have $\sqrt{\frac{11}{12}}\theta(\boldsymbol{u},\boldsymbol{v}) \le \|\boldsymbol{u}-\boldsymbol{v}\|_2 \le \theta(\boldsymbol{u},\boldsymbol{v})$. Choose $c'=c_1, c''=\sqrt{\frac{12}{11}}c_2$, we get the results proved. □

Lemma 2. *Let S be a set of subspaces. Let $X = \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x},\boldsymbol{Y}) \le r, \forall \boldsymbol{Y} \in S\}$. Here $B_k(0,1)$ is the unit ball in k-dimension. Then the set X is a symmetric convex set.*

Proof. For any $\boldsymbol{x} \in X, \forall \boldsymbol{Y} \in S$, since $d(\boldsymbol{x},\boldsymbol{Y}) = d(-\boldsymbol{x},\boldsymbol{Y})$, we have $-\boldsymbol{x} \in X$. So S is symmetric about the origin.

For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in X$, for any $\boldsymbol{Y} \in S$, we have $d(\boldsymbol{x}_1,\boldsymbol{Y}) \le r, d(\boldsymbol{x}_2,\boldsymbol{Y}) \le r$. Let $\boldsymbol{P}$ be the projection matrix of $\boldsymbol{Y}$,

$$\begin{aligned}
&\left(\|\boldsymbol{P}\boldsymbol{x}_1\|_2 + \|\boldsymbol{P}\boldsymbol{x}_2\|_2\right)^2 - \|\boldsymbol{P}\left(\boldsymbol{x}_1+\boldsymbol{x}_2\right)\|_2^2 \\
=&\boldsymbol{x}_1^\top \boldsymbol{P}^\top \boldsymbol{P}\boldsymbol{x}_1 + \boldsymbol{x}_2^\top \boldsymbol{P}^\top \boldsymbol{P}\boldsymbol{x}_2 + 2\|\boldsymbol{P}\boldsymbol{x}_1\|_2\|\boldsymbol{P}\boldsymbol{x}_2\|_2 - \left(\boldsymbol{x}_1+\boldsymbol{x}_2\right)^\top \boldsymbol{P}^\top \boldsymbol{P}\left(\boldsymbol{x}_1+\boldsymbol{x}_2\right) \\
=&\|\boldsymbol{P}\boldsymbol{x}_1\|_2\|\boldsymbol{P}\boldsymbol{x}_2\|_2 - \left(\boldsymbol{P}\boldsymbol{x}_1\right)^\top \left(\boldsymbol{P}\boldsymbol{x}_2\right) \ge 0
\end{aligned}$$

So we have

$$d\left(\frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}, \boldsymbol{Y}\right) = \left\|\boldsymbol{P}\left(\frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}\right)\right\|_2 \le \left\|\boldsymbol{P}\left(\frac{\boldsymbol{x}_1}{2}\right)\right\|_2 + \left\|\boldsymbol{P}\left(\frac{\boldsymbol{x}_2}{2}\right)\right\|_2 = \frac{1}{2}d\left(\boldsymbol{x}_1,\boldsymbol{Y}\right) + \frac{1}{2}d\left(\boldsymbol{x}_2,\boldsymbol{Y}\right) \le r$$

For a fixed $\boldsymbol{Y}$, $\{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x},\boldsymbol{Y}) \le r\}$ is closed and thus convex. Therefore

$$X = \bigcap_{\boldsymbol{Y} \in S} \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x},\boldsymbol{Y}) \le r\}$$

is a convex set. □

Lemma 3. *For any $\hat{k} \in [\tilde{k}]$, let*

$$Y_{i_{\hat{k}}} := \{\boldsymbol{V} | d(\tilde{\boldsymbol{w}}_{i_j}, \boldsymbol{V}) \le c_1\epsilon_{acc}, \forall j \le \hat{k}\},$$

$$X_{i_{\hat{k}}} := \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x},\boldsymbol{V}) \le (c_1 + \frac{1}{c'})\epsilon_{acc}, \forall \boldsymbol{V} \in Y_{i_{\hat{k}}}\},$$

where c_1, c' are small constants. Then for any $j \ge \hat{k}, \pm\boldsymbol{a}_{i_{\hat{k}}} \in X_{i_j}$.

Proof. For $\forall \boldsymbol{V} \in Y_{i_{\hat{k}}}$, $d(\tilde{\boldsymbol{w}}_{i_{\hat{k}}}, \boldsymbol{V}) \le c_1\epsilon_{acc}$. Since we learn the feature vector $\tilde{\boldsymbol{w}}_{i_{\hat{k}}}$ within error ϵ_{acc}, by Lemma 1, we have $d(\boldsymbol{a}_{i_{\hat{k}}}, \tilde{\boldsymbol{w}}_{i_{\hat{k}}}) \le \epsilon_{acc}/c'$. So $d(\boldsymbol{a}_{i_{\hat{k}}}, \boldsymbol{V}) \le (c_1 + \frac{1}{c'})\epsilon_{acc}$. Hence $\boldsymbol{a}_{i_{\hat{k}}} \in X_i$. Because $d(\boldsymbol{x},\boldsymbol{V}) = d(-\boldsymbol{x},\boldsymbol{V})$, we also know $-\boldsymbol{a}_{i_{\hat{k}}} \in X_i$. Also $Y_{i_1} \supseteq Y_{i_2} \supseteq \cdots \supseteq Y_{i_{\tilde{k}}}$, so $X_{i_1} \subseteq X_{i_2} \subseteq \cdots \subseteq X_{i_{\tilde{k}}}$. So for any $j \ge \hat{k}$, $\pm\boldsymbol{a}_{i_{\hat{k}}} \in X_{i_j}$. □

Corollary 1. *For any $\hat{k} \in [\tilde{k}]$, $\mathrm{conv}(\pm\boldsymbol{a}_{i_1}, \cdots, \pm\boldsymbol{a}_{i_{\hat{k}}}) \subseteq X_{i_{\hat{k}}}$.*

Proof. From Lemma 3, we know $\pm\boldsymbol{a}_{i_1}, \cdots, \pm\boldsymbol{a}_{i_{\hat{k}}} \in X_{i_{\hat{k}}}$. Combined with Lemma 2, we get the corollary. □

Lemma 4 (Max Ellipsoid). *Let $K \subset \mathbb{R}^k$ be a symmetric convex body and $E(K)$ be the maximum volume ellipsoid contained in K. For a vector $\boldsymbol{u}$ on the boundary of $E(K)$,* i.e.$\boldsymbol{u} \in \mathrm{bd}(E(K))$, *let $K' = \mathrm{conv}(K, 2\sqrt{k}\boldsymbol{u}, -2\sqrt{k}\boldsymbol{u})$. Then,*

$$\frac{\mathrm{vol}\left(E\left(K'\right)\right)}{\mathrm{vol}\left(E\left(K\right)\right)} \geq \frac{13}{10}.$$

Proof. Since $E(K) \subseteq K$, we have

$$K'' := \mathrm{conv}\left(E\left(K\right), 2\sqrt{k}\boldsymbol{u}, -2\sqrt{k}\boldsymbol{u}\right) \subseteq K'$$

So it suffices to prove that

$$\frac{\mathrm{vol}\left(E\left(K''\right)\right)}{\mathrm{vol}\left(E\left(K\right)\right)} \geq \frac{13}{10}$$

First, let's assume that $E(K) = B_k(0,1)$, *i.e.*, the unit ball around the origin, and the vector $\boldsymbol{u}$ is $(1, 0, \cdots, 0)^\top \in B_k(0,1)$. By symmetry, we can assume the ellipsoid

$$E(K'') = \left\{ \boldsymbol{x} \,\middle|\, \frac{x_1^2}{a^2} + \sum_{i=2}^{k} \frac{x_i^2}{b^2} \leq 1 \right\}$$

We consider the two-dimensional slice first, where $\boldsymbol{u} = (1,0)^\top$, $E(K'') = \{(x,y) | \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1\}$. Direct calculation shows that

$$\forall (x,y) \in K'', y \leq -\frac{x}{\sqrt{4k-1}} + \frac{2\sqrt{k}}{\sqrt{4k-1}}$$

where $y = -\frac{x}{\sqrt{4k-1}} + \frac{2\sqrt{k}}{\sqrt{4k-1}}$ is the line tangent to the unit ball and go across the point $(\frac{1}{2\sqrt{k}}, \frac{\sqrt{4k-1}}{2\sqrt{k}})$ and the point $(2\sqrt{k}, 0)$ (see Figure 3.2). So for any point on the boundary of $E(K'')$, it also satisfies

$$y^2 = b^2\left(1 - \frac{x^2}{a^2}\right) \leq \left(-\frac{x}{\sqrt{4k-1}} + \frac{2\sqrt{k}}{\sqrt{4k-1}}\right)^2$$

Simplifying the inequality we get

$$\left(\frac{b^2}{a^2} + \frac{1}{4k-1}\right) x^2 - \frac{4\sqrt{k}}{4k-1} x + \frac{4k}{4k-1} - b^2 \geq 0$$

To ensure that the quadratic inequality holds, let the determinant equal zero, and we get $a^2 = 4k - b^2(4k-1)$. So,

$$\left\{ (x,y) \,\middle|\, \frac{x^2}{4k - b^2(4k-1)} + \frac{y^2}{b^2} \leq 1 \right\} \subseteq K'' \text{ for } b < 1.$$

By symmetry, in k dimensions, we get that $E_b \subseteq E(K'')$ for $b < 1$.

$$E_b = \left\{ x \,\middle|\, \frac{x_1^2}{4k - b^2(4k-1)} + \sum_{i=2}^{k} \frac{x_i^2}{b^2} \leq 1 \right\}$$

The volume of this ellipsoid is

$$\mathrm{vol}\left(E_b\right) = \sqrt{\left(4k - b^2\left(4k-1\right)\right) b^{2k-2}} \cdot \mathrm{vol}\left(B_k\left(0,1\right)\right)$$

Let $f(b) = (4k - b^2(4k-1))b^{2k-2}$. Calculate its derivative and let it to be zero, so we get $\hat{b}^2 = 1 - \frac{3}{4k-1} < 1$.

$$f\left(\hat{b}\right) = 4\left(1 - \frac{3}{4k-1}\right)^{k-1} = 4\left(\left(1 - \frac{3}{4k-1}\right)^{\frac{4k-4}{3}}\right)^{\frac{3}{4}} \geq 4\left(\frac{1}{e}\right)^{\frac{3}{4}}$$

Hence we know

$$\operatorname{vol}\left(E_{\hat{b}}\right) \geq 2\left(\frac{1}{e}\right)^{\frac{3}{8}} \operatorname{vol}\left(B_k\left(0,1\right)\right) \geq \frac{13}{10}\operatorname{vol}\left(B_k\left(0,1\right)\right)$$

Finally for any symmetric ellipsoid $E(K) := \boldsymbol{A}^{1/2}B_k(0,1)$ and any $\boldsymbol{u} = \boldsymbol{A}^{1/2}\boldsymbol{u}_0$ on the boundary of $E(K)$, where $\boldsymbol{u}_0$ is the unit vector corresponding to $\boldsymbol{u}$. There exists an orthogonal matrix, say $\boldsymbol{Q}$, that rotates $\boldsymbol{u}_0$ to $(1,0,\cdots,0)^\top$. That is $\boldsymbol{Q}\boldsymbol{u}_0 = (1,0,\cdots,0)^\top$. Define an affine bijective transformation $T := \boldsymbol{A}^{1/2}\boldsymbol{Q}^\top\boldsymbol{x}$, with $T^{-1}(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{A}^{-1/2}\boldsymbol{x}$. Then

$$T\left((1,0,\cdots,0)^\top\right) = \boldsymbol{A}^{1/2}\boldsymbol{Q}^\top\boldsymbol{Q}\boldsymbol{u}_0 = \boldsymbol{A}^{1/2}\boldsymbol{u}_0 = \boldsymbol{u}$$

$$\begin{aligned}
T\left(B_k\left(0,1\right)\right) =& \{T\left(\boldsymbol{y}\right)|\boldsymbol{y}^\top\boldsymbol{y} \leq 1\} \\
=& \{\boldsymbol{x}|T^{-1}\left(\boldsymbol{x}\right)^\top T^{-1}\left(\boldsymbol{x}\right) \leq 1\} \\
=& \{\boldsymbol{x}|\boldsymbol{x}^\top\boldsymbol{A}^{-1/2}\boldsymbol{Q}^\top\boldsymbol{Q}\boldsymbol{A}^{-1/2}\boldsymbol{x} \leq 1\} \\
=& \{\boldsymbol{x}|\boldsymbol{x}^\top\boldsymbol{A}^{-1}\boldsymbol{x} \leq 1\} \\
=& E\left(K\right)
\end{aligned}$$

So we get $T(E_{\hat{b}}) \subset E(K'')$. Since the ratio of volumes is invariant under affine transformation, we have

$$\frac{\operatorname{vol}\left(E\left(K''\right)\right)}{\operatorname{vol}\left(E\left(K\right)\right)} \geq \frac{\operatorname{vol}\left(T\left(E_{\hat{b}}\right)\right)}{\operatorname{vol}\left(T\left(B_k\left(0,1\right)\right)\right)} = \frac{\operatorname{vol}\left(E_{\hat{b}}\right)}{\operatorname{vol}\left(B_k\left(0,1\right)\right)} \geq \frac{13}{10}.$$

□

The next lemma gives us a stopping condition.

Lemma 5. *Let $P = \operatorname{conv}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m)$ be a polytope in $\mathbb{R}^k$ with each $\boldsymbol{x}_i$ of unit Euclidean length. Then, the maximum volume ellipsoid contained in P satisfies*

$$\operatorname{vol}\left(E\left(P\right)\right) \leq 2\sqrt{2e}\left(\sqrt{\frac{2\log 2m}{k}}\right)^k \operatorname{vol}\left(B_k\left(0,1\right)\right).$$

Proof. Recall the polar of a convex body P is the convex body defined as

$$P^* = \{\boldsymbol{x} : \langle\boldsymbol{x},\boldsymbol{y}\rangle \leq 1 \text{ for all } \boldsymbol{y} \in P\}.$$

By the Blaschke-Santalo inequality, we have

$$\operatorname{vol}\left(P\right)\operatorname{vol}\left(P^*\right) \leq \operatorname{vol}\left(B_k\left(0,1\right)\right)^2.$$

Next we lower bound the volume of P^*. Note that P^* is the intersection of exactly m halfspaces, each tangent to the unit ball. Consider the ball $B_k(0,r)$ with $r = \sqrt{\frac{k-1}{2\log(2m)}}$. By Lemma 6, each halfpace cuts off a cap of this ball, of volume at most $e^{-(k-1)/2r^2} = \frac{1}{2m}$ of the volume of $B_k(0,r)$. Therefore, the volume that is in the intersection of all m halfspaces is at least $\operatorname{vol}(B_k(0,r))/2$ and hence, this is a lower bound on the volume of P^*. Using this, we have,

$$\operatorname{vol}\left(P\right) \leq \frac{\operatorname{vol}\left(B_k\left(0,1\right)\right)^2}{\operatorname{vol}\left(B_k\left(0,r\right)\right)/2} \leq 2r^{-k}\operatorname{vol}\left(B_k\left(0,1\right)\right)$$

Furthermore, since $1 + x \leq e^x$, we can derive the following and complete the proof.

$$2r^{-k} = 2(\sqrt{\frac{2\log(2m)}{k-1}})^k = 2(\sqrt{\frac{2\log(2m)}{k}})^k(1+\frac{1}{k-1})^{\frac{k-1}{2}}\sqrt{1+\frac{1}{k-1}} \leq 2\sqrt{2e}(\sqrt{\frac{2\log(2m)}{k}})^k$$

□

Lemma 6. *(Lemma 4.1 from [Lovász and Vempala, 2007])For any $\frac{1}{\sqrt{k}} < t < 1$ and halfspace H at distance tr from the origin,*

$$\text{vol}\left(B_k\left(0,r\right) \cap H\right) \leq e^{-t^2k/2}\text{vol}\left(B_k\left(0,r\right)\right).$$

We can now prove Theorem 4.

Theorem 4 (LLL with Representation Refinement)**.** *Consider the lifelong learning setting of input dimension d, m tasks with k common features. Suppose that the algorithm has access to an oracle that gives a constant-factor approximation of Optimization Problem 2.1. Set $\epsilon_{acc} = \frac{\epsilon}{c\sqrt{k}}$ for a sufficiently small constant $c > 0$. Under Assumption 1, the LLL-RR algorithm learns at most $O(k\log(\log(k)/\epsilon))$ new features, and the dimension of the feature space is $O(k)$. The total number of labeled examples to learn tasks to within error ϵ is $O(\frac{dk^{1.5}}{\epsilon}\log(\frac{\log(k)}{\epsilon})\log(\frac{k}{\epsilon}) + \frac{km}{\epsilon}\log(\frac{1}{\epsilon})) = \tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$.*

Proof. Let's first prove the linear case. Since the ground truth feature space lies in a k-dimensional subspace $\boldsymbol{V}^* = \text{span}(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m) \subseteq \mathbb{R}^k$, the span of the true features is $\mathbb{R}^k$. Along training, when we deal with the $i_{\hat{k}}$-th task, let $\hat{\boldsymbol{w}}_{i_{\hat{k}}}$ be the new feature we learn to ensure that the $i_{\hat{k}}$-th task has error no more than $\epsilon_{acc} = \frac{\epsilon}{2\sqrt{k}c''(c_1+1/c')}$, where c', c'' are universal constants defined in Lemma 1 and c_1 is the approximation constant of Optimization Problem 2.1. WLOG we assume $\hat{\boldsymbol{w}}_{i_{\hat{k}}}$ to be a unit vector. By Lemma 1,

$$d(\boldsymbol{a}_{i_{\hat{k}}}, \hat{\boldsymbol{w}}_{i_{\hat{k}}}) \leq \frac{\epsilon_{acc}}{c'} = \frac{\epsilon}{2\sqrt{k}c''(c_1c'+1)}$$

Denote $\boldsymbol{V}_{i_{\hat{k}}}$ be the feature subspace after fine-tuning (optimization). Since there exists a k-dimensional subspace $\boldsymbol{V}^*$ satisfying all constraints and the algorithm outputs a constant-factor approximation of Optimization Problem 2.1, we can get a c_1-approximation solution with dimension c_2k for constants c_1, c_2. So we know in the end the dimension of the feature subspace we get is $O(k)$.

Let $B_k(0,1)$ be the unit ball on the subspace. Denote the set of all possible solutions of the optimization as $Y_{i_{\hat{k}}} := \{\boldsymbol{V} | d(\tilde{\boldsymbol{w}}_{i_j}, \boldsymbol{V}) \leq c_1\epsilon_{acc}, \forall j \leq \hat{k}\}$. Let $X_{i_{\hat{k}}}$ be all the vectors in the unit ball that is within distance $(c_1 + \frac{1}{c'})\epsilon_{acc} = \frac{\epsilon}{2\sqrt{k}c''}$ to all of the subspaces in $Y_{i_{\hat{k}}}$, that is $X_{i_{\hat{k}}} := \{\boldsymbol{x} \in B_k(0,1) | d(\boldsymbol{x}, \boldsymbol{V}) \leq \frac{\epsilon}{2\sqrt{k}c''}, \forall \boldsymbol{V} \in Y_{i_{\hat{k}}}\}$. By Lemma 2 and Corollary 1, we know $X_{i_{\hat{k}}}$ is a convex set containing $\{\pm\boldsymbol{a}_{i_1}, \cdots, \pm\boldsymbol{a}_{i_{\hat{k}}}\}$. We will show next that after learning $\tilde{k} = O(k\log(\log(k)/\epsilon))$ new tasks, $X_{i_{\tilde{k}}}$ contains the ball $B_k(0, 1/2\sqrt{k})$.

In the initial step, Y_0 is the set of all subspaces. We naturally have $B(0, \frac{\epsilon}{2\sqrt{k}c''}) \subseteq X_0$ since for any $\boldsymbol{x} \in B(0,\epsilon)$, $d(\boldsymbol{x}, \boldsymbol{V}) \leq d(\boldsymbol{O}, \boldsymbol{V}) + d(\boldsymbol{x}, \boldsymbol{O}) \leq \frac{\epsilon}{2\sqrt{k}c''}$. Here $\boldsymbol{O}$ is the origin. So we know $\text{vol}(X_0) \geq (\frac{\epsilon}{2\sqrt{k}c''})^k V_0$, where V_0 is the volume of the unit ball in $\mathbb{R}^k$. Encountering the $i_{\hat{k}}$-th task, the current feature subspaces $\boldsymbol{V}_{i_{\hat{k}-1}}$ cannot ensure an ϵ error. By Lemma 1, $d(\boldsymbol{a}_{i_{\hat{k}}}, \boldsymbol{V}_{i_{\hat{k}-1}}) \geq \frac{\epsilon}{c''}$. Hence $d(\frac{\boldsymbol{a}_{i_{\hat{k}}}}{2\sqrt{k}}, \boldsymbol{V}_{i_{\hat{k}-1}}) \geq \frac{\epsilon}{2\sqrt{k}c''}$, which means $\pm\frac{\boldsymbol{a}_{i_{\hat{k}}}}{2\sqrt{k}} \notin X_{i_{\hat{k}-1}}$. Consequently, the vector $\boldsymbol{u} = u'\boldsymbol{a}_{i_{\hat{k}}} \in \text{bd}(E(X_{i_{\hat{k}-1}}))$ satisfies $\|\boldsymbol{u}\| < \frac{1}{2\sqrt{k}}$. According to lemma 4, we know that

$$\text{vol}\left(\text{conv}\left(X_{i_{\hat{k}-1}}, \pm\boldsymbol{a}_{i_{\hat{k}}}\right)\right) \geq \text{vol}\left(\text{conv}\left(X_{i_{\hat{k}-1}}, \pm 2\sqrt{k}\boldsymbol{u}\right)\right) \geq \frac{13}{10}\text{vol}\left(X_{i_{\hat{k}-1}}\right)$$

Also because of the convexity of $X_{i_{\hat{k}}}$, we have $X_{i_{\hat{k}}} \supseteq \text{conv}(X_{i_{\hat{k}-1}}, \pm\boldsymbol{a}_{i_{\hat{k}}})$. Therefore,

$$\frac{\text{vol}\left(X_{i_{\hat{k}}}\right)}{\text{vol}\left(X_{i_{\hat{k}-1}}\right)} \geq \frac{13}{10}$$

The algorithm will terminate when $X_{i_{\tilde{k}}} \supseteq E(X_{i_{\tilde{k}}}) \supseteq B_k(0, \frac{1}{2\sqrt{k}})$. So for any unit vector $\boldsymbol{a} \in B_k(0,1)$, $d(\boldsymbol{a}, \boldsymbol{V}_{i_{\tilde{k}}}) \leq \frac{\epsilon}{c''}$. This means that after learning $\tilde{k}$ new features, for any new tasks with weights lie in the same feature subspace $\boldsymbol{V}^*$, the current features can achieve error less than ϵ. By Lemma 5,we know the volume of $E(X_{i_{\tilde{k}}})$ is upper bounded by $2\sqrt{2e}\left(\sqrt{\frac{2\log 2\tilde{k}}{k}}\right)^k \text{vol}(B_k(0,1))$. It grows by a constant factor $\frac{13}{10}$ whenever we learn a new feature. So the number of tasks we learn with error ϵ_{acc} in the algorithm $\tilde{k}$ satisfies:

$$\left(\frac{\epsilon}{2c''\sqrt{k}}\right)^k \cdot \left(\frac{13}{10}\right)^{\tilde{k}} \leq 2\sqrt{2e}\left(\sqrt{\frac{2\log 2\tilde{k}}{k}}\right)^k$$

Simplify and take the log to both sides, so we will have $\tilde{k} - \frac{k}{2}\log(2\log(2\tilde{k})) \leq const + k\log(\frac{1}{\epsilon})$. This will lead to $\tilde{k} \leq O(k\log(\log(k)/\epsilon))$.

Moreover, the sample complexity [Balcan and Long, 2013] of learning one task with input dimension d up to ϵ error is $O(d\log(1/\epsilon)/\epsilon)$. So the sample complexity of our algorithm is

$$
\begin{aligned}
&O\left(\frac{dk}{\epsilon_{acc}}\log\left(1/\epsilon_{acc}\right)\log\left(\log\left(k\right)/\epsilon\right)\right) + O\left(\frac{km}{\epsilon}\log\left(1/\epsilon\right)\right) \\
&= O\left(\frac{dk\sqrt{k}}{\epsilon}\log\left(k/\epsilon\right)\log\left(\log\left(k\right)/\epsilon\right) + \frac{km}{\epsilon}\log\left(1/\epsilon\right)\right) \\
&= \tilde{O}\left(\frac{dk\sqrt{k}}{\epsilon} + \frac{km}{\epsilon}\right).
\end{aligned}
$$

Finally, for the nonlinear case, we consider the kernel of the features. These features live in a potentially infinite-dimensional space. If we assume there is an oracle to get a constant approximation for the optimization problem 2.1, the dimension of features will be $O(k)$ in the end. Other bounds follow precisely the same as the linear case. □

It is noteworthy that the convex set X_i (of feature vectors in $\mathbb{R}^k$) we keep in the proof is defined to be close to any possible subspaces that are close to the subspace spanned by new features $\tilde{\boldsymbol{w}}_i$. So our proof is quite general for any lifelong learning algorithm that dynamically expands the architecture, *e.g.*the basic LLL algorithm. Consequently, we can prove Theorem 3 as follows.

Theorem 3 (Basic LLL)**.** *Consider the lifelong learning setting of input dimension d, m tasks with k common features. Let $\epsilon_{acc} = \frac{\epsilon}{c\sqrt{k}}$ for a sufficiently small constant $c > 0$. Under Assumption 1, the basic LLL algorithm, learns new features at most $\tilde{k} = O(k\log(\log(k)/\epsilon))$ times and the dimension of the learned feature space is $O(k\log(\log(k)/\epsilon))$ for linear features and $O(k^2\log(\log(k)/\epsilon))$ for nonlinear features. The total number of labeled examples to learn all tasks to within error ϵ is $O(\frac{dk^{1.5}}{\epsilon}\log(\frac{\log(k)}{\epsilon})\log(\frac{k}{\epsilon}) + \frac{km}{\epsilon}\log(\frac{\log(k)}{\epsilon})\log(\frac{1}{\epsilon})) = \tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$ for linear features and a factor of k higher for nonlinear features.*

Proof. The proof exactly follows the proof of the Theorem 4. Without the refinement of the feature subspace, the subspace we get is still in the set $Y_{i_{\tilde{k}}}$. Since $X_{i_{\tilde{k}}}$ will eventually cover the $B_k(0, \frac{1}{2\sqrt{k}})$ after learning $O(k\log(\log(k)/\epsilon))$ features, the dimension of the feature subspace is at most $O(k\log(\log(k)/\epsilon))$ for the linear features and $O(k^2\log(\log(k)/\epsilon))$ for the nonlinear features. So the total sample complexity is $\tilde{O}(dk^{1.5}/\epsilon + km/\epsilon)$ for linear features and $\tilde{O}(dk^{2.5}/\epsilon + k^2m/\epsilon)$ for nonlinear features. □

B A Lower Bound for General Lifelong Learning Algorithms

In this section, we show that our sample complexity bound for *general* lifelong learning algorithms in the linear setting is asymptotically the best possible, assuming black-box access to a learner for a single linear target.

As a warm-up, we first show that the analysis of *our* lifelong algorithm is *tight*.

Theorem 6 (Tight Example)**.** *Using the same condition and algorithm as in Theorem 4, the total sample complexity is $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$.*

Proof. Assume the task vectors $\boldsymbol{a}_i = \boldsymbol{e}_i, 1 \leq i \leq k-1$, where $\boldsymbol{e}_i \in \mathbb{R}^k$ has 1 in i-th coordinate and 0 otherwise. Assume that our algorithm accurately learns these $k-1$ tasks and returns $\tilde{\boldsymbol{w}}_i = \boldsymbol{a}_i + \epsilon_{acc}\boldsymbol{a}_k$. Then for a new task's weight $\frac{1}{\sqrt{k}}\sum_{i=1}^{k-1}\alpha_i\boldsymbol{a}_i$, $\alpha_i \in \{1, -1\}$. Based on the current features, the error we make on this task is $O(\sqrt{k}\epsilon_{acc})$. If we assume that $\epsilon_{acc} = \omega(\epsilon/\sqrt{k})$, then we need to learn these 2^{k-2} tasks accurately as well given learning any of them will not help with others (except its negative). Then the total complexity will be exponential with respect to k. So $\epsilon_{acc} = O(\epsilon/\sqrt{k})$, and thus we have the sample complexity $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$. □

Theorem 6 shows that the analysis of our algorithm's sample complexity is tight for the linear setting. The main result of this section is a lower bound for general lifelong learning algorithms (Theorem 2). Based on the proof idea we state in the paper, we will show each step accordingly with the following lemmas.

Lemma 7. *For k orthonormal tasks $\boldsymbol{a}_i = \boldsymbol{e}_i, 1 \le i \le k$, for any algorithm that learns task i within error ϵ_i,* i.e., *$d_D(\boldsymbol{a}_i, \hat{\boldsymbol{a}}_i) \le \epsilon_i$. Let $U = \text{span}(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k)$ be original feature subspace, and $V = \text{span}(\hat{\boldsymbol{a}}_1, \cdots, \hat{\boldsymbol{a}}_k)$ be the learned subspace. Then there exist feasible outputs $\hat{\boldsymbol{a}}_i$ such that $\theta(U, V) = \Omega(\sqrt{\sum_{i=1}^k \epsilon_i^2})$.*

Proof. Since the algorithm learns task i within error ϵ_i, with Lemma 1, there exists a constant c such that $\|\boldsymbol{a}_i - \hat{\boldsymbol{a}}_i\| \le \epsilon_i / c$. Because we does not consider constant factor, we assume WLOG that $\|\boldsymbol{a}_i - \hat{\boldsymbol{a}}_i\| \le \epsilon_i$. Consider the case where all errors made by the algorithm concentrate on the $k+1$ coordinate. Then the features learned by the algorithm are $\hat{\boldsymbol{a}}_i = \begin{pmatrix} \boldsymbol{e}_i \\ \epsilon_i \end{pmatrix}, 1 \le i \le k$. Denote $\boldsymbol{A} = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k) = \begin{pmatrix} \boldsymbol{I}_k \\ \boldsymbol{0}^\top \end{pmatrix}$, columns of which are the task vectors. Denote $\boldsymbol{s} = (\epsilon_1, \cdots, \epsilon_k)^\top$. $\hat{\boldsymbol{A}} = (\hat{\boldsymbol{a}}_1, \cdots, \hat{\boldsymbol{a}}_k) = \begin{pmatrix} \boldsymbol{I}_k \\ \boldsymbol{s}^\top \end{pmatrix}$. By the definition of the angles between subspace, we have $\theta(U, V) = \max_{\boldsymbol{x} \in \boldsymbol{U}} \theta(\boldsymbol{x}, \boldsymbol{P}_V \boldsymbol{x})$, where $\boldsymbol{P}_V$ is the projection matrix of the subspace V.

$$
\begin{aligned}
\boldsymbol{P}_V =& \hat{\boldsymbol{A}} (\hat{\boldsymbol{A}}^\top \hat{\boldsymbol{A}})^{-1} \hat{\boldsymbol{A}}^\top \\
=& \begin{pmatrix} \boldsymbol{I}_k \\ \boldsymbol{s}^\top \end{pmatrix} \left(\boldsymbol{I}_k + \boldsymbol{s}\boldsymbol{s}^\top \right)^{-1} \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{s} \end{pmatrix} \\
=& \begin{pmatrix} \boldsymbol{I}_k \\ \boldsymbol{s}^\top \end{pmatrix} \left(\boldsymbol{I}_k - \boldsymbol{s}\boldsymbol{s}^\top + o((\boldsymbol{s}\boldsymbol{s})^3) \right) \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{s} \end{pmatrix} \\
\approx& \begin{pmatrix} \boldsymbol{I}_k - \boldsymbol{s}\boldsymbol{s}^\top & \boldsymbol{s}(1 - \boldsymbol{s}^\top \boldsymbol{s}) \\ \boldsymbol{s}^\top (1 - \boldsymbol{s}^\top \boldsymbol{s}) & \boldsymbol{s}^\top \boldsymbol{s} - (\boldsymbol{s}^\top \boldsymbol{s})^2 \end{pmatrix}
\end{aligned}
$$

Let $\boldsymbol{x} = \boldsymbol{s} = (\epsilon_1, \cdots, \epsilon_k)^\top \in U$. Since

$$
\boldsymbol{P}_V \boldsymbol{x} = \left(\left(1 - \sum_{i=1}^k \epsilon_i^2 \right) \epsilon_1, \cdots, \left(1 - \sum_{i=1}^k \epsilon_i^2 \right) \epsilon_k, \left(1 - \sum_{i=1}^k \epsilon_i^2 \right) \sum_{i=1}^k \epsilon_i^2 \right)^\top,
$$

we have

$$
\tan \theta \left(\boldsymbol{x}, \boldsymbol{P}_V \boldsymbol{x} \right) = \frac{\left(1 - \sum_{i=1}^k \epsilon_i^2 \right) \sum_{i=1}^k \epsilon_i^2}{\left(1 - \sum_{i=1}^k \epsilon_i^2 \right) \sqrt{\sum_{i=1}^k \epsilon_i^2}} = \sqrt{\sum_{i=1}^k \epsilon_i^2}
$$

So we know that $\theta(U, V) \ge \theta(\boldsymbol{x}, \boldsymbol{P}_V \boldsymbol{x}) = \Omega(\sqrt{\sum_{i=1}^k \epsilon_i^2})$. So we prove that the angle between the learned subspace V and the underlying one U is at lease $\Omega(\sqrt{\sum_{i=1}^k \epsilon_i^2})$.

□

Lemma 8. *For any vector $\boldsymbol{x} = \sum_{i \in S} x_i \boldsymbol{e}_i$, where $S \subset [k]$, $x_i \overset{i.i.d}{\sim}$ Bernoulli$(1/2)$. Assume that $\forall i \in S, 0 \le \epsilon_i \le 2\sqrt{\sum_{i \in S} \epsilon_i^2 / |S|}$. Let $V = \text{span}(\{\hat{\boldsymbol{a}}_i\}, i \in S)$, where $\hat{\boldsymbol{a}}_i = \begin{pmatrix} \boldsymbol{e}_i \\ \epsilon_i \end{pmatrix}$. Then with high probability, $\theta(\boldsymbol{x}, V) = \Omega(\sqrt{\sum_{i \in S} \epsilon_i^2})$.*

Proof. Denote $\boldsymbol{y} = \boldsymbol{P}_V(\boldsymbol{x})$, then $y_j = \begin{cases} x_j - \epsilon_j \sum_{i \in S} \epsilon_i x_i & \text{if } j \in S \\ (1 - \sum_{i=1}^k \epsilon_i^2) \sum_{i \in S} \epsilon_i x_i & \text{if } j = k+1 \end{cases}$. Then we can calculate the angle between $\boldsymbol{x}$ and the subspace V as follows.

$$
\left(\tan \left(\theta \left(\boldsymbol{x}, \boldsymbol{P}_V \left(\boldsymbol{x} \right) \right) \right) \right)^2 = \frac{\left(1 - \sum_{i \in S} \epsilon_i^2 \right)^2 \left(\sum_{i \in S} \epsilon_i x_i \right)^2}{\sum_{i \in S} x_i^2 + \left(\sum_{i \in S} \epsilon_i^2 \right) \left(\sum_{i \in S} \epsilon_i x_i \right)^2 - 2 \left(\sum_{i \in S} \epsilon_i x_i \right)^2} \ge \frac{\frac{1}{4} \left(\sum_{i \in S} \epsilon_i x_i \right)^2}{\sum_{i \in S} x_i^2}
$$

The last inequality is because we learn each task well, so we can assume $\sum_{i\in S}\epsilon_i^2 \le 1/2$. Then the probability that $\theta(\boldsymbol{x}, \boldsymbol{P}_V(\boldsymbol{x}))$ is greater than $O(\sqrt{\sum_{i\in S}\epsilon_i^2})$ is as follows.

$$\begin{aligned}
\mathbb{P}\left(\theta\left(\boldsymbol{x}, \boldsymbol{P}_V\left(\boldsymbol{x}\right)\right) \ge \frac{1}{16}\sqrt{\sum_{i\in S}\epsilon_i^2}\right) =& \mathbb{P}\left(\tan^2\theta(\boldsymbol{x}, \boldsymbol{P}_V\left(\boldsymbol{x}\right)\right) \ge \frac{1}{256}\sum_{i\in S}\epsilon_i^2) \\
\ge& \mathbb{P}\left(\frac{\frac{1}{4}\left(\sum_{i\in S}\epsilon_i x_i\right)^2}{\sum_{i\in S}x_i^2} \ge \frac{1}{256}\sum_{i\in S}\epsilon_i^2\right) \\
=& \mathbb{P}\left(\left(\sum_{i\in S}\epsilon_i x_i\right)^2 \ge \frac{1}{64}\sum_{i\in S}\epsilon_i^2\sum_{i\in S}x_i^2\right) \\
\ge& \mathbb{P}\left(\left(\sum_{i\in S}\epsilon_i x_i\right)^2 \ge \frac{1}{64}|S|\sum_{i\in S}\epsilon_i^2\right)
\end{aligned}$$

By Chernoff bound, for any $t > 0$,

$$\mathbb{P}\left(\sum_{i\in S}\epsilon_i x_i - \frac{1}{2}\sum_{i\in S}\epsilon_i \ge -t\sqrt{\sum_{i\in S}\epsilon_i^2}\right) \ge 1 - e^{-t^2/2}$$

Choose $t = \sqrt{|S|}/8$, we have

$$\mathbb{P}\left(\sum_{i\in S}\epsilon_i x_i \ge \frac{1}{2}\sum_{i\in S}\epsilon_i - \frac{1}{8}\sqrt{|S|\sum_{i\in S}\epsilon_i^2}\right) \ge 1 - e^{-|S|/128}$$

Note that with $\epsilon = \sqrt{\sum_{i\in S}\epsilon_i^2}$ and $0 \le \epsilon_i \le 2\epsilon/\sqrt{|S|}$ for all $i \in S$, we have

$$\sum_{i\in S}\epsilon_i \ge \frac{\sqrt{|S|}}{2}\epsilon.$$

Consequently, we have

$$\sqrt{\frac{1}{64}|S|\sum_{i\in S}\epsilon_i^2} \le \frac{1}{2}\sum_{i\in S}\epsilon_i - \frac{1}{8}\sqrt{|S|\sum_{i\in S}\epsilon_i^2}$$

$$\begin{aligned}
\mathbb{P}\left(\theta\left(\boldsymbol{x}, V\right) \ge \frac{1}{16}\sqrt{\sum_{i\in S}\epsilon_i^2}\right) \ge& \mathbb{P}\left(\left(\sum_{i\in S}\epsilon_i b_i\right)^2 \ge \frac{1}{64}|S|\sum_{i\in S}\epsilon_i^2\right) \\
\ge& \mathbb{P}\left(\sum_{i\in S}\epsilon_i b_i \ge \frac{1}{2}\sum_{i\in S}\epsilon_i - \frac{1}{8}\sqrt{|S|\sum_{i\in S}\epsilon_i^2}\right) \\
\ge& 1 - e^{-|S|/128}
\end{aligned}$$

□

Lemma 9. *For $b_1, \cdots, b_k \ge 0$, with $\bar{b} = \sqrt{\sum_{i=1}^k b_i^2/k}$, let $b_i \ge \bar{b}/C$ for some constant $C > 1$. Then there exists a subset $S \subset [k]$ with $|S| \ge k(1-p)$ s.t. for all $i \in S$, we have*

$$b_i \le \sqrt{\frac{1}{p}\ln\frac{C^2}{1-p}}\sqrt{\frac{\sum_{i\in S}b_i^2}{|S|}}.$$

Proof. Let $S_1 = \{1, \cdots, k\}$. Choose a constant $\gamma > 1$. We repeat the following procedure. For the j-th step, we are given a set $\{x_i, i \in S_j\}$, Let $S_{j+1} = \{i \in S, b_i \leq \gamma\sqrt{\sum_{i\in S_j} b_i^2/|S_j|}\}$. The algorithm terminates when $S_j = S_{j+1}$. Denote $p_j = 1 - |S_{j+1}|/|S_j|$. For the j-th step, we have

$$\sum_{i\in S_j} b_i^2 = \sum_{i\in S_{j+1}} b_i^2 + \sum_{i\in S_j\setminus S_{j+1}} b_i^2 \geq \sum_{i\in S_{j+1}} b_i^2 + p_j\gamma^2 \sum_{i\in S_j} b_i^2$$

This derives that

$$\sum_{i\in S_{j+1}} b_i^2 \leq (1 - p_j\gamma^2) \sum_{i\in S_j} b_i^2 \leq e^{-p_j\gamma^2} \sum_{i\in S_j} b_i^2$$

Accumulating all J steps, we have

$$\sum_{i\in S_J} b_i^2 \leq e^{-\gamma^2 \sum_{j=1}^{J-1} p_j} \sum_{i\in S_1} b_i^2 = e^{-\gamma^2 \sum_{j=1}^{J-1} p_j} k\bar{b}^2$$

From the condition that $b_i \geq \bar{b}/C$, we have

$$|S_J|\frac{\bar{b}^2}{C^2} \leq \sum_{i\in S_J} b_i^2 \leq e^{-\gamma^2 \sum_{j=1}^{J-1} p_j} k\bar{b}^2$$

From the definition of p_j, we know that

$$|S_J| = k\prod_{j=1}^{J-1}(1 - p_j) \geq k\left(1 - \sum_{j=1}^{J-1} p_j\right)$$

Denote $p = \sum_{j=1}^{J-1} p_j$, then we have

$$1 - p \leq e^{-\gamma^2 p} C^2$$

Choose $\gamma = \sqrt{\frac{1}{p}\ln\frac{C^2}{1-p}}$, then we get a set S_J with $|S_j| \geq k(1-p)$ satisfying for all $i \in S_J$,

$$b_i \leq \sqrt{\frac{1}{p}\ln\frac{C^2}{1-p}}\sqrt{\frac{\sum_{i\in S_J} b_i^2}{|S_j|}}$$

□

We now restate and prove Theorem 2 as follows.

Theorem 2 (Lower Bound)**.** *Suppose that a lifelong learner has black-box access to a single task learner that takes an error parameter ϵ as input and is allowed to return any vector that is within distance ϵ of the true target unit vector, using $\Theta(d/\epsilon)$ samples in $\mathbb{R}^d$. Then, there exists a distribution of m tasks, $m = 2^{\Theta(k)}$ such that for any lifelong learning algorithm, WHP, the total number of samples required to learn all m tasks up to error ϵ is $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$.*

Proof. Denote the underlying feature subspace as U. Consider a sequence of tasks with first k tasks the basis of the feature subspace, *i.e.*, $\boldsymbol{a}_i = \boldsymbol{e}_i, 1 \leq i \leq k$. The lifelong learning algorithm learns task i up to error ϵ_i, and get k features $\hat{\boldsymbol{a}}_1, \cdots, \hat{\boldsymbol{a}}_k$. If the number of tasks for which $\epsilon_i < \sqrt{2\sum_{i=1}^k \epsilon_i^2/3k}$ is more than $k/4$, then we have that the total sample complexity is already $\Omega(dk^{1.5}/\sqrt{\sum_{i=1}^k \epsilon_i^2})$ and the theorem follows. So we assume that for at least $3k/4$ tasks, we have

$$\epsilon_i \geq \sqrt{2\sum_{i=1}^k \epsilon_i^2/3k}.$$

Calling this subset S_1, it follows that for each $i \in S_1$, we have $\epsilon_i \geq \sqrt{\sum_{i\in S_1} \epsilon_i^2/2|S_1|}$. Next, applying Lemma 9 to the set S_1 with cardinality at least $3k/4$, using $p = 1/3$ and $C = \sqrt{2}$, we get that there exists a set $S \subseteq S_1$

with $|S| \geq k/2$ and $\epsilon_i \leq 2\sqrt{\frac{\sum_{i\in S}\epsilon_i^2}{|S|}}$ for all $i \in S$. Consider the span $V := \{\hat{\boldsymbol{a}}_i, i \in S\}$. By Lemma 7, we know there exists feasible $\hat{\boldsymbol{a}}_i$ such that $\theta(U, V) = \Omega(\sqrt{\sum_{i\in S}\epsilon_i^2})$.

Next we consider the following tasks as $\boldsymbol{a}_j = \sum_{i\in S} x_{ji}\boldsymbol{e}_i, j \geq k+1$, where $x_{ji} \overset{\text{i.i.d}}{\sim} \text{Bernoulli}(1/2)$. There are $2^{k/2}$ such tasks. By Lemma 8, we know that with high probability each new task is far from the learned subspace, *i.e.*,

$$\mathbb{P}\left(\theta\left(\boldsymbol{a}_j, V\right) \geq \frac{1}{16}\sqrt{\sum_{i\in S}\epsilon_i^2}\right) \geq 1 - e^{-k/256}$$

Assume that the single-task learning algorithm applied to this new task also induce error in the $(k+1)$'st coordinate. Say the new learned feature $\hat{\boldsymbol{a}}_j = (\boldsymbol{a}_j^\top, \sum_{i\in S} b_{ji}\epsilon_i)^\top$. Then the new learned feature is in the learned subspace V, which means that learning new tasks does not improve the learned subspace.

Since each new task is generated randomly, there are exponentially many new tasks that are far from the learned subspace but make no improvement by learning them. Therefore to ensure each task is learned with error ϵ, the only way is to let $\theta(U, V) \leq \epsilon$. This implies that $\sum_{i\in S}\epsilon_i^2 \leq c\epsilon^2$. By the generalized Hölder inequality [Finner, 1992], we have

$$\sum_{i\in S}\frac{1}{\epsilon_i}\sum_{i\in S}\frac{1}{\epsilon_i}\sum_{i\in S}\epsilon_i^2 \geq \left(\sum_{i\in S}1\right)^3 \geq \left(\frac{k}{2}\right)^3$$

So $\sum_{i\in k}\frac{1}{\epsilon_i} \geq c'k^{1.5}/\epsilon$. The number of the samples needed to learn the tasks in set S is $\sum_{i\in S}\frac{d}{\epsilon_i} = \Omega(dk^{1.5}/\epsilon)$. So the overall sample complexity for the sequence of tasks are $\Omega(dk^{1.5}/\epsilon + km/\epsilon)$. □

C Formal Algorithm of LLL-RR

For completeness, we describe the formal algorithm of LLL-RR. Different from the basic LLL algorithm, we are memorizing a list $\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_{\hat{k}k_0}$ all along with the algorithm. Each time when we need to learn the new features $\tilde{\boldsymbol{w}}_{\hat{k}k_0+1}, \cdots, \tilde{\boldsymbol{w}}_{\hat{k}k_0+k_0}$, we add them to the list, and feed the list to Algorithm 2 to get a new feature subspace. The formal algorithm is in Algorithm 4.

Algorithm 4 Lifelong Learning Algorithm with Representation Refinement (LLL-RR)

Input: d, m, k, labeled examples of m tasks , threshold parameters ϵ_{acc}, ϵ.

1. Using data from the first task to learn a set of features $\tilde{\boldsymbol{W}}_1(\cdot) = (\tilde{\boldsymbol{w}}_1(\cdot), \cdots, \tilde{\boldsymbol{w}}_{k_0}(\cdot))^\top$ and a linear function $\tilde{\boldsymbol{c}}_1$ such that $\boldsymbol{x} \to \text{sign}(\tilde{\boldsymbol{c}}_1^\top \tilde{\boldsymbol{W}}_1(\boldsymbol{x}))$ has error smaller than ϵ_{acc}.
   `/* Number of features` $1 \leq k_0 \leq k$. `For linear features,` $k_0 = 1$. `*/`
   Let $\tilde{k} = 1$. Set the feature subspace $\boldsymbol{V}_1 = \tilde{\boldsymbol{W}}_1$, and the temporary features $\tilde{\boldsymbol{V}}_1 = \tilde{\boldsymbol{W}}_1$.
2. For the task $i = 2, \cdots, m$
   - Using the data from the i task, attempt to learn the linear function $\tilde{\boldsymbol{c}}_i$ using the temporary features $\tilde{\boldsymbol{V}}_{i-1}$.
   - Check whether $\boldsymbol{x} \to \text{sign}(\tilde{\boldsymbol{c}}_i^\top \tilde{\boldsymbol{V}}_{i-1}(\boldsymbol{x}))$ has error less than ϵ.
     (a) If yes, set $\tilde{\boldsymbol{V}}_i = \tilde{\boldsymbol{V}}_{i-1}$. `// Small error with current features.`
     (b) Otherwise, learn a new set of features $\tilde{\boldsymbol{W}}_i(\cdot)$ and a linear function $\tilde{\boldsymbol{c}}_i$ such that the predictor $\boldsymbol{x} \to \text{sign}(\tilde{\boldsymbol{c}}_i^\top \tilde{\boldsymbol{W}}_i(\boldsymbol{x}))$ has error less than ϵ_{acc}.
     Update the feature subspace $\boldsymbol{V}_i = (\boldsymbol{V}_i; \tilde{\boldsymbol{W}}_i)$, and feed into Algorithm 2. It returns the refined subspace $\boldsymbol{V}'$. Set the temporary features $\tilde{\boldsymbol{V}}_i = \boldsymbol{V}'$. Let $\hat{k} = \hat{k} + 1$.

return m *predictors:* $\boldsymbol{x} \to \text{sign}(\tilde{\boldsymbol{c}}_i^\top \tilde{\boldsymbol{V}}_i(\boldsymbol{x}))$, $1 \leq i \leq m$.

D Extensions to Task-Incremental Regression and Class-Incremental Learning

In our main text, we study the setting of solving m tasks of binary classification incrementally. The classification error is defined as $err(\hat{l}) = \mathbb{P}_{(\boldsymbol{x},y)\sim P}[\hat{l}(\boldsymbol{x}) \neq y]$. By Assumption 1, we know the task error is small if and only

if the parameters are close to each other. Now we would like to extend to task-incremental regression and class-incremental classification by connecting the parameter's l_2 distance to the error of the model.

Task-incremental regression. Consider the regression tasks shared with low-dimensional common features. For $i \in [m], y = \langle \boldsymbol{c}_i^*, \boldsymbol{\sigma}^*(\boldsymbol{x})\rangle + \epsilon_i$. The regression error is $err(\hat{l}) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\|\hat{l}(\boldsymbol{x}) - y\|_2^2]$. We can further weaken our assumption to $c_1 I \preccurlyeq \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] \preccurlyeq c_2 I$ for $0 < c_1 < c_2$. Then for any unit vector $\boldsymbol{u}, \boldsymbol{v}$, we have

$$\mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\left(\boldsymbol{u}^\top \boldsymbol{x} - \boldsymbol{v}^\top \boldsymbol{x}\right)^2\right] = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[(\boldsymbol{u}-\boldsymbol{v})^T \boldsymbol{x}\boldsymbol{x}^T (\boldsymbol{u}-\boldsymbol{v})\right]$$

So we get a lemma similar to Assumption 1 that $c_1\|\boldsymbol{u}-\boldsymbol{v}\|^2 \le \mathbb{E}_{\boldsymbol{x}\in D}\|\boldsymbol{u}^\top\boldsymbol{x} - \boldsymbol{v}^\top\boldsymbol{x}\|^2 \le c_2\|\boldsymbol{u}-\boldsymbol{v}\|^2$, which is sufficient for analysis.

Class-incremental classification. Let $X = \mathbb{R}^d$ be the input space and $Y = \{1, 2, \cdots, m\}$ be the class labels. We assume that the labels can be recovered by passing the input through a linear/nonlinear layer and then taking the maximum of m linear combinations. Formally, the label is given by

$$\ell(\boldsymbol{x}) = \text{argmax}_{i\in[m]}\langle \boldsymbol{c}_i^*, \boldsymbol{\sigma}^*(\boldsymbol{x})\rangle$$

The classification error is $err(\hat{l}) = \mathbb{P}_{(\boldsymbol{x},y)\sim P}[\hat{l}(\boldsymbol{x}) \neq y]$. Noticing that, when we meet a new class, the classifier should determine whether the label belongs to the current class or not. In this sense, we regard the problem as a binary classification. To get the negative samples in the current class, we also need a small proportional of data from previous classes. Practically, we propose the heuristic lifelong learning (H-LLL) algorithm in Section 2.3 to solve the class-incremental learning. Experiments in Section 4.2 complement our results.

E Another Approach – Theoretical Guarantees for LLL

Here we give a simpler analysis for the basic LLL algorithm, along the lines of [Balcan et al., 2015]. The result is weaker than Theorem 3, but we include the proof here for completeness, along with an extension to nonlinear features.

Theorem 7 (Basic LLL). *Let $\gamma = c\epsilon$ and ϵ_{acc} s.t. $4k\frac{\epsilon_{acc}}{\gamma} + \gamma = c'\epsilon$ for sufficiently small constants $c, c' > 0$. Assume that all targets share k common features. Then, under Assumption 1, and sequential presentation of the tasks in any order, the basic LLL algorithm will incrementally learn a representation of dimension k for linear features and k^2 for nonlinear features with error at most ϵ on all tasks. The total number of samples used by the algorithm is $O(dk^2\log(k/\epsilon)/\epsilon^2 + km\log(1/\epsilon)/\epsilon) = \tilde{O}(dk^2/\epsilon^2 + mk/\epsilon)$ in the linear setting and a factor of k higher in the nonlinear setting.*

Before we prove the theorem, we define the γ-*separated* term. We use the definition γ-*separated* from [Balcan et al., 2015] that a subsequence of vectors $\boldsymbol{a}_{i_1}, \boldsymbol{a}_{i_2}, \cdots$ is γ-separated if for any $\boldsymbol{a}_{i_j}$, $\theta(\boldsymbol{a}_{i_j}, \text{span}(\boldsymbol{a}_{i_1}, \cdots, \boldsymbol{a}_{i_{j-1}})) \ge \gamma$. Define the γ-effective dimension of $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m$ as the size of the largest γ-separated subsequence. Note that when $\gamma = 0$, γ-effective dimension is exactly the dimension of the spanned subspace. We prove Theorem 7 by showing two facts: each target is far from the span of previous ones; we learn the new target accurately. (Lemma 11). We start with a helper lemma (Lemma 10).

Lemma 10. *Let $\boldsymbol{w}, \boldsymbol{v}$ be two unit vectors in $\mathbb{R}^d$ and $\boldsymbol{U}$ be a subspace. Then,*

$$\sin\theta(\text{span}(\boldsymbol{U},\boldsymbol{w}), \text{span}(\boldsymbol{U},\boldsymbol{v})) \le \frac{\sin\theta(\boldsymbol{w},\boldsymbol{v})}{\max\{\sin\theta(\boldsymbol{w},\boldsymbol{U}), \sin\theta(\boldsymbol{v},\boldsymbol{U})\}}.$$

Proof. If $\boldsymbol{w} \in \boldsymbol{U}$ or $\boldsymbol{v} \in \boldsymbol{U}$, $\theta(\text{span}(\boldsymbol{U},\boldsymbol{w}), \text{span}(\boldsymbol{U},\boldsymbol{v})) = 0$. The inequality becomes trivial. Now we assume that $\boldsymbol{w}, \boldsymbol{v} \notin \boldsymbol{U}$. From the symmetry of $\boldsymbol{w}$ and $\boldsymbol{v}$, we prove the following and then replacing $\boldsymbol{v}$ with $\boldsymbol{w}$ leads to the original inequality.

$$\sin\theta(\text{span}(\boldsymbol{U},\boldsymbol{w}), \text{span}(\boldsymbol{U},\boldsymbol{v})) \le \frac{\sin\theta(\boldsymbol{w},\boldsymbol{v})}{\sin\theta(\boldsymbol{v},\boldsymbol{U})}$$

By definition, $\exists \boldsymbol{x} \in \text{span}(\boldsymbol{U},\boldsymbol{w})$ s.t. $\theta(\boldsymbol{x}, \text{span}(\boldsymbol{U},\boldsymbol{v})) = \theta(\text{span}(\boldsymbol{U},\boldsymbol{w}), \text{span}(\boldsymbol{U},\boldsymbol{v}))$. Here $\boldsymbol{x}$ is a combination of $\boldsymbol{u}_1$ and $\boldsymbol{w}$, $\boldsymbol{u}_1$ is some vector in $\boldsymbol{U}$. Using the fact that $\theta(\boldsymbol{x}, \text{span}(\boldsymbol{U},\boldsymbol{v})) \le \theta(\boldsymbol{x}, \text{span}(\boldsymbol{u}_1,\boldsymbol{v})) \le$

$\theta(\text{span}(\boldsymbol{u}_1, \boldsymbol{w}), \text{span}(\boldsymbol{u}_1, \boldsymbol{v}))$ and $\theta(\boldsymbol{v}, \boldsymbol{U}) \leq \theta(\boldsymbol{v}, \boldsymbol{u}_1)$, it is sufficient to prove that

$$\sin\theta(\text{span}(\boldsymbol{u}_1, \boldsymbol{w}), \text{span}(\boldsymbol{u}_1, \boldsymbol{v})) \leq \frac{\sin\theta(\boldsymbol{w}, \boldsymbol{v})}{\sin\theta(\boldsymbol{v}, \boldsymbol{u}_1)}$$

Denote $\alpha = \theta(\text{span}(\boldsymbol{u}_1, \boldsymbol{w}), \text{span}(\boldsymbol{u}_1, \boldsymbol{v}))$, $\beta = \theta(\boldsymbol{v}, \boldsymbol{u}_1)$. WLOG we assume $\boldsymbol{u}_1 = (1, 0, 0)$ and $\text{span}(\boldsymbol{u}_1, \boldsymbol{w})$ is the x-y plane. Then we can write $\boldsymbol{v} = \cos(\beta)\boldsymbol{u}_1 + \sin(\beta)\boldsymbol{v}_1$, where $\boldsymbol{v}_1 = (0, \cos(\alpha), \sin(\alpha))$. Since $\sin\theta(\boldsymbol{w}, \boldsymbol{v}) \geq d(\boldsymbol{v}, x\text{-}y \text{ plane}) = \sin(\alpha)\sin(\beta)$, we get the lemma proved.

□

Lemma 11 (Kernel Subspace). *Let $\boldsymbol{U}_k, \boldsymbol{V}_k$ be two subspaces of $\mathbb{R}^d$. Let $\boldsymbol{U}_k = \text{span}\{\boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_k^*\}$, $\boldsymbol{V}_k = \text{span}\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$. Let $\epsilon, \gamma \geq 0$ and $\epsilon \leq \gamma^2/(10k)$. Assume that*

1. $\sin\theta(\boldsymbol{y}_i, \text{span}\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1}\}) \geq \gamma$, *for* $i = 2, \cdots, k$.
2. $\sin\theta(\boldsymbol{y}_i, \boldsymbol{y}_i^*) \leq \epsilon$, *for* $i = 1, \cdots, k$.

Then we have $\sin\theta(\boldsymbol{U}_k, \boldsymbol{V}_k) \leq 2k\epsilon/\gamma$. *In other words, for any point $\boldsymbol{y}^* \in \boldsymbol{U}_k$, there is a point $\boldsymbol{y} \in \boldsymbol{V}_k$ s.t.*

$$\sin\theta(\boldsymbol{y}^*, \boldsymbol{y}) \leq \frac{2\epsilon k}{\gamma}.$$

Proof. Here we use the strong induction on a stronger version of the conclusion where $\boldsymbol{U}_k = \text{span}\{\boldsymbol{W}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_k^*\}$, $\boldsymbol{V}_k = \text{span}\{\boldsymbol{W}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$ for some fixed subspace $\boldsymbol{W}$. The base case is $k = 1$. This follows directly from Lemma 10 with $\boldsymbol{U} = \boldsymbol{W}, \boldsymbol{w} = \boldsymbol{y}, \boldsymbol{v} = \boldsymbol{y}^*$. Now we prove the induction step on k with strong hypothesis. Let $\boldsymbol{U}_k' = \text{span}(\boldsymbol{U}_{k-1}, \boldsymbol{y}_k)$. By Lemma 10 and induction hypothesis, we have

$$\sin\theta(\boldsymbol{U}_k, \boldsymbol{V}_k) \leq \sin\theta(\boldsymbol{U}_k, \boldsymbol{U}_k') + \sin\theta(\boldsymbol{U}_k', \boldsymbol{V}_k) \leq \frac{\sin\theta(\boldsymbol{y}_k, \boldsymbol{y}_k^*)}{\sin\theta(\boldsymbol{y}_k, \boldsymbol{U}_{k-1})} + \frac{2(k-1)\epsilon}{\gamma}$$

By triangle inequality and induction hypothesis, we further have

$$\sin\theta(\boldsymbol{y}_k, \boldsymbol{U}_{k-1}) \geq \sin\theta(\boldsymbol{y}_k, \boldsymbol{V}_{k-1}) - \sin\theta(\boldsymbol{V}_{k-1}, \boldsymbol{U}_{k-1}) \geq \gamma - \frac{2\epsilon(k-1)}{\gamma}$$

Combining the two inequalities, we get

$$\sin\theta(\boldsymbol{U}_k, \boldsymbol{V}_k) \leq \frac{\epsilon}{\gamma - \frac{2\epsilon(k-1)}{\gamma}} + \frac{2(k-1)\epsilon}{\gamma} = \frac{\epsilon}{\gamma}\left(\frac{\gamma^2}{\gamma^2 - 2(k-1)\epsilon} + 2(k-1)\right) \leq \frac{2k\epsilon}{\gamma}$$

□

Now we put them together to analyze Algorithm 1.

Proof. We consider the kernel of nonlinear features $\boldsymbol{\sigma}(\boldsymbol{x})$. These features live in a potentially infinite-dimensional space (or exponential in d dimensional space if, e.g., the input is from the Boolean hypercube). Let $\boldsymbol{U}$ be the span of the nonlinear features (viewed as vectors) in the model used to label data.The γ-effective dimension of $\boldsymbol{U}$ is at most k. Let $y_i^* = \boldsymbol{c}_i^{*\top}\boldsymbol{\sigma}^*(\boldsymbol{x}) = \boldsymbol{a}_i^*(\boldsymbol{x}), y_i = \boldsymbol{c}_i^\top\boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{a}_i(\boldsymbol{x})$. WLOG let's assume $\boldsymbol{a}_i, \boldsymbol{a}_i^*$ be vectors of unit length. From the algorithm, if the current task i has already achieved ϵ error by current features, it's done. Otherwise, we learn a new set of k_0 features and a linear combination whose error is at most ϵ_{acc}. Denote the indices of tasks that we learn new features as $i_1, i_2, \cdots, i_{\tilde{k}}$. Encountering the task $i_{\hat{k}}$, denote $\boldsymbol{V}_{\hat{k}} = \text{span}(\boldsymbol{a}_{i_1}, \cdots, \boldsymbol{a}_{i_{\hat{k}}}), \boldsymbol{U}_{\hat{k}} = \text{span}(\boldsymbol{a}_{i_1}^*, \cdots, \boldsymbol{a}_{i_{\hat{k}}})$. We will prove by induction that for any $\hat{k} \in [\tilde{k}]$, (1) $\theta(\boldsymbol{a}_{i_{\hat{k}}}, \boldsymbol{V}_{\hat{k}-1}) \geq \gamma$; (2) $\theta(\boldsymbol{a}_{i_{\hat{k}}}^*, \boldsymbol{U}_{\hat{k}-1}) \geq \gamma$.

The base case $\hat{k} = 1$ holds immediately. For the inductive step $\hat{k} > 1$, the task $i_{\hat{k}}$ cannot achieve ϵ error with the current features $\boldsymbol{V}_{\hat{k}-1}$. By Assumption 1, $\theta(\boldsymbol{a}_{i_{\hat{k}}}^*, \boldsymbol{V}_{\hat{k}-1}) \geq \epsilon/c_2$. After learning a new set of features to ensure error less than ϵ_{acc}, we know there is a new linear combination $\boldsymbol{a}_{i_{\hat{k}}}$ such that $\theta(\boldsymbol{a}_{i_{\hat{k}}}^*, \boldsymbol{a}_{i_{\hat{k}}}) \leq \epsilon_{acc}/c_1$. So by triangle inequality,

$$\theta\left(\boldsymbol{a}_{i_{\hat{k}}}, \boldsymbol{V}_{\hat{k}-1}\right) \geq \epsilon/c_2 - \epsilon_{acc}/c_1 \geq \gamma$$

So we have shown that (1) holds for $i_{\hat{k}}$.

To prove (2), we suppose for contradiction that $\theta(\boldsymbol{a}^*_{i_{\hat{k}}}, \boldsymbol{U}_{\hat{k}-1}) < \gamma$. From induction hypothesis, for any $j \in [\hat{k}-1]$, $\sin\theta(\boldsymbol{a}_{i_j}, \boldsymbol{V}_{j-1}) \geq \gamma/2$. By construction, we also have for any $\sin\theta(\boldsymbol{a}_{i_j}, \boldsymbol{a}^*_{i_j}) \leq \epsilon_{acc}/c_1$. Apply Lemma 11, we have $\theta(\boldsymbol{U}_{\hat{k}-1}, \boldsymbol{V}_{\hat{k}-1}) \leq 8\epsilon_{acc}k/\gamma$. By triangle inequality, we further have

$$\theta\left(\boldsymbol{a}^*_{i_{\hat{k}}}, \boldsymbol{V}_{\hat{k}-1}\right) \leq \theta\left(\boldsymbol{a}^*_{i_{\hat{k}}}, \boldsymbol{U}_{\hat{k}-1}\right) + \theta\left(\boldsymbol{U}_{\hat{k}-1}, \boldsymbol{V}_{\hat{k}-1}\right) \leq \gamma + 4\epsilon_{acc}k/\left(c_1\gamma\right) \leq \epsilon/c_2$$

By Assumption 1, there exists $\boldsymbol{b}_{i_{\hat{k}}} \in \boldsymbol{V}_{\hat{k}-1}$ with error less than ϵ, and thus leads to contradiction. So (2) is also proved. Furthermore, since we have assume that the γ-effective dimension of the true targets is at most k, we have $\tilde{k} \leq k$. So the size of the internal representation is $k' = O(kk_0)$.

The sample complexity for learning one task in d-dimension up to error ϵ is $O(dk_0\log(1/\epsilon)/\epsilon)$. Here we learn $O(k)$ such tasks. All other tasks can be learned using the features of dimension $O(kk_0)$. Therefore the total sample complexity is $O(dkk_0/\epsilon_{acc}\log(1/\epsilon_{acc}) + kk_0m\log(1/\epsilon)/\epsilon) = \tilde{O}(dk^2k_0/\epsilon^2 + kk_0m/\epsilon)$.

□