# Efficient Representations for Life-Long Learning and Autoencoding

Maria-Florina Balcan
CMU
ninamf@cs.cmu.edu

Avrim Blum
CMU
avrim@cs.cmu.edu

Santosh Vempala
Georgia Tech
vempala@cc.gatech.edu

December 8, 2014

## 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.

# 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 [13], 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" [23, 25] or "transfer learning" [1, 19] or "learning to learn" [7, 24]) is to use information from previously-learned tasks to *improve the underlying representation* used by 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 [9, 10]. 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 that what their associated target vectors share in common is they lie in a low $k$ dimensional subspace of the ambient space $\Re^n$ (equivalently, there exist a small number of 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 [1, 19]. However, for the online setting, a key challenge is that we won't have perfectly learned the previous target functions when we set out to learn our next one.[2] For this problem we provide a sample-efficient 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 required for learning the tasks separately.

We then consider scenarios where the commonalities require a representation with more than one level of metafeatures and provide efficient algorithms for these settings as well. For linear metafeatures, we provide a natural framework where two levels of metafeatures can be efficiently extracted and provide substantial benefit: specifically, 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 low $k$-dimensional subspace).

In Section 4 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 [2], 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

---

[1]Except in Section 5 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.

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 can be reconstructed by superimposing only a few of them. In the first case our algorithm finds the optimal solution under the anchor-variable assumption (the problem is NP-hard in general) and in the second case it finds a bicriteria approximation (for a given sparsity level, approximates both the number and the sparsity to logarithmic factors) under the weaker anchor-set assumption.

In Section 5 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. Interestingly, as opposed to the algorithms for linear metafeatures, this algorithm periodically re-compactifies its current representation. In particular, whenever a new polynomial cannot be learned using the current representation and must be learned from scratch, we then revisit the previously-learned polynomials and optimally "compactify" them into 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 [7, 6] 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 appear in [8].

Recent work of [19, 16] 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 [20]. By contrast with these results, our setting is more demanding since we aim to achieve small error on all tasks and to do so online without keeping all training data from past learning tasks in memory.

[11] considers 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. Discussion in [26] hints toward the type of the algorithms we analyze in Section 3, but without formal analysis about how the error accumulation could harm the sample complexity (which, as we will see, is one of the central challenges in this setting).

The problem of trying to learn invariants or other commonalities when faced with a series of learning tasks arriving over time has a long history in applied machine learning (e.g., [23, 25]). Our work is the first to give provable efficiency guarantees for learning multi-layer representations in this life-long learning setting.

## 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, where $\text{err}(h_i) = \text{err}_{P_i}(h_i) = P_{(x,y) \sim P_i}[h_i(x) \neq y]$.

# 3 Life-long Learning of Halfspaces

We consider here the natural case that $X = \Re^n$ and each target function is a linear separator going 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 $\text{sign}(a_i \cdot x) = y$. To begin, we will assume that what the target functions have in common is that they 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 allows for the $a_i$ to be "near" to a low-dimensional subspace (see Theorem 1).

In this section we analyze the following online algorithm for this setting; note, the algorithm is very natural but the challenge will be to analyze it and control the propagation of error at reasonable sample sizes. Let $\varepsilon_{acc}$ be a quantity to be determined later. For the first learning problem we just learn it to error $\varepsilon_{acc}$ 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 $\epsilon$) 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 $\varepsilon_{acc}$ 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 $\varepsilon_{acc}$ needs to be for this to succeed.

We show in the following that if the $D_i$ are isotropic log-concave (which includes many distributions such as Gaussian and uniform, see, e.g., [18]), the above procedure will be successful and learn the target functions with much fewer labeled examples 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 $\mathrm{d}_{\tilde{D}}(a, b) = \mathbb{P}_{x \sim \tilde{D}}(\text{sign}(u \cdot x) \neq \text{sign}(v \cdot x))$. 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 $\theta(u, v) \leq \alpha$.

**Lemma 1** *Assume $D$ is an isotropic log-concave 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 $c\theta(v, u) \leq \mathrm{d}_D(u, v) \leq c'\theta(v, u)$.*

**Proof:** The proof of the lower bound appears in [5]. The proof of the upper bound is implicit in the earlier work of [27] – 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 [15] 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 $\alpha$ 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. ∎
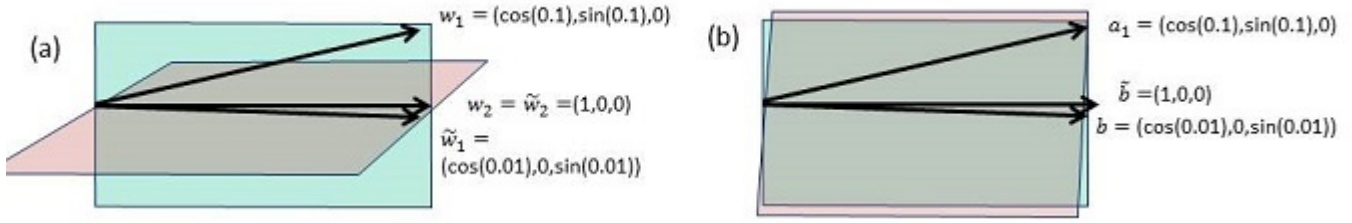
Figure 1: (a) Even though each $\tilde{w}_i$ is within angle 0.11 of its corresponding $w_i$ ($\theta(w_2, \tilde{w}_2) = 0, \theta(w_1, \tilde{w}_1) \le \theta(w_1, w_2) + \theta(w_2, \tilde{w}_1) = 0.11$), the two subspaces are orthogonal (span($w_1, w_2$) is the $x$-$y$ plane and span($\tilde{w}_1, \tilde{w}_2$) is the $x$-$z$ plane). (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 $\theta(\tilde{b}, a_1) = 0.1$.

Lemma 1 implies that if we learn some target $a_i$ to error $\epsilon_{acc}$, 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 $U = \text{span}\{a_1, \ldots, a_{k-1}\}, V = \text{span}\{a_1, a_2, \ldots, a_{k-1}, b\}$ and $\tilde{V} = \text{span}\{a_1, \ldots, a_{k-1}, \tilde{b}\}$ be sets of vectors in $\Re^n$. Then,*

$$\theta(V, \tilde{V}) \le \frac{\pi}{2} \frac{\theta(\tilde{b}, b)}{\theta(\tilde{b}, U)}.$$

**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 $\theta(V, \tilde{V}) = \theta(u, \tilde{V}) \le \theta(\text{span}(u_1, b), \text{span}(u_1, \tilde{b}))$ and the fact that $\theta(\tilde{b}, U) \le \theta(\tilde{b}, u_1)$, it is sufficient to prove that:

$$\theta(\text{span}(u_1, b), \text{span}(u_1, \tilde{b})) \le \frac{\pi}{2} \frac{\theta(b, \tilde{b})}{\theta(\tilde{b}, u_1)},$$

which is just a statement about 3-d space. Let $\alpha = \theta(\text{span}(u_1, b), \text{span}(u_1, \tilde{b}))$ and $\beta = \theta(\tilde{b}, u_1)$. We can wlog write $u_1 = (1, 0, 0)$ and assume $\text{span}(u_1, b)$ is the $x$-$y$ plane. Since $\text{span}(u_1, \tilde{b})$ has angle $\alpha$ 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 $\alpha\beta \le \frac{\pi}{2}\sin^{-1}(\sin(\alpha)\sin(\beta))$ for $0 \le \alpha, \beta \le \frac{\pi}{2}$. ∎

The key point of the next lemma is that the 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 sufficiently far from the previously-learned subspace. In contrast, a difficulty with the usual analysis of perturbation of matrices is that while we can assume that each new $\tilde{a}_i$ is far from the span of the previous $\tilde{a}_1, \ldots, \tilde{a}_{i-1}$, we do

4

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 the *height* of a matrix (e.g., [22]). Note also that even adding the *same* vector to two different subspaces can potentially increase their angle (e.g., in Figure 1(a), $\theta(w_1, \tilde{w}_1) < 0.11$ but $\theta(\text{span}(w_1, w_2), \text{span}(\tilde{w}_1, w_2) = \pi/2$).

**Lemma 3** *Let* $V_k = \text{span}\{a_1, \ldots, a_k\}$ *and* $\tilde{V}_k = \text{span}\{\tilde{a}_1, \ldots, \tilde{a}_k\}$. *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:** The proof is by induction on $k$, on the stronger hypothesis that the conclusion holds for $V_k = \text{span}\{W, a_1, \ldots, a_k\}$ and $\tilde{V}_k = \text{span}\{W, \tilde{a}_1, \ldots, \tilde{a}_k\}$ 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 $V_k' = \text{span}(V_{k-1}, \tilde{a}_k)$. Then we have:

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

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

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 $\gamma$-separated if each $a_{i_j}$ has angle greater than $\gamma$ 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 $\gamma$-separated subsequence. Our assumption will be that the $\gamma$-effective dimension of the targets is at most $k$ for $\gamma = c\epsilon$ for some absolute constant $c > 0$, where $\epsilon$ is our desired error rate per target. Note that for $\gamma = 0$, $\gamma$-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 1** *Assume that all marginals* $D_i$ *are isotropic log-concave. Choose* $\gamma = c_1\epsilon$ *and* $\epsilon_{acc}$ *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* $\epsilon$ *and* $\epsilon_{acc}$ *on any sequence of targets whose* $\gamma$-*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* $\epsilon$ *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 $\epsilon$ by using the previously learnt problems; the rest are of type (b).

---

**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 $\epsilon$ and $\epsilon_{acc}$.

1. Learn the first target to error $\epsilon_{acc}$ to get an $n$-dimensional vector $\alpha_1$. 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 $\epsilon$.
     (a) If yes, set $\tilde{c}_i = (\alpha_{i,1}, \ldots, \alpha_{i,\tilde{k}}, 0, \ldots, 0)$.
     (b) If not, learn a classifier $\alpha_i$ for problem $i$ of accuracy $\epsilon_{acc}$ 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 $v \to \text{sign}(\tilde{A}_i \cdot v)$

---

For problems of type (a) we achieve error $\epsilon$ 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 $\alpha_i$ has error at most $\epsilon_{acc}$, 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 $\gamma$-far from $\text{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$ and (2) $\tilde{w}_{\hat{k}}$ is $\gamma$-far from $\text{span}(\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1})$.

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 $\epsilon$ for problem $i_{\hat{k}}$.[3] That is $a_{i_{\hat{k}}}$ is at least $\epsilon/c'$-far from the $\text{span}\{\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1}\}$ for some absolute constant $c'$ (by Lemma 1). We also have $\theta(\tilde{w}_{\hat{k}}, a_{i_{\hat{k}}}) \leq \varepsilon_{acc}/c$, therefore, by triangle-inequality, we obtain

$$\theta(\tilde{w}_{\hat{k}}, \text{span}(\tilde{w}_1, ... \tilde{w}_{\hat{k}-1})) \geq \epsilon/c' - \varepsilon_{acc}/c \geq \gamma.$$

Thus $\tilde{w}_{\hat{k}}$ is $\gamma$-far from $\text{span}\{\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1}\}$. It remains to show that $a_{i_{\hat{k}}}$ is $\gamma$-far from the span of $\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$. Suppose for contradiction that $\theta(a_{i_{\hat{k}}}, \{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}) \leq \gamma$. We will show that this implies there exists $\tilde{b}_{i_{\hat{k}}} \in \text{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}$ with error at most $\epsilon$, contradicting the fact that no such vector exists.

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 $\gamma$-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

$$\theta(\text{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}, \text{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}) \leq 2k\varepsilon_{acc}/(c\gamma).$$

These together with triangle inequality imply that

$$\theta(a_{i_{\hat{k}}}, \text{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}) \leq \gamma + 2k\frac{\varepsilon_{acc}}{c\gamma} \leq \epsilon/c'.$$

So by Lemma 1 there exist $\tilde{b}_{i_{\hat{k}}} \in \text{span}\{\tilde{w}_1, \cdots, \tilde{w}_{\hat{k}-1}\}$ of error at most $\epsilon$, 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 $\gamma$-effective dimension of the targets). ∎

By setting $\gamma = O(\epsilon)$ and $\epsilon_{acc} = O(\epsilon^2/k)$ the total number of labeled examples needed to learn all the problems to error $\epsilon$ is $\tilde{O}(nk^2/\epsilon^2 + km/\epsilon)$, which could be significantly lower than learning each problem separately. In this case the sample complexity would be $\Omega(mn/\epsilon)$ even under log-concave distributions [5].

---

[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 $c, c'$.

**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 $\epsilon$ under the assumption that the optimal separator has error $\eta = \epsilon/\log^2(1/\epsilon)$ [4], and with near-optimal sample complexity [14, 28]. Thus, by reducing $\varepsilon_{acc}$ by an $O(\log^2(1/\epsilon))$ factor, one can achieve the bounds of Theorem 1 efficiently.*

## 3.1 Halfspaces with more complex common structure

In this section we 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, the numbers of nodes in the network do not tell the whole story: 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 $\tau$-dimensional spaces. This naturally models settings where there are really $r$ different types of learning problems but they share some commonality across type (given by the common $k$-dimensional subspace).[4] 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 layer given by $r$ $\tau$-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 $\tau$-dimensional spaces. In other words, the first hidden layer captures the overall low dimensionality and the second hidden layer captures sparsity. We assume $r \ll m$ and $k \ll n$ and that $\tau$ is a constant.

Algorithmically, given a new problem we first try to learn well via a sparse linear combination of only $\tau$ 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, 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 1 we can show we have $\tilde{k} \le k$ and $\tilde{r} \le \tau r$. Formally:

**Theorem 2** *Assume all marginals $D_i$ are isotropic log-concave and the target functions satisfy the above conditions. Consider $\tilde{\gamma} \le c\epsilon$, $\tilde{\epsilon}_{acc} \le c\frac{\tilde{\gamma}\epsilon}{\tau}$, $\gamma \le c\tilde{\epsilon}_{acc}$, and $\epsilon_{acc} \le c\frac{\gamma\tilde{\epsilon}_{acc}}{k}$ for (sufficiently small) constant $c > 0$. Consider running Algorithm 2 (see appendix) with parameters $\epsilon$, $\epsilon_{acc}$, and $\tilde{\epsilon}_{acc}$. Then $\tilde{k} \le k$ and $\tilde{r} \le \tau r$. Moreover the total number of examples needed to learn all the problems to error $\epsilon$ is $\tilde{O}(nk/\epsilon_{acc} + kr/\tilde{\epsilon}_{acc} + m\log(r)/\epsilon)$.*

(Proof in appendix). 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 $\epsilon$ 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 1 (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 $\epsilon$ is constant,

---

[4]For instance, imagine a job-placement company whose goal is to decide which people would do well in which job. In this setting, we can measure a large number of features for each person (e.g., based on how well they do on various tests). There are then $k$ "intrinsic qualities" that are linear combinations of these features. E.g., "quantitative reasoning" might be one linear combination, "people skills" and "time management" might be others, etc., and really what is important about each person is where they sit in this $k$-dimensional subspace. Then, different jobs might belong in different low-dimensional spaces within this k-dimensional space, based on what is important for that job. I.e., there are $r$ "kinds" of jobs, each of which has a $\tau$-dimensional subspace that is relevant for it.

$k = \sqrt{n}$, $r = n^2$ 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.

# 4 Life-long Learning of Monomials

We now consider a nonlinear case where the metafeatures will be products and combined via products. Specifically, we assume that the instance space is $X = \{0, 1\}^n$, 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 conjunctions (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 can be converted to an integer weight for a dummy variable $x_0$). More importantly, if the metafeatures do not overlap, then the indicator vectors for all the *targets* are in a space of rank $k$ with basis given by the indicator vectors of the metafeatures. If furthermore the underlying distribution is one for which, when learning from scratch, we can learn the target functions exactly (e.g., a product distribution where each variable is set to 0 some non-negligible fraction of the time) then we can directly apply the analysis for linear case. In fact, the overall analysis is much simpler since we have the targets exactly that were learned from scratch.

So, the interesting case is when metafeatures may overlap (it is easy to construct examples where this produces a space of dimension $\Theta(2^k)$). 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 (it is called the "set-basis problem" [12]). For this reason, we will make a natural *anchor-variable* assumption that each metafeature $m_i$ has at least one 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 can think of $y_i$ as an "anchor variable" for metafeature $m_i$.

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 4.4 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 5, we build on some of these results to give an algorithm for life-long learning of polynomials.

## 4.1 Solving the Consistency Problem

We now show that we can use Algorithm 3 (below) for solving the consistency problem under the anchor variable assumption. That is, given a collection of conjunctions, the goal is to find the fewest monomial metafeatures needed to reconstruct all of them as products of subsets of the metafeatures. Given a conjunction $T$ we denote by $\text{vars}(T)$ the variables appearing in $T$. Given a variable $z$ and a set of conjunctions TS we denote by $N(\text{TS}, z)$ the set of conjunctions in TS that contain $z$.

**Lemma 4** *Let* TS *be a set of conjunctions such that each of them is a conjunction of some subset of metafeatures* $m_1, \ldots, m_k$ *satisfying the anchor variable condition. We can use Algorithm 3 to find* $\tilde{m}_1, \ldots, \tilde{m}_i$,

---

**Algorithm 3** Consistency problem for monomial metafeatures with anchor variables

---

**Input**: set $\text{TS} = \{T_1, \ldots, T_r\}$ 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. $N(\text{TS}, z') \subset N(\text{TS}, z)$. If there are multiple options, choose $z_{i+1}$ to be the option of least index.

    (3) Let $\text{vars}(\tilde{m}_{i+1})$ be the intersection of $\text{vars}(T)$ for all $T$ in TS that contain $z_{i+1}$. That is
$$\text{vars}(\tilde{m}_{i+1}) = \bigcap_{T \in \text{TS}, z_{i+1} \in \text{vars}(T)} \text{vars}(T).$$

    (4) 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.

---

$i \leq k$ s.t. each $T_j \in \text{TS}$ is a conjunction of a subset of them. Moreover each $\tilde{m}_i$ is associated to a metafeature $m_{t_i}$ s.t. the following conditions are satisfied:

(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:** Note that for any $i$ for any $T \in \text{TS}$ we have $\text{vars}(h(T)) = \cup\{\text{vars}(\tilde{m}_j) : j \leq i, \text{vars}(\tilde{m}_j) \subseteq \text{vars}(T)\}$; that is, $\text{vars}(h(T))$ represents all variables from $T$ that are already used by the previous hypothesized metafeatures $\tilde{m}_j$ whose relevant variables are contained in $T$.

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 $i' \leq i$. To see this assume by contradiction that $m_{t_{i+1}} = m_{t_{i'}}$ for $i' \leq i$; so $z_{i+1} \in m_{t_{i'}}$. By condition $(a)$ we know $\text{vars}(m_{t_{i'}}) \subseteq \text{vars}(\tilde{m}_{t_{i'}})$ and since $\text{vars}(m_{t_{i'}}) \subseteq \text{vars}(T)$ by condition $(b)$ we have $\text{vars}(\tilde{m}_{t_{i'}}) \subseteq \text{vars}(T)$, so $z_{i+1} \in \text{vars}(h(T))$, contradiction.

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 $N(\text{TS}, y_{t_{i+1}}) \subset N(\text{TS}, z_{i+1})$, and so would have been chosen instead of $z_{i+1}$ in step 3(1). ∎

## 4.2 An Abstract Online Problem

Building on Algorithm 3 and Lemma 4, we now describe an algorithm for the following abstract online setting. 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. To do so we need to argue that each time we pay 1, we can use $T_r$ to make progress.

---

**Algorithm 4** Lifelong Learning of Conjunctions with Monomial Metafeatures

---

**Input**: Targets $T_1, T_2, \ldots, T_m$ provided online.
  1. Initialize TS $= \emptyset$ and $\tilde{M} = \emptyset$.
  2. For r $= 1$ to $m$ do:
      - If we cannot represent $T_r$ as conjunction of hypothesized metafeatures $\tilde{M}$ then
          - Add $T_r$ to TS.
          - Run Algorithm 3 with input TS to produce hypothesized metafeatures $\tilde{M}$.

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

---

**Theorem 3** *The number of targets that need to be learned from scratch in in Algorithm 4 is at most $n^2 + k$.*

**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 TS $\subseteq \tilde{TS}$ we have $E(G_{TS}) \subseteq E(G_{\tilde{TS}})$. We start with the complete directed graph (corresponding to 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 3 . 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 3. 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. ∎

## 4.3 Applications

As one immediate application of the above abstract online problem, 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 1** *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.*

As another application of the above abstract online problem, we now show we can learn with good sample complexity over any product distribution $D$.

**Theorem 4** *Assume that all $D_\mathrm{r} = D$ which is a product distribution, that the metafeatures $m_i$ satisfy the anchor variable assumption and all the target functions $c_\mathrm{r}$ are balanced. We can learn hypotheses $h_1, \ldots, h_m$ of error at most $\epsilon$ by using Algorithm 5 with parameters $s_1(n, \epsilon, \delta) = O(n/\epsilon \log(n/\delta))$, $s_2(n, \epsilon, \delta) = k/\epsilon \log(m/\delta)$, and $s_3(n, \epsilon, \delta) = n/\epsilon \log(nk/\delta)$. The total number of labeled examples needed is $\tilde{O}((n^2 + k)n/\epsilon \log(n/\delta) + km/\epsilon)$.*

**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 $\epsilon$ 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 $c_\mathrm{r}$ 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 $c_\mathrm{r}$ 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 3 gives the desired result. ∎

## 4.4 Sparse Boolean Autoencoders and Relaxing the Anchor-Variable Assumption

The above results (and in particular, Lemma 4) have an interesting interpretation as constructing a minimal feature space for Boolean, or superimposition-based, autoencoding.

Specifically, consider a collection of black-and-while pixel images $\{T_\mathrm{r}\}$ where each $T_\mathrm{r} \in \{0,1\}^n$. Our goal is to contruct a 2-level auto-encoder $\mathcal{A}$ (for each r, we want $\mathcal{A}(T_\mathrm{r}) = T_\mathrm{r}$) 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 $T_\mathrm{r}$ should be equal to the bitwise-OR of all pieces contained within it (i.e., superimposing them together). Formally, for each

**Algorithm 5** Transfer Learning of Conjunctions with Monomial Metafeatures

---

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

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 TS $= \{h_1\}$.

3. Run Algorithm 3 with input TS to produce hypothesized metafeatures $\tilde{M}$.

4. For the learning problem $r = 2$ 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 3 with input TS to produce hypothesized metafeatures $\tilde{M}$.

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

---

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 $m_j \preceq T_r$ if each bit set to 1 in $m_j$ is also set to 1 in $T_r$; we then require $T_r$ to be the bitwise-OR of all $m_j \preceq T_r$. Lemma 4 then shows that given a collection of images $\{T_r\}$, Algorithm 3 finds the smallest number of hidden nodes needed to perform this autoencoding, under the assumption that each metafeature $m_j$ contains some anchor-variable (some pixel set to 1 that no other metafeature sets to 1).

We now consider the problem of *sparse* Boolean autoencoding. That is, given a set TS $= \{T_r\}$, with each $T_r \in \{0,1\}^n$, our goal is to find a collection of metafeatures $\tilde{m}_j$ (perhaps more than $n$ of them) such that each $T_r \in$ 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 $T_r$, 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 $T_r$ that $|\{j : \tilde{m}_j \preceq T_r\}|$ should be small as well.

This problem has two motivations. From the perspective of autoencoding, this corresponds to finding a sparse autoencoder (viewing the $T_r$ as pixel images). From the perspective of life-long learning, if this can be done online then (viewing the $T_r$ as conjunctions) it will allow for fast learning, since conjunctions of $k$ out of $N$ variables can be learned with sample complexity (or equivalence queries) 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 once the number of metafeatures exceeds the number of input features $n$) which is that each metafeature should have a *set* of $\leq c$ variables (for some constant $c$) such that any $T_r$ 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 the $c$-anchor-set assumption 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 1** *A set of metafeatures $M = \{m_j\}$ and set of targets $\mathrm{TS} = \{T_r\}$ satisfy the $c$-anchor-set assumption at sparsity level $k$ if*

1. *for each $T_r \in \mathrm{TS}$ there exists a set $R_r$ of at most $k$ "relevant" metafeatures in $M$ such that $T_r$ is the bitwise-OR of the metafeatures in $R_r$, 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 r, if $y_j \preceq T_r$ then $m_j \in R_r$. Note that in particular this implies that $|\{j : m_j \preceq T_r\}| \leq k$.*

We now prove that under this assumption, we can solve for a near-optimal set of metafeatures $\{\tilde{m}_j\}$.

**Theorem 5** *Given a set of targets $\mathrm{TS} = \{T_r\}$ 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 $\mathrm{poly}(n^c)$ we can:*

1. *Find a set of $O(n^c)$ metafeatures such that each $T_r \in \mathrm{TS}$ 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 $\mathrm{TS}$ at sparsity level $O(k \log(n|\mathrm{TS}|))$.*

**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 $T_r \in \mathrm{TS}$ such that $y \preceq T_r$. 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 $y_j \preceq T_r$ then $m_j \in R_r$. Therefore we have both (a) $m_j \preceq \tilde{m}_{y_j}$ and (b) $\tilde{m}_{y_j} \preceq T_r$ for all r such that $m_j \in R_r$. Therefore each $T_r$ is the bitwise-OR of the (at most $k$) metafeatures $\tilde{m}_{y_j}$ such that $m_j \in R_r$.

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

$$\text{Minimize:} \quad \sum_y Z_y,$$

$$\begin{aligned}
\text{Subject to :} \quad (1) \quad &\text{for all } y: \quad 0 \leq Z_y \leq 1 \\
(2) \quad &\text{for all r}, i: \quad \textstyle\sum_{y : e_i \preceq \tilde{m}_y \preceq T_r} Z_y \geq 1 \quad (e_i \text{ is the unit vector in coordinate } i) \\
(3) \quad &\text{for all r:} \quad \textstyle\sum_{y : \tilde{m}_y \preceq T_r} Z_y \leq k
\end{aligned}$$

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 $\tilde{R}_r = \{\tilde{m}_y \in \tilde{M} : \tilde{m}_y \preceq T_r\}$. Note that item (2) of Definition 1 will be satisfied by how the $\tilde{m}_y$ were constructed (taking the bitwise-AND of all $T_r$ such that $y \preceq T_r$). 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 $e_i \preceq \tilde{m}_y \preceq T_r$ 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 $\tilde{R}_r$ is at most $1/n$. Now, to

address the size of the sets $\tilde{R}_r$, the expected size of each $\tilde{R}_r$ by constraint (3) and the rounding step is at most $k \ln(n^2|\text{TS}|) \leq \max[k, 3] \ln(n^2|\text{TS}|)$. By Chernoff bounds, the probability any given $\tilde{R}_r$ has size more than twice this value is at most $e^{-\max[k,3]\ln(n^2|\text{TS}|)/3} \leq 1/(n^2|\text{TS}|)$. So, by the union bound, the probability that any $\tilde{R}_r$ is too large is at most $1/n^2$. ∎

Theorem 5 shows that we can efficiently find a near-optimal sparse autoencoder for any set of targets in $\{0, 1\}^n$ having an optimal encoder satisfying the $c$-anchor-set assumption for constant $c$. Theorem 5 also has the following corollary for online learning from equivalence queries, similar to Corollary 1.

**Corollary 2** *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 only $O(mk \log(n) + n^2|M|)$ equivalence queries total.*

**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 $\tilde{m}_y \leftarrow \tilde{m}_y \;\&\; T_r$ (where "&" denotes bitwise-AND) for all $\tilde{m}_y$ such that $y \preceq T_r$. 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. ∎

# 5 Life-long Learning of Polynomials

We now show an application of the results in Section 4 to the case where the target functions are polynomials from $\{0, 1\}^n$ to $\mathbb{R}$, whose terms "share" a not too large number of pieces. Specifically, we assume there exist $k$ distinguished monomials (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 distinguished monomials are $\{x_1 x_2 x_3, x_3 x_4 x_5, x_5 x_6 x_7, x_7 x_8 x_1\}$ then we might have polynomials such as $4x_3 x_4 x_5 x_6 x_7 - 2x_5 x_6 x_7 x_8 x_1$ and $3x_1 x_2 x_3 x_4 x_5 + 3x_1 x_2 x_3 x_7 x_8$. 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) in addition to equivalence queries or random examples even in the single task setting [21]. 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 [21] that learns polynomials exactly. Any function from $\{0, 1\}^n$ to $\mathbb{R}$ has a unique representation as a polynomial over $\{0, 1\}^n$, so learning exactly means learning the exact functional form of the target function as a polynomial.

As a warmup, let's first consider a simple case. Assume that the target functions are polynomials that simply use at most $k$ distinct monomials in total. This corresponds to a network with with only one hidden layer of $k$ nodes. In this case, there is a very simple algorithm that exploits the structure of the problem. Let $\tilde{M}$ be the set of hypothesized monomials. Given a new target function, we try to learn a linear function over the monomials in $\tilde{M}$. If we succeed, we are done and move on to the next problem. If not, we learn from scratch using queries; we will clearly get at least one new monomial we have not seen, and add it to the set $\tilde{M}$. So, we only need to learn $k$ problems from scratch.

We now provide an algorithm for the general, more interesting case. Our theoretical guarantees are under the assumptions that each polynomial in our family has $L_1$ norm bounded by $B$ and the number of terms in

14

each is bounded by $t$. If the target function has an $L_1$ norm bounded by $B$ and its monomials 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 [17], we can achieve low mean squared error using only $O(B^2 \log(2^k)) = O(B^2 k)$ examples.

---

**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 r $= 1$ to $m$

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

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

   - If we succeed, record the hypothesis.
   - Otherwise, run the algorithm of Schapire and Sellie [21] 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 3 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.

---

**Theorem 6** *Assume that the monomials corresponding to the first network layer satisfy the anchor assumption and the $L_1$ norm of the target polynomials is bounded by $B$. Consider running Algorithm 6. The number of targets needed to learn from scratch is $n^2 + k$. Furthermore the number of hypothesized metafeatures satisfies $|\tilde{M}| \leq k$ at any time, thus the sample complexity of learning problems in Step 2(b) is $O(B^2 k)$ per problem.*

**Proof:** In Algorithm 6, $\tilde{M}$ represents the set of hypothesized metafeatures for the first hidden layer – they are learned using Algorithm 3; let $k' = |\tilde{M}|$. Let $\mathrm{conj}(\tilde{M})$=all possible conjunctions of hypothesized metafeatures in $\tilde{M}$; so $TS \subseteq \mathrm{conj}(\tilde{M})$, $|\mathrm{conj}(\tilde{M})| = 2^{k'}$.

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 conjunction of hypothesized first level metafeatures $\tilde{M}$. Since we create $\tilde{M}$ by using Algorithm 3, by Theorem 3 we only need to learn at most $n^2 + k$ problems from from scratch (that is $|TS| \leq n^2 + k$), and furthermore, $k' \leq k$.  ∎

Note that while the sample complexity of Algorithm 6 is linear in $k$ for problems learned from scratch, its *running time* is exponential in $k$, due to the work in creating the set $\mathrm{conj}(\tilde{M})$. 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

even if the true metafeatures were given to us in advance, one possibility is that the targets could be arbitrary polynomials over $x_1, \ldots, x_k$). Thus, for this problem one should think of $k$ as small.

# 6  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 the case of product metafeatures, our results have natural interpretations as autoencoders, which interestingly do not require assumptions such as the problem matrix being 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 [3].

# References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

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

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

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

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

[12] 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.

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

[14] S. Hanneke. Personal communication. 2013.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

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

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

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

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

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

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

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

---

**Algorithm 2** 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 $\epsilon$, $\epsilon_{acc}$, $\tilde{\epsilon}_{acc}$.

1. Learn the first target to error $\epsilon_{acc}$ to get an $n$-dimensional vector $\alpha_1$.
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 $\tau$-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 $\tau$-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 $\epsilon$.

     (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 $\alpha_i$ for problem $i$ of accuracy $\epsilon_{acc}$ 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 $v \to \text{sign}(\tilde{A}_i \cdot v)$

---

## A   Proofs for halfspaces with more complex common structure

We now provide the algorithm and proof for Theorem 2.

**Theorem 2** *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 2 with parameters $\epsilon$, $\epsilon_{acc}$, 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 $\epsilon$ 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 $\epsilon$ 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 $\epsilon$ 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 $r$ $\tau$-dimensional subspaces that $a_{j_{\hat{r}}}$ belongs to. (Otherwise if $\tilde{u}_{\hat{r}}$ is $\tilde{\gamma}$-close we would have been able to learn sparsely to error $\epsilon$ 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 $\tau$ 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 $\tau$-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 $\epsilon_{acc} \leq c\frac{\gamma\tilde{\epsilon}_{acc}}{k}$, as in the proof of Theorem 1, 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 $\gamma$-far from $\mathsf{span}\{a_{i_1}, \cdots, a_{i_{\hat{k}-1}}\}$ and $\tilde{w}_{\hat{k}}$ is $\gamma$-far from $\mathsf{span}(\tilde{w}_1, ..., \tilde{w}_{\hat{k}-1})$; this implies $\tilde{k} \leq k$. ∎