Proceedings of Machine Learning Research 76:1–38, 2017 Algorithmic Learning Theory 2017

Lifelong Learning in Costly Feature Spaces

Maria-Florina Balcan NINAMF@CS.CMU.EDU
Carnegie Mellon University

Avrim Blum AVRIM@TTIC.EDU
TTI-Chicago

Vaishnavh Nagarajan VAISHNAVH@CS.CMU.EDU
Carnegie Mellon University

Editors: Steve Hanneke and Lev Reyzin

Abstract

An important long-term goal in machine learning systems is to build learning agents that, like humans, can learn many tasks over their lifetime, and moreover use information from these tasks to improve their ability to do so efficiently. In this work, our goal is to provide new theoretical insights into the potential of this paradigm. In particular, we propose a lifelong learning framework that adheres to a novel notion of resource efficiency that is critical in many real-world domains where feature evaluations are costly. That is, our learner aims to reuse information from previously learned related tasks to learn future tasks in a *feature-efficient* manner. Furthermore, we consider novel combinatorial ways in which learning tasks can relate. Specifically, we design lifelong learning algorithms for two structurally different and widely used families of target functions: decision trees/lists and monomials/polynomials. We also provide strong feature-efficiency guarantees for these algorithms; in fact, we show that in order to learn future targets, we need only slightly more feature evaluations per training example than what is needed to predict on an arbitrary example using those targets. We also provide algorithms with guarantees in an agnostic model where not all the targets are related to each other. Finally, we also provide lower bounds on the performance of a lifelong learner in these models, which are in fact tight under some conditions.

Keywords: lifelong learning; costly features; representation learning

1. Introduction

Machine learning algorithms have found widespread use in solving naturally occurring tasks in domains like medical diagnosis, autonomous navigation and document classification. Accompanying this rapid growth, there has been remarkable progress in theoretically understanding how machine learning can solve single tasks in isolation. However, real-world tasks rarely occur in isolation. For example, an autonomous robot may have to accomplish a series of control learning tasks during its life, and to do so well it should employ methods that improve its ability to learn as it does so, needing less resources as it learns more (Thrun and Pratt, 1997; Thrun and Mitchell, 1995). As we scale up our goals from learning a single function to learning a stream of many functions, we need to develop sound theoretical foundations to analyze these large-scale learning settings.

Broadly, the goal of a *lifelong learner* is to solve a series of many tasks over its lifetime by a) extracting succinct and useful representations about the relations among previously learned tasks, and then b) using these representations to learn future tasks more efficiently. In this work, we provide new insights into this paradigm by first proposing a metric for lifelong learning that exposes

an important type of resource efficiency gain. Then we design algorithms for important and widely used classes of functions with strong theoretical guarantees in this metric.

In particular, we consider a setting where evaluating the features of data points is costly and hence the learner wishes to exploit task relations to improve its *feature-efficiency* over time. Feature-efficiency is critical in applications such as medical diagnosis and high-dimensional data domains where evaluating feature values of a data point might involve performing expensive or intrusive medical tests or accessing millions of values. In fact, one of the reasons decision trees (which is one of the important function classes we study in this paper) are commonly used in medical diagnosis (Podgorelec et al., 2002) is that once the trees are learned, one can then make *predictions* on new examples by evaluating very few features—at most the depth of the tree. We consider lifelong learning from the perspective of this feature evaluation cost, and show how we can use commonalities among previously-learned target functions to perform much better in learning new related targets according to this cost. Specifically, if we face a stream of m *adversarially chosen* related learning tasks over the same set of N features, each with about S training examples, we will make $O(SmN)$ feature evaluations if we learn each task from scratch individually. Our goal will be to leverage task relatedness to learn very few tasks from scratch and learn the rest in a feature-efficient manner, making as few as $O(S(m+N))$ feature evaluations in total.

We study two structurally different classes of target functions. In Section 3 we focus on decision trees (and lists) which are a widely used class of target functions (Wu et al., 2008; Rokach and Maimon, 2008; Quinlan, 1986; Breiman et al., 1984) popular because of their naturally interpretable structure – to make a prediction one has to simply make a sequence of feature evaluations – and their usefulness in the context of prediction in costly feature spaces. In Section 4 we analyze monomial and polynomial functions, an expressive family that can approximate many realistic functions (e.g., Lipschitz functions (Andoni et al., 2014)) and is relevant in common machine learning techniques like polynomial regression, curve fitting and basis expansion (Ziegel, 2003). Our study of polynomials also demonstrates how feature-efficient learning is possible even when the function class is not intrinsically feature-efficient for prediction. The non-linear structure of both of these function classes poses interesting technical challenges in modeling their relations and proposing feature-efficient solution strategies. Indeed our algorithms will use their learned information to determine an adaptive feature-querying strategy that significantly minimizes feature evaluations.

In Section 3, we present our results for decision trees and lists. First, we describe intuitive relations among our targets in terms of a small *unknown* set of K "metafeatures" or parts of functions common to all targets (think of K much less than N). More specifically, these metafeatures are subtrees that can be combined sequentially to represent the target tree. We then present our feature-efficient lifelong learning protocol which involves addressing two key challenges. First, we need a computationally-efficient strategy that can recover useful metafeatures from previously learned targets (Algorithm 2). Interestingly, we show that the learned metafeatures can be useful even if they do not exactly match the unknown K metafeatures, so long as they "contain" them in an appropriate sense. Second, we need a feature-efficient strategy that can learn new target functions using these learned metafeatures (Algorithm 1). Making use of these two powerful routines, we present a lifelong learning protocol that learns only at most K out of m targets from scratch and for the remaining targets examines only Kd features per example (where d is the depth of the targets), thus making $O(S(NK+mKd))$ feature evaluations in total (Theorem 2).

In Section 4, we study monomials and polynomials which are similarly related through K unknown metafeatures. We adopt a natural model where the metafeatures are monomials themselves,

so that the monomial targets are simply products of metafeatures. In the case of polynomials, this defines a two-level relation, where each polynomial is a sum of products of metafeatures. For polynomials, we present an algorithm that learns only K of m targets from scratch and on the remaining targets, evaluate s$O(K+d)$ features per example (where d is the degree of the target), thus making only $O(S(KN+m(K+d)))$ feature evaluations over all tasks. More interestingly, in the case of large-degree monomials, our algorithm may need fewer feature evaluations per example (K) to learn the monomial than that needed (d) to evaluate the monomial on an input point.

Next in Section 5, we consider a relaxation of the original model, more specifically, an agnostic case where the learner faces $m+r$ targets, r of which are "bad" targets adversarially chosen to be unrelated to the other m interrelated "good" targets. As a natural goal, we want the learner to minimize the feature evaluations made on the training data of the m good targets. We show that when r is not too large, the above lifelong learners can be easily made to work as well as they would when $r=0$. To address greater values of r, we first highlight a trade-off between allowing the learner to learn more targets from scratch and learning the remaining targets with more feature evaluations. We then present a technique that strikes the right balance between the two.

Finally, in Section 6 we present lower bounds on the performance of a lifelong learner for all values of r, including $r=0$ by designing randomized adversaries. Ignoring the sample size S and other problem-specific parameters, for small r we prove a lower bound of $\Omega(KN+mK)$ feature evaluations which proves that our above approaches are in fact tight. For sufficiently large r, we prove a bound of $\Omega(mN)$, thereby demarcating a realm of r where lifelong learning is simply futile.

We present a summary of our results in Tables 1 and 2 in Appendix A.

1.1. Related Work

Related work in multi-task or transfer learning (Kumar and III, 2012; Maurer and Pontil, 2013; Pan and Yang, 2010) considers the case where tasks are drawn from an easily learnable distribution or are presented to the learner all at once. The theoretical results in that setting are sample complexity results that guarantee low error averaged over all tasks (Baxter, 1997, 2000). On the other hand, research in lifelong learning has been mostly empirical (Thrun and Mitchell, 1995; Koenig et al., 2004; Drachsler et al., 2008; Thrun and Pratt, 1997). There has been a small amount of recent theoretical work (Balcan et al., 2015; Pentina and Urner, 2016). Balcan et al. (2015) consider fairly simple targets and commonalities such as linear separators that lie in a common low-dimensional subspace. Pentina and Urner (2016) consider a setting where except for a small subset of target functions, each target can be written as a weighted majority vote over the previous ones. Balcan et al. (2015) also consider conjunctions that share a set of conjunctive metafeatures, but assume that the metafeatures contain a unique "anchor variable". Though decision trees have a more elaborate combinatorial structure than conjunctions, in this work we are able to achieve strong guarantees for lifelong learning of decision trees (and other classes) without making such unrealistic assumptions about the metafeatures. We also note that one of main technical challenges addressed by Balcan et al. (2015) is that of controlling error propagation during lifelong learning. However, for the problems considered in this paper, it is possible to learn targets exactly from scratch, so we do not have to deal with error propagation.

Feature-efficiency has been considered in the single-task setting, often under the name of budgeted learning (Lizotte et al., 2003; Kapoor and Greiner, 2005; BLw, 2010), where one has to learn an accurate model subject to a limit on feature evaluations, somewhat like bandit algorithms.

Obozinski and Taskar (2006); Argyriou et al. (2006) consider a related problem in a multi-task setting with all tasks present up-front, where the learner has free access to all features but uses commonalities between targets to identify useful common features in order to be sample-efficient.

2. Preliminaries

In this section, we define our notations and present a high level protocol which will provide a framework for presenting our algorithms in the later sections. We consider a setting in which the learner faces a sequence of m related target functions $g^{(j)}$ over the same set of N features/variables (where both m and N are very large). The target functions arrive one after the other, each with its own set of training data $\mathcal{S}^{(j)}$ with at most S examples to learn from. Also, feature evaluation (or equivalently, feature query or feature examination) is costly: if we view our training data for $g^{(j)}$ as an $S \times N$ matrix, we pay a cost of 1 for each cell probed in the matrix.

Our belief is that the targets are related to each other through an unknown set $\mathcal{F}$ of *metafeatures* that are parts of functions. More specifically, all targets in the series can be expressed by combining metafeatures in $\mathcal{F}$ using a known set of legal combination rules, such as concatenating lists or trees. Our algorithms will learn a set of hypothesized metafeatures $\tilde{\mathcal{F}}$ that allows them to learn new targets using a small number of feature evaluations except for a limited number of targets learned from scratch i.e., by examining all features on all examples. We call $\tilde{\mathcal{F}}$ our *learned representation*. Note that we will refer to $\tilde{\mathcal{F}}$ as just metafeatures if it is clear from context that it does not refer to the true metafeatures $\mathcal{F}$.

Then, our lifelong protocol is as follows. We make use of two basic subroutines: a UseRep routine that uses $\tilde{\mathcal{F}}$ to learn new related targets, and an ImproveRep routine that improves our representation $\tilde{\mathcal{F}}$ whenever the first subroutine fails. We begin with an empty $\tilde{\mathcal{F}}$. On task j, using $\tilde{\mathcal{F}}$ and $\mathcal{S}^{(j)}$, we attempt to cheaply learn target $g^{(j)}$ with UseRep. If UseRep fails to learn the target, we evaluate all features in $\mathcal{S}^{(j)}$ and learn $g^{(j)}$ from scratch. Then, we provide $\tilde{\mathcal{F}}$ and $g^{(j)}$ as input to ImproveRep to update $\tilde{\mathcal{F}}$. For clarity, we present this generic approach, which we will call as (UseRep, ImproveRep)-protocol, in Algorithm 3 in Appendix B. In the following sections, we will present concrete approaches for these subroutines, specific to each class of targets. We will then analyze the performance of the protocol in terms of the total number of feature evaluations (across all samples over all the tasks) given an adversarial stream of tasks.

Our setting can be viewed as analogous to that of dictionary learning (Lewicki and Sejnowski, 2000; Elad and Aharon, 2006; Arora et al., 2014) in which the goal is to find a small set of vectors that can express a given set of vectors via sparse linear combinations. Here, we will be interested in broader classes of objects and richer types of combination rules.

3. Decision Trees

We first formally define decision tree metafeatures and describe our learning model. Based on this we describe our problem concretely in Problem Setup 1. To simplify our discussion, we consider decision trees over Boolean features, though we later present a simple extension to real values. Formally, in a decision tree $g : \{0,1\}^N \to \{+,-\}$, each internal node corresponds to a split over one of N variables and each leaf node corresponds to one of the two labels $\{+,-\}$. No internal node and its ancestor split on the same variable.

Now, we define a metafeature to be an *incomplete* decision tree, a tree where any of the leaf nodes can be empty i.e., the labels of some leaf nodes are left unspecified. Then, there are two natural ways of combining metafeatures to form a (complete) decision tree. Let u be one of the empty leaf nodes of a metafeature f. We may combine f with another incomplete tree f' using an $\textsf{AFFIX}(f, u, f')$ operation which simply affixes the root node of f' at u (as illustrated in Figure 1). As a result, u now becomes an internal node of a larger incomplete tree. The variable at u and its descendants correspond to the variables in f'. Alternatively, we may perform a $\textsf{LABEL}(f, u, l)$ operation which assigns a label $l \in \{+, -\}$ to the empty node u in f. We can then pick an arbitrary element $f \in \mathcal{F}$, apply an arbitrary sequence of LABEL and AFFIX operations (affixing only trees from $\mathcal{F}$) and eventually grow f into a decision tree. In this manner, we define below what it means to be able to represent a decision tree using a set of metafeatures $\mathcal{F}$. Both LABEL and AFFIX are described for completeness in Appendix C.

Definition 1 *Let $\mathcal{F} = \{f_1, f_2, \ldots\}$ where each metafeature f_i is an incomplete decision tree. We define $\mathbb{DT}(\mathcal{F})$ to be the set of all decision trees that can be grown by using the elements of $\mathcal{F}$ and sequentially applying* LABEL *and* AFFIX *operations on them. We say that a decision tree g can be expressed using $\mathcal{F}$ if $g \in \mathbb{DT}(\mathcal{F})$.*

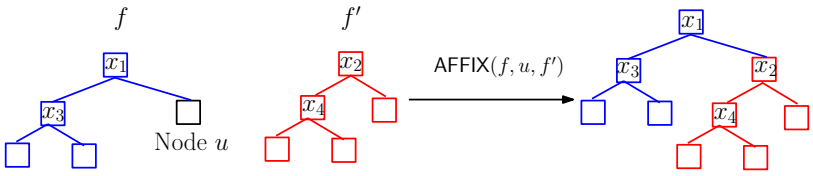

Figure 1: Illustration of AFFIX

A modeling challenge here is that there are no known polynomial-time algorithms to learn decision trees, even ignoring the issue of costly features and even for trees of depth $d = O(\log N)$. On the other hand, there are popular top-down tree-learning algorithms (like ID3 and C4.5) that work well empirically (Rokach and Maimon, 2008; Quinlan, 1986; Breiman et al., 1984). Therefore, we will assume that we are given such an algorithm that indeed correctly produces $g^{(j)}$ from $\mathcal{S}^{(j)}$ if we are willing to evaluate all the features in all the examples. More specifically, these methods are defined by a "gain function" $\texttt{Gain}(\mathcal{S}, i)$ that given a set of labeled examples $\mathcal{S}$ and a feature i, returns a score indicating the desirability of splitting the set $\mathcal{S}$ using feature i. For instance, ID3 uses *information gain* as its splitting criterion,[1] and an elegant theoretical analysis of the use of different such gain functions is given in Kearns and Mansour (1996). The algorithm begins at the root, chooses the variable of highest gain to put there, and then recurses on the nodes on each side. This process continues until all leaves are pure (all positive or all negative).

Problem Setup 1 *The decision tree targets $g^{(1)}, \ldots g^{(m)}$ and data sets $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(m)}$, each of at most S examples, satisfy the following conditions:*

1. *There exists an unknown set $\mathcal{F}$ of K metafeatures ($K \ll N$) such that $\forall j$, $g^{(j)} \in \mathbb{DT}(\mathcal{F})$.*

1. If feature i splits data set S into two sets L and R, its information gain of feature i is then $Ent(S) - [\frac{|L|}{|S|}Ent(L) + \frac{|R|}{|S|}Ent(R)]$. Here, Ent is the binary entropy of the label proportions in the given set; that is, if a p fraction of the labels in S' are positive, then $Ent(S') = p\log_2(1/p) + (1-p)\log_2(1/(1-p))$.

2. *The target $g^{(j)}$ can be learned by running top-down decision-tree learning on $\mathcal{S}^{(j)}$ using a given* `Gain` *function. In other words,* always choosing to recursively split on the variable of highest `Gain` based on $\mathcal{S}^{(j)}$ produces $g^{(j)}$.
3. *We are given s, d ($d \ll N$) such that $g^{(j)}$ has at most s internal nodes and depth at most d. Then, $S = O(s \log N)$ examples are sufficient to guarantee that $g^{(j)}$ has high accuracy over the underlying distribution over data.*

A straightforward lifelong learning approach would be as follows: IMPROVEREP simply adds to $\tilde{\mathcal{F}}$ features seen in tasks learned from scratch and USEREP examines only those features in $\tilde{\mathcal{F}}$ when learning a target. Since each metafeature in $\mathcal{F}$ can have at most s distinct features, we show in Appendix C that this learns at most K targets from scratch and evaluates only Ks features per example on the rest i.e., $O(S(KN + mKs))$ feature queries overall. However, when $s = \Omega(N)$ this is no better than learning all tasks individually from scratch. In this section, we will present a significantly better protocol:

Theorem 2 *The (USEREP Algorithm 1, IMPROVEREP Algorithm 2)-protocol for decision trees makes $O(S(KN + mKd))$ feature evaluations overall and runs in time $poly(m, N, K, S, s, d)$.*[2]

This is a significant improvement especially in the case of shallow bushy trees for which $d \ll s$ e.g., when $d = O(\log N)$ but $s = \Omega(N)$. To achieve this improvement, we need a computationally efficient approach that extracts bigger decision tree substructures from previous tasks and also knows how to learn future tasks using such a representation. We first address the latter problem: we present an USEREP routine, Algorithm 1, that takes as input a set of hypothesized metafeatures $\tilde{\mathcal{F}}$ and a training dataset $\mathcal{S}$ consistent with an unknown tree g and either outputs a consistent tree $\tilde{g}$ or halts with failure. To appreciate its guarantees, define $\mathsf{Pref}(f)$ to denote the set of all "prefix" trees (prunings) of some incomplete tree f. For any set of hypothesized metafeatures $\tilde{\mathcal{F}}$, let $\mathsf{Pref}(\tilde{\mathcal{F}}) = \{\mathsf{Pref}(\tilde{f}) \mid \tilde{f} \in \tilde{\mathcal{F}}\}$. We show that Algorithm 1, given $\tilde{\mathcal{F}}$, can effectively learn a target that can be represented using not only $\tilde{\mathcal{F}}$, but also the exponentially larger metafeature set $\mathsf{Pref}(\tilde{\mathcal{F}})$. That is, our USEREP algorithm can effectively learn trees from a much larger space $\mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$ compared to just $\mathbb{DT}(\tilde{\mathcal{F}})$.

Though we limit our discussion of Algorithm 1 to Boolean feature values for simplicity, we later extend it to real values. In Algorithm 1, we basically grow an incomplete decision tree $\tilde{g}$ one node at a time, by picking one of its empty leaf nodes u, and either assigning a label to u or splitting u on a particular feature. Before doing so, we first make sure that we have not failed already (Step 4). More specifically, if u is at a depth greater than d or if $\tilde{g}$ already has more than s nodes, we halt with failure because we were not able to find a small tree consistent with the data. If not, we proceed to examine samples from the training set that have reached u, which we will denote by $\mathcal{S}_u$. If all $x \in \mathcal{S}_u$ have the same label, we make u a leaf with that label and proceed to other nodes in $\tilde{g}$.

Otherwise, we evaluate a small set of features on $\mathcal{S}_u$ to compute their `Gain` and pick the best of those features to be the variable at u (denoted by $\texttt{var}(u)$). The way we pick this set of features at u, which we will call $\mathcal{I}$, is based on the following intuition. Assume we have grown $\tilde{g}$ identically to g so far and let u' be the node in g that corresponds to u. Then the correct variable to be assigned

2. It may seem that this result can be equivalently stated in terms of the average number of features examined per example i.e., $O(KN + mKd)$. However, such a performance metric is different from what we defined. Under certain independence conditions it may be possible to learn a target simply by drawing a large number of examples and examining only a single feature per example while still making many feature evaluations in total.

at u is $\texttt{var}(u')$ which is in fact the gain maximizing variable on $\mathcal{S}_u$ (as assumed in the second point of Problem Setup 1). Thus, our goal is to ensure $\texttt{var}(u') \in \mathcal{I}$.

If indeed $g \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$, this variable must in fact correspond to the variable in some node in some $\tilde{f} \in \tilde{\mathcal{F}}$. In other words, we should be able to "superimpose" some $\tilde{f}$ over $\tilde{g}$ with the root of $\tilde{f}$ at either u or one of its ancestors such that the variable in $\tilde{f}$ that has been superimposed over u is in fact the correct variable for u. Additionally, the variables in $\tilde{f}$ should not conflict with those that have already been assigned to the ancestors of u in $\tilde{g}$. Since we do not know which $\tilde{f}$ and which superimposition of $\tilde{f}$ "induces" the correct variable at u, we add to $\mathcal{I}$ the variable induced at u by every possible superimposition: we pick every $\tilde{f} \in \tilde{\mathcal{F}}$ and every node w that is either an ancestor of u or u itself, and then superimpose $\tilde{f}$ over $\tilde{g}$ with its root at w. We add to $\mathcal{I}$ the variable thus induced at u, provided the variables in $\tilde{f}$ do not conflict with those in the ancestors of u. In Algorithm 1, we use helper routines, $\textsf{INDUCE}(\tilde{g}, w, u, \tilde{f})$ which outputs the induced variable and $\textsf{CONFLICT}(\tilde{g}, w, u, \tilde{f})$ which outputs false if there is no conflict (both these simple subroutines are described for completeness in Appendix C and illustrated in Figure 2). Finally, since no variable should repeat along any path down the root, we remove from $\mathcal{I}$ any variable already assigned to an ancestor of u. Then, we assign the gain maximizing feature from $\mathcal{I}$ to u. Observe that, at u, in total over all $\tilde{f}$ we may examine $O(|\tilde{\mathcal{F}}|d)$ features on $\mathcal{S}_u$. Therefore, for a particular sample, considering all nodes along a path from the root, we may examine $O(|\tilde{\mathcal{F}}|d^2)$ features. However, with a more rigorous analysis we prove a tighter bound:

Theorem 3 USEREP *Algorithm 1 has the property that given $\tilde{\mathcal{F}}$ and data $\mathcal{S}$, a) if the underlying target $g \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$, the algorithm outputs g and b) conversely, if the algorithm outputs $\tilde{g}$ without halting on failure, then $\tilde{g}$ has depth at most d, size at most s and is consistent with $\mathcal{S}$, c) the algorithm evaluates $O(|\tilde{\mathcal{F}}| + d)$ features per example.*

Algorithm 1 USEREP - Learning a decision tree using metafeatures

Input: Metafeatures $\tilde{\mathcal{F}}$, samples $\mathcal{S}$ consistent with unknown g, depth bound d, size bound s.
Initialize the tree $\tilde{g}$ to be an empty leaf node. Let $\mathcal{Z}$ be the set of empty leaf nodes in $\tilde{g}$.
while $\exists\ u \in \mathcal{Z}$ **do**
  Halt with failure if a) u is at depth $> d$ or b) the size of $\tilde{g}$ is $> s$.
  Let $\mathcal{S}_u$ be the examples that have reached u.
  if all $x \in \mathcal{S}_u$ have the same label l **then**
    Make u a leaf with the label l.
  else
    Let $\mathcal{I}$ be the set of features to be examined at u. Initialize $\mathcal{I}$ to be empty.
    for each $\tilde{f} \in \tilde{\mathcal{F}}$ and each node w in the path starting from the root of $\tilde{g}$ to u **do**
      If $\textsf{CONFLICT}(\tilde{g}, w, u, \tilde{f})$ is false, add $\textsf{INDUCE}(\tilde{g}, w, u, \tilde{f})$ to $\mathcal{I}$.
    Remove from $\mathcal{I}$ any variable assigned to an ancestor of u.
    Evaluate only the features $\mathcal{I}$ on $\mathcal{S}_u$. Assign $\texttt{var}(u) \leftarrow \arg\max_{i \in \mathcal{I}} \texttt{Gain}(\mathcal{S}_u, i)$.
Output $\tilde{g}$.

Proof Sketch a) We show by induction that $\tilde{g}$ is always grown correctly i.e., $\tilde{g} \in \mathsf{Pref}(g)$. This is trivially true at the beginning. For the general case, let u be the node of $\tilde{g}$ that is to be grown and let u' be its counterpart in g. By induction, note that $\mathcal{S}_{u'} = \mathcal{S}_u$. Thus, if u' is a leaf, all $x \in \mathcal{S}_{u'}$ have the same label. Therefore, all $x \in \mathcal{S}_u$ have the same label, and hence we will label u as a

leaf with the correct label. If u' is an internal node, let $\texttt{var}(u')$ be the variable it splits on i.e., $\arg\max_{i\in[N]} \texttt{Gain}(\mathcal{S}_{u'}, i)$. To show that we assign $\texttt{var}(u')$ to u in Step 13, we only need to prove that $\texttt{var}(u') \in \mathcal{I}$. To prove this, we use the fact that $g \in \mathbb{DT}(\tilde{\mathcal{F}})$. Hence in g, $\texttt{var}(u')$ belongs to the prefix of some metafeature $\tilde{f}^*$ from $\tilde{\mathcal{F}}$ that is rooted either at u' or at one of its ancestors. Thus, during some execution of Step 11 we will add $\texttt{var}(u')$ to $\mathcal{I}$.

b) This follows immediately from the Algorithm, more specifically from Step 4 and 6. We need this guarantee so that when the learner does not fail, its output is guaranteed to be correct.

c) Each example corresponds to a particular path in $\tilde{g}$. Hence, the features examined on that example correspond to $\textsf{INDUCE}(\tilde{g}, w, u, \tilde{f})$ for different nodes v and u on that path, computed in Step 11 at different points during the run of the algorithm. Firstly, in the case that $w = u$, this could have been computed to be one of the $|\tilde{\mathcal{F}}|$ fixed set of features that occur at the root of each element of $\tilde{f} \in \tilde{\mathcal{F}}$. Now, to count the number of features examined for the cases where w is an ancestor of u, we prove the following. If for a particular u^*, we examine k_{u^*} different features $\textsf{INDUCE}(\tilde{g}, w, u^*, \tilde{f})$ each corresponding to different ancestor-metafeature pairs $(u, \tilde{f})$, then we will "eliminate" at least $k_{u^*} - 1$ metafeatures from resulting in feature evaluations for nodes u that are descendants of u^*. We use this conclusion to prove that we make only $O(|\tilde{\mathcal{F}}| + d)$ such queries down any path.

We present the full proofs in Lemma 10 and 11 in Appendix C. ■

Now, to provide a lifelong learning protocol for Problem Setup 1, the challenge is to design a computationally efficient IMPROVEREP routine[3]. To this end, we present Algorithm 2 that creates useful metafeatures by adding to $\tilde{\mathcal{F}}$ well-chosen subtrees from target functions. In particular, after learning a target g from scratch, we identify a root-to-leaf path in g that we failed to learn using $\tilde{\mathcal{F}}$. We add to $\tilde{\mathcal{F}}$ the subtrees rooted at every node in that path. The intuition is that one of these subtrees makes the representation more useful. To describe how the path is chosen, let $\tilde{g}$ be the incomplete tree learned using $\tilde{\mathcal{F}}$ just before we halted with failure. Since either the depth or the node count was exceeded in $\tilde{g}$, there must be a path from the root of $\tilde{g}$ longer than the corresponding path in g. We pick the corresponding path in g which was incorrectly learned in $\tilde{g}$ (see Figure 3).

Finally, as we see below in the proof sketch for Theorem 2, the resulting protocol evaluates only $O(Kd)$ features per example when learning from $\tilde{\mathcal{F}}$, besides learning K trees from scratch. Recall that this is a significant improvement of our straightforward USEREP which evaluates $O(Ks)$ features per example.

Algorithm 2 IMPROVEREP - Decision Trees

1: Input: Old representation $\tilde{\mathcal{F}}_{\text{old}}$ and a tree $g \in \mathbb{DT}(\mathcal{F})$ learned from scratch and the (incorrect) incomplete tree $\tilde{g}$ learned using $\tilde{\mathcal{F}}_{\text{old}}$.
2: $\tilde{\mathcal{F}} \leftarrow \tilde{\mathcal{F}}_{\text{old}}$
3: Identify a path from root of $\tilde{g}$ such that the corresponding path in g has fewer internal nodes.
4: For each node in the corresponding path in g, add the subtree rooted at that node to $\tilde{\mathcal{F}}$.
5: Output $\tilde{\mathcal{F}}$

3. As a warm-up, consider a semi-adversarial scenario where each element of $\mathcal{F}$ has a reasonable chance of being the topmost metafeature in any target. We can then learn the first few targets from scratch and simply add them to $\tilde{\mathcal{F}}$ so that with high probability, each metafeature from $\mathcal{F}$ is guaranteed to be the prefix of some element in $\tilde{\mathcal{F}}$. Then we can use Algorithm 1 to learn the remaining targets using $\tilde{\mathcal{F}}$ as all those targets will lie in $\mathbb{DT}(\textsf{Pref}(\tilde{\mathcal{F}}))$. We provide a careful analysis of this simpler case in Appendix C Theorem 12.

Proof Sketch (for Theorem 2) We show by induction that at any point during the protocol, if k targets have been learned from scratch, then $\exists\ \mathcal{F}' \subseteq \mathcal{F}$ such that $|\mathcal{F}'| = k$ and for each $f \in \mathcal{F}'$, some element from $\tilde{\mathcal{F}}$ contains f as its prefix. Then, after K failures of USEREP, $\mathbb{DT}(\mathcal{F}') \subseteq \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. Then from Lemma 10, USEREP is guaranteed not to fail. To prove the induction hypothesis, we note that when USEREP fails to learn g, there must be an element from $\mathcal{F} - \mathcal{F}'$ rooted somewhere in any root-to-leaf path in g that was not correctly learned. On adding to $\tilde{\mathcal{F}}$ the subtrees rooted at each node in one such path, we are sure to add an element whose prefix is a metafeature from $\mathcal{F} - \mathcal{F}'$. Thereby we "learn" a new metafeature from $\mathcal{F}$. Now, each time USEREP fails, we add at most d elements to $\tilde{\mathcal{F}}$, so $|\tilde{\mathcal{F}}| \leq Kd$. From Theorem 3, our result follows. ■

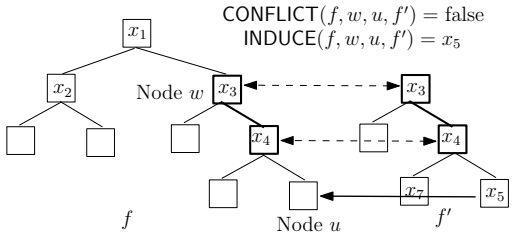

Figure 2: Superimposing f' over f with its root at w

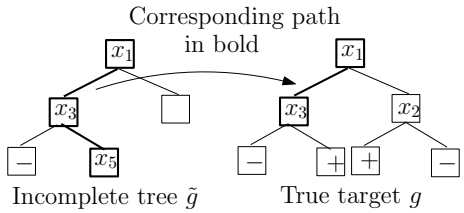

Figure 3: Path chosen by IMPROVEREP Algorithm 2.

Extension to real-valued features: Our results hold also for decision trees over real-valued features, where nodes contain binary splits such as "$x_1 \geq 7$". In particular, we reduce this to the Boolean case by viewing each such split as a Boolean variable. While this reduction involves an implicitly infinite number of Boolean variables, our bounds still apply. This is because we make only N feature evaluations per example when learning from scratch (and not infinitely many). Also, the feature evaluations made by our USEREP is independent of the number of Boolean variables.

Decision Lists While we can use the above protocol for decision lists too, it does not effectively provide any improvement over the baseline approach because for lists, $s = d$. However, by making use of the structure of decision lists, we provide a protocol that learns K^2 lists from scratch and on the rest examines only $O\left(K^2 + d\right)$ features per example. The high level idea is that when we fail to learn a target using $\tilde{\mathcal{F}}$, we add to $\tilde{\mathcal{F}}$ only a single suffix of the target list as a new metafeature instead of adding all d suffixes like in Algorithm 2. To prove our bound, we have a technically interesting argument which we present in the appendix.

Theorem 4 *The (USEREP Algorithm 1, IMPROVEREP Algorithm 7)-protocol for decision lists makes $O\left(S(K^2N + m(K^2 + d))\right)$ feature evaluations overall and runs in time $poly(m, N, K, S, d)$.*

4. Monomials

We consider lifelong learning of degree-d monomials under the belief that there exists a set of K monomial metafeatures like $\{x_1x_2, x_1^2x_3, \ldots\}$ and each target can be expressed as a product of powers of these metafeatures e.g., $(x_1x_2)^2(x_1^2x_3)$. This is similar to the lifelong Boolean monomial learning discussed in Balcan et al. (2015) where each monomial is a conjunction of monomial

metafeatures. Since that is an NP-hard problem, they assume that the metafeatures have so-called "anchor" variables unique to each. We will however not need this assumption.

We first describe our setup more formally. For any input $\mathbf{x} = (x_1, x_2, \dots x_N) \in \mathbb{R}^N$, we denote the output of a d-degree target monomial $\mathbf{g} = (g_1, g_2, \dots, g_N)$ by the function $P_{\mathbf{g}}(\mathbf{x}) = x_1^{g_1} x_2^{g_2} \dots x_N^{g_N}$ where $g_i \in \mathbb{N} \cup \{0\}$ and the degree $\sum_i g_i \leq d$. The unknown metafeature set $\mathcal{F} = \{\mathbf{f}_1, \mathbf{f}_2, \dots \mathbf{f}_K\}$ consists of K monomials. To simplify notations, we also consider $\mathcal{F}$ to be a matrix where column i is $\mathbf{f}_i$. Therefore, if $\mathbf{g}$ can be expressed using $\mathcal{F}$, then $\mathbf{g}$ lies in the column space of $\mathcal{F}$ denoted by $\mathbb{C}(\mathcal{F})$. Then, our problem setup is as follows:

Problem Setup 2 *The m d-degree targets $\mathbf{g}^{(1)}, \dots \mathbf{g}^{(m)}$ and the training data (each of at most S examples) drawn from unknown distributions $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(2)}$ satisfy the following conditions:*

1. *There exists an unknown $N \times K$ matrix $\mathcal{F}$ ($K \ll N$) such that $\mathbf{g}^{(j)} \in \mathbb{C}(\mathcal{F})$.*
2. *Each $\mathcal{D}^{(j)}$ is a product distribution (as assumed in Balcan et al., 2015; Andoni et al., 2014) that is not too concentrated (explained in Appendix D).*

Unlike the decision tree problem, where we only consider an abstraction of the learning routine, here we present a particular technique for learning a monomial exactly. We show that under product distributions that are not too concentrated, it is possible to *exactly* learn the power of a given feature in a target by examining only that feature on polynomially many samples (Lemma 16). Naturally, we can learn the monomial exactly from scratch (Algorithm 8 in Appendix D). Then, in the lifelong learning model, by merely keeping a record of the features that have been seen so far, it is fairly straightforward to learn only K targets from scratch while learning the rest by examining $O(Kd)$ features per example (Theorem 18). Here, we present a significantly better protocol that learns only K targets from scratch and on the rest, evaluates only $O(K)$ features on all examples and d features on one example. This is an improvement especially for cases where d is large. We present a summary of our approach in the following proof sketch.

Theorem 5 *The (*USEREP *Algorithm 10,* IMPROVEREP *Algorithm 9)-protocol for monomials makes $O(S(KN + mK) + md)$ feature evaluations overall and runs in time $poly(m, N, K, S, d)$.*

Proof Sketch The key idea is that we store a list of targets that have been learned from scratch as columns of the matrix $\tilde{\mathcal{F}}$. Also, we learn a new target from scratch only if $g \notin \mathbb{C}(\tilde{\mathcal{F}})$. Therefore, after learning K targets from scratch, we can show that we have a K rank matrix $\tilde{\mathcal{F}}$ such that $\mathbb{C}(\tilde{\mathcal{F}}) = \mathbb{C}(\mathcal{F})$. Therefore all future targets can be learned using $\tilde{\mathcal{F}}$.

Now, the idea for USEREP is as follows. If we have learned k targets from scratch, then $\tilde{\mathcal{F}}$ is of rank k. Then, we identify a set of k features $\mathcal{I}$ that correspond to linearly independent rows in $\tilde{\mathcal{F}}$. We first learn only the powers of $\mathcal{I}$ (which we will denote by $\mathbf{g}[\mathcal{I}]$) by examining $\mathcal{I}$ on all samples. Then we learn a monomial $\tilde{\mathbf{g}}$ by using the equation $\tilde{\mathbf{g}} = \tilde{\mathcal{F}}(\tilde{\mathcal{F}}[\mathcal{I}])^{-1}\mathbf{g}[\mathcal{I}]$. If indeed $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$, then $\tilde{\mathbf{g}}$ equals $\mathbf{g}$. This is because the power of each monomial metafeature in $\mathbf{g}$ is recovered through $(\tilde{\mathcal{F}}[\mathcal{I}])^{-1}\mathbf{g}[\mathcal{I}]$. However, we do not know if $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$ and it may be that $\tilde{\mathbf{g}} \neq \mathbf{g}$. To address this, we can show using Lemma 20 that we only need to draw a single sample $\mathbf{x}$, examine d features relevant to $\tilde{\mathbf{g}}$ and check if our prediction $P_{\tilde{\mathbf{g}}}(\mathbf{x})$ equals the true label $P_{\mathbf{g}}(\mathbf{x})$. If this fails, we conclude that $\tilde{\mathbf{g}} \neq \mathbf{g}$ and therefore, $\mathbf{g} \notin \mathbb{C}(\tilde{\mathcal{F}})$. We learn $\mathbf{g}$ from scratch and add it to $\tilde{\mathcal{F}}$. Thus, USEREP examines only at most K features on all but one sample and d features on one final sample. In fact, after learning K targets from scratch, we do not need to examine the d features and do the verification step because we are guaranteed that $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$. ■

4.1. Polynomials

We consider lifelong learning of real-valued polynomial targets each of which is a sum of at most t degree-d monomials. Similar to the Boolean model in Balcan et al. (2015), our belief is that there exists a set of monomial metafeatures such that each monomial in the polynomial can be expressed as a product of these metafeatures like we described in the previous section. As an example, given $\mathcal{F} = \{x_1x_2, x_1^2x_3, \ldots\}$, one possible target is $3(x_1x_2)(x_1^2x_3) - 5(x_1x_2)^2(x_1^2x_3)$. Again, we assume that each $\mathcal{D}^{(j)}$ is a product distribution over $\mathbb{R}^N$. Since polynomial learning is a hard problem, we will have to make a strong assumption that each $\mathcal{D}^{(j)}$ is *known*, which then enables us to adopt the polynomial learning technique from Andoni et al. (2014). Note that we can relax this assumption when all the distributions are common (like it is assumed in Balcan et al., 2015), so that the common distribution can first be learned using $O\left(poly(N)\right)$ feature evaluations. However, if the distributions were all different, learning them may need $O\left(poly(mN)\right)$ feature evaluations, which would be feature-inefficient.

We now briefly discuss the algorithm in Andoni et al. (2014) for learning a polynomial from scratch from a known distribution. Note that we have modified the algorithm slightly for our purposes. The basic idea is to perform t iterations extracting one of the t monomials from the given polynomial at a time. After a few iterations, consider the residual set of monomials that have not yet been extracted yet. For the next iteration, we extract the "lexicographically largest" monomial from this set; namely, the monomial that has the highest power of x_1, say $x_1^{d_1}$, the highest power of x_2 of all monomials in the residual set containing $x_1^{d_1}$ and so on. To extract the monomial identified this way, we compute the value of this monomial on each training sample and subtract the value from the output of that sample. Note that the running time of this approach is exponential in d (as is the case in other polynomial learning approaches).

Then, in the lifelong learning model, we can learn the targets by making $O\left(S(KN + mKd)\right)$ feature evaluations by simply remembering what features have been seen so far (Theorem 21). We present an approach in Appendix D that makes only $O\left(S(KN + m(K + td))\right)$ feature evaluations. This is an improvement for sparse polynomials $t < K$ e.g., when $t = O\left(1\right)$.

Theorem 6 *The (*USEREP *Algorithm 12,* IMPROVEREP *Algorithm 11)-protocol for polynomials makes* $O\left(S(KN + m(K + dt))\right)$ *feature evaluations overall and runs in time* $poly(m, N, K, S, t)$.

Proof The high level idea is to maintain a metafeature set of "linearly independent monomials" picked from previously seen targets, like we did in the previous section. When learning a target using $\tilde{\mathcal{F}}$, we perform t iterations to extract the monomials, but now in each iteration we find the lexicographically largest power restricted to at most K features. These K features correspond to linearly independent rows in $\tilde{\mathcal{F}}$. Given the powers of these features, we can determine powers of all the features like we did before. Based on this we can extract a complete monomial from the residual polynomial. Note that the restricted lexicographic search examines only a fixed set of K features per example. Besides this, in each of the t iterations, we evaluate d features relevant to the extracted monomial, accounting for $K + td$ feature evaluations per example. ■

5. The Agnostic Case

We propose a novel agnostic lifelong learning model where the learner faces $m + r$ learning tasks of which m tasks are guaranteed to be related through the K metafeatures in $\mathcal{F}$ while the other r tasks

are arbitrary. Note that this is different from the conventional sense of agnostic learning where each individual task may involve model misspecification or noisy labels. What makes this challenging is that the r "bad" targets can be chosen and placed adversarially in the stream of tasks. Since in the worst case there is no hope of minimizing feature evaluations done on the bad targets, we adopt the natural goal of reducing the feature evaluations on the training data of the m good targets.

We focus our discussion on learning decision trees with depth $d = O(1)$ noting that it is straightforward to extend it to learning more general decision trees and to other targets. In fact, in the following discussion, it may be helpful to imagine the targets to be decision stumps over just one feature and the metafeature set $\tilde{\mathcal{F}}$ to simply be a set of K features. Now, recall that in the original setup, $\tilde{\mathcal{F}}$ consisted of $O(K)$ useful metafeatures from at most K targets that were learned from scratch USEREP failed to learn them. A problem that arises now is that $\tilde{\mathcal{F}}$ may have been updated with metafeatures from bad targets. Then, even if $\tilde{\mathcal{F}}$ contained K metafeatures, we cannot guarantee that future good targets can be learned using $\tilde{\mathcal{F}}$. What should we do then?

To address this, we present two simple computationally-efficient solutions (described in detail in Appendix E) that highlight an interesting trade-off between the number of targets learned from scratch and the number of features evaluated on the remaining targets. In the *r-expansion* technique, we allow the learner to update $\tilde{\mathcal{F}}$ on every failure of USEREP. Doing so, we learn at most K good targets from scratch (and possibly all the r bad targets too, which we do not care about). Since $\tilde{\mathcal{F}}$ can expand to as many as $O(K + r)$ metafeatures this way, we learn the remaining targets using $\tilde{\mathcal{F}}$ evaluating $O(K + r)$ features per example. In the *r-restart* technique, the key idea is to restrict the representation to $O(K)$ metafeatures. Whenever the representation reaches this limit and still USEREP fails, we restart lifelong learning with an empty representation. This technique learns more targets from scratch, $O(rK)$ targets in particular, but evaluating only $O(K)$ features per example on the remaining targets. When $r = O\left(\max\left(\frac{m}{N}, \frac{KN}{m}, K\right)\right)$, it is easy to see that one of these two techniques makes only $O(S(KN + mK))$ feature evaluations, which is as good as the performance when $r = 0$. To deal with larger values of r, in Appendix E, we describe a combined technique that deals with the trade off carefully and does better than both the above:

Theorem 7 *In the agnostic model where we face $m + r$ decision tree targets such that m trees belong to $\mathbb{DT}(\mathcal{F})$, the number of feature evaluations on the training data for the m trees:*

- *the r-expansion technique is $O(S(KN + m(K + r)))$.*
- *the r-restart technique is $O(S(rKN + mK))$.*
- *a combination of c-expansion and r/c-restart is $O(S(\sqrt{rKNm} + Km))$, for $c = \sqrt{rKN/m}$ provided $r = \Omega(\max(m/n, KN/m, K))$.*

6. Lower bounds

We prove lower bounds on the performance of any lifelong learner under different ranges of r in the agnostic model. In particular, we prove tight lower bounds for sufficiently small and large values of r, ignoring other problem-specific parameters and the sample size parameter S (that scaled only logarithmically with N for most of our target classes). An interesting insight here is that when r is too large, we prove that no learner is guaranteed to succeed by making $O(mN)$ feature queries, which means that lifelong learning is no longer meaningful for really large values of r. We state our results formally below, with more discussion in Appendix F.

Theorem 8 *Let* $r_{\min} = \max\left(\frac{m}{N}, \frac{KN}{m}, K\right)$, $r_{\max} = \min\left(\frac{mN}{K}, \frac{(N-K)^2 m}{KN}\right)$. *In the agnostic model of Section 5, there exists an adversary such that, on the* m *good trees, any lifelong learner makes:*

- $\Omega\left(NK + Km\right)$ *feature evaluations when* $0 \leq r \leq r_{\min}$.
- $\Omega\left(\max\left(\frac{r}{N-K}, 1\right) KN + Km\right)$ *feature evaluations when* $r_{\min} \leq r \leq r_{\max}$.
- $\Omega\left(mN\right)$ *feature evaluations when* $r_{\max} \leq r$.

Acknowledgements. This work was supported in part by the National Science Foundation under grants CCF-1535967, CCF-1525971, CCF-1422910, IIS-1618714, a Sloan Research Fellowship, a Microsoft Faculty Fellowship, and a Google Research Award. This work was conducted while the second author was at Carnegie Mellon University.

References

International Conference on Machine Learning (ICML) Workshop on budgeted learning, 2010.

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 500–510, 2014.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 41–48, 2006.

Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 779–806, 2014.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Efficient representations for lifelong learning and autoencoding. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 191–210, 2015.

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

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research (JAIR)*, 12, 2000.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

Hendrik Drachsler, Hans G. K. Hummel, and Rob Koper. Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model. *International Journal of Learning Technology (IJLT)*, 3(4):404–423, 2008.

Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 895–900, 2006.

Aloak Kapoor and Russell Greiner. Learning and classifying under hard budgets. In *Machine Learning: 16th European Conference on Machine Learning (ECML)*, pages 170–181, 2005.

Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 459–468, 1996.

Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A_*. *Artificial Intelligence*, 155(1-2):93–146, 2004.

A. Kumar and H. Daume III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the International Conference on Machine Learning (ICML)* , 2012.

Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

Daniel J. Lizotte, Omid Madani, and Russell Greiner. Budgeted learning of naive-bayes classifiers. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 378–385, 2003.

A. Maurer and M. Pontil. Excess risk bounds for multitask learning with trace norm regularization. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, 2013.

Guillaume Obozinski and Ben Taskar. Multi-task feature selection. In *Workshop of Structural Knowledge Transfer for Machine Learning in the International Conference on Machine Learning (ICML)*, 2006.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

A. Pentina and R. Urner. Lifelong learning with weighted majority votes. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, 26(5):445–463, 2002.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. ISSN 0885-6125.

Lior Rokach and Oded Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., 2008. ISBN 9789812771711, 9812771719.

S. Thrun and L.Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, 1997.

Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

Eric R. Ziegel. The elements of statistical learning. *Technometrics*, 45(3):267–268, 2003.

Appendix A. Summary of results

Problem	**Total number of feature evaluations**
Decision trees of depth d	$O\left(S(KN + mKd)\right)$
Decision trees of depth d in semi-adversarial model	$O\left(S(\frac{\log K}{p_{\min}}N + m(K + d))\right)$
Decision trees of depth d with anchor variables	$O\left(S(KN + m(K + d))\right)$
Decision lists of depth d	$O\left(S(K^2N + m(K^2 + d))\right)$
Monomials of degree d	$\tilde{O}\left(KN + m(K + d)\right)$
Polynomials of degree d, sparsity t	$O\left(S(KN + m(K + td))\right)$

Table 1: Performance of our approaches

Range of r	Performance of algorithm	Lower bound
$0 \leq r \leq r_{\min}$	$O(S(NK + Km))$	$\Omega\left(NK + Km\right)$
$r \in [r_{\min}, r_{\max}]$	$O(S(\underbrace{\sqrt{rKNm}}_{\leq\sqrt{\frac{r_{\max}}{r}}\max\left(\frac{r}{N-K},1\right)KN} + Km))$	$\Omega\left(\max\left(\frac{r}{N-K}, 1\right)KN + Km\right)$
$r \geq r_{\max}$	$O\left(SmN\right)$	$\Omega\left(mN\right)$

Table 2: Performance of our algorithms for different values vs the lower bounds for different values of r. Recall that $r_{\min} = \max\left(\frac{m}{N}, \frac{KN}{m}, K\right)$ and $r_{\max} = \min\left(\frac{mN}{K}, \frac{(N-K)^2m}{KN}\right)$

Appendix B. Notations

Notation	**Meaning**
m	No. of targets in sequence
N	No. of features/variables
$\mathcal{F}$	True metafeature set/representation
$\tilde{\mathcal{F}}$	Learned representation
K	No. of true metafeatures
S	No. of samples for each task
$\mathcal{S}^{(j)}$	Training data for task j

Table 3: Important notations

Algorithm 3 $(\mathcal{A}_{\mathsf{UR}}, \mathcal{A}_{\mathsf{IR}})$-protocol for lifelong learning

Input: A sequence of m training sets $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots,$ corresponding to targets $g^{(1)}, g^{(2)}, \ldots,$ each of which can be represented using an unknown set $\mathcal{F}$ of K metafeatures.
Let $\tilde{\mathcal{F}}$ be our current learned representation. Initialize $\tilde{\mathcal{F}}$ to be empty.
for $j = 1, 2, \ldots m$ **do**
    Using $\tilde{\mathcal{F}}$ and $S^{(j)}$, attempt to cheaply learn $g^{(j)}$ with USEREP algorithm $\mathcal{A}_{\mathsf{UR}}$.
    If learning was not successful, extract all features in $\mathcal{S}^{(j)}$ and learn $g^{(j)}$ from scratch; provide $\tilde{\mathcal{F}}$ and $g^{(j)}$ as input to IMPROVEREP algorithm $\mathcal{A}_{\mathsf{IR}}$ to update $\tilde{\mathcal{F}}$.

Appendix C. Decision Trees

We first present proofs from Section 3. In Appendix C.2 we prove our main result for decision lists.

C.1. Proofs from Section 3

Theorem 9 (Naive lifelong learning of decision trees) *There exists a naive lifelong learning protocol for decision trees in the model of Problem Setup 1 evaluates $O\left(S(KN + mKs)\right)$ features overall.*

Proof The naive approach follows from a simple observation. If we knew beforehand the set of features that are involved in a tree $g^{(j)}$, then in order to learn the tree, at any given node we require the learner to evaluate `Gain` only over these features to determine the best split at that node. Thus, our protocol will just maintain the set of features present in any tree learned from scratch so far, so that USEREP can use these as "metafeatures" to carry out its evaluations limited to these features. Then, any target that can be represented using metafeatures $f \in \mathcal{F}$ that have been seen before in some other target, will be learned using our metafeatures. In other words when USEREP fails, the target is guaranteed to contain an "unseen" metafeature from $\mathcal{F}$. Thus, we will learn targets from scratch at most $|\mathcal{F}| = K$ times. Since each metafeature in $\mathcal{F}$ has at most s distinct features, we will have to evaluate only at most Ks features when not learning from scratch. ■

We now present the pseudocode for the different subroutines described informally in our discussion.

Algorithm 4 AFFIX(f, u, f'): Affix f' to f at empty leaf node u in f

Input: Incomplete decision trees f, f', empty leaf node u in f
Assign to $\texttt{var}(u)$ the root variable of f'.
Create descendants nodes of u and assign variables to them such that the tree rooted at u is identical to f'.

Algorithm 5 LABEL(f, u, l): Assign l to u in f

Input: Incomplete decision tree f, empty leaf node u in f, label $l \in \{+, -\}$
Assign to leaf node u the label l.

Algorithm 6 CONFLICT(f, w, u, f') and INDUCE(f, w, u, f')

1: **Input:** Incomplete decision trees f, f', node w in f, node u that is a descendant of w or equal to w itself.
2: Let $\mathcal{V}$ be the set of nodes in f that are ancestors of u but not of w.
3: Map w in f to the root node of f'.
4: Similarly map all descendant nodes of w from $\mathcal{V}$ to the nodes in the corresponding path in f'.
5: **Output of CONFLICT**(f, w, u, f'): If there are two internal nodes $v \in f$ and $v' \in f'$ mapped to each other but $v \in \mathcal{V}$, $\texttt{var}(v) \neq \texttt{var}(v')$, output true. Else output false.
6: **Output of INDUCE**(f, w, u, f'): Let u' be the node from f' mapped to u. Output $\texttt{var}(v')$.

Now we prove Lemma 10 and Lemma 11, which together prove guarantees about our powerful USEREP Algorithm 1. Lemma 10 states that as long as the underlying target g can be constructed using the exponentially large representation $\mathsf{Pref}(\tilde{\mathcal{F}})$, Algorithm 1 will output g. Lemma 11 states that this algorithm examines $O\left(|\tilde{\mathcal{F}}| + d\right)$ features per example.

Lemma 10 *If $g \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$, Algorithm 1 outputs $\tilde{g} = g$.*

Proof We are given that $g \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. We will show by induction that $\tilde{g}$ is always grown correctly i.e., $\tilde{g} \in \mathsf{Pref}(g)$. This is trivially true at the beginning. Consider the general case. Let u be the node in $\tilde{g}$ that is chosen in Step 3 to be grown. By our induction hypothesis that $\tilde{g}$ is a prefix of g, there exists u' in g that corresponds to u and furthermore, $\mathcal{S}_u = \mathcal{S}_{u'}$. Now to show that u will be grown identical to u', since $\tilde{g}$ is only a prefix, the size and depth constraints will be satisfied and so we are guaranteed to not halt with failure at this node. Next, if u' was a leaf node, since $\mathcal{S}_u = \mathcal{S}_{u'}$, we are guaranteed to label u as a leaf and assign it the correct label.

If u' is not a leaf node, let $\texttt{var}(u')$ be the variable present in u' i.e., $\texttt{var}(u') = \arg\max_{i \in [N]} \texttt{Gain}(\mathcal{S}_{u'}, i)$. Therefore, to show that we assign $\texttt{var}(u')$ to u in Step 13, we only need to prove that $\texttt{var}(u') \in \mathcal{I}$ i.e., we consider this feature for examination. To prove this, note that in g, $\texttt{var}(u')$ belongs to the prefix of some metafeature $\tilde{f}^*$ from $\tilde{\mathcal{F}}$ that is rooted either at some v' which is either u' itself or at one of its ancestors (because $g \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$). We can show that in Step 11, when $w = v$ and $\tilde{f} = \tilde{f}^*$, we end up adding $\texttt{var}(u')$ to $\mathcal{I}$. First, if v is the corresponding node in $\tilde{g}$ we will have that $\textsf{CONFLICT}(\tilde{g}, v, u, \tilde{f}^*)$ is false. Furthermore, clearly $\textsf{INDUCE}(\tilde{g}, v, u, \tilde{f}) = \texttt{var}(u')$. Now since g has no variable repeating along any root-to-leaf path, $\texttt{var}(u')$ does not occur in any of the ancestor nodes of u', and similarly in $\tilde{g}$, it does not occur in any of the ancestor nodes of u. Thus, the conditions in Step 11 succeed, following which $\texttt{var}(u')$ is added to $\mathcal{I}$. ■

Lemma 11 *Algorithm 1 makes at most $O(|\tilde{\mathcal{F}}| + d)$ feature queries per example.*

Proof First of all note that each example corresponds to a particular path in $\tilde{g}$. Thus, the features examined on that example as $\tilde{g}$ was grown, correspond to the different features computed from $\textsf{INDUCE}(\tilde{g}, w, u, \tilde{f})$ for different nodes v and u on that path. These feature queries can be classified into two types depending on whether A) $w = u$ or B) w is an ancestor of u. For type A, since $w = u$, $\textsf{INDUCE}(\tilde{g}, w, u, \tilde{f})$ can only be one of the fixed set of features that occur at the root of metafeatures in $\tilde{\mathcal{F}}$. In total this may account for at most $|\tilde{\mathcal{F}}|$ feature examinations.

Now consider the type B features queries corresponding to $w \neq u$. Each feature examined in this case corresponds to a 3-tuple $(w, u, \tilde{f})$ where w is an ancestor of u. We claim that for a given $\tilde{f}$, w has to be unique in this path. This is because if such a 3-tuple results in a feature query, it should also satisfy the condition in Step 11 that $\textsf{CONFLICT}(\tilde{g}, w, u, \tilde{f})$ is false. Then, this means that $\texttt{var}(w)$ is equal to the root variable in $\tilde{f}$. However since the algorithm never considers a feature already assigned to an ancestor (as seen in Step 12), there can only be at most one node w in this path that contains the variable $\texttt{var}(w)$.

Thus type B feature query effectively corresponds to a 2-tuple $(u, \tilde{f})$ instead of a 3-tuple $(w, u, \tilde{f})$ because $\tilde{f}$ corresponds to a unique w. Let $w_{\tilde{f}}$ denote this unique node for $\tilde{f}$. Now, let k_u be the number of type B feature queries made at u. We can divide this case further into type B(a) consisting of nodes u, such that $k_u = 1$ and type B(b) corresponding to $k_u > 1$. In total over the d nodes in $\tilde{g}$, we would examine only d type B(a) features. Now, for type B(b), at node u, where we evaluate k_u features at u, we claim that this effectively "eliminates" at least $k_u - 1$ different metafeatures from resulting in feature examinations of type B further down this path. This is because each of the k_u features that we examine at u correspond to $\textsf{INDUCE}(\tilde{g}, w_{\tilde{f}}, u, \tilde{f})$ for some $\tilde{f} \in \tilde{\mathcal{F}}$. Let this set of metafeatures be $\tilde{\mathcal{F}}_u$, where $|\tilde{\mathcal{F}}_u| = k_u$. Now, we assign only one feature to u that corresponds to say, $\tilde{f}^* \in \tilde{\mathcal{F}}_u$. After this, when we are growing a descendant node v, for the $k_u - 1$ other metafeatures $\tilde{f} \in \tilde{\mathcal{F}}_u$ and $\tilde{f} \neq \tilde{f}^*$, $\textsf{CONFLICT}(\tilde{g}, w_{\tilde{f}}, v, \tilde{f})$ will be true as there will be a conflict at u. However, since $\textsf{CONFLICT}(\tilde{g}, w_{\tilde{f}}, v, \tilde{f})$ needs to be false in Step 11 for $\tilde{f}$ to result in a feature query, we conclude that there are $k_u - 1$ different metafeatures that do not result in a feature query beyond this point.

Using the above claim, we can now bound $\sum_{u:k_u>1} k_u$, which will account for the total feature queries of type B(b) along the path. Since $k_u - 1$ denotes the number of eliminated metafeatures beyond u, and since only at most $|\tilde{\mathcal{F}}|$ can be eliminated, we have $\sum_{u:k_u>1}(k_u - 1) \leq |\tilde{\mathcal{F}}|$. Now, since $\sum_{u:k_u>1} 1 \leq d$, we have that $\sum_{u:k_u>1} k_u \leq |\tilde{\mathcal{F}}| + d$ i.e., we make at most $|\tilde{\mathcal{F}}| + d$ type B(b) feature queries of the last kind on this path. Thus, in summary, we examine at most $O(|\tilde{\mathcal{F}}| + d)$ features on each example. ■

We now prove our result for the semi-adversarial model, where in any given target, each $f \in \mathcal{F}$ has at least a p_{min} probability of being the topmost metafeature.

Theorem 12 (Lifelong learning of decision trees in semi-adversarial model) *There exists a lifelong learning protocol for decision trees that evaluates $O\left(\frac{1}{p_{\min}} \log \frac{K}{\delta} \cdot N + m(K + d)\right)$ features overall in a semi-adversarial model where each element of $\mathcal{F}$ has at least a $p_{\min}$ probability of being the topmost element of any target. The protocol learns only the first $O\left(\frac{1}{p_{\min}} \log \frac{K}{\delta}\right)$ targets from scratch, adds them to $\tilde{\mathcal{F}}$ and then uses* USEREP *Algorithm 1 to learn all the subsequent targets from $\tilde{\mathcal{F}}$.*

Recall that direct application of Lemma 11 implies that we will learn the subsequent targets examining $O\left(\frac{1}{p_{\min}} \log K + d\right)$ features per example. However, a more careful analysis making use of the fact that each element in $\tilde{\mathcal{F}}$ is in fact from $\mathbb{DT}(\mathcal{F})$ shows that we will examine only $O(K + d)$ features per example. Note that this is an improvement because $\frac{1}{p_{\min}} \log K \geq K \log K$.

Proof (for Theorem 12) Consider the protocol from Theorem 12 that learns the first $O\left(\frac{1}{p_{\min}} \log \frac{K}{\delta}\right)$ targets from scratch, and adds them all to $\tilde{\mathcal{F}}$. Then with probability at least $1 - \delta$, each metafea-

ture from $\mathcal{F}$ will be at the top of some metafeature from $\tilde{\mathcal{F}}$. That is, $\mathbb{DT}(\mathcal{F}) \subseteq \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. Then, from Theorem 3 clearly Algorithm 1 can learn any future target from $\mathbb{DT}(\mathcal{F})$ as the target will also lie in $\mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. Now, by a direct application of Theorem 3 this means we evaluate $O\left(\frac{1}{p_{\min}} \log \frac{K}{\delta} + d\right)$ features per example.

However, we can prove a tighter bound of $O(K + d)$ by following the proof technique for Lemma 11 but using to our advantage the fact that the metafeatures in $\tilde{\mathcal{F}}$ are not arbitrary trees, but in fact members of $\mathbb{DT}(\mathcal{F})$. First of all, observe that the number of type A costs along any path is in fact K and not $|\tilde{\mathcal{F}}|$ because the metafeatures in $\tilde{\mathcal{F}}$ can have only one of at most K variables at its root. Now, for the first case within type B, we will pay a cost of d as before. However, for the second case, observe that any variable that is induced at u by a metafeature $\tilde{f} \in \tilde{\mathcal{F}}$, is in effect induced by a metafeature $f \in \mathcal{F}$. That is, when we compute $\mathsf{INDUCE}(\tilde{g}, w_{\tilde{f}}, u, \tilde{f})$ for some metafeature $\tilde{f} \in \tilde{\mathcal{F}}$, we effectively compute $\mathsf{INDUCE}(\tilde{g}, w_f, u, f)$ for some metafeature $f \in \mathcal{F}$. Similarly we ca argue that whenever we make k_u distinct feature queries at a particular node u during the algorithm, for all nodes beyond u in that path, we effectively eliminate queries arising from $k_u - 1$ metafeatures from $\mathcal{F}$ (and not $\tilde{\mathcal{F}}$ as before). This will result in a total cost of $|\mathcal{F}| = K$ for this case. ■

We now prove our main result, Theorem 2 for decision trees which presented a protocol for learning decision trees which learns at most K targets from scratch, and learns the rest examining only $O(Kd)$ features per example.

Theorem 2 *The (*USEREP *Algorithm 1,* IMPROVEREP *Algorithm 2)-protocol for decision trees makes $O(S(KN + mKd))$ feature evaluations overall and runs in time $poly(m, N, K, S, s, d)$.*[4]

Proof (for Theorem 2) We will show by induction that at any point during a run of the protocol, if k targets have been learned from scratch, then there exists a subset of k true metafeatures $\mathcal{F}' \subseteq \mathcal{F}$ that have been "learned" in the sense that $f \in \mathcal{F}'$ is the prefix of some metafeature in $\tilde{\mathcal{F}}$, implying that $\mathbb{DT}(\mathcal{F}') \subseteq \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. Then after learning K targets from scratch, it has to be the case that $\mathcal{F}' = \mathcal{F}$ after which $\mathbb{DT}(\mathcal{F}) \subseteq \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$ and hence from Lemma 10 it follows that the protocol can never fail while learning from $\tilde{\mathcal{F}}$.

The base case is when $\tilde{\mathcal{F}}'$ is empty for which the induction hypothesis is trivially true. Now, assume at some point we have metafeatures $\tilde{\mathcal{F}}_{\text{old}}$ and these correspond to true metafeatures $\mathcal{F}'_{\text{old}} \subseteq \mathcal{F}$ such that $\mathbb{DT}(\mathcal{F}'_{\text{old}}) \subseteq \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$ and $|\mathcal{F}'_{\text{old}}| = k$. Now, from Theorem 3, we can conclude that any target that lies in $\mathbb{DT}(\mathcal{F}'_{\text{old}})$ will be successfully learned by USEREP Algorithm 1. Hence, when USEREP does fail on a new target g, it means that the g contains metafeatures from $\mathcal{F} - \mathcal{F}'_{\text{old}}$. In fact, along any path in g in which learning failed (that is, the tree $\tilde{g}$ that is output differs from g on this path), there must be a node at which some metafeature from $\mathcal{F} - \mathcal{F}'_{\text{old}}$ is rooted. If this was not true for a particular failed path, we can show using an argument similar to Lemma 10 that this path would have been learned correctly. Therefore, when we add to $\tilde{\mathcal{F}}$ all the subtrees rooted at the nodes in some failed path in g, we are sure to add a tree which has some $f \in \mathcal{F} - \mathcal{F}'_{\text{old}}$ as one of its prefixes. This means that for the updated set of metafeatures, there exists $\mathcal{F}' = \mathcal{F}'_{\text{old}} \cup \{f\}$ of cardinality $k + 1$ that satisfies the induction hypothesis.

4. It may seem that this result can be equivalently stated in terms of the average number of features examined per example i.e., $O(KN + mKd)$. However, such a performance metric is different from what we defined. Under certain independence conditions it may be possible to learn a target simply by drawing a large number of examples and examining only a single feature per example while still making many feature evaluations in total.

Now, each time USEREP fails, we add at most d metafeatures to $\tilde{\mathcal{F}}$, so $|\tilde{\mathcal{F}}| \leq Kd$. From Theorem 3, it follows that we evaluate only $O(Kd)$ features when learning using $\tilde{\mathcal{F}}$. ■

C.2. Decision Lists

Recall that we have a protocol for decision lists (Theorem 4) that learns only $O\left(K^2\right)$ targets from scratch and on the remaining targets, evaluates $O\left(K^2 + d\right)$ features per example. In this section, we present the corresponding IMPROVEREP algorithm. In the following discussion, we will use the term suffix to denote a subtree (i.e., sublist) of a list. In other words, a suffix of a list would be a path beginning anywhere on the list and ending at the leaf. Similarly, we use the term prefix to denote a path beginning at the root of the list and ending anywhere on the list.

Recall that for USEREP we use the same algorithm we did for decision trees, namely Algorithm 1. However, for IMPROVEREP we present Algorithm 7 which is simpler than Algorithm 2. More specifically, given a decision list g learned from scratch (that we could not learn from $\tilde{\mathcal{F}}$), we examine g and the actual list $\tilde{g}$ we learned from $\tilde{\mathcal{F}}$. Then we simply ignore the first few nodes of g that we managed to learn using $\tilde{\mathcal{F}}$, and add the remaining suffix to $\tilde{\mathcal{F}}$. The intuition is that the representation is improved by introducing a part of g that we could not learn using $\tilde{\mathcal{F}}$. Note that here $\tilde{g}$ might not even be a complete decision list as USEREP may have simply failed in finding a decision list using $\tilde{\mathcal{F}}$ that fits the data. However, it may have still been successful in learning the first few nodes of g.

Algorithm 7 IMPROVEREP - Decision Lists

1: **Input**: Old representation $\tilde{\mathcal{F}}_{\text{old}}$, target g learned from scratch, $\tilde{g}$ learned using $\tilde{\mathcal{F}}_{\text{old}}$.
2: Let $g = (g_p, g_s)$ where g_p is the longest common prefix of $\tilde{g}$ and g.
3: $\tilde{\mathcal{F}} \leftarrow \tilde{\mathcal{F}}_{\text{old}} \cup \{g_s\}$
4: Return $\tilde{\mathcal{F}}$

We now show that the resulting protocol learns only $O\left(K^2\right)$ targets from scratch. We will use the notation $u \in_{\texttt{dl}} f$ to denote that node u is present in the list f, and $f' \subset_{\texttt{dl}} f$ to denote that f' is an incomplete list (like an incomplete decision tree) which corresponds to a path within f, not necessarily a prefix or a suffix. Furthermore, if g is a concatenation of other lists $g_1, g_2, \ldots$ we will say $g = (g_1, g_2, \ldots)$.

Lemma 13 *In the model of Problem Setup 1 for decision lists, the (*USEREP *Algorithm 1 ,* IMPROVEREP *Algorithm 7)-protocol for decision lists learns $O\left(K^2\right)$ targets from scratch.*

Proof

We need to understand how adding the suffix g_s from a target g on which USEREP failed, makes the representation more useful. As a warm up, we can show that when the protocol faces the same target g in the future, the updated representation $\tilde{\mathcal{F}} = \tilde{\mathcal{F}}_{\text{old}} \cup \{g_s\}$ will be able to learn it. A crucial fact from which this follows is that USEREP Algorithm 1 *learns any list if and only if the list can be represented as a concatenation of prefixes of elements from $\tilde{\mathcal{F}}$.* This fact holds because Lemma 10 and the way the algorithm works. Thus, since we were able to learn g_p when we first saw g, g_p is a concatenation of prefixes from $\tilde{\mathcal{F}}_{\text{old}}$ i.e., $g_p \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}_{\text{old}}))$. Then, since $g = (g_p, g_s) \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}_{\text{old}} \cup \{g_s\}))$, we can learn g using $\tilde{\mathcal{F}}$.

Of course, we should show that the updated representation is more powerful than just allowing us to learn repeated tasks in the future. To see how, note that since the target g is a concatenation of metafeatures from $\mathcal{F}$, its suffix g_s must begin with the suffix of a metafeature from $\mathcal{F}$. More formally, since $g \in \mathbb{DT}(\mathcal{F})$, g_s must begin with a suffix f_s of an element $f \in \mathcal{F}$. Let f_p be the corresponding prefix of f. Now, consider a future target that contains f. If the learner is able to identify all nodes in the target upto the end of prefix f_p, the learner is also guaranteed to identify f completely in the target. This tells us a little bit more about the power of the updated representation.

Now, to prove our lemma, we use the fact that each failure of USEREP Algorithm 1 must correspond to a specific element $f \in \mathcal{F}$ as seen above. That is, there must exist an $f = (f_p, f_s) \in \mathcal{F}$ such that $f \subseteq_{\texttt{dl}} g$ and furthermore, USEREP was able to learn upto a prefix f_p of f after which it failed. We claim that there can only be $O(K)$ failures of USEREP that corresponds to a particular f in this manner. From here, our lemma immediately follows. To prove this claim, we will categorize the failures of USEREP corresponding to f into two different cases and bound the number of failures in each case. Throughout the following discussion, we will simply use the term failure to denote failure of USEREP.

We will divide failures corresponding to f based on whether f_p can be represented as a concatenation of prefixes from $\tilde{\mathcal{F}}_{\text{old}}$ or not. If it can be, we show that it is easy to argue that in any future target there will not be a failure corresponding to f. If not, we present a more involved argument to show that there can be at most K failure events corresponding to a particular f. Then, the bound of $O\left(K^2\right)$ on the total number of failures follows.

Case 1: For the first case we assume that $f_p \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}_{\text{old}}))$. Then, clearly, this is true for the new representation $\tilde{\mathcal{F}}$ i.e., $f_p \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. Furthermore, since there is a new element g_s with f_s as its prefix, $f_s \in \mathsf{Pref}(\tilde{\mathcal{F}})$. This implies that $f \in \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}))$. This means that we can henceforth learn an occurrence of f in a new target if learning has been successful until the beginning of f in that target. In other words, there can never be another failure that corresponds to f. This case can hence occur only once.

Case 2: The second case corresponds to $f_p \notin \mathbb{DT}(\mathsf{Pref}(\tilde{\mathcal{F}}_{\text{old}}))$. We will now subdivide this case further based on another metafeature $f' \in \mathcal{F}$, a part of which lies in some hypothesized metafeature in $\tilde{\mathcal{F}}_{\text{old}}$ and was used to learn/match a part of f in g_p. We will fix f' and argue that there can be at most two failure events characterized by f and f' during the lifelong learning protocol. Since there are only K different f', then for a fixed f, there can only be $2K$ failure events of this type, thus completing our proof.

We begin by informally explaining how we choose f' to classify a given failure event. We first note that there are two ways in which g_p can be represented in terms of the true metafeatures $\mathcal{F}$. The "direct" representation corresponds to the fact that $g \in \mathbb{DT}(\mathcal{F})$. On the other hand, there is also an "indirect" representation: since Algorithm 1 could learn the prefix g_p using $\tilde{\mathcal{F}}_{\text{old}}$, g_p can be represented as a sequence of prefixes from $\tilde{\mathcal{F}}_{\text{old}}$. Since each element in $\tilde{\mathcal{F}}_{\text{old}}$ are parts of older targets from $\mathbb{DT}(\mathcal{F})$, we can represent this sequence of prefixes in terms of parts of true metafeatures (that are not necessarily prefix/suffix parts).

Now, let the root variable of f be i_f. There must be a unique element in the sequence of prefixes that contains i_f. *We let f' be the metafeature in $\mathcal{F}$ that contributes to the last bit of this unique element in the above-described indirect representation.* Before we proceed to describe this more formally, we note that this is all possible only because i_f indeed belongs to f_p. If it did not, it means f_p is an empty string, which we have dealt with in Case 1.

We now state our choice of f' more formally. Since we were able to learn g_p using $\tilde{\mathcal{F}}_{\text{old}}$ we can write $g_p = (\mathsf{Pref}_\star(\tilde{f}_{l_1}), \mathsf{Pref}_\star(\tilde{f}_{l_2}), \ldots)$ for $\tilde{f}_{l_1}, \tilde{f}_{l_2}, \ldots \in \tilde{\mathcal{F}}_{\text{old}}$ where we use the notation $\mathsf{Pref}_\star(\tilde{f})$ to denote a particular prefix of $\tilde{f}$. Let $\mathsf{Pref}_\star(\tilde{f}_{l_r})$ be the unique element in the above sequence that contains i_f (we use the index r to denote that it contains the root). Like we stated before, since $\tilde{f}_{l_r}$ is also the suffix of some old target in $\mathbb{DT}(\mathcal{F})$, $\tilde{f}_{l_r}$ must be made up of parts of true metafeatures $\mathcal{F}$. The same holds for $\mathsf{Pref}_\star(\tilde{f}_{l_r})$ too. We will focus on the true metafeature that makes up the last bit of $\mathsf{Pref}_\star(\tilde{f}_{l_r})$. That is, let $f' \in \mathcal{F}$ be the metafeature that occurs in an older target, such that a non-empty suffix of $\mathsf{Pref}_\star(\tilde{f}_{l_r})$ comes from f' i.e., there exists suffix $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ such that $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r})) \subseteq_{\texttt{dl}} f'$. Here, again $\mathsf{Suff}_\star(\tilde{f})$ is used to denote a particular suffix of $\tilde{f}$. Thus each failure event in this case can be characterized by a particular f and f'.

Note that $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ need not necessarily be a suffix of f' because $\tilde{f}_{l_r}$ may have stopped matching with g somewhere in the middle of f'. It need not necessarily be a prefix of f' either because $\tilde{f}_{l_r}$ is only a suffix of some target in $\mathbb{DT}(\mathcal{F})$ and this suffix may have begun somewhere in the middle of f' in that target.

To show that there are at most two failure events for a given f and f', we will consider two subcases depending on whether $i_f \notin_{\texttt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$. That is, when we use a part of f' to learn g_p, we see whether we learn i_f or not. These two cases are illustrated in Figure 4 and Figure 5. For both these scenarios, we first analyze the structure behind the failure i.e., the locations of the different variables and how the different metafeatures align with each other. Based on this, we show that for each type, there can be at most one failure.

Case 2a: $i_f \notin_{\texttt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$. Let us call this an $(f, f')^{(1)}$-type failure event. We first look at how the different elements are positioned when such a failure occurs, by aligning the elements in a way that the variables match. First, recall that by the definition of $\mathsf{Pref}_\star(\tilde{f}_{l_r})$, $i_f \in_{\texttt{dl}} \mathsf{Pref}_\star(\tilde{f}_{l_r})$. Thus, $i_f \in_{\texttt{dl}} \tilde{f}_{l_r}$. Furthermore, by definition of f', and because $\tilde{f}_{l_r}$ is the suffix of an older target from $\mathbb{DT}(\mathcal{F})$, either a suffix or the whole of f' must occur in $\tilde{f}_{l_r}$. We claim that 1) it is the latter, i.e., $f' \subseteq_{\texttt{dl}} \tilde{f}_{l_r}$ and furthermore, 2) the root of f' is located below i_f in $\tilde{f}_{l_r}$ (as illustrated in Figure 4). If only a suffix of f' occurred in $\tilde{f}_{l_r}$, it means that $\tilde{f}_{l_r}$ begins with that particular suffix and therefore by definition of $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ being the last bit of $\mathsf{Pref}_\star(\tilde{f}_{l_r})$ that comes from f', $\mathsf{Pref}_\star(\tilde{f}_{l_r}) = \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$. Then, since $i_f \in_{\texttt{dl}} \mathsf{Pref}_\star(\tilde{f}_{l_r})$, $i_f \in_{\texttt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ which is a contradiction. Now, if indeed $f' \subseteq_{\texttt{dl}} \tilde{f}_{l_r}$ but the root of f' was not located below i_f in $\tilde{f}_{l_r}$ again by definition of $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ being the last bit of $\mathsf{Pref}_\star(\tilde{f}_{l_r})$ that comes from f', $i_f \in_{\texttt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ which is a contradiction. Note that conclusions 1) and 2) above mean that $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ is a prefix of f'.

Given this, assume on the contrary that we do face a later target g' with an $(f, f')^{(1)}$-type failure event. Then, we can define notations similar to the first failure. Let g'_p be the prefix we were able to learn correctly using $\tilde{\mathcal{F}}$. Then, g'_p can be similarly expressed as a sequence of prefixes from $\tilde{\mathcal{F}}$, say $(\mathsf{Pref}'_\star(\tilde{f}_{l'_1}), \mathsf{Pref}'_\star(\tilde{f}_{l'_2}), \ldots)$. By definition of this failure type, $f \subseteq_{\texttt{dl}} g'$. So consider the prefix that contains i_f, call it $\mathsf{Pref}'_\star(\tilde{f}_{l'_{r'}})$. Furthermore, $\mathsf{Pref}'_\star(\tilde{f}_{l'_{r'}})$ has a suffix $\mathsf{Suff}'_\star(\mathsf{Pref}'_\star(\tilde{f}_{l'_{r'}}))$ that is also a part of f' but is not necessarily the same as $\mathsf{Suff}'_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$.

We will now show that a prefix longer than g'_p that includes f completely can be represented using prefixes from $\tilde{\mathcal{F}}$ which contradicts the fact that the algorithm failed somewhere in between f. To do this, we will make use of the fact that the algorithm was able to learn until $i_{f'}$ in the second failure, beyond which it can learn the rest of the target until the end of f_p like it did the previous time, after which we can append f_s from the representation. More specifically, observe that there

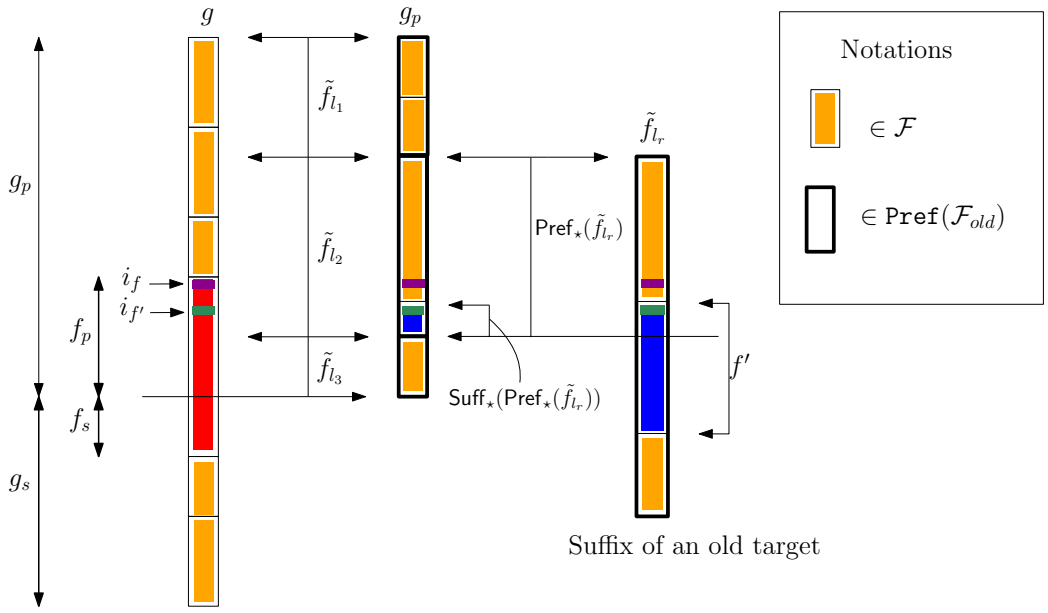

Figure 4: $(f, f')^{(1)}$**-type failure where** $i_f \notin_{\mathtt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$**:** We represent the decision list g on the left. Each subrectangle in this corresponds to some element from $\mathcal{F}$ with f marked in red. In the middle column, we represent the prefix of g, g_p in terms of the elements of $\mathsf{Pref}(\tilde{\mathcal{F}}_{\text{old}})$ each denoted by a thick subrectangle. We can do this because we were able to learn g_p from $\tilde{\mathcal{F}}$. Now each of these thick subrectangles can in turn be represented using parts of metafeatures from $\mathcal{F}$ because these are suffixes of actual targets. In particular, we choose the thick subrectangle that matched with the root of f and show the complete metafeature from $\tilde{\mathcal{F}}_{\text{old}}$ on the right. In this metafeature, the thin rectangles correspond to its representation in $\mathcal{F}$. Observe that we have marked f' in blue, and a part of it is what makes the last bit in the rectangle marked as $\mathsf{Pref}_\star(f_{l_r})$ in g_p. Also i_f is marked in magenta below which $i_{f'}$ is marked in green.

is exactly one position at which $i_{f'}$ in f' can match with f and hence the failure will look similar to Figure 4 again; f' will be contained in $\tilde{f}_{l'_{r'}}$ and i_f will be located above $i_{f'}$. Now, since we also know that $f' \subseteq_{\texttt{dl}} \tilde{f}_{l'_{r'}}$, we can extend/shorten the prefix $\mathsf{Pref}'_\star(\tilde{f}_{l'_{r'}})$ that is used to match with g'_p to another prefix $\mathsf{Pref}''_\star(\tilde{f}_{l'_{r'}})$ that has the same suffix as before, $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$. On doing this, the rest of f_p in g'_p can be represented using the same prefixes from $\tilde{\mathcal{F}}$ used to represent that part in g_p. Furthermore, we can append f_s to this sequence because f_s is a prefix of g_s that was added to the representation. Thus, we take the sequence $(\mathsf{Pref}'_\star(\tilde{f}_{l'_1}), \mathsf{Pref}'_\star(\tilde{f}_{l'_2}), \ldots)$ 1) we retain the first $r'-1$ elements, 2) modify the r'th element so that its suffix matches with $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$, 3) append the rth, $r+1$th, ... elements from the representation for g_p, 4) and finally append f_s. This represents a larger prefix of g that includes f completely, using only prefixes from $\tilde{\mathcal{F}}$. Namely, this is $(\mathsf{Pref}'_\star(\tilde{f}_{l'_1}), \mathsf{Pref}'_\star(\tilde{f}_{l'_2}), \ldots \mathsf{Pref}''_\star(\tilde{f}_{l'_{r'}}), \mathsf{Pref}_\star(\tilde{f}_{l_{r+1}}), \mathsf{Pref}_\star(\tilde{f}_{l_{r+2}}), \ldots, f_s)$. This contradicts the fact we failed to learn f completely in g'.

Case 2b: $i_f \in_{\texttt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$. Let us call this an $(f, f')^{(2)}$-type failure event. We now make a similar argument. The only difference is that now $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ is not necessarily a prefix of f' and therefore, $i_{f'}$ is not necessarily present in $\mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$ (see Figure 5. However it is guaranteed that a suffix of f' containing i_f is present in $\tilde{f}_{l_r}$. Now let $\mathsf{Suff}_{\star\star}(\mathsf{Pref}_\star(f_{l_r}))$ be an alternative shorter suffix of $\mathsf{Pref}_\star(f_{l_r})$ that begins only at i_f.

Now, consider a new target with a similar failure with a similar $\mathsf{Suff}'_{\star\star}(\mathsf{Pref}'_\star(f_{l'_{r'}}))$ that begins with i_f. We will again show how we can use the updated representation to represent a larger prefix of g' , specifically a prefix that extends until the end of f in g'. In particular, we make use of the fact that the algorithm was able to learn at least before i_f in this target, beyond which we can learn f_p the way we did in the previous target, and then append f_s from the representation. More specifically, we first extend/shorten the prefix $\mathsf{Pref}'_\star(f_{l'_{r'}})$ that is used to match with g'_p to another prefix $\mathsf{Pref}''_\star(f_{l'_{r'}})$ that it has the suffix $\mathsf{Suff}_{\star\star}(\mathsf{Pref}_\star(f_{l_r}))$ (which is only possible because $i_f \in_{\texttt{dl}} \mathsf{Pref}''_\star(f_{l'_{r'}})$). On doing this, we can represent the rest of f using $\tilde{\mathcal{F}}$ like in the previous case.

Thus, we take the sequence $(\mathsf{Pref}'_\star(\tilde{f}_{l'_1}), \mathsf{Pref}'_\star(\tilde{f}_{l'_2}), \ldots)$ 1) we retain the first $r'-1$ elements, 2) modify the r'th element, 3) append the rth, $r+1$th, ... elements from the representation for g_p, 4) and finally append f_s. This represents a larger prefix of g that includes f completely, using only prefixes from $\tilde{\mathcal{F}}$. Namely, this is $(\mathsf{Pref}'_\star(\tilde{f}_{l'_1}), \mathsf{Pref}'_\star(\tilde{f}_{l'_2}), \ldots \mathsf{Pref}''_\star(\tilde{f}_{l'_{r'}}), \mathsf{Pref}_\star(\tilde{f}_{l_{r+1}}), \mathsf{Pref}_\star(\tilde{f}_{l_{r+2}}), \ldots, f_s)$. This contradicts the fact that we failed to learn f completely in g'.

■

Appendix D. Monomials

In this section, we elaborate on our results for lifelong learning of monomials and polynomials. In Appendix D.1, we present a simple algorithm for learning monomials exactly from scratch under some assumptions. Then in Appendix D.2, we present proofs from Section 4. Finally, in Appendix D.3 we provide a detail discussion of sparse polynomials.

Recall that for any input $\mathbf{x} = (x_1, x_2, \ldots x_N) \in \mathbb{R}^N$, we denote the output of a d-degree target monomial $\mathbf{g} = (g_1, g_2, \ldots, g_N)$ by the function $P_{\mathbf{g}}(\mathbf{x}) = x_1^{g_1} x_2^{g_2} \ldots x_N^{g_N}$ where $g_i \in \mathbb{N} \cup \{0\}$ and the degree $\sum_i g_i \leq d$. We denote the unknown metafeature set $\mathcal{F} = \{f_1, f_2, \ldots\}$ also as a matrix where column i is f_i. Therefore, saying that $\mathbf{g}$ can be expressed using $\mathcal{F}$ is equivalent to saying $\mathbf{g}$

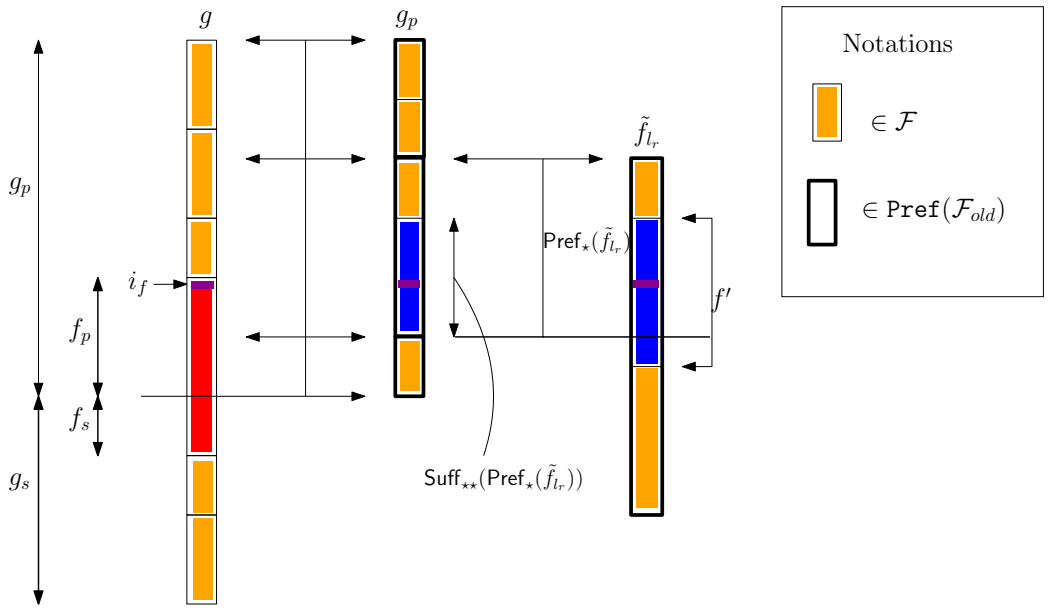

Figure 5: $(f, f')^{(2)}$**-type failure where** $i_f \in_{\mathtt{dl}} \mathsf{Suff}_\star(\mathsf{Pref}_\star(\tilde{f}_{l_r}))$

lies in the column space of $\mathcal{F}$ denoted by $\mathbb{C}(\mathcal{F})$. Then for any $k-$rank ($k \leq K$), $N \times k$ matrix $\tilde{\mathcal{F}}$ and for any $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$, we define $\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g}) \in \mathbb{R}^k$ to denote the unique vector of column weights such that $\tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g}) = \mathbf{g}$.

D.1. Learning Monomials from scratch

Recall that for each monomial target, we assumed that $\mathcal{D}^{(j)}$ is a product distribution i.e., the features are independent. We now state some specific assumptions about $\mathcal{D}^{(j)}$. In particular, we assume that the variance of each variable x_i is not too small. The rationale is that if the variance was very small (in the extreme case, imagine x_i being a constant), the factor $x_i^{g_i}$ would essentially be a constant factor in the monomial target. While it may be possible to design a more careful learning algorithm that can extract these nearly constant factors, that is beyond the scope of our discussion.

Secondly, we assume that the probability density function is finite at every point i.e., the probability distribution is not too concentrated at any point. We will use this assumption to apply Lemma 20 when we draw a single sample to verify whether the monomial we have learned matches the true monomial.

Finally, we assume that the support of x_i is $[1, 2]$. While the upper bound of 2 is to simplify our discussion, the lower bound is to avoid dealing with values of x_i that are close to zero. This is essential because as we will see later, we will deal with logarithmic values of x_i in the learning process. We now state our assumptions formally.

Assumption 1 *Each $\mathcal{D}^{(j)}$ is a product distribution. Let $\mathcal{D}^{(j)} = \mu_1^{(j)} \times \cdots \times \mu_N^{(j)}$. We assume that for all features i:*

- ***Minimum variance*** $Var_{\mu_i^{(j)}}(\log x_i) \geq c$.
- ***Bounded probability density*** $\forall x_i \in \mathbb{R}$, $\mu_i^{(j)}(x_i) \in \mathbb{R}$.

- ***Bounded support*** *The support of* $\mu_i^{(j)}$ *is* $[1, 2]$.

We now present our simple poly-time technique for learning monomials from scratch with polynomially many samples. Recall that the output of the monomial $\mathbf{g}$ on an input $\mathbf{x}$ is denoted by $P_{\mathbf{g}}(\mathbf{x})$. Let us denote the logarithm of this output $\log |P_{\mathbf{g}}|$ by $Q_{\mathbf{g}}$. Observe that learning $\mathbf{g}$ is equivalent to learning the coefficients of the 'linear' function $Q_{\mathbf{g}}$. To see how this can be done, we will define a notion of correlation/inner product of two functions $h(\mathbf{x})$ and $h'(\mathbf{x})$:

$$\langle h(\mathbf{x}), h'(\mathbf{x}) \rangle \triangleq \mathbb{E}[h(\mathbf{x})h'(\mathbf{x})].$$

Then, we claim that g_i can be expressed as the following inner product.

Lemma 14

$$\frac{\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)] \rangle}{\mathbb{E}[\log^2 x_i] - \mathbb{E}^2[\log x_i]} = g_i$$

Proof Since x_i is picked independent of the other variables, so is the random variable $(\log x_i - \mathbb{E}[\log(x_i)])$. Thus, when $j \neq i$

$$\mathbb{E}[\log x_j (\log x_i - \mathbb{E}[\log(x_i)])] = \mathbb{E}[\log x_j] \times \mathbb{E}[\log x_i - \mathbb{E}[\log(x_i)]] = 0$$

However,

$$\mathbb{E}[\log x_i (\log x_i - \mathbb{E}[\log(x_i)])] = \mathbb{E}[\log^2 x_i] - \mathbb{E}^2[\log x_i]$$

Then, the claim follows from our definition of $Q_{\mathbf{g}}$. ■

Observe that using the above fact, we can calculate g_i for each $i \in [N]$ exactly if we were provided the exact values of each correlation term in the equality. However, the best we can hope for is to approximate these terms using sufficiently many samples. Fortunately, we can actually approximate each of these correlation terms to a small constant error such that these errors together imply a constant error smaller than $1/2$ in estimating g_i. Then we can round off our estimate to the closest natural number to find the exact value of g_i. We now summarize our simple algorithm for learning a monomial from scratch, and then prove our polynomial sample complexity bound.

Algorithm 8 Learning a monomial from scratch

1: Input: Distribution $\mathcal{D}$ over $\mathbb{R}^N$
2: Draw S samples $(\mathbf{x}, P_{\mathbf{g}}(\mathbf{x}))$ from $\mathcal{D}$ and query all the features on all samples.
3: **for** $i = 1, 2, \ldots N$ **do**
4: Estimate $\mathbb{E}[\log^2 x_i]$, $\mathbb{E}[\log^2 x_i] - \mathbb{E}^2[\log x_i]$, and $\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)] \rangle$ empirically.
5: Round off

$$\frac{\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)] \rangle}{\mathbb{E}[\log^2 x_i] - \mathbb{E}^2[\log x_i]}$$

to estimate g_i.
6: Return $\tilde{\mathbf{g}}$

Clearly the above algorithm has polynomial running time and sample complexity as long as S is polynomial. The crucial guarantee we need now is that polynomially many samples are sufficient

to estimate each g_i exactly, which we show in Theorem 17. We first begin by bounding the error in estimating the numerator $\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)]\rangle$ in Lemma 15. Then, in Lemma 16 we show how this error and the error in the denominator terms, add up to result in an error of at most $1/2$ in estimating g_i. Using these, we prove in Theorem 17 that the algorithm estimates each power exactly. In the following notation we will use $\tilde{\mathbb{E}}$ to denote the empirical estimate of an expected value.

Lemma 15 *Using a sample set $\mathcal{S}$ of size $O\left(\frac{d}{\epsilon_3^2}\log\frac{1}{\delta'}\right)$, for a given $i \in [N]$, if $|\tilde{\mathbb{E}}[\log x_i] - \mathbb{E}[\log x_i]| \leq \epsilon_1$, then we can guarantee that*

$$Pr\left[\left|\frac{1}{|S|}\sum_{\mathbf{x}\in S} Q_{\mathbf{g}}(\mathbf{x})(\log(x_i) - \tilde{\mathbb{E}}[\log(x_i)]) - \langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)]\rangle\right| \leq d\epsilon_1 + \epsilon_3\right] = O\left(\delta'\right)$$

.

Proof Consider the random variable $Q_{\mathbf{g}}(\mathbf{x}) \cdot (\log(x_i) - \tilde{\mathbb{E}}[\log(x_i)])$. It is easy to show that $Q_{\mathbf{g}}(\mathbf{x})\log(x_i) \in [0, d]$ with the extreme values attained at $\mathbf{x} = (2, 2, \ldots)$ and $\mathbf{x} = (1, 1, \ldots)$. Then, $Q_{\mathbf{g}}(\mathbf{x})\mathbb{E}[\log(x_i)] \in [0, d]$. Thus, the random variable $Q_{\mathbf{g}}(\mathbf{x}) \cdot (\log(x_i) - \tilde{\mathbb{E}}[\log(x_i)])$ lies in a range of size $2d$. Then, by Chernoff bounds, we can show that

$$Pr\left[\left|\frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}} Q_{\mathbf{g}}(\mathbf{x})(\log(x_i) - \tilde{\mathbb{E}}[\log(x_i)]) - \langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \tilde{\mathbb{E}}[\log(x_i)]\rangle\right| \leq \epsilon_3\right] = O\left(\delta'\right)$$

from which the above claim follows because the absolute difference between $\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)]$ and $\langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \tilde{\mathbb{E}}[\log(x_i)]$ is at most $\left|\max_{\mathbf{x}} Q_{\mathbf{g}}(\mathbf{x}) \cdot (\mathbb{E}[\log(x_i)]) - \tilde{\mathbb{E}}[\log(x_i)]))\right| \leq d\epsilon_1$ (because the first term is at most d and the next is at most ϵ_1). ■

Lemma 16 *Using a sample set $\mathcal{S}$ of size $O\left(\frac{d}{\left(\min(\frac{c^2}{d}, \frac{c}{d}, 1)\right)^2}\log\frac{1}{\delta'}\right)$ with a high probability of $1 - \delta'$ for a given $i \in [N]$ we can learn $\tilde{g}_i$ such that $|\tilde{g}_i - g_i| \leq \frac{1}{2}$.*

Proof Let ϵ_1 and ϵ_3 be as defined in Lemma 15. Additionally let $|\tilde{\mathbb{E}}[\log^2 x_i] - \mathbb{E}[\log^2 x_i]| \leq \epsilon_2$. From the previous results and from Chernoff bounds, we have that $\epsilon_1, \epsilon_2, \epsilon_3$ are all $O\left(\min(\frac{c^2}{d}, \frac{c}{d}, 1)\right)$ given the size of $\mathcal{S}$. We now have a fractional expression on the right hand side of the equation in Lemma 14 for which we can derive the error in estimating the numerator and the denominator individually. We need to show that the overall error in estimating the fraction is $1/2$ i.e., $O(1)$. Now, the error in estimating some fraction $\frac{G}{H}$ using $\frac{\tilde{G}}{\tilde{H}}$ given that $|G - \tilde{G}| \leq \epsilon_G$ and $|H - \tilde{H}| \leq \epsilon_H$ can be upper bounded by:

$$\begin{aligned}\left|\frac{G\pm\epsilon_G}{H\pm\epsilon_H} - \frac{G}{H}\right| &= \left|\frac{\epsilon_G}{H} \pm \frac{G\epsilon_H}{(H-\epsilon_H)H}\right| \\ &\leq \frac{\epsilon_G}{\min H} + \frac{(\max G+\epsilon_G)\epsilon_H}{(\min H-\epsilon_H)\min H}\end{aligned}$$

In our case, we have $H = \mathbb{E}[\log^2 x_i] - \mathbb{E}^2[\log x_i]$ and $G = \langle Q_{\mathbf{g}}(\mathbf{x}), \log(x_i) - \mathbb{E}[\log(x_i)]\rangle$, $\min H = c$ and $\max G = d$. Also, $\epsilon_G = \epsilon_1 d + \epsilon_3$ and $\epsilon_H \leq \epsilon_2 + 2\epsilon_1 + \epsilon_1^2$. The latter inequality follows from

the fact that the error in estimating $\mathbb{E}[\log^2 x_i]$ is ϵ_2 and the error in estimating $\mathbb{E}^2[\log x_i]$ is at most $(\mathbb{E}[\log x_i] + \epsilon_1)^2 - \mathbb{E}^2[\log x_i] \leq \epsilon_1(2\mathbb{E}[\log x_i] + \epsilon_1) \leq \epsilon_1(2 + \epsilon_1)$. By a simple calculation, it can be verified that this results in a total error of $O(1)$ in estimating g_i. ■

Theorem 17 *Algorithm 8 exactly learns a target* $\mathbf{g}$ *from scratch with high probability* $1 - O\left(\frac{\delta}{K}\right)$ *with* $S = O\left(\frac{d}{\left(\min(\frac{c^2}{d}, \frac{c}{d}, 1)\right)^2} \log \frac{Nm}{\delta}\right)$ *samples.*

Proof From Lemma 16 we have that each g_i is accurately estimated with probability at least $1 - O\left(\frac{\delta}{Nm}\right)$. By a union bound, $\mathbf{g}$ is accurately estimated with probability at least $1 - O\left(\frac{\delta}{m}\right)$. ■

D.2. Proofs from Section 4

We first present our straightforward approach for lifelong learning which merely keeps a record of features that have been seen in earlier targets.

Theorem 18 (Naive lifelong learning of monomials) *In the model of Problem Setup 2, there exists a naive algorithm for lifelong learning of monomials that evaluates* $O(S(KN + mKd))$ *features overall.*

Proof Sketch (Theorem 18) We use IMPROVEREP Algorithm 9 that essentially stores the list of targets that have been learned from scratch as the columns of the matrix $\tilde{\mathcal{F}}$. Now, consider the set of features that have been "seen" so far i.e., these correspond to rows in $\tilde{\mathcal{F}}$ that have at least one non-zero entry. Then, for a new target $\mathbf{g}$, we define a USEREP algorithm that determines the powers of only these features. This can be done by evaluating only those features on the data set using the technique in Algorithm 8. The unseen features are assumed to have zero power.

Now, consider a new target $\mathbf{g}$ that is "linearly dependent" on the targets that have been learned so far i.e., $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$. In this case, the unseen features should have a zero exponent in $\mathbf{g}$ as it is zero in all earlier targets. Thus, our USEREP technique would not fail on such targets. Now, if $\mathbf{g}$ was linearly independent, it is possible that an unseen feature has a non-zero exponent in $\mathbf{g}$. To verify whether this is the case, we can draw a single sample and check whether our prediction matches the true output. If this fails, we learn the target correctly from scratch and add it to $\tilde{\mathcal{F}}$.

Thus, since we add only linearly independent targets to $\tilde{\mathcal{F}}$, in a manner similar to the proof of Theorem 5, we can show that USEREP will not fail more than K times. Our result follows from here because each of the targets that we learn from scratch have at most d non-zero exponents. Then, in total we only have at most Kd "seen" features i.e., features with non-zero powers that we always examine. ■

We now prove our main result for monomials, Theorem 5 where we presented a lifelong learning protocol that makes $O(S(KN + mK) + md)$ feature evaluations overall. We first prove guarantees about USEREP Algorithm 10 below in Lemma 19. Note that this is similar in spirit to Theorem 3 for decision trees. The reader can recall the proof sketch of Theorem 5 for an informal description of USEREP Algorithm 10.

Lemma 19 *Let $\tilde{\mathcal{F}}$ be an $N \times k$ matrix. Then, with high probability $1 - O\left(\frac{\delta}{m}\right)$, a) if $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$, then Algorithm 10 correctly learns and outputs $\tilde{\mathbf{g}} = \mathbf{g}$ b) if Algorithm 10 does output some $\tilde{\mathbf{g}}$, then $\tilde{\mathbf{g}} = \mathbf{g}$, c) Algorithm 10 examines only at most k features per sample point and at most d features on a single sample.*

Proof a. Given that $\tilde{\mathcal{F}}$ is of rank k, then if $\mathbf{g} \in \mathbb{C}(\tilde{\mathcal{F}})$, there exists a unique solution for $\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g})$ in $\tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g}) = \mathbf{g}$. Note that this is a system of N linear equations in k. Therefore, if the Algorithm picked any set of k linearly independent rows $\mathcal{I} = \{i_1, i_2, \ldots i_k\}$ from $\tilde{\mathcal{F}}$, there must exist a unique solution to $\tilde{\mathcal{F}}[\mathcal{I}]\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) = \mathbf{g}[\mathcal{I}]$ where the solution is $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) = \mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g})$. Thus, solving this system will give us the value of $\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g})$ from which we can compute $\mathbf{g}$ correctly using $\tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g}) = \mathbf{g}$. This however requires that we determine the values of $g_{i_1}, g_{i_2}, \ldots, g_{i_k}$ from scratch, which we can do accurately with high probability of $1 - O\left(\frac{\delta}{m}\right)$ from Lemma 16 using polynomially many samples.

b. To prove our second claim, observe that the only event in which the learner may potentially have an incorrect output is when $\mathbf{g} \notin \mathbb{C}(\tilde{\mathcal{F}})$ but we still do learn a $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}$ because it so happens that $\mathbf{g}[I] \in \mathbb{C}(\tilde{\mathcal{F}}[\mathcal{I}])$. However, $\tilde{\mathbf{g}} = \tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) \neq \mathbf{g}$. If $\tilde{g}$ has a degree greater than d, the algorithm halts with failure. Otherwise, we can show using Lemma 20 that by drawing a single sample and checking whether $P_{\tilde{\mathbf{g}}}(\mathbf{x}) = P_{\mathbf{g}}(\mathbf{x})$ we can conclude whether $\mathbf{g} = \tilde{\mathbf{g}}$.

c. This follows directly from the design of the algorithm: we examine only K features on all samples, and then on a single new sample we examine features relevant to $\tilde{\mathbf{g}}$ provided $\tilde{\mathbf{g}}$ has degree at most d. ■

Recall that for monomials we stated that using USEREP Algorithm 10 and IMPROVEREP Algorithm 9 we learn at most K targets from scratch. On each of the remaining targets, we examine only K features per example, and d features on exactly one sample.

Theorem 5 *The (USEREP Algorithm 10, IMPROVEREP Algorithm 9)-protocol for monomials makes $O\left(S(KN + mK) + md\right)$ feature evaluations overall and runs in time $poly(m, N, K, S, d)$.*

Proof Applying Lemma 19 over at most m problems, we have that with probability $1 - O\left(\delta\right)$, every target added to $\tilde{\mathcal{F}}$ increases the rank of $\tilde{\mathcal{F}}$ by one as it does not lie in the column space of the rest. Assume we fail to learn from our representation on more than K targets. This means that there will be at least $K + 1$ targets (that is the columns of $\tilde{\mathcal{F}}$) that are linearly independent. However, since all targets belong to $\mathbb{C}(\mathcal{F})$, there cannot be more than K targets that are linearly independent. Thus, we achieve a contradiction. Now, since we learn only at most K targets from scratch, applying Theorem 17 over these we get that we learn them correctly with probability $1 - O\left(\delta\right)$. Also, since $\tilde{\mathcal{F}}$ has at most K columns, from Lemma 19 we have that each time we learn using the representation, we examine K features per example. Besides, we examine d features that are relevant to $\mathbf{g}$ in Step 8. ■

We note that it is easy to refine our application of union bounds to use slightly fewer samples than in the bound of Theorem 17. In particular, it is possible bring the $\log Nm$ factor down to $\log NK$ while learning from scratch, and to $\log Km$ on all other targets.

Algorithm 9 IMPROVEREP - Monomials

1: Input: Old representation $\tilde{\mathcal{F}}_{\text{old}}$ and $\mathbf{g}$ learned from scratch
2: Return $\tilde{\mathcal{F}} = [\tilde{\mathcal{F}}_{\text{old}}, \mathbf{g}]$

Algorithm 10 USEREP - Learning a Monomial from Metafeatures

1: Input: Metafeatures $\tilde{\mathcal{F}} = [\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_k]$ ($k \leq K$), sample set $\mathcal{S}$ of size S.
2: Halt with failure if $\tilde{\mathcal{F}}$ is empty.
3: Let $\mathcal{I}$ be the indices of those rows in $\tilde{\mathcal{F}}$ that are linearly independent and let $\tilde{\mathcal{F}}[\mathcal{I}]$ be the corresponding $k \times k$ sub-matrix of $\tilde{\mathcal{F}}$.
4: Examine features $\mathcal{I}$ on all samples and use Lemma 16 to learn and round off estimates $\tilde{g}_i$ for each $i \in \mathcal{I}$.
5: Solve for $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$ in $\tilde{\mathcal{F}}[\mathcal{I}]\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) = \mathbf{g}[\mathcal{I}]$. If no solution exists, halt with failure.
6: Estimate $\tilde{\mathbf{g}} \leftarrow \tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$.
7: Halt with failure if the degree of $\tilde{\mathbf{g}}$ is greater than d.
8: Draw a single sample $(\mathbf{x}, P_{\mathbf{g}}(\mathbf{x}))$, examine the features relevant to $\tilde{\mathbf{g}}$. If $P_{\mathbf{g}}(\mathbf{x}) \neq P_{\tilde{\mathbf{g}}}(\mathbf{x})$, halt with failure.
9: Return $\tilde{\mathbf{g}}$.

D.2.1. MONOMIAL IDENTITY TESTING

Lemma 20 *If for every feature i, the marginal probability density function at x_i is finite for all values of x_i then we have that for any $\mathbf{g}' \neq \mathbf{g}$, $Pr[P_{\mathbf{g}'}(\mathbf{x}) \neq P_{\mathbf{g}}(\mathbf{x})] = 1$.*

Proof We will prove by induction on $N' \leq N$ and $d' \leq d$ that for any polynomial P' of degree d' over N' variables $Pr[P'(\mathbf{x}) = 0] = 0$. Then, we only need to plug in $P' = P_{\mathbf{g}} - P_{\mathbf{g}'}$ to complete the proof.

For the base case assume the polynomial is only over one variable and any degree i.e., $N' = 1$ and any $d' \leq d$. Then the event $[P'(\mathbf{x}) = 0]$ corresponds to picking one of at most d' zeroes of P' from $\mathbb{R}$ (since $N' = 1$), which amounts to a probability of 0 according to the assumption on the probability density function.

Now assume for all $N' < N$ and $d' \leq d$, our induction hypothesis is true. The polynomial P' can be expressed as a summation of terms in x_1: $\sum_{i=0}^{k} P_i''(x_2, \ldots x_n)x_1^i$ where k is the highest degree of x_1 and P_i'' is the coefficient of x_1^i. Then, for a fixed value of $x_2, \ldots x_N$, P' reduces to a polynomial of degree $k \leq d$ over one variable. Then, our induction assumption implies that conditioned on some arbitrary values of $x_2, \ldots, x_N$, the polynomial in x_1 attains zero with probability 0 i.e., $Pr[P'(\mathbf{x}) = 0 \,|\, x_2, \ldots x_N] = 0$. Then it follows that $Pr[P'(\mathbf{x}) = 0] = 0$. ■

D.3. Polynomials

In this section we present a more elaborate discussion of our results for lifelong learning of polynomials. We first set up our notations. For any input $\mathbf{x} \in \mathbb{R}^N$, we denote the output of a t-sparse d-degree target polynomial $\mathcal{G} = \{(\mathbf{g}_1, a_{\mathbf{g}_1}), (\mathbf{g}_2, a_{\mathbf{g}_2}), \ldots\}$ ($|\mathcal{G}| \leq t$) by the function $P_{\mathcal{G}}(\mathbf{x}) = \sum_{(\mathbf{g}, a_{\mathbf{g}}) \in \mathcal{G}} a_{\mathbf{g}} P_{\mathbf{g}}(\mathbf{x})$ where for each $(\mathbf{g}, a_{\mathbf{g}}) \in \mathcal{G}$, $\mathbf{g}$ is a monomial of degree d and

co-efficient $a_{\mathbf{g}} \in \mathbb{R}$. Our belief is that there exists a set of monomial metafeatures $\mathcal{F}$, and each polynomial can be represented as a sum of monomials, each of which can be represented using $\mathcal{F}$ as described in Section 4. More formally, a polynomial $\mathcal{G}$ can be represented using $\mathcal{F}$ if for each $(\mathbf{g}, a_{\mathbf{g}}) \in \mathcal{G}$, $\mathbf{g} \in \mathbb{C}(\mathcal{F})$. More compactly, $\mathcal{G}^{(j)} \subset \mathbb{C}(\mathcal{F}) \times \mathbb{R}$. Then, our problem setup is as follows.

Problem Setup 3 (Lifelong polynomial learning) *The m d-degree t-sparse targets $\mathcal{G}^{(j)}$ and data $\mathcal{S}^{(j)}$ (each of at most S examples) satisfy the following conditions:*

1. *There exists an unknown $N \times K$ matrix $\mathcal{F}$ ($K \ll N$) such that each $\mathcal{G}^{(j)} \in \mathbb{C}(\mathcal{F}) \times \mathbb{R}$.*
2. *The samples in $\mathcal{S}^{(j)}$ are drawn i.i.d from a **known** product distribution $\mathcal{D}^{(j)}$* [5].

We will now describe the method for learning polynomials from scratch, after which we describe our baseline lifelong learning result in Theorem 21 and the proof for our main result.

We adopt a modification of the approach in Andoni et al. (2014) to learn polynomials from scratch. The basic idea is to use correlations between the target and some cleverly chosen functions to detect the presence of different monomials in $\mathcal{G}$. For the sake of convenience, assume there exist correlation oracles that when provided as input some function P', return the exact value of the correlations $\langle P'(\mathbf{x}), P_{\mathcal{G}}(x) \rangle$, $\langle P'(\mathbf{x}), P^2_{\mathcal{G}}(x) \rangle$ etc., In practice these oracles can be replaced by approximate estimates based on the sample $\mathcal{S}$. We will limit our analysis to the exact scenario noting that it can be extended to the sample-based approach in a manner similar to Andoni et al. (2014). Our guarantees will then hold good with high probability, given sufficiently many samples.

To simplify the discussion we will assume like in Andoni et al. (2014) that the distribution over each variable is identical i.e., $\mathcal{D} = \mu^N$. Then, as a first step, given $\mathcal{D}$, the learner creates an inventory of polynomials in each variable x_i such that these polynomials represent an "orthornormal bases" with respect to $\mathcal{D}$. More formally, the inventory will consist of polynomials $H_{d'}(x_i)$ of degree d' (identical for each $i \in [N]$) for each $0 \leq d' \leq d$, such that $\mathbb{E}[H_{d'}(x_i)H_{d''}(x_i)]$ is zero when $d' \neq d''$ and is one when $d' = d''$.

Equipped with this inventory, we then set out to perform t iterations extracting one monomial from $\mathcal{G}$ at a time. Assume that from the iterations performed so far, we have extracted a set of monomials and their coefficients $\tilde{\mathcal{G}} \subseteq \mathcal{G}$. Now, for the next iteration, we first find the largest power of x_1 that is present in $\mathcal{G} - \tilde{\mathcal{G}}$ by testing whether $\langle H_{2d'}(x_1), (P_{\mathcal{G}} - P_{\tilde{\mathcal{G}}})^2 \rangle > 0$ for $d' = d, d-1, \ldots$ to detect the presence of x_1^d, $x_1^{d-1}, \ldots$ in that order, respectively. We stop when the test is positive for some $x_1^{d_1}$. The curious reader can refer Andoni et al. (2014) to understand why this particular test works, but all we need to know for our discussion is that if these tests are done in this particular order, we are guaranteed to find the highest power of $x_1^{d_1}$ in $\mathcal{G} - \tilde{\mathcal{G}}$. Then, we find the largest power of x_2 that "co-occurs" with $x_1^{d_1}$ in some monomial, by testing whether $\langle H_{2d_1}(x_1)H_{2d'}(x_2), (P_{\mathcal{G}} - P_{\tilde{\mathcal{G}}})^2 \rangle > 0$ for $d' = d, d-1, \ldots$ to detect the presence of $x_1^{d_1}x_2^d, x_1^{d_1}x_2^{d-1}, \ldots$ and so on in that particular order. In this manner, the algorithm builds a monomial over N sub-iterations which turns out to be the *lexicographically largest* $\mathbf{g}$ present in $\mathcal{G} - \tilde{\mathcal{G}}$. Now, to compute the co-efficient $a_{\mathbf{g}}$ we find $\langle \prod_{i=1}^{N} (b_{g_i} H_{g_i}(x_i)), P_{\mathcal{G}} \rangle$ where b_{g_i} is the co-efficient of $x_i^{g_i}$ in $H_{g_i}(x_i)$. The algorithm then adds $(a_{\mathbf{g}}, \mathbf{g})$ to $\tilde{\mathcal{G}}$ before proceeding to the next of t iterations.

The above summary differs from that original algorithm presented in Andoni et al. (2014) in the precise quantity that it extracts in each iteration. Andoni et al. (2014) consider a representation of the polynomial in the orthornormal bases such that it is a weighted sum of terms of the form $H_{d_1}(x_1)H_{d_2}(x_2)\ldots H_{d_N}(x_N)$, and in each iteration they extract one such term. We however use

5. This is the model considered in Andoni et al. (2014). An upper bound on S can be found in Andoni et al. (2014).

the representation in the orthonormal bases only to detect the lexicographically largest monomial and its corresponding co-efficient and then remove the monomial itself.

We now describe our straightforward lifelong learning approach for polynomials which remembers only the features that have been seen so far.

Theorem 21 (Naive lifelong learning of polynomials) *In the model of Problem Setup 3, there exists a naive algorithm for lifelong learning of t-sparse polynomials that makes $O\left(S(KN + mKd)\right)$ feature evaluations in total.*

Proof Sketch This approach is very similar to the naive approach for lifelong learning of monomials. IMPROVEREP Algorithm 11 maintains a list of linearly independent monomial targets that have been seen in the polynomials learned from scratch so far. Now, for a new target $\mathcal{G}$, we will perform the "lexicographic search" method from Andoni et al. (2014) over only the features that have been seen i.e., during the search we skip features that correspond to an all zero row in $\tilde{\mathcal{F}}$. Essentially, we assume that the unseen features do not occur in the target polynomial. We again check whether the polynomial computed this way is correct by verifying it on a single sample.

Using this approach we are guaranteed that if $\mathcal{G} \subset \mathbb{C}(\tilde{\mathcal{F}}) \times \mathbb{R}$, USEREP does not fail because such a target will not contain unseen features in any of its monomials. Then, we can use an argument similar to Theorem 18 and show by contradiction that USEREP can fail at most K times, and hence evaluate only Kd features per example. ■

We now prove our main result, Theorem 6. Recall using IMPROVEREP Algorithm 11 and USEREP Algorithm 12, we claimed we learn at most K polynomials from scratch, and on the rest we examine at most $O\left(K + td\right)$ features per example. We first prove guarantees about USEREP Algorithm 12.

Recall that our representation consists of linearly independent monomials from previously seen polynomials. Then our main idea was to run the lexicographic search restricted to K features that are chosen based on K linearly independent rows in the learned representation. More specifically, if $\mathcal{I}$ is the set of chosen features, in each iteration we find the lexicographically largest powers restricted to $\mathcal{I}$, say $\mathbf{g}[\mathcal{I}]$. Then, we use our technique for monomials, to estimate the remaining powers in the monomial $\mathbf{g}$ that contains $\mathbf{g}[\mathcal{I}]$. Then, as before, we extract $\mathbf{g}$ from the polynomial and proceed to the next iteration. After t iterations, our estimate of the polynomial is complete, so we draw a single example to verify it. If our verification fails, we learn the polynomial from scratch and update the representation with more linearly independent monomials from the learned polynomial.

Lemma 22 *Let $\tilde{\mathcal{F}}$ be an $N \times k$ matrix. Then, with high probability a) if $\mathcal{G}^{(j)} \in \mathbb{C}(\tilde{\mathcal{F}})$, then Algorithm 12 correctly learns and outputs $\tilde{\mathcal{G}}^{(j)} = \mathcal{G}^{(j)}$ b) if Algorithm 12 does output some $\tilde{\mathcal{G}}^{(j)}$, then $\tilde{\mathcal{G}}^{(j)} = \mathcal{G}^{(j)}$. Also, Algorithm 12 examines only at most $k + td$ features per sample point.*

Proof a. Assume $\mathcal{G}^{(j)} \in \mathbb{C}(\tilde{\mathcal{F}})$. The fact that in each iteration, we find the lexicographically largest value for the features $\mathcal{I}$ follows directly from the discussion in Andoni et al. (2014). However, we do have to prove that there is a unique $\mathbf{g}$ in $\mathcal{G}$ such that $\mathbf{g}[\mathcal{I}]$ corresponds to the above value. This follows from the proof of Lemma 19 where we showed that for $\mathcal{I}$ corresponding to linearly

independent rows, $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) = \mathbf{w}_{\tilde{\mathcal{F}}}(\mathbf{g})$ and hence given $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$ there is a unique $\mathbf{g} \in \mathbb{C}(\mathcal{F})$ defined by $\mathbf{g} = \tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$.

Now, we need to prove that we find a co-efficient $a_{\tilde{\mathbf{g}}}$ for the to-be-extracted monomial, that satisfies $a_{\tilde{\mathbf{g}}} = a_{\mathbf{g}}$. We first note that $\langle \prod_{i=1}^{N} H_{g_i}(x_i), (P_{\mathcal{G}} - P_{\tilde{\mathcal{G}}}) \rangle$ returns the co-efficient of $\prod_{i=1}^{N} H_{g_i}(x_i)$ in $(P_{\mathcal{G}} - P_{\tilde{\mathcal{G}}})$, say $a'_{\mathbf{g}}$, in the basis representation of the polynomial. Next, we claim that the co-efficient $a'_{\mathbf{g}}$ in the bases representation is contributed to purely by the co-efficient a_g in the monomial representation. If there was any other monomial that contributed to $a'_{\mathbf{g}}$, then it had to have a lexicographically larger value than $\mathbf{g}$ with respect to $\mathcal{I}$ or equal to $\mathbf{g}$ with respect to $\mathcal{I}$. However, this contradicts the fact that $\mathbf{g}$ was chosen to be the unique lexicographically largest value with respect to $\mathcal{I}$. Thus, we only need to account for the contribution of the co-efficient of $\prod_{i=1}^{N} H_{g_i}(x_i)$ with an extra factor of b_{g_i} which corresponds to the co-efficient of $x_i^{g_i}$ within $H_{g_i}(x_i)$.

b. This follows from the proof of Lemma 19 and Lemma 20 applied to polynomials.

c. First of all, we examine k features when we query $\mathcal{I}$ on all samples. Now, note that when we execute the algorithm using samples for the correlation oracles, we will have to compute $P_{\tilde{\mathcal{G}}}(\mathbf{x})$ on each sample $\mathbf{x}$. This however will only require evaluation of features relevant to $\tilde{\mathcal{G}}$. Since $\mathcal{G}$ consists of at most t monomials each of degree at most d, this can be only as large as td. ■

Now, we prove Theorem 6 in which we stated that the (USEREP Algorithm 12, IMPROVEREP Algorithm 11)-protocol for polynomials makes $O\left(S(KN + m(K + dt))\right)$ feature evaluations overall.

Proof (for Theorem 6) From Lemma 22, we have that we increase the rank of $\tilde{\mathcal{F}}$ by at least one every time we fail to learn using $\tilde{\mathcal{F}}$ on some target. If USEREP has failed on more than K targets it means that there are at least $K + 1$ monomials from $\mathbb{C}(\mathcal{F})$ that were added as columns to $\tilde{\mathcal{F}}$ and are linearly independent. However, since $\mathbb{C}(\mathcal{F})$ is a K-dimensional subspace in $\mathbb{R}^N$, this results in a contradiction, thus proving that at most K failures of USEREP can occur. The result then follows from Lemma 22 and the fact that $|\tilde{\mathcal{F}}|$ contains only at most K targets. ■

We note that when we replace the oracles by estimation using random samples, we should be careful about approximation errors that may affect the lifelong learning protocol. For example, if we were to infer that a monomial term exists in $\mathcal{G}$, when in reality it does not, we may incorrectly add it to our representation $\tilde{\mathcal{F}}$ when it should not be. However, if the co-efficients of each term in the polynomial were not too small, we can overcome this problem by learning the co-efficient of the monomial, and checking whether it is above a small threshold, before deducing that it indeed is a term in the polynomial.

Algorithm 11 IMPROVEREP - Polynomials

1: Input: Representation $\tilde{\mathcal{F}}_{\text{old}}$ and a target $\mathcal{G}$ learned from scratch.
2: $\tilde{\mathcal{F}} \leftarrow \tilde{\mathcal{F}}_{\text{old}}$
3: **for** $\mathbf{g} \in \mathcal{G}$ **do**
4: If $\mathbf{g} \notin \mathbb{C}(\tilde{\mathcal{F}})$, add $\mathbf{g}$ as a column to $\tilde{\mathcal{F}}$.
5: Return $\tilde{\mathcal{F}}$

Algorithm 12 USEREP - Learning Polynomial from Metafeatures

Input: Metafeatures $\tilde{\mathcal{F}} = [\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_k]$ ($k \leq K$), distribution $\mathcal{D}$
Halt with failure if $\tilde{\mathcal{F}}$ is empty.
Let $\mathcal{I}$ be the indices of those rows in $\tilde{\mathcal{F}}$ that are linearly independent and let $\tilde{\mathcal{F}}[\mathcal{I}]$ be the corresponding $k \times k$ sub-matrix of $\tilde{\mathcal{F}}$.
Query for only the features $\mathcal{I}$ on all samples.
Initialize $\tilde{\mathcal{G}}$ to be empty.
for t iterations **do**
    Let $\mathbf{g}$ be the lexicographically largest monomial in $\mathcal{G} - \tilde{\mathcal{G}}$ with respect to $\mathcal{I}$. Find $\mathbf{g}[\mathcal{I}]$ using the lexicographic search technique from Andoni et al. (2014) using the correlation oracle (in practice, estimate this using the $\mathcal{S}$).
    Solve for $\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$ in $\tilde{\mathcal{F}}[\mathcal{I}]\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}]) = \mathbf{g}[\mathcal{I}]$. If no solution exists, halt with failure.
    Estimate $\tilde{\mathbf{g}} \leftarrow \tilde{\mathcal{F}}\mathbf{w}_{\tilde{\mathcal{F}}[\mathcal{I}]}(\mathbf{g}[\mathcal{I}])$.
    Halt with failure if the degree of $\tilde{\mathbf{g}}$ is greater than d.
    $a_{\tilde{\mathbf{g}}} \leftarrow \langle \prod_{i=1}^{N}(b_{g_i} H_{g_i}(x_i)), (P_{\mathcal{G}} - P_{\tilde{\mathcal{G}}}) \rangle$
    $\tilde{\mathcal{G}} \leftarrow \tilde{\mathcal{G}} \cup \{\tilde{\mathbf{g}}\}$
Draw a single sample $(\mathbf{x}, P_{\mathcal{G}}(\mathbf{x}))$ from $\mathcal{D}$, query the td features that are relevant to $\tilde{\mathcal{G}}$. If $P_{\mathcal{G}}(\mathbf{x}) \neq P_{\tilde{\mathcal{G}}}(\mathbf{x})$, halt with failure.
Return $\tilde{\mathcal{G}}$.

Appendix E. The agnostic case

Let us recall the scenario in the agnostic case:

Problem Setup 4 *In the agnostic model, the learner is faced with a series of $m + r$ targets such that:*

1. *m (good) targets are guaranteed to be related to each other through a set of at most K metafeatures, while the remaining r (bad) targets can be adversarially chosen and placed.*
2. *the learner has to reduce the feature evaluations done on the samples for the m related targets.*

Recall that we presented two simple techniques to address this. For the sake of simplicity, we consider decision trees of constant depth.

***r*-expansion technique** Since m targets belong to $\mathbb{DT}(\mathcal{F})$, observe there exists a representation of at most $O(K + r)$ metafeatures that is sufficient to describe all the $m + r$ targets: a representation that is the union of $\tilde{\mathcal{F}}$ and the r bad targets as they are. Thus, we allow the lifelong learner to update $\tilde{\mathcal{F}}$ whenever its USEREP fails. Essentially, we allow $\tilde{\mathcal{F}}$ to expand to $O(K + r)$ metafeatures; USEREP will fail on at most K good targets (and possibly on all the r bad targets which we do not care about) and learn the rest successfully evaluating $O(K + r)$ features per example.

***r*-restart technique** Alternatively, we enforce $|\tilde{\mathcal{F}}| \leq K$ as before but USEREP fails on a $K + 1^{th}$ target, we learn that target from scratch after which we simply erase $\tilde{\mathcal{F}}$ and effectively restart our lifelong learning from the next task. Every time USEREP fails on a $K + 1$th target after the most

recent restart, we restart similarly.

Below we prove Theorem 7 which formally describes how well the above techniques work and in what ranges of r.

Theorem 7 *In the agnostic model where we face $m + r$ decision tree targets such that m trees belong to $\mathbb{DT}(\mathcal{F})$, the number of feature evaluations on the training data for the m trees:*

- *the r-expansion technique is $O\left(S(KN + m(K + r))\right)$.*
- *the r-restart technique is $O\left(S(rKN + mK)\right)$.*
- *a combination of c-expansion and r/c-restart is $O(S(\sqrt{rKNm}+Km))$, for $c = \sqrt{rKN/m}$ provided $r = \Omega\left(\max\left(m/n, KN/m, K\right)\right)$.*

Proof (for Theorem 7) In r-expansion, we allow $\tilde{\mathcal{F}}$ to have as many as $O\left(K + r\right)$ metafeatures. Now, every bad target may result in adding $O\left(1\right)$ metafeatures to $\tilde{\mathcal{F}}$ while the m bad targets will result in adding $O\left(K\right)$ metafeatures to $\tilde{\mathcal{F}}$. Thus, we will be able to learn all but m good targets using $\tilde{\mathcal{F}}$ by examining only $O\left(K + r\right)$ features per example i.e., $O\left(S(rKN + mK)\right)$ features overall.

In r-restart, every time USEREP fails on a $K + 1$th target, we learn that target from scratch and then erase $\tilde{\mathcal{F}}$ effectively restarting our lifelong learning. Now, at least one of the $K+1$ trees learned from scratch must be a bad target. This is because if none of the K trees that were used to update $\tilde{\mathcal{F}}$ were bad, $\tilde{\mathcal{F}}$ would have been rich enough to represent all the good targets. This means that the $K + 1$th target has to be a bad target. Thus, every restart corresponds to a failure of USEREP on at least one bad target and at most K good targets. Then, we will face at most r such restarts, learning at most rK targets from scratch during the process and the rest from only $O\left(K\right)$ features per example i.e., $O\left(S(KN + m(K + r))\right)$ features overall.

Now when $r = O\left(\max\left(\frac{KN}{m}, K\right)\right)$ observe that r-expansion makes only $O\left(S(KN + mK)\right)$ feature evaluations. Similarly, when $r = O\left(\frac{m}{N}\right)$, r-restart makes $O\left(S(KN + mK)\right)$ feature evaluations. This is as good as our performance when $r = 0$.

To deal with $r = \Omega\left(\max\left(\frac{m}{N}, \frac{KN}{m}, K\right)\right)$, we can combine the above techniques, in particular, we combine $\frac{r}{c}$-restart with c-expansion. That is, between every restart we allow $\tilde{\mathcal{F}}$ to accommodate $O\left(K + c\right)$ metafeatures and when USEREP fails on the $K + c + 1$th target we restart the representation. Recall that each bad target may contribute $O\left(1\right)$ metafeatures while all the good targets contribute to $O\left(K\right)$ metafeatures. Thus, between every restart USEREP would have failed on at most K good targets and at least $c + 1$ bad targets. Since there are only r bad targets, we then face only $O\left(\frac{r}{c}\right)$ restarts. Since we learn only $O\left(\frac{r}{c}\right) \cdot K$ targets from scratch and learn the rest by examining only $O\left(K + c\right)$ features per example, we evaluate $O\left(S(\frac{r}{c}KN + m(K + c))\right)$ features overall.

The value of c that optimizes the above bound is $c^* = \sqrt{\frac{rKN}{m}}$ and the minimum is $O\left(S(\sqrt{rKNm} + mK)\right)$. But note that c^* must take a meaningful value for this bound to hold good. That is, for c-expansion to make sense, we need $c^* \geq 1$ and for $\frac{r}{c^*}$-restart to make sense, $\frac{r}{c^*} \geq 1$. That is, we need $c^* \in [1, r]$, which can be verified to hold good when $r = \Omega\left(\max\left(\frac{m}{N}, \frac{KN}{m}, K\right)\right)$. ■

Appendix F. Lower Bounds

We now present detailed proof for our lower bound results:

Theorem 8 *Let* $r_{\min} = \max\left(\frac{m}{N}, \frac{KN}{m}, K\right)$, $r_{\max} = \min\left(\frac{mN}{K}, \frac{(N-K)^2 m}{KN}\right)$. *In the agnostic model of Section 5, there exists an adversary such that, on the* m *good trees, any lifelong learner makes:*

- $\Omega\left(NK + Km\right)$ *feature evaluations when* $0 \leq r \leq r_{\min}$.
- $\Omega\left(\max\left(\frac{r}{N-K}, 1\right) KN + Km\right)$ *feature evaluations when* $r_{\min} \leq r \leq r_{\max}$.
- $\Omega\left(mN\right)$ *feature evaluations when* $r_{\max} \leq r$.

Our main idea is a randomized adversary that poses decision stumps (trees with only the root node) or degree-1 monomials to the learner. In particular, we use Lemma 23 where we show that when the adversary picks one feature at random from a pool of N' features to be the decision stump/monomial, if the learner examines only $o(N')$ features, the learner will fail to identify the correct feature for the target with probability $\Omega\left(1\right)$. Thus, for the learner to successfully complete the task, it must examine $\Omega\left(N'\right)$ features. Then to force a learner to examine $O\left(KN + mK\right)$ features, the adversary picks K distinct features at random from the pool of N features for the first K targets. Then it assigns these K features as the metafeatures and picks the remaining targets at random from this chosen set of K features.

Lemma 23 ***(Randomized adversary)*** *For a particular task, if the adversary picks a feature from a pool of* N' *features* ($N' \leq N$) [6] *to pose a single-feature target, if the learner examines only* $o(N')$ *features, the learner will fail (i.e., pick the wrong feature) with probability* $\Omega\left(1\right)$.

Proof Let i^* be the feature chosen by the adversary at random from a pool of N' features $\mathcal{I}^*$, and $\mathcal{I}$ be the set of features examined by the learner. The random choice of i^* corresponds to different possible outcome events. But observe that from the perspective of the learner the events corresponding to $i^* \notin \mathcal{I}$ (the adversary picking a feature not examined by the learner) are all indistinguishable. This crucial observation tells us that in all such events, the learner will adopt the same strategy. Let $Pr_l(i)$ denote the probability that the learner outputs feature i in this strategy. Let $Pr_a(i)$ denote the probability that the adversary chose feature i at random from its pool of N' features.

Then, the probability that the learner fails is at least the sum of probability of the event that the adversary picks an i from $\mathcal{I}^* - \mathcal{I}$ and the learner does not pick i. We lower bound this probability $\sum_{i \in \mathcal{I}^* - \mathcal{I}} Pr_a(i)(1 - Pr_l(i))$ as follows:

$$\sum_{i \in \mathcal{I}^* - \mathcal{I}} \underbrace{Pr_a(i)}_{\frac{1}{N'}}(1 - Pr_l(i)) = \frac{1}{N'} \sum_{i \in \mathcal{I}^* - \mathcal{I}} (1 - Pr_l(i)) \geq \frac{1}{N'} \left(|\mathcal{I}^* - \mathcal{I}| - \sum_{i \in \mathcal{I}^* - \mathcal{I}} Pr_l(i) \right)$$

$$\geq \frac{1}{N'} \left(N' - o(N) - 1\right) = \Omega\left(1\right)$$

The second inequality follows from the fact that $\sum_{i \in \mathcal{I}^* - \mathcal{I}} Pr_l(i) \leq 1$ and the number of examined features $|\mathcal{I}| = O\left(N'\right)$.

6. It does not matter if the learner knows these N' features or not.

■

Now we prove Theorem 8 in the following three lemmas one for each range of r. First in Lemma 24 we prove a lower bound of $\Omega(KN + mK)$ that holds for any value of r. Then in Lemma 25 we prove a lower bound for intermediate values of r and finally in Lemma 26, we prove a lower bound for large values of r.

Lemma 24 *There exists an adversary such that any lifelong learning algorithm makes $\Omega(KN + mK)$ feature evaluations.*

Proof For the first K single-feature targets, our adversary randomly picks K distinct features which will be the metafeatures. Each of the remaining $m - K$ tasks are targets that correspond to one of these K chosen features at random. Now note that for a task j where $j \leq K$, the adversary effectively picks a feature at random from a pool of $N - j + 1$ features (which excludes the $j - 1$ features already chosen). Thus, the learner has to examine $\Omega(N - j + 1)$ features in order to not fail in this task with probability $\Omega(1)$. Thus, over the first K tasks, the learner has to examine $O\left(\sum_{j=1}^{K} N - j + 1\right) = \Omega(KN)$ features over all. Then, in each of the following $m - K$ tasks, the learner has to examine $\Omega(K)$ features per task i.e., $\Omega((m - K)K)$ features overall, which is $\Omega(mK)$ since m is large.

■

Now we prove a better bound for values of r greater than $r_{\min} = \max\left(\frac{m}{N}, \frac{KN}{m}, K\right)$ but less than $r_{\max} = \min\left(\frac{mN}{K}, \frac{(N-K)^2 m}{KN}\right)$. Here, instead of precisely choosing m good targets and r targets, the adversary will pose a set of targets and then choose K features to be the metafeatures. We then show that $\Theta(m)$ of the targets are good targets and $\Theta(r)$ targets are bad targets that correspond to the remaining $N - K$ features.

Lemma 25 ***(Lower bound for intermediate values of** r**)*** *When $r \leq r_{\max}$, there exists an adversary such that any lifelong learning algorithm makes $\Omega\left(\max\left(\frac{r}{N-K}, 1\right) KN + Km\right)$ feature evaluations.*

Proof When $\frac{r}{N-K} \leq 1$, the lower bound of $\Omega(KN + Km)$ follows from Lemma 24. Hence, consider $\frac{r}{N-K} > 1$. Let $m' = \frac{rN}{(N-K)}$. The adversary first presents m' single-feature targets picked at random from the pool of all N features. Then the adversary chooses K random features to be the metafeatures, hence marking targets corresponding to these K features as good targets, and the rest as bad.

Now, we can show that there are in fact $\Theta(m)$ good targets and $\Theta(r)$ bad targets, thus ensuring that this is a legal sequence of adversarial targets. Since $m' = \frac{r}{N-K} N \geq N$, using Chernoff bounds, with high probability $1 - O(1)$, we have $\Theta\left(m' \frac{N-K}{N}\right) = \Theta(r)$ bad targets and $\Theta\left(m' \frac{K}{N}\right) = \Theta\left(\frac{rK}{(N-K)}\right)$ good targets. Since, $r \leq \frac{(N-K)^2 m}{KN}$, this translates to $\Theta\left(\frac{(N-K)m}{N}\right) = O(m)$ good targets. Thus, this is a valid sequence of targets.

Now, from Lemma 23, we get that the learner has to evaluate $\Omega\left(\frac{rK}{(N-K)} \cdot N\right)$ features overall. In addition to this, the adversary presents a sequence of m good targets chosen at random from the

K metafeatures. Note that this is legal because we still pose only θm good targets. This accounts for $\Omega(mK)$ more feature evaluations.

In total, the learner examines $\Omega\left(\frac{rK}{(N-K)} \cdot N + mK\right)$ features. ■

We finally show that for sufficiently large r i.e., $r \geq r_{\max}$ and $r \geq r_{\min}$, the learner has to evaluate $\Omega(mN)$ features.

Theorem 26 ***(For large** r**)*** *Given $r \geq r_{\max}$ and $r \geq r_{\min}$, there exists an adversary such that any lifelong learning algorithm makes $\Omega(mN)$ feature evaluations.*

Proof The range of values of r such that $r \geq r_{\max} = \min\left(\frac{mN}{K}, \frac{(N-K)^2 m}{KN}\right)$ can be split into the interval $r \geq \frac{mN}{K}$ and the interval $\frac{(N-K)^2 m}{KN} \leq r \frac{mN}{K}$. We will consider these two intervals separately and provide adversarial strategies for both.

Case 1: $r \geq \frac{mN}{K}$. Let $m' = \frac{mN}{K}$. The adversary poses m' targets to the learner chosen at random from all the N features. Thus, the learner is forced to examine $\Omega(N)$ features on each target. Then, the adversary chooses K features to be good features, thereby marking some of the targets as good targets. We show that, of the m' targets, there are $\Theta(m)$ good targets and only $O(r)$ bad targets. Therefore, this is a valid sequence of targets and furthermore, on this sequence the learner examines $\Omega(m \cdot N)$ features.

To count the number of good targets, we observe that $m' = \Omega\left(\frac{N}{K}\right)$. Then from Chernoff bounds, with high probability $1 - O(1)$, we have that $\Theta\left(m' \frac{K}{N}\right)$ i.e., $\Theta(m)$ targets are good targets. Since $m' \leq r$, we have only $O(r)$ bad targets.

Case 2: $r < \frac{mN}{K}, r \geq \frac{(N-K)^2 m}{KN}$. Now, we set $m' = \sqrt{\frac{rNm}{K}}$ and sample m' targets at random from the pool of all N features. Then we pick K random features to be the metafeatures and then present m good targets choosing randomly from the pool of K metafeatures.

To count the number of good targets in the first sequence of m' targets, observe that $m' \geq N$ because $r \geq \frac{KN}{m}$. Hence, with high probability $1 - O(1)$, the number of good targets is $\Theta\left(m' \frac{K}{N}\right) = \Theta\left(\sqrt{\frac{rKm}{N}}\right)$. Since $r \leq \frac{mN}{K}$, this is $O(m)$. Similarly, with high probability $1 - O(1)$, the number of bad targets is $\Theta\left(m' \frac{N-K}{N}\right) = \Theta\left(\sqrt{\frac{rNm}{K}} \cdot \frac{N-K}{N}\right) = \Theta\left(\sqrt{r} \cdot \sqrt{\frac{(N-K)^2 m}{KN}}\right)$. Then using the inequality $r \geq \frac{(N-K)^2 m}{KN}$, we get that the number of bad targets is $O(r)$. Thus, this is a valid sequence of targets. Furthermore, on $\Theta\left(\sqrt{\frac{rKm}{N}}\right)$ good targets, the learner is forced to examine $\Omega(N)$ features. Thus, on the first sequence the learner examines $\Omega\left(\sqrt{rKmN}\right)$ features overall. Since $r \geq \frac{(N-K)^2 m}{KN}$, this is $\Omega(m(N-K))$. On the second sequence the learner examines $O(mK)$ features overall. In total, this is $\Omega(mN)$ feature evaluations.

■