JMLR: Workshop and Conference Proceedings vol 40:1–20, 2015

Efficient Representations for Lifelong Learning and Autoencoding

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

Avrim Blum AVRIM@CS.CMU.EDU
Carnegie Mellon University

Santosh Vempala VEMPALA@GATECH.EDU
Georgia Tech

Abstract

It has been a long-standing goal in machine learning, as well as in AI more generally, to develop life-long learning systems that learn many different tasks over time, and reuse insights from tasks learned, "learning to learn" as they do so. In this work we pose and provide efficient algorithms for several natural theoretical formulations of this goal. Specifically, we consider the problem of learning many different target functions over time, that share certain commonalities that are initially unknown to the learning algorithm. Our aim is to learn new internal representations as the algorithm learns new target functions, that capture this commonality and allow subsequent learning tasks to be solved more efficiently and from less data. We develop efficient algorithms for two very different kinds of commonalities that target functions might share: one based on learning common low-dimensional and unions of low-dimensional subspaces and one based on learning nonlinear Boolean combinations of features. Our algorithms for learning Boolean feature combinations additionally have a dual interpretation, and can be viewed as giving an efficient procedure for constructing near-optimal sparse Boolean autoencoders under a natural "anchor-set" assumption.

Keywords: Life-long learning, multi-task learning, shared representations, auto-encoding.

1. Introduction

Machine learning has developed a deep mathematical understanding as well as powerful practical methods for the problem of learning a single target function from large amounts of labeled data. Yet if we wish to produce machine learning systems that persist in the world, we need methods for continually learning many tasks over time and that, like humans (Gopnik et al., 2001), improve their ability to learn as they do so, needing less data per task as they learn more. A natural approach for tackling this goal, called "life-long learning" (Thrun, 1996; Thrun and Mitchell, 1995a) or "transfer learning" (Argyriou et al., 2008; Maurer and Pontil, 2013) or "learning to learn" (Baxter, 1997; Thrun and Pratt, 1997), is to use information from previously-learned tasks to *improve the underlying representation* of the learning algorithm, under the hope or belief that some kinds of commonalities across tasks exist. These commonalities could be a single low-dimensional or sparse representation, a collection of multiple low-dimensional or sparse representations, or some combination or hierarchy, such as in Deep Learning (Bengio, 2013; Bengio and Delalleau, 2011). In this paper, we develop algorithms with provable efficiency and sample size guarantees for several interesting categories of commonalities, considering both linear and Boolean transfer functions, under natural distributions on the data points.

Specifically, we consider a setting where we are trying to solve a large number of binary classification problems that arrive one at a time. Each classification problem will individually be learnable from a polynomial-size sample,[1] but our goal will be to learn new internal representations that will allow us to learn new target functions faster and from less data. We will furthermore aim to do this in a streaming setting in which we cannot keep the labeled data for problem t in memory when we move on to problem $t+1$, only the learned hypotheses (which will be required to have a compact description) and the current internal representation.

We start by considering a conceptually simple case that each classification problem is a linear separator, and their defining vectors lie in a common k dimensional subspace of the ambient space $\Re^n$, for $k \ll n$ (equivalently, there exist k hidden linear metafeatures and each target is a linear separator over these metafeatures). This case has been considered in the "batch" setting in which one has data available for all target functions at the same time and therefore can solve a joint optimization problem (Argyriou et al., 2008; Maurer and Pontil, 2013). However, for the online setting, a key challenge is that we will not have perfectly learned the previous target functions when we set out to learn the next one.[2] For this problem we provide a polynomial time algorithm that, when the underlying data distributions for our m learning problems are log-concave, has labeled sample complexity much better than the $\Omega(nm/\epsilon)$ sample complexity to learn the tasks separately.

We then consider scenarios where the commonalities involve more than one level of metafeatures and provide efficient algorithms for these settings as well. Specifically, for linear metafeatures, we analyze a scenario where the target functions all lie in a k dimensional space and furthermore within that k-dimensional space, each target lies in one of r different constant dimensional spaces, where r could be large. This models situations where there are really r different *types* of learning problems but they do share some commonalities across types (given by a k-dimensional subspace).

In Section 3 we develop algorithms for a scenario where the metafeatures are non-linear, in particular where features are boolean and the metafeatures are products. We give an efficient algorithm for finding the fewest product-based metafeatures for a given set of target monomials under an "anchor-variable" assumption analogous to the anchor-word assumption of Arora et al. (2012), and prove bounds on its performance for learning a series of target functions arriving online. We then give an extension that learns an approximately-optimal overcomplete sparse representation (we may have more metafeatures than input features, but each target should have a sparse representation) under a weaker form of assumption we call the "anchor-set" assumption (anchor variables no longer make sense in the overcomplete case). These results can be viewed as giving efficient algorithms for a Boolean autoencoding where given a set of black-and-white pixel images (vectors in $\{0,1\}^n$) we want to find either (a) the fewest "basic objects" (also vectors in $\{0,1\}^n$) such that each given image can be reconstructed by superimposing some subset of them (taking their bitwise-OR), or (b) a larger number of such objects such that each image has a sparse bitwise-OR reconstruction. In the first case our algorithm finds the optimal solution under the anchor-variable assumption (this is NP-hard in general) and in the second case it finds a bicriteria approximation (for a given sparsity level, approximates both number and sparsity to log factors) under the weaker anchor-set assumption.

In Section 4 we show how our results can be applied to the case that target functions are polynomials of low L_1 norm whose terms share pieces in common (within and across polynomials), a scenario that can be expressed via two levels of product-based metafeatures. As opposed to the

1. Except in Section 4 where we consider learning multivariate polynomials and allow membership queries.
2. In particular, because of this we will need to be particularly careful with which targets we use in constructing our metafeatures, as well as in controlling the propagation of errors. See, e.g., Lemma 3 and Figure 1.

algorithms for linear metafeatures, this algorithm periodically re-compactifies its current representation, revisiting the previously-learned polynomials and optimally constructing the fewest number of (possibly overlapping) conjunctive metafeatures that can be used to recreate all their monomials.

1.1. Related Work

Most related work in multi-task or transfer learning considers the case that all target functions are present simultaneously or that target functions are drawn from some easily learnable distribution. Baxter (1997, 2000) developed some of the earliest foundations for transfer learning, by providing sample complexity results for achieving low average error in such settings. Other related sample complexity results are given by Ben-David and Schuller (2003). Recent work of Maurer and Pontil (2013) and Kumar and Daume (2012) considers the problem of learning multiple linear separators that share a common low-dimensional subspace in the batch setting where all tasks are given up front. They specifically provide guarantees for a natural ERM algorithm with trace norm regularization. There has also been work on applying the Group Lasso method to batch multi-task learning which solves a specific multi-task optimization problem (Roth and Vogt, 2012).

Cavallanti et al. (2010) consider multi-task learning where explicit known relationships among tasks are exploited for faster learning. In their setting each learning problem is an online problem but the collection of learning problems are all occurring simultaneously. Valiant, in presenting his neuroidal model (Valiant, 2000), says, *"the intention of our architecture is that each new function that is learned or programmed be expressive in terms of old ones already represented in the system."* Our algorithm in Section 2 is of this type, with formal analysis about how the error accumulation affects the sample complexity (which, as we will see, is one of the central challenges in this setting). In earlier work, Rivest and Sloan (1988) also considered a formal setting of learning multiple target functions, where each builds on previously-learned rules. Work of Maurer (2005), building on Edelman (1995), and of Fahlman and Lebiere (1989), is also of this flavor, where previously-learned functions are viewed as features for new learning problems.

The problem of trying to learn invariants or other commonalities when faced with a series of related learning tasks arriving over time also has a long history in applied machine learning dating to work of Thrun (1996); Thrun and Mitchell (1995a,b).

1.2. Preliminaries

We assume we have m learning (binary classification) problems that arrive online over time. The learning problems are all over a common instance space X of dimension n (e.g., we will consider $X = \Re^n$ and $X = \{0,1\}^n$) but each has its own target function and potentially its own distribution D_i over X. Formally, learning problem i is defined by a distribution P_i over $X \times Y$ where $Y = \{-1, 1\}$ is the label space and D_i is the marginal over X of P_i, and the goal of the learning algorithm on problem i is to produce a hypothesis function h_i of small error over P_i.

2. Life-long Learning of Halfspaces

We consider here the case that $X = \Re^n$ and each target function is a linear separator through the origin; that is, for all i there exists a_i of unit length such that for all (x, y) drawn from P_i we have $\text{sign}(a_i \cdot x) = y$. We assume the a_i all lie in some common k-dimensional subspace of $\Re^n$ for $k \ll \min(n, m)$. In particular, let A be a m by n matrix whose rows are $a_1, \ldots, a_m$; then our assumption is that A has rank k. This implies that there exist a decomposition $A = CW$, where W

is a $k \times n$ matrix and C is a $m \times k$ matrix. The rows $w_1, \ldots, w_k$ of W can be viewed as k *linear metafeatures* that are sufficient to describe all m learning problems, or equivalently we can view this as a network with one middle layer of k hidden linear units. In fact, our algorithms will work under a more robust condition that the a_i are just "near" to a low-dimensional subspace (Theorem 4).

In this section we analyze the following natural online algorithm for this setting. Let ε_{acc} be a quantity to be determined later. For the first learning problem we just learn it to error ε_{acc} using the original input features and let the resulting weight vector be $\tilde{w}_1$. Suppose now we have produced weight vectors $\tilde{w}_1, \ldots, \tilde{w}_{k'}$ and we are considering problem i. We will first see if we can learn problem i well (to error ϵ) as a linear combination of the $\tilde{w}_j$. If so, then we mark this as a success and go on to problem $i+1$. If not, then we will learn it to error ε_{acc} using the input features and add the hypothesis weight vector as $\tilde{w}_{k'+1}$. (See Algorithm 1 for formal details). The challenge is how small ε_{acc} needs to be for this to succeed.

We show that if the D_i are isotropic log-concave (which includes many distributions such as Gaussian and uniform, see, e.g., Lovász and Vempala (2007)), the above procedure will be successful and learn with much less labeled data in total than by learning each function separately. We start with some useful facts and present a lemma (Lemma 3) that is crucial for our analysis.

Given two vectors a and b and a distribution $\tilde{D}$, let $\mathrm{d}_{\tilde{D}}(a,b) = \mathbb{P}_{x\sim\tilde{D}}(\mathrm{sign}(a\cdot x) \neq \mathrm{sign}(b\cdot x))$. Let $\theta(a,b)$ be the angle between two vectors a and b. For a vector a and a subspace V, let $\theta(a,V) = \min_{b\in V}\theta(a,b)$ be the angle between a and its closest vector in V (in angle). For subspaces U and V, let $\theta(U,V) = \max_{u\in U}\theta(u,V)$. That is, $\theta(U,V) \leq \alpha$ iff for all $u \in U$ there exists $v \in V$ such that $\theta(u,v) \leq \alpha$. The proof of Lemma 1 below appears in Appendix A.

Lemma 1 *Assume D is an isotropic log-concave distribution in R^n. Then there exist constants c and c' such that for any two unit vectors u and v in $\mathbb{R}^d$ we have $c\theta(v,u) \leq \mathrm{d}_D(u,v) \leq c'\theta(v,u)$.*

Lemma 1 implies that if we learn some target a_i to error ϵ_{acc}, then the angle between our learned vector and the target will be $O(\epsilon_{acc})$. In the other direction, if the target lies in subspace W and we have learned a subspace $\tilde{W}$ such that $\theta(W,\tilde{W})$ is small, then there will exist a low-error weight vector in $\tilde{W}$. Ideally, we would therefore like to say that if we construct a subspace $\tilde{W}$ out of vectors $\tilde{w}_i$ that individually are close to their associated targets w_i, then $\tilde{W}$ is close to the span W of the w_i. Unfortunately, this is not in general true if the targets are close to each other, e.g., see Figure 1(a). We will address this by using the fact that each $\tilde{w}_i$ was *not* learnable using the span of the previous $\tilde{w}_j$. We begin with a helper lemma (Lemma 2), which can be viewed as addressing the special case that all previous targets have been learned *perfectly*, and then present our main lemma (Lemma 3).

Lemma 2 *Let $U = \mathsf{span}\{a_1, \ldots, a_{k-1}\}, V = \mathsf{span}\{a_1, a_2, \ldots, a_{k-1}, b\}$ and $\tilde{V} = \mathsf{span}\{a_1, \ldots, a_{k-1}, \tilde{b}\}$ be sets of vectors in $\Re^n$. Then, $\theta(V,\tilde{V}) \leq \frac{\pi}{2}\frac{\theta(\tilde{b},b)}{\theta(\tilde{b},U)}$.*

Proof (For intuition, see Figure 1(b).) First, we may assume $b, \tilde{b} \notin U$ because if $b \in U$ then $\theta(V,\tilde{V}) = 0$ and if $\tilde{b} \in U$ then $\theta(\tilde{b},U) = 0$. Additionally we may assume b and $\tilde{b}$ are unit-length vectors since we are interested only in angles. Next, let $u \in V$ be the unit-length vector in V of farthest angle from $\tilde{V}$, i.e., $\theta(u,\tilde{V}) = \theta(V,\tilde{V})$. We can write u as a linear combination of b and some vector $u_1 \in U$, and will prove the lemma by showing there must be some nearby vector in the span of u_1 and $\tilde{b}$. Specifically, using the fact that $\theta(V,\tilde{V}) = \theta(u,\tilde{V}) \leq \theta(\mathsf{span}(u_1,b), \mathsf{span}(u_1,\tilde{b}))$ and the fact that $\theta(\tilde{b},U) \leq \theta(\tilde{b},u_1)$, it is sufficient to prove that: $\theta(\mathsf{span}(u_1,b), \mathsf{span}(u_1,\tilde{b})) \leq \frac{\pi}{2}\frac{\theta(b,\tilde{b})}{\theta(\tilde{b},u_1)}$, which is just a statement about 3-d space.

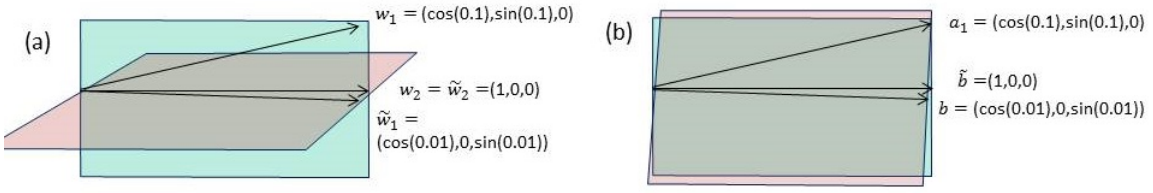

Figure 1: (a) Even though each $\tilde{w}_i$ is within angle 0.11 of its corresponding w_i ($\theta(w_2, \tilde{w}_2) = 0$, $\theta(w_1, \tilde{w}_1) \leq \theta(w_1, w_2) + \theta(w_2, \tilde{w}_1) = 0.11$), the two subspaces are orthogonal ($\mathsf{span}(w_1, w_2)$ is the x-y plane and $\mathsf{span}(\tilde{w}_1, \tilde{w}_2)$ is the x-z plane). This example also shows that even adding the *same* vector to two different subspaces can potentially *increase* their angle ($\theta(w_1, \tilde{w}_1) < \theta(\mathsf{span}(w_1, w_2), \mathsf{span}(\tilde{w}_1, w_2))$). (b) For intuition for Lemma 2: now, the two subspaces $V, \tilde{V}$ *are* close in angle, with angle at most $\frac{\pi}{2}$ times the ratio of $\theta(\tilde{b}, b) = 0.01$ to $\theta(\tilde{b}, a_1) = 0.1$.

Let $\alpha = \theta(\mathsf{span}(u_1, b), \mathsf{span}(u_1, \tilde{b}))$ and $\beta = \theta(\tilde{b}, u_1)$. We can wlog write $u_1 = (1, 0, 0)$ and assume $\mathsf{span}(u_1, b)$ is the x-y plane. Since $\mathsf{span}(u_1, \tilde{b})$ has angle α with the x-y plane and intersects the x-axis, we can write $\tilde{b} = \cos(\beta)u_1 + \sin(\beta)u_2$ for $u_2 = (0, \cos(\alpha), \sin(\alpha))$. Now, $\theta(b, \tilde{b})$ is at least $\sin^{-1}$ of the Euclidean distance of $\tilde{b}$ to the x-y plane, which is $\sin(\alpha)\sin(\beta)$. The lemma then follows from the fact that $\alpha\beta \leq \frac{\pi}{2}\sin^{-1}(\sin(\alpha)\sin(\beta))$ for $0 \leq \alpha, \beta \leq \frac{\pi}{2}$. ■

The key point of the next lemma is that errors (angles between a_i and $\tilde{a}_i$) only contribute additively to the overall angle gap between subspaces so long as each new learned vector is far from the previously-learned subspace. In contrast, a difficulty with the usual matrix perturbation analysis is that while we can assume each new $\tilde{a}_i$ is far from the span of the previous $\tilde{a}_1, \ldots, \tilde{a}_{i-1}$, we do not have control over its distance to the span of the past *and future* vectors $\{\tilde{a}_1, \ldots, \tilde{a}_{i-1}, \tilde{a}_{i+1}, \ldots, \tilde{a}_k\}$ as in the definition of matrix height (e.g., Spielman (2002)). Note that even adding the *same* vector to two different subspaces can potentially increase their angle (e.g., Figure 1(a)).

Lemma 3 *Let* $V_k = \mathsf{span}\{a_1, \ldots, a_k\}$ *and* $\tilde{V}_k = \mathsf{span}\{\tilde{a}_1, \ldots, \tilde{a}_k\}$. *Let* $\epsilon_{acc}, \gamma \geq 0$ *and* $\epsilon_{acc} \leq \gamma^2/(10k)$. *Assume for* $i = 2, \ldots, k$ *that* $\theta(\tilde{a}_i, \tilde{V}_{i-1}) \geq \gamma$, *and for* $i = 1, \ldots, n$, $\theta(a_i, \tilde{a}_i) \leq \epsilon_{acc}$. *Then* $\theta(V_k, \tilde{V}_k) \leq 2k\frac{\epsilon_{acc}}{\gamma}$.

Proof Fixing ϵ_{acc}, the proof is by induction on k, on the stronger hypothesis that the conclusion holds for $V_k = \mathsf{span}\{W, a_1, \ldots, a_k\}$ and $\tilde{V}_k = \mathsf{span}\{W, \tilde{a}_1, \ldots, \tilde{a}_k\}$ for any fixed subspace W. Note that the base case ($k = 1$), follows directly from Lemma 2, using $W = \tilde{V}_{k-1} = U$, $\tilde{a}_1 = \tilde{b}$, and $a_1 = b$. Now, let $V'_k = \mathsf{span}(V_{k-1}, \tilde{a}_k)$. Then we have:

$$
\begin{aligned}
\theta(V_k, \tilde{V}_k) &\leq \theta(V_k, V'_k) + \theta(V'_k, \tilde{V}_k) \quad \text{[by triangle inequality]} \\
&\leq \tfrac{\pi}{2}\tfrac{\theta(\tilde{a}_k, a_k)}{\theta(\tilde{a}_k, V_{k-1})} + \tfrac{2(k-1)\epsilon_{acc}}{\gamma} \\
&\quad \text{[the first term is by Lemma 2, and the second term is by induction using } W = \mathsf{span}(\tilde{a}_k)\text{]} \\
&\leq \tfrac{\pi}{2}\tfrac{\epsilon_{acc}}{\theta(\tilde{a}_k, \tilde{V}_{k-1}) - \theta(V_{k-1}, \tilde{V}_{k-1})} + \tfrac{2(k-1)\epsilon_{acc}}{\gamma} \\
&\quad \text{[by triangle inequality: } \theta(\tilde{a}_k, \tilde{V}_{k-1}) \leq \theta(\tilde{a}_k, V_{k-1}) + \theta(V_{k-1}, \tilde{V}_{k-1})\text{]} \\
&\leq \tfrac{\pi}{2}\tfrac{\epsilon_{acc}}{\gamma - \frac{2(k-1)\epsilon_{acc}}{\gamma}} + \tfrac{2(k-1)\epsilon_{acc}}{\gamma} \quad \text{[by assumption and by induction]} \\
&\leq \tfrac{\epsilon_{acc}}{\gamma}\left(\tfrac{\pi}{2}\tfrac{\gamma^2}{\gamma^2 - 2(k-1)\epsilon_{acc}} + 2(k-1)\right) \leq 2k\epsilon_{acc}/\gamma,
\end{aligned}
$$

where the last step comes from using $\epsilon_{acc} \leq \gamma^2/(10(k-1))$. ■

Algorithm 1 Life-long learning of halfspaces sharing a common low-dimensional subspace

Input: n,m,k, access to labeled examples for problems $i \in \{1, \ldots, m\}$, parameters ϵ and ϵ_{acc}.

1. Learn the first target to error ϵ_{acc} to get a vector $\alpha_1 \in R^n$. Set $\tilde{w}_1 = \alpha_1$; $\tilde{k} = 1$ and $i_1 = 1$.
2. For the learning problem $i = 2$ to m
   - Attempt to learn using the representation $v \to (\tilde{w}_1 \cdot v, ..., \tilde{w}_{\tilde{k}} \cdot v)$. I.e., check if for learning problem i there exists a hypothesis $\text{sign}(\alpha_{i,1}(\tilde{w}_1 \cdot v) + \cdots + \alpha_{i,\tilde{k}}(\tilde{w}_{\tilde{k}} \cdot v))$ of error at most ϵ.
     (a) If yes, set $\tilde{c}_i = (\alpha_{i,1}, \ldots, \alpha_{i,\tilde{k}}, 0, \ldots, 0)$.
     (b) If not, learn a classifier α_i for problem i of accuracy ϵ_{acc} by using the original features. Set $\tilde{k} = \tilde{k} + 1$, $i_{\tilde{k}} = i$, $\tilde{w}_{\tilde{k}} = \alpha_i$, and $\tilde{c}_i = e_{\tilde{k}}$.
3. Let $\tilde{W}$ be an $\tilde{k} \times n$ matrix whose rows are $\tilde{w}_1, \ldots, \tilde{w}_{\tilde{k}}$ and let $\tilde{C}$ be the matrix $m \times \tilde{k}$ matrix whose rows are $\tilde{c}_1, \ldots, \tilde{c}_{\tilde{k}}$. Compute $\tilde{A} = \tilde{C}\tilde{W}$.

Output: m predictors; predictor i is $v \to \text{sign}(\tilde{A}_i \cdot v)$

We now put these together to analyze Algorithm 1 when target functions lie on, or close to, a low-dimensional subspace. Specifically, say that a subsequence of target functions $a_{i_1}, a_{i_2}, \ldots$ is γ-separated if each a_{i_j} has angle greater than γ from the span of the previous $a_{i_1}, \ldots, a_{i_{j-1}}$. Define the *$\gamma$-effective dimension* of targets $a_1, a_2, \ldots, a_m$ as the size of the largest γ-separated subsequence. Our assumption will be that the γ-effective dimension of the targets is at most k for $\gamma = c\epsilon$ for some absolute constant $c > 0$, where ϵ is our desired error rate per target. Note that for $\gamma = 0$, γ-effective dimension equals the dimension of the subspace spanned, and for $\gamma > 0$ this allows the targets to just be "near" to a low-dimensional subspace.

Theorem 4 *Assume that all marginals D_i are isotropic log-concave. Choose $\gamma = c_1\epsilon$ and ϵ_{acc} s.t. $2k\frac{\epsilon_{acc}}{\gamma} + \gamma = c_2\epsilon$ for sufficiently small constants $c_1, c_2 > 0$. Consider running Algorithm 1 with parameters ϵ and ϵ_{acc} on any sequence of targets whose γ-effective dimension is at most k. Then $\tilde{k} \leq k$ (the rank of $\tilde{A}$ is at most k). Moreover the total number of labeled examples needed to learn all the problems to error ϵ is $\tilde{O}(nk/\epsilon_{acc} + km/\epsilon) = \tilde{O}(nk^2/\epsilon^2 + km/\epsilon)$.*

Proof We divide problems in two types: problems of type (a) are those for which we can learn a classifier of error at most ϵ by using the previously learnt problems; the rest are of type (b).

For problems of type (a) we achieve error ϵ by design. For each problem i of type (b) we open a new row in $\tilde{W}$, and set $\tilde{w}_{\hat{k}} = \alpha_i$, where $\hat{k}$ is such that $i_{\hat{k}} = i$. We also set $\tilde{c}_i = e_{\hat{k}}$, so $\tilde{a}_i = \alpha_i$. Since α_i has error at most ϵ_{acc}, we have $\theta(\tilde{w}_{\hat{k}}, a_{i_{\hat{k}}}) \leq \epsilon_{acc}/c$ for some absolute constant c (by Lemma 1).

We next show that $\tilde{k} \leq k$. We prove by induction that for each $\tilde{w}_{\hat{k}}$ we create for a problem $i = i_{\hat{k}}$, we have both (1) $a_{i_{\hat{k}}}$ is γ-far from $\mathsf{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$ and (2) $\tilde{w}_{\hat{k}}$ is γ-far from $\mathsf{span}(\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1})$. Step $\hat{k} = 1$ follows immediately. For the inductive step $\hat{k} > 1$: if we create $\tilde{w}_{\hat{k}}$ for a problem $i = i_{\hat{k}}$, this only happens if there is no vector in the span of the previous

metafeatures $\tilde{w}_j$, $j < i$ that has error less than ϵ for problem $i_{\hat{k}}$.[3] That is $a_{i_{\hat{k}}}$ is at least ϵ/c'-far from $\mathsf{span}\{\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1}\}$ for some constant c' (by Lemma 1). We also have $\theta(\tilde{w}_{\hat{k}}, a_{i_{\hat{k}}}) \leq \varepsilon_{acc}/c$. So, by triangle-inequality, $\theta(\tilde{w}_{\hat{k}}, \mathsf{span}(\tilde{w}_1, ...\tilde{w}_{\hat{k}-1})) \geq \frac{\epsilon}{c'} - \frac{\varepsilon_{acc}}{c} \geq \gamma$, for $c_1 = \frac{1}{2c'}$, c_2 sufficiently small.

Thus $\tilde{w}_{\hat{k}}$ is γ-far from $\mathsf{span}\{\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1}\}$. It remains to show that $a_{i_{\hat{k}}}$ is γ-far from the span of $\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$. Suppose for contradiction that $\theta(a_{i_{\hat{k}}}, \{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}) \leq \gamma$. By construction we have $\theta(a_{i_j}, \tilde{w}_j) \leq \epsilon_{acc}/c$ for $j \in \{1, \ldots, \hat{k}-1\}$; also by induction we have $\tilde{w}_j$ is γ-far from the span of $\{\tilde{w}_1, \cdots, \tilde{w}_{j-1}\}$ for $j \in \{1, \ldots, \hat{k}-1\}$. By Lemma 3 we obtain that $\theta(\mathsf{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}, \mathsf{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}) \leq 2k\varepsilon_{acc}/(c\gamma)$. These together with triangle inequality imply that $\theta(a_{i_{\hat{k}}}, \mathsf{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}) \leq \gamma + 2k\frac{\varepsilon_{acc}}{c\gamma} \leq \epsilon/c'$. So by Lemma 1 there exist $\tilde{b}_{i_{\hat{k}}} \in \mathsf{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}$ of error at most ϵ, which contradicts our assumption. Therefore, our induction is maintained (by condition (2)) and so we have $\tilde{k} \leq k$ (by condition (1) and our assumption on the γ-effective dimension of the targets). ■

By setting $\gamma = O(\epsilon)$ and $\epsilon_{acc} = O(\frac{\epsilon^2}{k})$ the total number of labeled examples needed to learn all the problems to error ϵ is $\tilde{O}(\frac{nk^2}{\epsilon^2} + \frac{km}{\epsilon})$, which could be significantly less than the $\Omega(\frac{mn}{\epsilon})$ needed to learn each problem separately, even under log-concave distributions (Balcan and Long, 2013).

Note 1 *As stated, Algorithm 1 is not efficient because it requires finding an optimal linear separator in Step 2, which in general is hard. However, for log-concave distributions, there exist algorithms running in time poly$(k, 1/\epsilon)$ that find a near-optimal linear separator: in particular, one of error ϵ under the assumption that the optimal separator has error $\eta = \epsilon/\log^2(1/\epsilon)$ (Awasthi et al., 2014), and with near-optimal sample complexity (Hanneke, 2013; Yang, 2013). Thus, by reducing ε_{acc} by an $O(\log^2(1/\epsilon))$ factor, one can achieve the bounds of Theorem 1 efficiently.*

2.1. Halfspaces with more complex common structure

We now consider life-long learning of halfspaces with more complex common structure, corresponding to a multi-layer network of linear metafeatures. It is at first not obvious how multiple levels of linear nodes could help: if the target vectors span a k-dimensional subspace, then to represent them with a multi-layer linear network, each layer would need to have at least k nodes. However, sample complexity of learning can also be reduced via sparsity.

Specifically, we assume now that the target functions all lie in a k dimensional space and that furthermore within that k-dimensional space, each target lies in one of r different τ-dimensional spaces. This naturally models settings where there are really r different types of learning problems but they share some overall commonalities (given by the common k-dimensional subspace). We can view this as a network with two hidden layers: the first layer given by vectors $w_1, w_2, \ldots, w_k$, and the second given by r τ-tuples of vectors, $u_1^1, \ldots, u_1^\tau$, ..., $u_r^1, \ldots, u_r^\tau$, where $u_i^1, \ldots, u_i^\tau$ span one of τ-dimensional spaces. In other words, the first hidden layer captures the overall low dimensionality and the second captures sparsity. We assume $r \ll m$, $k \ll n$ and that τ is a constant.

Algorithmically, given a new problem we first try to learn well via a sparse linear combination of only τ second level metafeatures. If we fail, we try to learn based on the first level metafeatures and if successful we add a new second level metafeature corresponding to this target. If that fails,

3. Technically, since we are learning over a finite sample, we can only be confident that there is no vector in the span of error at most $\epsilon/2$. However, we can absorb these factors of 2 into the constants c, c'.

we learn using the input features and then we add both a first and second level metafeature corresponding to this target. For log-concave distributions, by using the subspace lemma and an error analysis similar to that for Theorem 4 we can show we have $\tilde{k} \leq k$ and $\tilde{r} \leq \tau r$. Formally:

Theorem 5 *Assume all marginals D_i are isotropic log-concave and the target functions satisfy the above conditions. Consider $\tilde{\gamma} \leq c\epsilon$, $\tilde{\epsilon}_{acc} \leq c\frac{\tilde{\gamma}\epsilon}{\tau}$, $\gamma \leq c\tilde{\epsilon}_{acc}$, and $\epsilon_{acc} \leq c\frac{\gamma\tilde{\epsilon}_{acc}}{k}$ for (sufficiently small) constant $c > 0$. Consider running Algorithm 4 (see appendix) with parameters ϵ, ϵ_{acc}, and $\tilde{\epsilon}_{acc}$. Then $\tilde{k} \leq k$ and $\tilde{r} \leq \tau r$. Moreover the total number of examples needed to learn all the problems to error ϵ is $\tilde{O}(nk/\epsilon_{acc} + kr/\tilde{\epsilon}_{acc} + m\log(r)/\epsilon)$.*

(Proof in Appendix B). By setting $\tilde{\gamma} = \epsilon/2$, $\tilde{\epsilon}_{acc} = \Theta(\epsilon^2/\tau)$, $\gamma = \Theta(\epsilon^2/\tau)$, $\epsilon_{acc} = \Theta(\epsilon^4/\tau^2 k)$ we get that the total number of labeled examples needed to learn all the problems to error ϵ is $\tilde{O}(nk^2\tau^2/\epsilon^4 + kr\tau^2/\epsilon^2 + m\tau\log(r)/\epsilon)$. This could be significantly lower than learning each problem separately or by learning the problems together but only using one layer of metafeatures. Specifically, if we used one layer of metafeatures as in Theorem 4 (corresponding to the k-dimensional subspace) the sample complexity would be $O(nk^2/\epsilon^2 + mk/\epsilon)$. Alternatively we could have just one middle layer of size $r\tau$ and learn sparsely within that, but this would also give worse bounds if r is large. As a concrete example, if ϵ is constant, $k = \sqrt{n}$, $r = n^2$ and $m = n^{2.5}$, we get that the two-layer algorithm requires only $O(\tau^2/\epsilon^2 + \tau\log(r)/\epsilon)$ examples per target. On the other hand, the other two options require at least $O(k/\epsilon)$ examples per target, which could be much worse.

3. Life-long Learning of Monomials

We now consider a nonlinear case where the metafeatures are products and combined via products. Specifically, we assume an instance space $X = \{0,1\}^n$, that the m target functions are conjunctions (i.e., products) of features, and that there exist k monomial metafeatures such that all the target functions can be expressed as products over them. Our goal will be to learn them efficiently.

If the metafeatures do not overlap, then this can be viewed as an instance of the linear case. Each target function can be described by an indicator vector with coefficients in $\{0,1\}$ (plus a threshold) that all lie in a space of rank k with basis given by the indicator vectors of the metafeatures.

So, the interesting case is when metafeatures may overlap. Unfortunately, without any additional assumptions, even just the consistency problem is now NP-hard. That is, given a collection of conjunctions, it is NP-hard to determine whether there exist k monomials such that each can be written as a product of subsets of those monomials (the "set-basis problem" (Garey and Johnson, 1979)). For this reason, we will make a natural *anchor-variable* assumption that each metafeature m_i has at least one "anchor" variable (call it y_i) that is not in any other metafeature m_j. So this is a generalization of the disjoint case where *every* variable in m_i is not inside any other m_j.

We now show how with this assumption we can efficiently solve the consistency problem (and *find* the smallest set of monomials for which one can reconstruct each target). Using this as a subroutine, we then show how to solve an abstract online learning problem where at each stage we must propose a set of at most k monomial metafeatures and then pay a cost of 1 if the next target cannot be written as a product over them. This can then be applied to give efficient life-long learning of related conjunctions over product distributions. In Section 3.3 we give an application to constructing Boolean superimposition-based autoencoders. We then relax the anchor-variable assumption and show how under this relaxed condition we can solve for near-optimal *sparse* autoencoders as well as life-long learning of conjunctions under relaxed conditions. In Section 4, we build on some of these results to give an algorithm for life-long learning of polynomials.

3.1. Solving the Consistency Problem

We now show how given a collection of conjunctions, we can efficiently find the fewest monomial metafeatures needed to reconstruct all of them as products of metafeatures, under the anchor variable assumption. First, given a conjunction T, let $\text{vars}(T)$ denote the variables appearing in T. Given a variable z and a set of conjunctions TS, let $S(\text{TS}, z)$ denote the set of conjunctions in TS that contain z, and let $conj(\text{TS}, z)$ denote the conjunction of all variables that appear in every conjunction in $S(\text{TS}, z)$; that is, $\text{vars}(conj(\text{TS}, z)) = \bigcap_{T \in S(\text{TS},z)} \text{vars}(T)$. For intuition, suppose the true metafeatures are m_1, m_2, m_3 and the conjunctions observed are $T_1 = m_1 m_2$, $T_2 = m_2 m_3$, and $T_3 = m_2$. In this case, variables z whose sets $S(\text{TS}, z)$ are minimal will be anchors for m_1 and m_3; however, only after "pulling them out" will we be able to identify an anchor for m_2.

Algorithm 2 Consistency problem for monomial metafeatures with anchor variables

Input: set $\text{TS} = \{T_1, \ldots, T_{\text{r}}\}$ of conjunctions.

1. Let $i = 0$.
2. Let $h(T)$ denote the conjunction of all metafeatures $\tilde{m}_j$ produced so far that are fully contained in T. I.e., $\text{vars}(h(T)) = \cup\{\text{vars}(\tilde{m}_j) : \text{vars}(\tilde{m}_j) \subseteq \text{vars}(T)\}$.
3. While there exists $T \in \text{TS}$ s.t. $\text{vars}(T) \neq \text{vars}(h(T))$ do:
   (1) Let T be the target of least index in TS s.t. $\text{vars}(T) \neq \text{vars}(h(T))$.
   (2) Choose z_{i+1} to be a minimal variable in $\text{vars}(T) \setminus \text{vars}(h(T))$; that is, there is no other variable $z' \in \text{vars}(T) \setminus \text{vars}(h(T))$ s.t. $S(\text{TS}, z') \subset S(\text{TS}, z)$. If there are multiple options, choose z_{i+1} to be the option of least index.
   (3) Let $\tilde{m}_{i+1} = conj(\text{TS}, z_{i+1})$. Let $i = i + 1$.

Output: Conjunctions $\tilde{m}_1, \ldots, \tilde{m}_i$ s.t. each T_j is a conjunction of a subset of them.

Lemma 6 *Let* TS *be a set of conjunctions each of which is a conjunction of some subset of metafeatures $m_1, \ldots, m_k$ satisfying the anchor variable condition with anchor variables $y_1, \ldots, y_k$. We can use Algorithm 2 to find $\tilde{m}_1, \ldots, \tilde{m}_i$, $i \leq k$ s.t. each $T_j \in$ TS is a conjunction of a subset of them. Moreover each $\tilde{m}_i$ is associated to a distinct metafeature m_{t_i} such that:*

(a) $\text{vars}(m_{t_i}) \subseteq vars(\tilde{m}_i)$*; that is, $\tilde{m}_i$ is more specific than m_{t_i}.*

(b) For all targets T in TS *such that* $\text{vars}(m_{t_i}) \subseteq \text{vars}(T)$ *we have* $\text{vars}(\tilde{m}_i) \subseteq vars(T)$*; that is, $\tilde{m}_i$ is not too specific.*

(c) For any j, if $y_j \in \text{vars}(\tilde{m}_i)$ then $\text{vars}(m_j) \subseteq \text{vars}(\tilde{m}_i)$*.*

Proof We prove the desired statement by induction. Assume inductively that $\tilde{m}_1, \ldots, \tilde{m}_i$ satisfy conditions (a),(b),(c). We show that $\tilde{m}_{i+1}$ satisfies these conditions as well.

Consider the target T we choose in step 3(1) in round $i + 1$. We know $z_{i+1} \in \text{vars}(T) \setminus \text{vars}(h(T))$ and that T is a conjunction of the true metafeatures. So z_{i+1} belongs to some metafeature $m_{t_{i+1}}$ s.t. $\text{vars}(m_{t_{i+1}}) \subseteq \text{vars}(T)$. From the induction hypothesis, by conditions (a),(b) we know that $m_{t_{i+1}} \neq m_{t_{i'}}$ for $i' \leq i$. Now, consider $T \in \text{TS}$ such that $\text{vars}(m_{t_{i+1}}) \subseteq \text{vars}(T)$. Since $z_{i+1} \in \text{vars}(m_{t_{i+1}})$ and we create $\tilde{m}_{i+1}$ by intersecting the variables in every target T containing z_{i+1}, we clearly have $\text{vars}(\tilde{m}_{i+1}) \subseteq \text{vars}(T)$, satisfying condition (b). Also if any target T contains an anchor variable y_j, then it must contain m_j, so condition (c) is satisfied as well.

We now show that (a) is satisfied, namely that $\text{vars}(m_{t_{i+1}}) \subseteq \text{vars}(\tilde{m}_{i+1})$. This could only fail if z_{i+1} is not an anchor for $m_{t_{i+1}}$, so in step 2 of the algorithm we intersected some target T that contains z_{i+1} but does not contain $m_{t_{i+1}}$. This can only happen if z_{i+1} also belongs to some other m_j. But then z_{i+1} is not minimal since $y_{t_{i+1}}$ (the true anchor variable for $m_{t_{i+1}}$, which is also contained in $\text{vars}(T) \setminus \text{vars}(h(T))$ by (c)) satisfies $S(\text{TS}, y_{t_{i+1}}) \subset S(\text{TS}, z_{i+1})$, and so would have been chosen instead of z_{i+1} in step $3(1)$. ■

3.2. Solving the Lifelong Learning Problem

Building on Algorithm 2 and Lemma 6, we now consider the lifelong-learning setting. We begin with an abstract setting where at each time-step r we propose a set $\tilde{M}$ of at most k hypothesized metafeatures and are provided with a target conjunction T_{r}. If T_{r} can be written as a conjunction of metafeatures in $\tilde{M}$ then we pay 0. If not, then we pay 1 and may update our set $\tilde{M}$ using T_{r} (this corresponds to the case of learning T_{r} from scratch). Our goal is to bound our total cost, under the assumption that there *exists* a set of k metafeatures for all targets.

Algorithm 3 Lifelong Learning of Conjunctions with Monomial Metafeatures

Input: Targets $T_1, T_2, \ldots, T_m$ provided online.

1. Initialize $\text{TS} = \emptyset$ and $\tilde{M} = \emptyset$.
2. For $\text{r} = 1$ to m do:
   - If we cannot represent T_{r} as conjunction of hypothesized metafeatures $\tilde{M}$ then add T_{r} to TS and run Algorithm 2 with input TS to produce hypothesized metafeatures $\tilde{M}$.

Output: Hypothesized metafeatures $\tilde{M}$.

Theorem 7 *The number of targets learned from scratch in Algorithm 3 is at most $n^2 + k$.*

(Proof in Appendix C.) Since conjunctions over $\{0,1\}^n$ can be exactly learned in the Equivalence Query model with at most n equivalence queries (and conjunctions over $\{0,1\}^k$ can be learned from at most k equivalence queries), we immediately have the following:

Corollary 8 *Let* TS *be a sequence of m conjunctions such that each is a conjunction of some subset of metafeatures $m_1, \ldots, m_k$ satisfying the anchor variable condition. Then this sequence can be learned using only $O(mk + n^3)$ equivalence queries total.*

We can also use Theorem 7 to learn with good sample complexity over any product distribution D.

Theorem 9 *Assume that all D_{r} are the same product distribution D, the metafeatures satisfy the anchor variable assumption, and all the target functions c_{r} are balanced. We can learn hypotheses $h_1, \ldots, h_m$ of error at most ϵ by using Algorithm 5 with parameters $s_1(n, \epsilon, \delta) = O(\frac{n}{\epsilon}\log(n/\delta))$, $s_2(n, \epsilon, \delta) = \frac{k}{\epsilon}\log(m/\delta)$, and $s_3(n, \epsilon, \delta) = \frac{n}{\epsilon}\log(nk/\delta)$. The total number of labeled examples needed is $\tilde{O}((n^2 + k)\frac{n}{\epsilon}\log(n/\delta) + \frac{km}{\epsilon})$. (Algorithm and Proof in Appendix C.)*

3.3. Sparse Boolean Autoencoders and Relaxing the Anchor-Variable Assumption

The above results (in particular, Lemma 6) can be interpreted as constructing a minimal feature space for AND/OR autoencoding. Specifically, consider a collection of black-and-while pixel images $\{T_{\text{r}}\}$ where each $T_{\text{r}} \in \{0,1\}^n$. Our goal is to contruct a 2-level auto-encoder $\mathcal{A}$ (for each r, we want $\mathcal{A}(T_{\text{r}}) = T_{\text{r}}$) with as few nodes in the middle (hidden) level as possible, such that nodes

in the hidden level compute the AND of their inputs, and nodes in the output level compute the OR of their inputs. We can view each hidden node in such a network as representing a "piece" of an image, with the autoencoding property requiring that each $T_{\rm r}$ should be equal to the bitwise-OR of all pieces contained within it (i.e., superimposing them together). Formally, for each hidden node j, let $m_j \in \{0,1\}^n$ denote the indicator vector for the set of inputs to that node (which without loss of generality will also be the set of outputs of that node), and say that $m_j \preceq T_{\rm r}$ if each bit set to 1 in m_j is also set to 1 in $T_{\rm r}$; we then require $T_{\rm r}$ to be the bitwise-OR of all $m_j \preceq T_{\rm r}$. Lemma 6 implies that given a collection of images $\{T_{\rm r}\}$, Algorithm 2 finds the fewest hidden nodes needed to perform this autoencoding, under the assumption that each metafeature m_j contains some anchor-variable.

We now consider the problem of *sparse* Boolean autoencoding. That is, given a set $\mathrm{TS} = \{T_{\rm r}\}$, with each $T_{\rm r} \in \{0,1\}^n$, our goal is to find a collection of metafeatures $\tilde{m}_j$ (perhaps more than n of them) such that each $T_{\rm r} \in \mathrm{TS}$ can be written as the bitwise-OR of at most k of the $\tilde{m}_j$ (where $k \ll n$). Clearly this is trivial by having one metafeature $\tilde{m}_j$ for each $T_{\rm r}$, so our goal will be to have the (approximately) fewest of them subject to this condition. Additionally, because we want sparse reconstruction, we want for each $T_{\rm r}$ that $|\{j : \tilde{m}_j \preceq T_{\rm r}\}|$ should be small as well. This problem has two motivations. From the perspective of autoencoding, this corresponds to finding a sparse autoencoder (viewing the $T_{\rm r}$ as pixel images). From the perspective of life-long learning, if this can be done online then (viewing the $T_{\rm r}$ as conjunctions) it will allow for fast learning, since conjunctions of k out of N variables can be learned with sample complexity only $O(k \log N)$; in this case we would actually not need the additional "sparse reconstruction" property above.

To solve this problem, we make a relaxed version of the anchor-variable assumption (anchor-variables no longer make sense when there are more than n metafeatures) which is that each metafeature should have a *set* of $\leq c$ variables (for some constant c) such that any $T_{\rm r}$ containing that set should have the metafeature as one of its k "relevant metafeatures". We call this the *c-anchor-set* assumption. Note that metafeatures satisfying the anchor-*variable* assumption will also satisfy this condition for $c = 1$. Note also that in general the c-anchor-set assumption is a requirement on both the metafeatures and on the set TS. Formally, we make the following definition:

Definition 10 *A set of metafeatures $M = \{m_j\}$ and set of targets $\mathrm{TS} = \{T_{\rm r}\}$ satisfy the* c-anchor-set *assumption at sparsity level k if*

1. *for each $T_{\rm r} \in \mathrm{TS}$ there exists a set $R_{\rm r}$ of at most k "relevant" metafeatures in M such that $T_{\rm r}$ is the bitwise-OR of the metafeatures in $R_{\rm r}$, and*
2. *For each $m_j \in M$ there exists $y_j \preceq m_j$ of Hamming weight at most c such that for all $\rm r$, if $y_j \preceq T_{\rm r}$ then $m_j \in R_{\rm r}$. Note that in particular this implies that $|\{j : m_j \preceq T_{\rm r}\}| \leq k$.*

Under this assumption, we can solve for a near-optimal set of metafeatures (proofs in Appendix D).

Theorem 11 *Given a set of targets $\mathrm{TS} = \{T_{\rm r}\}$ in $\{0,1\}^n$, suppose there exists a set of metafeatures M satisfying the c-anchor-set assumption at sparsity level k. Then in time $poly(n^c)$ we can:*

1. *Find a set of $O(n^c)$ metafeatures such that each $T_{\rm r} \in \mathrm{TS}$ can be written as the bitwise-OR of at most k of them, and*
2. *Find a set of $O(|M| \log(n|\mathrm{TS}|))$ metafeatures that satisfy the c-anchor-set assumption with respect to TS at sparsity level $O(k \log(n|\mathrm{TS}|))$.*

Corollary 12 *Let TS be a sequence of m conjunctions for which there exists a set M of conjunctive metafeatures satisfying the c-anchor-set assumption at sparsity-level k for some constant c. Then this sequence can be efficiently learned using $O(mk \log(n) + n^2|M|)$ equivalence queries total.*

4. Life-long Learning of Polynomials

We now show an application of the results in Section 3 to the case where the target functions are polynomials from $\{0,1\}^n$ to $\mathbb{R}$, whose terms "share" a small number of pieces. Specifically, we assume there exist k product metafeatures (which might overlap) such that each monomial in each target polynomial can be written as a product of some subset of them. For example, if our metafeatures are $\{x_1x_2x_3, x_3x_4x_5, x_5x_6x_7, x_7x_8x_1\}$ then we might have polynomials such as $4x_3x_4x_5x_6x_7 - 2x_5x_6x_7x_8x_1$ and $3x_1x_2x_3x_4x_5 + 3x_1x_2x_3x_7x_8$. If the target polynomials use r distinct monomials in total, then viewed as a network we have k nodes in a first hidden layer, where each is a *product* of some of the inputs, r nodes in a second hidden layer, where each is a *product* of outputs of the first hidden layer, and then the final outputs (our target functions) are weighted *linear* functions of the second hidden layer. Efficiently learning polynomials requires membership queries (under the assumption that juntas are hard to learn) even in the single task setting (Schapire and Sellie, 1993). So we will assume access to membership queries as well. However, our goal will be to use these sparingly, only when we need to learn a new function from scratch. When learning from scratch we use an algorithm of Schapire and Sellie (1993) that learns polynomials exactly.

We assume that each target polynomial has L_1 norm bounded by B. If the monomials in such a target can indeed be written as products of our metafeatures, then by considering all products of metafeatures and running an L_1-based algorithm for learning linear functions (Littlestone et al., 1995), we can achieve low mean squared error using only $O(B^2 \log(2^k)) = O(B^2k)$ examples.

Theorem 13 *Assume target functions are polynomials satisfying the above assumptions. Running Algorithm 6 (Appendix E), the total number of targets learned from scratch using membership queries is at most $n^2 + k$. For the rest of the targets—learned from random examples only—sample complexity is $O(B^2k)$ per problem and running time is polynomial in n and 2^k.*

Note that while the sample complexity is linear in k for problems learned from random examples only, the *running time* is exponential in k. However, a poly(k) bound seems unachievable because it would require solving the junta learning problem. In particular, the problem of learning polynomials over k metafeatures is at least as hard as learning polynomials over $\{0,1\}^k$ (because the metafeatures could just be $x_1, \ldots, x_k$). Thus, for this problem one should think of k as small.

5. Discussion and Open Problems

In this work we present algorithms for learning new internal representations when presented with a series of learning problems arriving online that share different types of commonalities. For the case of linear threshold functions sharing linear subspaces, we require log-concave distributions to ensure that error can be both upper-bounded and lower-bounded by some "nice" function of angle: the lower bound helps to ensure that the span of accurate hypotheses is close to the span of their corresponding true targets (though one must be careful with error accumulation), and the upper-bound ensures that a sufficiently-close approximation to the span of the true targets is nearly as good as the span itself. It is an interesting question whether one can extend these results to distributions that do not have such properties while still maintaining the streaming nature of the algorithms (i.e., remembering only the learned rules and not the data from which they were generated). For product metafeatures, our results have natural interpretations as autoencoders, which interestingly do not require the metafeatures to be incoherent or a generative model, only the anchor-variable or anchor-set assumption. It would be interesting to see whether an analog of the anchor-set assumption could be applied to dictionary learning problems such as in Arora et al. (2014); Bansal et al. (2014).

References

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning Journal*, 2008.

Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models - going beyond SVD. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2012.

Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 779–806, 2014.

Pranjal Awasthi, Maria-Florina Balcan, and Philip M. Long. The power of localization for efficiently learning linear separators with noise. In *Symposium on Theory of Computing (STOC)*, pages 449–458, 2014.

M.-F. Balcan and P. M. Long. Active and passive learning of linear separators under log-concave distributions. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

T. Bansal, C. Bhattacharyya, and R. Kannan. A provable SVD-based algorithm for learning topics in dominant admixture corpus. *ArXiv e-prints*, October 2014.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 2000.

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

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT*, 2003.

Y. Bengio. Deep learning of representations: Looking forward, 2013. arXiv report 1305.0445.

Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *ALT*, 2011.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 2010.

Shimon Edelman. Representation, similarity, and the chorus of prototypes. *Minds and Machines*, 5 (1):45–68, 1995. URL http://dx.doi.org/10.1007/BF00974189.

Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 524–532, 1989. URL http://papers.nips.cc/paper/207-the-cascade-correlation-learning-architecture.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

A. Gopnik, A. Meltzoff, and P. Kuhl. *How babies think*. Orion, 2001.

S. Hanneke. Personal communication. 2013.

A. R. Klivans, P. M. Long, and A. Tang. Baum's algorithm learns intersections of halfspaces with respect to log-concave distributions. In *RANDOM*, 2009.

A. Kumar and H. Daume. Learning task grouping and overlap in multi-task learning. In *NIPS*, 2012.

Nick Littlestone, Philip M. Long, and Manfred K. Warmuth. On-line learning of linear functions. *Computational Complexity*, 5(1):1–23, 1995.

László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358, 2007.

A. Maurer and M. Pontil. Excess risk bounds for multitask learning with trace norm regularization. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

Andreas Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6:967–994, 2005. URL http://www.jmlr.org/papers/v6/maurer05a.html.

Ronald L. Rivest and Robert H. Sloan. Learning complicated concepts reliably and usefully. In *COLT*, pages 69–79, 1988.

Volker Roth and Julia E Vogt. A complete analysis of the l_1, p group-lasso. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 185–192, 2012.

R. E. Schapire and L. M. Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. In *Proceedings of the 6th Annual Conference on Computational Learning Theory*, 1993.

D. Spielman. Lecture notes for 18.409: The behavior of algorithms in practice. Lecture 2: On the condition number. 2002.

S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

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

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

Sebastian Thrun and Tom M. Mitchell. Learning one more thing. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1217–1225, 1995b.

L. G. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 2000.

S. Vempala. A random-sampling-based algorithm for learning intersections of halfspaces. *JACM*, 57(6), 2010.

L. Yang. *Mathematical Theories of Interaction with Oracles*. PhD thesis, CMU Dept. Machine Learning, 2013.

Acknowledgments

This work was supported in part by NSF grants CCF-0953192, CCF-1451177, CCF-1422910, CCF-1217793, EAGER-1415498, IIS-1065251, AFOSR grant FA9550-09-1-0538, ONR grant N00014-09-1-0751, and a Microsoft Research Faculty Fellowship.

Appendix A. Proof of Lemma 1

Proof The proof of the lower bound appears in Balcan and Long (2013). The proof of the upper bound is implicit in the earlier work of Vempala (2010) – we provide it here for completeness. The key idea is to project the region of disagreement in the space given by the two normal vectors, and then using properties of log-concave distributions in 2-dimensions. Specifically, consider the plane determined by u and v, and let g be the 2-dimensional marginal of the density function over this plane. Then g is an isotropic and log-concave density function over R^2.

It is known Klivans et al. (2009) that for some constants k_3, k_4 we have $g(z) \leq k_3 e^{-k_4||z||}$. Given this fact, we just need to show that the integral of $k_3 e^{-k_4||z||}$ over the region $\{z : u \cdot z \geq 0, v \cdot z \leq 0\}$ is at most $c'\alpha$ for some constant c', where α is the angle between u and v (the integral over $\{z : u \cdot z \leq 0, v \cdot z \geq 0\}$ is analogous). In particular, using polar coordinates, we can write the integral as:

$$\int_{\theta=0}^{\alpha} \int_{r=0}^{\infty} f(r\cos\theta, r\sin\theta)\ r\ dr d\theta \leq \int_{\theta=0}^{\alpha} \int_{r=0}^{\infty} r k_3 e^{-k_4 r} dr d\theta.$$

The inner integral evaluates to a constant and therefore the entire integral is bounded by $c'\alpha$ for some constant c' as desired. ■

Appendix B. Proofs for halfspaces with more complex common structure

We now provide the algorithm and proof for Theorem 5.

Theorem 5 *Assume all marginals D_i are isotropic log-concave and the target functions satisfy the above conditions. Consider $\tilde{\gamma} \leq c\epsilon$, $\tilde{\epsilon}_{acc} \leq c\frac{\tilde{\gamma}\epsilon}{\tau}$, $\gamma \leq c\tilde{\epsilon}_{acc}$, and $\epsilon_{acc} \leq c\frac{\gamma\tilde{\epsilon}_{acc}}{k}$ for (sufficiently small) constant $c > 0$. Consider running Algorithm 4 with parameters ϵ, ϵ_{acc}, and $\tilde{\epsilon}_{acc}$. Then $\tilde{k} \leq k$ and $\tilde{r} \leq \tau r$. Moreover the total number of examples needed to learn all the problems to error ϵ is $\tilde{O}(nk/\epsilon_{acc} + kr/\tilde{\epsilon}_{acc} + m\log(r)/\epsilon)$.*

Proof We divide problems in two types: problems of type (a) are those for which we can learn a classifier of desired error at most ϵ by using the previously learnt metafeatures at the second middle level; the rest are of type (b).

For problems of type (a) we achieve error at most ϵ by design. For each problem i of type (b) we have either opened a new row in $\tilde{U}$, and we have set $\tilde{w}_{\hat{r}} = \alpha_i$, where $\hat{r}$ is such that $j_{\hat{r}} = i$ or we have opened both a new row in $\hat{r}$ in $\tilde{U}$ and a new row $\hat{k}$ in $\tilde{W}$, and set $j_{\hat{r}} = i$ and $i_{\hat{k}} = i$. In both cases, by design and Lemma 1 (and the fact that $\epsilon_{acc} \leq \tilde{\epsilon}_{acc}$) we have $\theta(\tilde{u}_{\hat{r}}\tilde{W}, a_{j_{\hat{r}}}) = O(\tilde{\epsilon}_{acc})$; furthermore since $\tilde{c}_i = e_{\hat{r}}$ we also have $\theta(\tilde{a}_i, a_i) = O(\tilde{\epsilon}_{acc})$. Furthermore, for each $\tilde{u}_{\hat{r}}$ we create for a problem $j_{\hat{r}}$ we have that $\tilde{u}_{\hat{r}}\tilde{W}$ is $\tilde{\gamma}$-far from the span of those vectors in $\{\tilde{u}_1, ..., \tilde{u}_{j_{\hat{r}-1}}\}$ whose corresponding targets lie in space U_s, where U_s is one of the r τ-dimensional subspaces that $a_{j_{\hat{r}}}$

Algorithm 4 Life-long Learning with two levels of linear shared metafeatures

Input: n,m,k, access to labeled examples for problems $i \in \{1, \ldots, m\}$, parameters ϵ, ϵ_{acc}, $\tilde{\epsilon}_{acc}$.

1. Learn the first target to error ϵ_{acc} to get a an n-dimensional vector α_1.
2. Set $\tilde{w}_1 = \alpha_1$, $\tilde{k} = 1$, $\tilde{u}_1 = (1)$, $\tilde{r} = 1$, $\tilde{c}_1 = (1)$, and $i_1 = 1$.
3. For the learning problem $i = 2$ to m
   - Try to learn τ-sparsely by using the $\tilde{r}$ dimensional representation given by the second level meta-features $v \to \tilde{U}\tilde{W}v$. I.e., check whether for the learning problem i there exists a τ-sparse hypothesis $\text{sign}(\alpha_{i,1}(\tilde{u}_1\tilde{W}v) + \cdots + \alpha_{i,\tilde{r}}(\tilde{u}_{\tilde{r}}\tilde{W}v))$ of error at most ϵ.
     (a) If yes, set $\tilde{c}_i = (\alpha_{i,1}, \ldots, \alpha_{i,\tilde{k}})$.
     (b) Otherwise, check whether for learning problem i there exists a hypothesis in the $\tilde{k}$ dimensional representation given by the first level meta-features $v \to \tilde{W}v$ of error at most $\tilde{\epsilon}_{acc}$, i.e., a hypothesis $\text{sign}(\alpha_{i,1}(\tilde{w}_1 \cdot v) + \cdots + \alpha_{i,\tilde{k}}(\tilde{w}_{\tilde{k}} \cdot v))$ of error $\leq \tilde{\epsilon}_{acc}$.
        (a) If yes, set $\tilde{r} = \tilde{r} + 1$, $\tilde{u}_{\tilde{r}} = (\alpha_{i,1}, \ldots, \alpha_{i,\tilde{k}})$, $j_{\tilde{r}} = i$. Extend all rows of $\tilde{C}$ with one zero, set $\tilde{c}_i = e_{\tilde{r}}$.
        (b) If not, learn a classifier α_i for problem i of accuracy ϵ_{acc} by using the original features. Set $\tilde{k} = \tilde{k} + 1$, $i_{\tilde{k}} = i$, $\tilde{w}_{\tilde{k}} = \alpha_i$. Extend all rows of $\tilde{U}$ by one zero, set $\tilde{r} = \tilde{r} + 1$, $\tilde{u}_{\tilde{r}} = e_{\tilde{k}}$. Extend all rows of $\tilde{C}$ with one zero, set $\tilde{c}_i = e_{\tilde{r}}$, $j_{\tilde{r}} = i$.
4. Let $\tilde{W}$ be an $\tilde{k} \times n$ matrix whose rows are $\tilde{w}_1, \ldots, \tilde{w}_{\tilde{k}}$; let $\tilde{U}$ be an $\tilde{r} \times \tilde{k}$ matrix whose rows are $\tilde{u}_1, \ldots, \tilde{u}_{\tilde{r}}$; and let $\tilde{C}$ be the matrix $m \times \tilde{k}$ matrix whose rows are $\tilde{c}_1, \ldots, \tilde{c}_{\tilde{k}}$. Compute $\tilde{A} = \tilde{C}\tilde{U}\tilde{W}$.

Output: m predictors; predictor i is $v \to \text{sign}(\tilde{A}_i \cdot v)$

belongs to. (Otherwise if $\tilde{u}_{\hat{r}}$ is $\tilde{\gamma}$-close we would have been able to learn sparsely to error ϵ based on the second level metafeatures.)

Using this together with the fact that $\tilde{\epsilon}_{acc} = O(\frac{\tilde{\gamma}\epsilon}{\tau})$, we obtain (by Lemma 3) that once we have τ second level meta-features $\tilde{u}_{j_{l_1}}, \ldots, \tilde{u}_{j_{l_\tau}}$ whose corresponding targets $a_{l_1}, \ldots, a_{l_\tau}$ lie in the same τ-dimensional space U_s, we have

$$\theta(U_s, \mathsf{span}(\tilde{u}_{j_{l_1}}\tilde{W}, \ldots, \tilde{u}_{j_{l_\tau}}\tilde{W})) = O(\tau\tilde{\epsilon}_{acc}/\tilde{\gamma}) \leq \epsilon.$$

Therefore we will be able to learn based on second level metafeatures any future target belonging to that subspace. This implies $\tilde{r} \leq \tau r$.

Using the fact that $\epsilon_{acc} \leq c\frac{\tilde{\gamma}\tilde{\epsilon}_{acc}}{k}$, as in the proof of Theorem 4, we can prove by induction that for each $\tilde{w}_{\hat{k}}$ we create for a problem $i_{\hat{k}}$, we have $a_{i_{\hat{k}}}$ is γ-far from $\mathsf{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$ and $\tilde{w}_{\hat{k}}$ is γ-far from $\mathsf{span}(\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1})$; this implies $\tilde{k} \leq k$. ■

Appendix C. Proofs for Lifelong Learning of Conjunctions (Theorems 7 and 9)

C.1. Proof of Theorem 7

Proof For any given set of targets TS learnt from scratch, we define a directed graph G_{TS} on the variables, by adding an edge (x_i, x_j) if every target in TS that has x_i also has x_j. Note that if $\mathrm{TS} \subseteq \tilde{\mathrm{TS}}$ we have $E(G_{\mathrm{TS}}) \subseteq E(G_{\tilde{\mathrm{TS}}})$. We start with the complete directed graph (corresponding to $\mathrm{TS} = \emptyset$), and then we argue that each time we are forced to learn a new target from scratch and increase TS we either delete at least one edge from the graph or we increment the number of hypothesized metafeatures by 1.

Suppose the new target T_{r} cannot be represented using the current hypothesis metafeatures. So we add T_{r} into TS and re-run Algorithm 2 . Let us look at the first time the new run differs from the old run. There are three possibilities for this difference.

(1) It could be that we choose a different z_{i+1} in step $3(2)$ of Algorithm 2. There are two ways this can happen: (a) the old z_{i+1} is not minimal any more or (b) it could be some z' (of lower index than the old z_{i+1}) was not minimal before but is minimal now. In case (a) we have some z' is now in a strict subset of the targets in TS that contain z_{i+1} but this was not the case before adding T_{r}. This means the new target T_{r} must contain the old z_{i+1} but not z', and all previous targets that contained either z' or z_{i+1} contained both of them. That means we cut the edge (z_{i+1}, z'). In case (b), some z' (of lower index than the old z_{i+1}) was not minimal before but is minimal now. This means that before there was some z'' that was in a strict subset of the targets as z', but it is not anymore. Now, z' is minimal, z'' is no longer in a strict subset of the targets containing z'; so the new target contains z'' but not z'. So we cut the edge (z'', z').

(2) It could be that we get the same z_{i+1} but different $\tilde{m}_{i+1}$ in step $3(3)$; this means $\mathrm{vars}(\tilde{m}_{i+1})$ is smaller. Thus we cut the edges between z_{i+1} and all the variables in the old $\tilde{m}_{i+1}$ that are not in the new $\tilde{m}_{i+1}$.

(3) It could be that we use the new target T_{r} in step $3(1)$. Since we go through the targets in order, the only way that the first difference can be when the new target is used in 3(1) is if every previous metafeature is created the same as before. Therefore, in this case we create a new metafeature. So, the number of metafeatures is increasing and we make progress as desired. ■

C.2. Proof of Theorem 9

Proof Let us call a variable i *insignificant* if over a sample of size $\Theta((n/\epsilon)\log(n/\delta))$ appears set to 0 less than $\epsilon/4n$ fraction of the time. Let I be the set of insignificant variables and let S be the set of significant variables. Let D_S be the distribution D restricted to examples that are set to 1 on all variables in I. We can show that error at most $\epsilon/2$ over D_S implies error at most ϵ over D. This is true, since by Chernoff bounds for every variable i we have $\mathsf{P}_{x\sim D}[x_i = 0] \leq \epsilon/2n$ if i appears set to 0 less than $\epsilon/4n$ fraction of the time over a sample of size $\Theta(n\log(n)/\delta)$. So, by union bound $\mathsf{P}_{x\sim D}[\exists i \in I, x_i = 0] \leq \epsilon/2$.

It remains to show that hypotheses $h_1, \ldots, h_m$ have error at most $\epsilon/2$ over D_S. First note that for any label r if $x_i \notin c_{\mathrm{r}}$ and $i \in S$, then $\mathsf{P}_{x\sim D_S}[x_i = 0 | c_{\mathrm{r}}(x) = 1] = \mathsf{P}_{x\sim D_S}[x_i = 0]$. This follows from two facts. First, since the target c_{r} is a conjunction we have $\mathsf{P}_{x\sim D_S}[x_i = 0 | c_{\mathrm{r}}(x) = 1] = \mathsf{P}_{x\sim D_S}[x_i = 0 | x_j = 1 \forall x_j \in c_{\mathrm{r}}]$. Second, because D is a product distribution and D_S be the distribution D restricted to examples that are set to 1 on all variables in I, we have $\mathsf{P}_{x\sim D_S}[x_i =$

$0|x_j = 1 \forall x_j \in c_{\mathrm{r}}] = \mathsf{P}_{x \sim D_S}[x_i = 0]$. Furthermore since c_{r} is balanced over D and so over D_S we get $\mathsf{P}_{x \sim D_S}[x_i = 0, c_{\mathrm{r}}(x) = 1] \geq c\epsilon/n$.

Note that every time we learn we learn a problem from scratch (by using the original variables), we get $n/\epsilon \log(n/\delta)$ labeled examples from D_S. Therefore significant variables that are not in the target will appear set to 0 in at least one positive example. Therefore for every problem i learned based on the original features (via case 1 or 3(b)), we learn the target, that is $h_i = c_i$.

These together with the argument in the Theorem 7 gives the desired result. ■

Algorithm 5 Transfer Learning of Conjunctions with Monomial Metafeatures

Input: parameters n,m,k, ϵ, δ; $s_1(n, \epsilon, \delta)$, $s_2(n, \epsilon, \delta)$, $s_3(n, \epsilon, \delta)$, access to unlabeled examples from D_i and label oracles for problems $\mathrm{r} \in \{1, \ldots, m\}$, .

1. Draw $s_1(n, \epsilon, \delta)$ unlabeled examples and identify the set of variables I that are set to 0 less than $\epsilon/4n$ fraction of the times.
2. Draw a set S_1 of $s_1(n, \epsilon, \delta)$ examples from D_1, remove from S_1 those examples for which not all features in I are set to 1. Label S_1 according to problem 1. Find a conjunction h_1 consistent with S_1. Initialize $\mathrm{TS} = \{h_1\}$.
3. Run Algorithm 2 with input TS to produce hypothesized metafeatures $\tilde{M}$.
4. For the learning problem $\mathrm{r} = 2$ to m
   - Draw a set S_{r} of $s_2(n, \epsilon, \delta)$, examples from D_{r}, remove from S_{r} those examples in S_{r} for which not all features in I is set to 1; re-represent each example in S_{r} using meta-features in $\tilde{M}$ and check if we can find a conjunction consistent with S_{r},
     (a) If yes, let h_{r} be its representation over the original features and record it.
     (b) If not, draw a set S_{r} of $s_3(n, \epsilon, \delta)$, examples from D_{r}, remove from S_{r} those examples for which a feature in I is set to 1; find a conjunction mh_{r} consistent with S_{r}.
        - Add h_{r} to TS.
        - Run Algorithm 2 with input TS to produce hypothesized metafeatures $\tilde{M}$.

Output: Conjunctions $h_1, \ldots, h_m$.

Appendix D. Proofs for Sparse Boolean Autoencoding (Theorem 11, Corollary 12)

D.1. Proof of Theorem 11

Proof Item (1) is the easier of the two. For each $y \in \{0,1\}^n$ of Hamming weight at most c, define $\tilde{m}_y$ to be the bitwise-AND of all $T_{\mathrm{r}} \in \mathrm{TS}$ such that $y \preceq T_{\mathrm{r}}$. By definition of the anchor-set assumption, for each $m_j \in M$ there exists $y_j \preceq m_j$ of Hamming weight at most c such that for all r, if $y_j \preceq T_{\mathrm{r}}$ then $m_j \in R_{\mathrm{r}}$. Therefore we have both (a) $m_j \preceq \tilde{m}_{y_j}$ and (b) $\tilde{m}_{y_j} \preceq T_{\mathrm{r}}$ for all r such that $m_j \in R_{\mathrm{r}}$. Therefore each T_{r} is the bitwise-OR of the (at most k) metafeatures $\tilde{m}_{y_j}$ such that $m_j \in R_{\mathrm{r}}$.

For item (2), we begin by creating $O(n^c)$ metafeatures $\tilde{m}_y$ as above. We next set up a linear program to find an optimal fractional subset of these metafeatures, and then round this fractional solution to a set of metafeatures $\tilde{M}$ satisfying (2). Specifically, the LP has one variable Z_y for each $\tilde{m}_y$ with objective

$$\begin{array}{lllll} & & \text{Minimize:} & \sum_y Z_y, & \\ \text{Subject to :} & (1) & \text{for all } y: & 0 \leq Z_y \leq 1 & \\ & (2) & \text{for all r}, i: & \sum_{y: e_i \preceq \tilde{m}_y \preceq T_{\mathrm{r}}} Z_y \geq 1 & (e_i \text{ is the unit vector in coordinate } i) \\ & (3) & \text{for all r:} & \sum_{y: \tilde{m}_y \preceq T_{\mathrm{r}}} Z_y \leq k & \end{array}$$

Here, constraint (2) ensures that each T_{r} is fractionally covered by all the metafeatures contained inside it, and constraint (3) ensures that each T_{r} fractionally contains at most k metafeatures. Note also that setting $Z_{y_j} = 1$ for each $m_j \in M$ (and setting all other $Z_y = 0$) satisfies all constraints at objective value $|M|$.

We now produce our output set of metafeatures $\tilde{M}$ by independently rounding each Z_y to 1 with probability $\min[1, Z_y \ln(n^2|\mathrm{TS}|)]$. Clearly $\mathsf{E}[|\tilde{M}|] = O(|M| \log(n|\mathrm{TS}|))$ so the key issue is the coverage of each T_{r} and the size of the set $\tilde{R}_{\mathrm{r}} = \{\tilde{m}_y \in \tilde{M} : \tilde{m}_y \preceq T_{\mathrm{r}}\}$. Note that item (2) of Definition 10 will be satisfied by how the $\tilde{m}_y$ were constructed (taking the bitwise-AND of all T_{r} such that $y \preceq T_{\mathrm{r}}$). First, for coverage, for each r and i such that variable i is set to 1 by T_{r}, the probability that $\tilde{M}$ does not contain some $\tilde{m}_y$ such that $e_i \preceq \tilde{m}_y \preceq T_{\mathrm{r}}$ is maximized when constraint (2) is satisfied at equality and all associated Z_y are equal (by concavity). This in turn is at most $\lim_{\epsilon \to 0}(1 - \epsilon \ln(n^2|\mathrm{TS}|))^{1/\varepsilon} = 1/(n^2|\mathrm{TS}|)$. Thus, by the union bound, the probability that any T_{r} fails to be completely covered by $\tilde{R}_{\mathrm{r}}$ is at most $1/n$. Now, to address the size of the sets $\tilde{R}_{\mathrm{r}}$, the expected size of each $\tilde{R}_{\mathrm{r}}$ by constraint (3) and the rounding step is at most $k \ln(n^2|\mathrm{TS}|) \leq \max[k, 3] \ln(n^2|\mathrm{TS}|)$. By Chernoff bounds, the probability any given $\tilde{R}_{\mathrm{r}}$ has size more than twice this value is at most $e^{-\max[k,3] \ln(n^2|\mathrm{TS}|)/3} \leq 1/(n^2|\mathrm{TS}|)$. So, by the union bound, the probability that any $\tilde{R}_{\mathrm{r}}$ is too large is at most $1/n^2$. ■

D.2. Proof of Corollary 12

Proof We instantiate $O(n^c)$ metafeatures $\tilde{m}_y$, one for each $y \in \{0,1\}^n$ of Hamming weight at most c, setting each $\tilde{m}_y$ initially to the conjunction of all variables. Given a new target T_{r}, we try to learn it as a conjunction of at most k of these metafeatures using at most $O(k \log n^c)$ equivalence queries using the Winnow algorithm. If we are unsuccessful, we learn T_{r} from scratch using at most n equivalence queries. We then (viewing T_{r} and the $\tilde{m}_y$ as their indicator vectors) let $\tilde{m}_y \leftarrow \tilde{m}_y \,\&\, T_{\mathrm{r}}$ (where "&" denotes bitwise-AND) for all $\tilde{m}_y$ such that $y \preceq T_{\mathrm{r}}$. This maintains the invariant that for each $m_j \in M$, we have $m_j \preceq \tilde{m}_{y_j}$, which implies that each time we learn some T_{r} from scratch we shrink at least one $\tilde{m}_{y_j}$ by at least one variable. This can happen at most $n|M|$ times. ■

Algorithm 6 Multi-task learning for polynomial target functions

Input: n,m.

1. Let $\tilde{M} = \emptyset$. $\tilde{M}$ is the set of hypothesized metafeatures for the first hidden layer.
   Let $TS = \emptyset$. TS is the set of terms used to create the hypothesized metafeatures in $\tilde{M}$.

2. For the learning problem $\mathrm{r} = 1$ to m

   (a) Create the set $\mathrm{prod}(\tilde{M})$ of terms obtained by taking all possible products of subsets of the hypothesized metafeatures in $\tilde{M}$.

   (b) Attempt to learn problem r as a linear function over the terms in $\mathrm{prod}(\tilde{M})$ to low mean squared error (quadratic loss) using $O(B^2k)$ examples.

   - If we succeed, record the hypothesis.
   - Otherwise, run the algorithm of Schapire and Sellie (1993) to learn the target T_{r} for problem r exactly based on the original feature representation with equivalence and membership queries.
     i. Expand TS by adding any term in T_{r} that was not in TS.
     ii. Run Algorithm 2 with input TS to "compactify" it into the fewest number of (possibly overlapping) conjunctive metafeatures that can be used to recreate all the terms in TS. Let $\tilde{M}$ be the resulting metafeatures.

Output: Hypothesis functions of low error for each learning task.

Appendix E. Algorithm and Proofs for Life-Long Learning of Polynomials

Proof of Theorem 13. In Algorithm 6, $\tilde{M}$ represents the set of hypothesized metafeatures for the first hidden layer – they are learned using Algorithm 2; let $k' = |\tilde{M}|$. Let $\mathrm{prod}(\tilde{M})$=all possible products of hypothesized metafeatures in $\tilde{M}$; so $TS \subseteq \mathrm{prod}(\tilde{M})$, $|\mathrm{prod}(\tilde{M})| = 2^{k'}$.

We know that in the true underlying network the metafeatures in the first middle layer are monomials satisfying the anchor assumption and the metafeatures in the second middle layer are monomials of meta-features in the first layer. Note that every time we fail to learn in Step 2(b) we know that at least one of the monomials that can make up the target polynomial (which is a metafeature second level of the true network) cannot be written as a product of hypothesized first level metafeatures $\tilde{M}$. Since we create $\tilde{M}$ by using Algorithm 2, by Theorem 7 we only need to learn at most $n^2 + k$ problems from from scratch (that is $|TS| \leq n^2 + k$), and furthermore, $k' \leq k$. ■