Efficient Estimators for Generalized Additive Models

Adam Kalai
TTI-Chicago
http://people.cs.uchicago.edu/~kalai

December 16, 2005

Abstract

Generalized additive models are a powerful generalization of linear and logistic regression models. In this paper we show that a natural *regression graph* learning algorithm efficiently learns generalized additive models. Efficiency is proven in two senses: the estimator's future prediction accuracy approaches optimality at rate inverse polynomial in the size of the training data, and its runtime is polynomial in the size of the training data. Furthermore, the guarantees are nearly linear in terms of the dimensionality (number of regressors) of the problem, and hence the algorithm does not suffer from the "curse of dimensionality." The algorithm is a simple generalization of Mansour and McAllester's classification algorithm that generates decision graphs, i.e., decision trees with merges.

Our analysis is also viewed as defining a natural extension of the original classification *boosting* theorems (Schapire, 1990) to the regression setting. Loosely speaking, we define a weak correlator to be a real-valued predictor that has a correlation coefficient with the target function that is bounded from zero. We show how to efficiently boost weak correlators to get predictions with correlation arbitrarily close to 1 (error arbitrarily close to 0). Our boosting analysis is a natural extension of the classification boosting analysis of Kearns and Mansour (1999) and Mansour and McAllester (2002).

1 Introduction

Generalized Additive Models (see, e.g., Hastie and Tibshirani, 1990), are an expressive generalization of linear and logistic regression models. In a Generalized Additive Model (GAM) there are random variables (X, Y), where $X = (X_1, X_2, \dots, X_d)$ is a d-dimensional real vector, and Y is one-dimensional. Our goal is to predict the function

$f : \mathcal{X} \to \mathbb{R}$, where f is defined by $f(x) \triangleq \mathbf{E}[Y|X = x]$. A GAM stipulates that,

$$f(x) = \mathbf{E}[Y|X = x] = u\left(v_1(x_1) + v_2(x_2) + \ldots + v_d(x_d)\right), \tag{1}$$

where $u : \mathbb{R} \to \mathbb{R}$ is nondecreasing, $v_i : [0, 1] \to \mathbb{R}$, and x_i denotes the ith component of vector x. For simplicity, we make the normalizing assumption that $X \in \mathcal{X} \subseteq [0, 1]^d$ and $Y \in \mathcal{Y} \subseteq [0, 1]$. For the purposes of the introduction, we further assume that u and each v_i have bounded derivatives (in Section 4 we present more general results for Lipschitz-continuous u the v_i's of bounded variation). It is easy to see that this is a generalization of linear and logistic regression, where the v_i's are linear. In terms of classification, where $Y, f(X) \in \{0, 1\}$, this also generalizes a linear threshold function with a *margin* that is inversely proportional to the bound on the derivative.

As an example, Hastie and Tibshirani (1990) consider the problem of predicting whether an individual will develop diabetes, based on several real-valued regressors $X_1, X_2, \ldots, X_d$, such as weight, age, etc. In this case, $\mathcal{Y} = \{0, 1\}$ and $f(X)$ is the probability that an individual with attributes X will develop diabetes. The advantage of restricting $f(X)$ to be of the form (1) is that it allows for avoiding the curse of dimensionality. Without some such assumption, any provably successful learning algorithm would have exponential dependence on the number of regressors d, as, even for $\mathcal{X} = \{0, 1\}^d$, one would have to learn $f(X)$ on 2^d potentially unrelated inputs, requiring prohibitively many training examples.

A learning algorithm receives as input training data $\mathcal{Z} = \left\langle (X^1, Y^1), \ldots, (X^n, Y^n)\right\rangle$ consisting of n independent samples distributed identically to the random variable (X, Y). Our goal is to output a function $h : \mathcal{X} \to [0, 1]$ that will predict $f(X)$ well on future, unseen examples, as measured by $\mathbf{E}[(h(X) - Y)^2]$ or $\mathbf{E}[(h(X) - f(X))^2]$.

Theorem 1. *Let $(X, Y) \in \mathcal{X} \times \mathcal{Y} \subseteq [0, 1]^d \times [0, 1]$ be random variables distributed according to a GAM $f(x) = \mathbf{E}[Y|X = x] = u\left(\sum_{i=1}^d v_i(x_i)\right)$ with nondecreasing differentiable $u : \mathbb{R} \to [0, 1]$ and arbitrary differentiable $v_i : [0, 1] \to \mathbb{R}$, where $|u'(a)| \leq \alpha$ and $|v_i'(b)| \leq \beta$ for all $a \in \mathbb{R}$, $b \in [0, 1]$, and $1 \leq i \leq d$. For any $\delta > 0$, with probability $1 - \delta$ over a training data $\mathcal{Z} = \left\langle (X^1, Y^1), \ldots, (X^n, Y^n)\right\rangle$ of n i.i.d. samples of (X, Y), the regression-graph learning algorithm of Section 3 outputs a regression graph $G : \mathcal{X} \to [0, 1]$ with,*

$$\mathbf{E}[(G(X) - Y)^2] \leq \mathbf{E}[(f(X) - Y)^2] + \frac{3(1 + \alpha\beta d)\ln(5nd/\delta)}{n^{1/7}}.$$

The runtime of the algorithm is $O(dn^{10/7} \log n)$.

For $f = \mathbf{E}[Y|X = x]$, a useful elementary identity is the following, for any function $h : \mathcal{X} \to \mathbb{R}$,

$$\mathbf{E}[(h(X) - Y)^2] = \mathbf{E}[(h(X) - f(X))^2] + \mathbf{E}[(f(X) - Y)^2]. \tag{2}$$

Let us define three types of error of a function $h : \mathcal{X} \to [0, 1]$,

$$\epsilon(h) \triangleq \mathbf{E}[(h(X) - Y)^2] \qquad \text{(generalization error)}$$

$$\epsilon(h, \mathcal{Z}) \triangleq \frac{1}{n}\sum_{i=1}^{n}(h(X^i) - Y^i)^2 \qquad \text{(training error)}$$

$$\xi(h) \triangleq \mathbf{E}[(h(X) - f(X))^2] \qquad \text{(true error)}$$

In these terms, equation (2) states that $\epsilon(h) = \xi(h) + \epsilon(f)$. Since $\xi(h) \geq 0$, $\epsilon(f)$ is the minimum possible generalization error.

Theorem 1 thus states that the future error rate of the algorithms predictions on unseen examples, i.e., the generalization error $\epsilon(G)$, approaches the optimal value $\epsilon(f)$ (equivalently $\xi(G)$ approaches 0) at an inverse polynomial rate. Moreover, the algorithm is computationally efficient. Also essential is the fact that the above guarantee has very good dependence on the dimensionality d, i.e., the number of regressors. Thus, in both the convergence rate and computation time, we have avoided the curse of dimensionality. For the class of GAMs defined above, this algorithm meets the computationally efficient definition of "probabilistic-concept learner" of Kearns and Schapire (1994).

The algorithm we describe generates simple regression graphs with axis-parallel splits (of the form $x_i < \theta$). The algorithm is a generalization of that of Mansour and McAllester (2002) (hereinafter MM) and Kearns and Mansour (1999) (hereinafter KM) to the regression setting. Each split is shown to reduce the error of the regression graph. We show that this can be extended to a general boosting procedure, for combining predictors with slight positively correlations with Y, to achieve a near optimal hypothesis.

1.1 Correlation boosting

More generally, the regression graph learning can be viewed as a *boosting* procedure, which can be used to attempt to improve the performance of any regression algorithm. Boosting is especially natural in the regression framework. For example, various predictors are known correlate with diabetes, and new predictors continue to be discovered. One would like to somehow combine these results to get very accurate predictions. Clearly, this is impossible in general as all the studies may be uncovering the same factors. What we show here is that if these studies are done in a divide (and merge) and conquer fashion, then as long as they achieve positive correlation with the data, the error of the combined estimator will decrease.

In particular, suppose that one has a weak regression algorithm that can do reasonably well at estimating $f(X)$, say it outputs $h : \mathcal{X} \to [0, 1]$ with positive correlation $\mathrm{cor}(h(X), f(X)) > 0$. Then we can find a split of the form $h(X) < \theta$ for some $\theta \in [0, 1]$

and use it in constructing a regression graph. Using recursion, we prove that a suitably good weak correlator can be boosted to achieve a hypothesis of arbitrarily good accuracy.

In binary classification, Kearns and Valiant (1988) suggested the idea that a weak learner, one that guarantees error bounded below 1/2, can be used to build a strong learner, with error arbitrarily close to 0. Schapire (1990) proved that this is possible via an efficient tree-like algorithm, and boosting is now a practical and popular technique. Kearns and Mansour (1999) showed how to view decision tree learners as a boosting procedure and Mansour and McAllester (2002) showed that decision graph learners are efficient boosters. What we do here is the natural analog of these earlier theorems. We define a weak correlator as an algorithm that is guaranteed to output h with $\mathrm{cor}(h(X), f(X)) \geq \rho$ (provided $\mathrm{var}(f(X)) > \sigma^2$) for some $\rho, \sigma > 0$, and we prove a regression boosting theorem in the spirit of Schapire's theorem.

1.2 Related work

KM and MM designed classification boosting algorithms using decision trees (also called classification trees) and decision graphs, respectively. Our algorithm and boosting results are a generalization of theirs. The idea of decision graphs has independently been suggested by practitioners, as well (Kohavi, 1995; Oliver, 1993). Related computationally efficient algorithms for classification of noisy halfspaces (Blum et al., 1997) and other interesting classes of functions have been considered. Perhaps most related, Bylander (1993) considers an arbitrary GLM $u(\sum w_i x_i)$ where u can be arbitrary and monotonic but must be symmetric, $u(0) = 1/2$ and $u(x) + u(-x) = 1$.

The idea of boosting in a regression setting as opposed to a classification setting is very natural, as evidenced by a large body of work on the subject (see, e.g., Buhlmann and Yu, 2003). While much of this work is experimental, some of it is theoretical. However, to the best of our knowledge, none of the prior work has, in the sense that we do, provided theoretical guarantees relating weak to strong learning in the same spirit as Schapire's original boosting theorem.

Much work has been done on estimating GAM's using other approaches. To the best of our knowledge, none of this prior work has been proven to have avoid the curse of dimensionality and have an inverse polynomial convergence rate, like our algorithm. Recently, Horowitz and Mammen (2004) have given an efficient algorithm for learning GAMs in with *known* link function, i.e., if u were known in advance. Their algorithm does not suffer from the curse of dimensionality and has impressive optimal convergence rates. Horowitz and Mammen also give a thorough summary of known results for other procedures, such as backfitting.

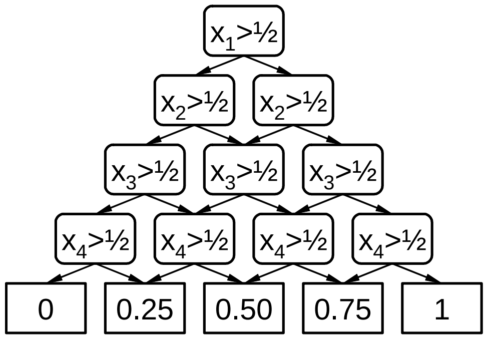

Figure 1: A regression-graph representation of $f(x) = \sum_{i=1}^{d} x_i/d$ over the Boolean hypercube $\mathcal{X} = \{0,1\}^d$, for $d = 4$. The size of the regression graph, for general d, is $O(d^2)$ but any regression *tree* requires a complete tree of size $2^{d+1} - 1$, because every attribute must be tested on every path from root to leaf.

While our algorithm does not have optimal convergence rates, the ability to learn with an unknown link function is, of course, an interesting generalization. We believe that the inverse polynomial guarantees from an computationally efficient algorithm are somewhat surprising. This is especially so in light of the fact that removing any of the assumptions in our main theorem most likely make the problem computationally intractable, as argued in the next section.

2 GAMs and the curse of dimensionality

It is well-known that regression *trees* suffer from the curse of dimensionality. Figure 1 illustrates how a very simple GAM can be implemented by a regression graph (regression graphs are formally defined in the next section). Even for this trivial d-dimensional Generalized Linear Model, a regression graph of size $O(2^d)$ can represent the function, while a regression tree would require size exponential in d. (It would require size $\Omega(2^d)$ even to approximate it to accuracy $1/d$ over the uniform distribution on $\{0,1\}^d$.) To learn such a large regression tree would require a number of training examples and computational complexity exponential in d.

Conversely, in low dimension, one can use more obvious approaches to learn GAMs. For example, for small d, one can discretize the cube $[0,1]^d$ into b blocks in each dimension. For each little cube, one computes the sample mean of those examples that lie in that cube. Due to the bounded derivatives, the function cannot change significantly in any block, for large enough b. Hence, for appropriate choice of b, simply estimating the sample mean of the examples each blocks and outputting the corresponding piecewise constant can be shown to be achieve an error rate that decreases to 0 at a rate that is inversely polynomial in n, but exponential in d.

We now point out some inherent computational limitations of learning GAMs. There are two assumptions in our main theorem that one might hope to remove. Unfortunately, under standard assumptions, removing either of these assumptions would require super-polynomial time in d. The first is the standard assumption that u is monotonically nondecreasing. One would like to learn *non-monotonic GAMs* (removing the monotonic assumption on u) as well. The second is the requirement that the data be truly of the GAM form; it would be more natural to attempt to learn as well as the best GAM without restricting $f(x) = E[Y|X = x]$ to be a GAM. In other words, one would like to achieve error $\epsilon(f^*_{\alpha\beta})$ for any random variables $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, where $f^*_{\alpha\beta}$ is a minimal error GAM with derivatives bounded by α and β. We refer to this as *learning as well as the best GAM*.

A classic learning problem is that of *learning parity with noise*. While the noiseless version of this problem is efficiently solved using Gaussian elimination, the noisy version has been conjectured to be computationally intractable in the number of dimensions d, i.e., requiring time larger than any fixed polynomial in d. Moreover, it has been proposed as the basis of cryptographic systems (Blum et al., 1994; Regev, 2005). The fastest currently known algorithm for this problem has very poor runtime dependence on d, namely $2^{O(d/\log d)}$ (Blum et al., 2003). In Appendix A, we define the hardness of learning parity with noise assumption, and observe that it implies the difficulty of removing either of the two assumptions:

Observation 1. *Efficiently learning GAMs without the monotonicity assumption is at least as difficult as learning parity with noise. Similarly, learning as well as the best GAM is at least as difficult as learning parity with noise.*

3 Regression Graphs: Definitions and Learning

A regression graph is a function $G : \mathcal{X} \to [0, 1]$. With a slight abuse of notation, we let G also denote its representation as an annotated graph. It is a directed acyclic graph with a root node. The size of G, $|G|$, is defined to be the number of nodes in the graph. A *leaf* is a node with no outgoing edges. Each leaf ℓ is annotated with a prediction value $p_\ell \in [0, 1]$. Each internal node v has exactly two outgoing edges, labeled 0 and 1, and is annotated with a binary split predicate $P_v : \mathcal{X} \to \{0, 1\}$. Each $x \in \mathcal{X}$ is mapped to a leaf, by beginning at the root, and at internal node v, following the edge labeled by $P_v(x)$, until a leaf is reached. We define the set $\mathcal{X}_\ell \subseteq \mathcal{X}$ to be the set of x that are mapped to leaf ℓ. Hence G naturally partitions $\mathcal{X}$ into sets $\mathcal{X}_\ell$. Finally, $G(x) = p_\ell$ for

$G \in \mathcal{G}$	A regression graph from the set $\mathcal{G}$ of regression graphs.
$\lvert G \rvert \in \mathbb{N}$	The size (number of nodes) of regression graph $G \in \mathcal{G}$.
$G(x) \in [0,1]$	The prediction of regression graph G on $x \in \mathcal{X}$.
$\mathcal{X}_\ell \subseteq \mathcal{X}$	The subset of $\mathcal{X}$ that is mapped to leaf ℓ.
(X_ℓ, Y_ℓ)	The random variables (X, Y) conditioned on $X \in \mathcal{X}_\ell$.
p_ℓ	The leaf prediction value of leaf ℓ, i.e. $G(x) = p_\ell$ for all $x \in \mathcal{X}_\ell$.
w_ℓ	The weight of leaf ℓ (for sample $\mathcal{Z}$), $w_\ell = \left\lvert \{ i \mid X^i \in \mathcal{X}_\ell \} \right\rvert / n$.

Figure 2: Regression graph notation.

the unique leaf ℓ such that $x \in \mathcal{X}_\ell$.

In the regression graph in Figure 1, the 0-labeled edges are those going left and the 1-labeled edges are those going right. The split predicates are indicator functions $\mathbf{I}[x_i > 1/2]$, which is 1 if $x_i > 1/2$ (equivalently $x_i = 1$), and 0 otherwise.

Define a *data-calibrated* regression graph G to be one where for each leaf ℓ, p_ℓ is the empirical mean of the samples Y^i such that $X^i \in \mathcal{X}_\ell$. The algorithm will generate a data-calibrated graph. Define an *axis-parallel* split to be one of the form $x_i < \theta$ for some $i \in [d]$ and $\theta \in [0,1]$. The algorithm will employ only axis-parallel splits. Moreover, for each split, it will greedily choose an axis-parallel split that minimizes the empirical error of the resulting data-calibrated regression graph. For simplicity, the algorithm's splits will all be of the form $x_i < X_i^j$ for some X^j from the data set. The algorithm is presented in Figure 3.

To merge two leaves, we simply remove one from the graph and move all incoming edges from that leaf to the remaining one, and reset the new leaf's prediction value to maintain the data-calibration property. Hence, the merger of two leaves reduces the size of the graph by 1. Each iteration $t = 1, 2, \ldots,$ increases the size of the graph by at most one. Furthermore, by design it also does not increase the empirical error on the sample. This is because the first step cannot increase the empirical error and the second step increases the empirical error by an amount no larger than (one third of) the decrease achieved by the first step.

We note that the algorithm described in Figure 3 differs from that of MM in a few ways. First, MM uses a slightly different splitting criterion. Second, our algorithm is designed for regression, outputting $G : \mathcal{X} \to [0,1]$. In contrast, MM was concerned with classification and rounded all predictions to output $G : \mathcal{X} \to \{0,1\}$. Third, MM merged based on buckets: $0 = s_0 < s_1 \ldots < s_m = 1$ were selected ahead of time, and all leaves with p_ℓ in the same interval $[s_i, s_{i+1})$ were merged together. Our algorithm performs greedy but adaptive merging like the classification algorithm of Kalai and Servedio (2003). This has the advantage that we do not need to specify the buckets s_i,

```
Input: data $\mathcal{Z} = (X^1, Y^1), \ldots, (X^n, Y^n) \in ([0,1]^d \times [0,1])^n$.
Begin with the trivial one-node graph ($\mathcal{X}_\ell = \mathcal{X}$). For each leaf $\ell$ created by alg.:

  • Compute $w_\ell := \left|\{i \mid X^i \in \mathcal{X}_\ell\}\right| / n$ and $p_\ell := \sum_{i:X^i \in \mathcal{X}_\ell} Y^i / \left|\{i \mid X^i \in \mathcal{X}_\ell\}\right|$.

  • For each $i \in [d], j \in [n]$, compute $\Delta_{\ell,i,j}$, the decrease in $\epsilon(G, \mathcal{Z})$ (equivalently
    in $\sum_\ell w_\ell p_\ell (1 - p_\ell)$) due to the potential split of leaf $\ell$ using predicate $x_i < X_i^j$.

For $t := 1, 2, \ldots, n^{3/7}$ :

  1. Split: Let $\Delta^t := \max_{\ell, i \in [d], j \in [n]} \Delta_{\ell,i,j}$, and perform the corresponding current
     best axis-parallel split of $G$, i.e., one with maximal $\Delta_{\ell,i,j}$.

  2. Merge(s):

     (a) Let $\nabla^t := 0$.

     (b) Sort leaves so that $p_{\ell_1} \leq p_{\ell_2} \leq \ldots$. For each $i$, let $\nabla_i^t$ be the increase in
         $\epsilon(G, \mathcal{Z})$ that *would result* from merging leaves $\ell_i$ and $\ell_{i+1}$.

     (c) If $\nabla^t + \min_i \nabla_i^t \leq \Delta^t/3$ then: for the minimizing $i$, merge $\ell_i$ and $\ell_{i+1}$,
         set $\nabla^t := \nabla^t + \nabla_i^t$, and GOTO (b).

Return the resulting graph $G$ after $t = n^{3/7}$ splits.
```

Figure 3: The regression graph learning algorithm.

whose separation depends on parameters not know in advance. In addition, the merging follows a greedy approach in the same spirit as the splitting. Lastly, MM consider abstract classes of splits and not axis-parallel splits in particular.

Note that the empirical error of a data-calibrated G obeys the following identity,

$$\epsilon(G, \mathcal{Z}) + \frac{1}{n}\sum_{i=1}^{n} Y^i(1-Y^i) = \sum_{\ell} w_\ell p_\ell(1-p_\ell). \tag{3}$$

This can be shown by verifying that, for any leaf ℓ,

$$\frac{1}{n}\sum_{i:X^i\in\mathcal{X}_\ell}\big((Y^i-p_\ell)^2 + Y^i(1-Y^i)\big) = w_\ell p_\ell(1-p_\ell).$$

Equation (3) is simply the sum, over all leaves ℓ, of the above. Hence, a decrease of Δ in $\epsilon(G,\mathcal{Z})$ is equivalent to a decrease of Δ in $\sum_\ell w_\ell p_\ell(1-p_\ell)$. This latter measure, known as the "Gini" splitting criterion, is one of the most common in practice. Thus, the algorithm we propose is very similar to the most common regression tree learning algorithms, except for the merges and hard stopping rule. Stopping is more commonly performed by building a complete tree and then pruning using an independent held-out data set. We chose the fixed-size stopping rule for ease of presentation, though an analysis with pruning could be performed, as well.

Runtime: Regression tree learning algorithms are known for their speed. However, one may be concerned that the regression graph algorithm of Figure 3 is significantly slower due to merging. In this section, we argue that the above algorithm can be implemented in time $O(dn^{10/7}\log n)$. Each iteration of the loop is extremely fast, and can be performed in a single pass through the at most $n^{3/7}$ leaves (as long as the leaves are maintained in sorted order), thus they contribute at most $n^{6/7}$ to the runtime. The expensive step is computing all of the $\Delta_{\ell,i,j}$ for each new leaf ℓ (the computation of p_ℓ and w_ℓ is straightforward in $O(n)$ time). This can be computed in time $O(dn\log n)$. To do this, for each $i\in[d]$, we take the $\leq n$ data that fall into this leaf and sort them by their ith coordinate, X_i^j, in time $O(n\log n)$. We now iterate through the leaf's data in this order, maintaining running totals of $\sum Y^k$ and $\sum (Y^k)^2$. From this, it is straightforward to compute $\Delta_{\ell,i,j}$ for each j. Iterating over $i\in[d]$ can be done in time $O(dn\log n)$. Since there are at most $3n^{3/7}$ new leaves created (each of the $n^{3/7}$ splits creates two new leaves, each merge creates one new leaf, and the number of merges is less than the number of splits). Hence, the total runtime is $O(dn^{10/7}\log n)$, in the RAM model of computing where arithmetic operations can be carried out in unit time.

4 GAM Analysis

In this section, we prove of the following which, with the above discussion of runtime, immediately implies Theorem 1.

Theorem 2. *Let $(X,Y) \in \mathcal{X} \times \mathcal{Y} \subseteq [0,1]^d \times [0,1]$ be random variables distributed according to a GAM $f(x) = \mathbf{E}[Y|X=x] = u\left(\sum_{i=1}^d v_i(x_i)\right)$ with nondecreasing k-Lipschitz $u : \mathbb{R} \to [0,1]$ and $v_i : [0,1] \to \mathbb{R}$ of bounded variation V_{v_i}. For any $\delta > 0$, with probability $1-\delta$ over a training data $\mathcal{Z} = \langle (X^1,Y^1), \ldots, (X^n,Y^n) \rangle$ of n i.i.d. samples of (X,Y), the regression-graph learning algorithm of Section 3 outputs a regression graph $G : \mathcal{X} \to [0,1]$ with,*

$$\epsilon(G) \le \epsilon(f) + \frac{3(1+kV)\ln(5nd/\delta)}{n^{1/7}},$$

where $V = \sum_{i=1}^d V_{v_i}$.

The function $u : \mathbb{R} \to [0,1]$ is k-Lipschitz continuous if $|u(a) - u(b)| \le k\,|a-b|$ for all $a, b \in \mathbb{R}$. A Lipschitz function is one that is 1-Lipschitz continuous. A differentiable function u is α-Lipschitz if $|u'(a)| \le \alpha$ for all $a \in \mathbb{R}$. The *total variation* of $v_i : [0,1] \to \mathbb{R}$, V_{v_i}, is defined to be the following maximum over all increasing sequences of $a_i \in \mathbb{R}$.

$$V_{v_i} = \sup_{m \in \mathbb{Z}} \sup_{0=a_0<a_1<\ldots<a_m=1} \sum_{j=0}^{m-1} |v_i(a_{j+1}) - v_i(a_j)|.$$

For monotonic v_i, $V_{v_i} = |v_i(0) - v_i(1)|$, and for differentiable v_i, $V_{v_i} = \int_0^1 |v_i'(a)|da$. Hence a differentiable function v_i with $|v_i'(a)| \le \beta$ has $V_{v_i} \le \beta$. Thus it is clear that Theorem 2 implies Theorem 1, because $k \le \alpha$ and $V = \sum_{i=1}^d V_{v_i} \le d\beta$.

WLOG we can assume that $k = 1$. This is because we can always replace $u(a)$ with $u(a/k)$, and $v_i(a)$ with $kv_i(a)$, without changing f, yet the new functions u and v_i are 1-Lipschitz and of bounded variation kV_i, respectively. Thus the quantity kV remains fixed but u has become Lipschitz. For the remainder of the analysis, we refer to a GAM with parameter V to be one in which u is Lipschitz and $V = \sum V_{v_i}$. Also, it may appear that the definition of a GAM requires X of full support, otherwise $E[Y|X=x]$ is not well defined. However, we only require that there exists an f of the above form such that f agrees with $E[Y|X=x]$ where defined.

While the regression graph learning algorithm attempts to minimize empirical error $\epsilon(G, \mathcal{Z})$, we would like to prove a bound on the generalization error $\epsilon(G)$. To this end, we show a uniform convergence result, also called a generalization bound: with high probability, every regression graph (of every size) has empirical error near its generalization error. More precisely,

Theorem 3. *For any $n \geq 1$, $\delta > 0$,*

$$\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G} \text{ s.t. } |\epsilon(G, \mathcal{Z}) - \epsilon(G)| > 3.8\sqrt{\frac{|G|\ln(3dn|G|/\delta)}{n}}\right] \leq \delta.$$

We prove the above Theorem in Appendix C. We will use this theorem in combination with showing that the generalization error decreases rapidly. To this end, we first bound how much merging affects the empirical error.

Lemma 1. *Given a regression graph $G \in \mathcal{G}$ with N leaves, there exists a merger of adjacent leaves that increases $\epsilon(G, \mathcal{Z})$ by at most $8/N^3$.*

Proof. Order the leaves $\ell_1, \ell_2, \ldots, \ell_N$ so that $p_{\ell_1} \leq p_{\ell_2} \leq \ldots \leq p_{\ell_N}$. Imagine merging leaves p_{ℓ_i} and p_{ℓ_j} into a single leaf. The weight would be $w_{\ell_i} + w_{\ell_j}$ and prediction value $(w_{\ell_i}p_{\ell_i} + w_{\ell_j}p_{\ell_j})/(w_{\ell_i} + w_{\ell_j})$. Using the fact from (3) that $\epsilon(G, \mathcal{Z}) = c - \sum_\ell w_\ell p_\ell^2$, simple algebra shows that the merger increases $\epsilon(G, \mathcal{Z})$ by,

$$w_{\ell_i}p_{\ell_i}^2 + w_{\ell_j}p_{\ell_j}^2 - (w_{\ell_i} + w_{\ell_j})\left(\frac{w_{\ell_i}p_{\ell_i} + w_{\ell_j}p_{\ell_j}}{w_{\ell_i} + w_{\ell_j}}\right)^2 = \frac{w_{\ell_i}w_{\ell_j}}{w_{\ell_i} + w_{\ell_j}}(p_{\ell_i} - p_{\ell_j})^2 \quad (4)$$

Since $\sum_i w_{\ell_i} = 1$, less than half of the leaves can have $w_{\ell_i} > 2/N$. Similarly, since $\sum_i |p_{\ell_{i+1}} - p_{\ell_i}| \leq 1$, less than half of the leaves have $|p_{\ell_{i+1}} - p_{\ell_i}| > 2/N$. Hence there must be some leaf ℓ_i with both $w_{\ell_i} \leq 2/N$ and $|p_{\ell_{i+1}} - p_{\ell_i}| \leq 2/N$.

Thus the amount of the increase is at most:

$$\frac{w_{\ell_i}w_{\ell_{i+1}}}{w_{\ell_i} + w_{\ell_{i+1}}}(p_{\ell_i} - p_{\ell_{i+1}})^2 \leq \frac{(2/N)w_{\ell_{i+1}}}{w_{\ell_i} + w_{\ell_{i+1}}}\left(\frac{2}{N}\right)^2 \leq \frac{8}{N^3}. \qquad \square$$

4.1 Existence of a correlated split

We denote the covariance of random variables A and B by $\text{cov}(A, B) = \mathbf{E}[AB] - \mathbf{E}[A]\mathbf{E}[B]$, the variance by $\text{var}(A) = \text{cov}(A, A)$, and their correlation coefficient by $\text{cor}(A, B) = \text{cov}(A, B)/\sqrt{\text{var}(A)\text{var}(B)}$. We will use the symmetry and bilinearity of covariance, $\text{cov}(A, B) = \text{cov}(B, A)$ and $\text{cov}(A, B + C) = \text{cov}(A, B) + \text{cov}(A, C)$. We will also use the following identity. Let $(\hat{A}, \hat{B})$ be distributed independently and exactly as (A, B). Then,

$$\mathbf{E}[(A - \hat{A})(B - \hat{B})] = \mathbf{E}[AB] + \mathbf{E}[\hat{A}\hat{B}] - \mathbf{E}[\hat{A}]\mathbf{E}[B] - \mathbf{E}[A]\mathbf{E}[\hat{B}] = 2\text{cov}(A, B) \quad (5)$$

From this, we can see that if A is binary, $A \in \{0, 1\}$, then,

$$\text{cov}(A, B) = \frac{1}{2}\mathbf{E}[(A - \hat{A})(B - \hat{B})] = \mathbf{P}[A = 1]\mathbf{P}[A = 0](\mathbf{E}[B|A = 1] - \mathbf{E}[B|A = 0]) \quad (6)$$

This also implies the usual identity for binary A, $\text{var}(A) = \mathbf{P}[A = 1]\mathbf{P}[A = 0]$.

In this section, we show that, for any GAM, there is an axis-parallel initial split $h : \mathcal{X} \to \{0, 1\}$ that has large covariance $\text{cov}(h(X), Y) = \text{cov}(h(X), f(X))$.

Lemma 2. *Let $u : \mathbb{R} \to \mathbb{R}$ be any monotonically nondecreasing Lipschitz function, and Z be any real-valued random variable. Then, $cov(u(Z), Z) \geq var(u(Z))$.*

Proof. Let $t : \mathbb{R} \to \mathbb{R}$ be defined by $t(a) = a - u(a)$. Note that, since u is Lipschitz, t is nondecreasing as well. By the bilinearity of covariance, and since $\text{var}(u(Z)) = \text{cov}(u(Z), u(Z))$, the statement of the lemma can be rewritten as $\text{cov}(u(Z), t(Z)) \geq 0$. Using (5) with $\hat{Z}$ distributed independently and exactly as Z,

$$2\text{cov}(u(Z), t(Z)) = \mathbf{E}[\big(u(Z) - u(\hat{Z})\big)\big(t(Z) - t(\hat{Z})\big)] \geq 0.$$

This holds because $u(Z) - u(\hat{Z})$ and $t(Z) - t(\hat{Z})$ have the same sign, by monotonicity. $\square$

Finally, we can state and prove a key property of GAMs.

Lemma 3. *Let $f : \mathcal{X} \to \mathbb{R}$ be of the form $f(x) = u\left(\sum_{i=1}^{d} v_i(x_i)\right)$, where $u : \mathbb{R} \to \mathbb{R}$ is nondecreasing and Liptschitz, each $v_i : [0,1] \to \mathbb{R}$ is a function of bounded variation V_{v_i}, and $V = \sum V_{v_i}$. Then, for any random variable $X \in \mathcal{X}$, there exists an axis-parallel split $h : \mathcal{X} \to \{0,1\}$, such that $|cov(h(X), f(X))| \geq var(f(X))/V$.*

Proof. The proof is by the probabilistic method. We argue that a *random* axis-parallel split drawn from a certain distribution has the desired property, in expectation. Hence, there must be some axis-parallel split that has the desired property.

A theorem from real analysis states that every function v of bounded variation V_v can be written as the sum of a monotonically nondecreasing function $s : [0,1] \to \mathbb{R}$ and a monotonically nonincreasing function $t : [0,1] \to \mathbb{R}$ with $V_v = V_s + V_t$ (e.g. Royden, 1988). Hence, we can assume each v_i is either monotonically nonincreasing or monotonically decreasing, by increasing d to $d' = 2d$, without changing V. Since the theorem has no dependence on d, this is without loss of generality.

Also without loss of generality, by monotonicity, we can translate each v_i (and u) so that $v_i : [0,1] \to [0, V_{v_i}]$. Now we argue that a random threshold function of a random attribute will have large covariance. Observe that for any fixed $z \in [0,1]$ and uniformly random $r \in [0,1]$, $E_{r\in[0,1]}[\mathbf{I}\left[z \geq r\right]] = z$. Then, since $v_i(x_i)/V_{v_i} \in [0,1]$,

$$\frac{v_i(x_i)}{V_{v_i}} = \mathbf{E}_{r\in[0,1]}\left[\mathbf{I}\left[\frac{v_i(x_i)}{V_{v_i}} \geq r\right]\right]$$

Choose $i \in [d']$ from the discrete distribution with $P(i) = V_{v_i}/V$. Then,

$$\begin{aligned}\mathbf{E}_{i\leftarrow P, r\in[0,1]}\left[\mathbf{I}\left[\frac{v_i(x_i)}{V_{v_i}} \geq r\right]\right] &= \sum_{i=1}^{d'} \frac{V_{v_i}}{V}\mathbf{E}_{r\in[0,1]}\left[\mathbf{I}\left[\frac{v_i(x_i)}{V_{v_i}} \geq r\right]\right] \\ &= \sum_{i=1}^{d'} \frac{v_i(x_i)}{V} = \frac{1}{V}\sum_{i=1}^{d'} v_i(x_i).\end{aligned}$$

By the bilinearity of covariance, the above, and the fact that covariance is immune to shifts, for any random variable $X \in \mathcal{X}$,

$$\begin{aligned}\mathbf{E}_{i \leftarrow P, r \in [0,1]} \left[\operatorname{cov}(f(X), \mathbf{I}\left[v_i(X_i) \geq rV_i\right])\right] &= \operatorname{cov}(f(X), \mathbf{E}_{i,r}[\mathbf{I}\left[v_i(X_i) \geq rV_i\right]]) \\ &= \operatorname{cov}(f(X), \frac{1}{V}\sum v_i(X_i)) \\ &= \frac{\operatorname{cov}(f(X), \sum v_i(X_i))}{V}\end{aligned}$$

From Lemma 2, the last quantity is at least $\operatorname{var}(f(X))/V$. Since the above holds in expectation, there must be an i and r for which it holds instantaneously. Finally, since WLOG v_i is monotonic, $\mathbf{I}\left[v_i(x_i) \geq rV_{v_i}\right]$ is equivalent to an axis-parallel split $\mathbf{I}\left[x_i \diamond \theta\right]$ for some $\theta \in \mathbb{R}$ and $\diamond \in \{<, >, \leq, \geq\}$. $\square$

The dependence on $\operatorname{var}(f(X))$ in the above lemma is necessary. For example, if $\operatorname{var}(f(X)) = 0$, then $\operatorname{cov}(h(X), f(X))$ must also be 0 for any $h : \mathcal{X} \to \mathbb{R}$.

4.2 Reducing generalization error by axis-parallel splits

Analogous to data-calibrated regression graphs, define a *calibrated* regression graph (with respect to the distribution over (X, Y)) to be one for which $p_\ell = \mathbf{E}[Y|X \in \ell]$ for each leaf ℓ. We denote the set of calibrated regression graphs by $\bar{\mathcal{G}} \subset \mathcal{G}$. For any graph $G \in \mathcal{G}$, we denote by $\bar{G} \in \bar{\mathcal{G}}$ the calibrated version of G, i.e., identical to G except with leaf predictions $p_\ell = \mathbf{E}[Y|X \in \ell]$. We use overbars to indicate the fact that a graph $\bar{G}$ is calibrated (with respect to the underlying distribution over (X, Y)).

Even though we cannot generate such perfectly calibrated graphs, it will be helpful to consider them for the purposes of analysis. In this section, we show:

Lemma 4. *For any GAM with parameter V and any calibrated regression graph $\bar{G} \in \bar{\mathcal{G}}$, there is a graph $\bar{H} \in \bar{\mathcal{G}}$ formed by splitting each leaf of $\bar{G}$ once by an axis-parallel split, with*

$$\epsilon(\bar{G}) - \epsilon(\bar{H}) = \xi(\bar{G}) - \xi(\bar{H}) \geq \left(2\xi(\bar{G})/V\right)^2.$$

Hence by at most tripling the size of a graph G, we can reduce its generalization error by $(2\xi(\bar{G})/V)^2$. For any $\ell \subseteq \mathcal{X}$, let (X_ℓ, Y_ℓ) be random variables distributed like (X, Y) according to the restriction $X \in \ell$. That is,

$$\mathbf{P}[(X_\ell, Y_\ell) \in S] = \frac{\mathbf{P}[(X, Y) \in S \wedge X \in \ell]}{\mathbf{P}[X \in \ell]} \text{ for all } S \subseteq \mathcal{X} \times \mathcal{Y}.$$

(Of course the above should hold for measurable sets S. We remark that, for readability, we have left off measurability requirements in several places. Readers can easily verify that the appropriately qualified theorems and arguments hold for measurable functions

u and v_i. More simply, they hold without further conditions for finite $\mathcal{X} \subset [0,1]$ since each v_i and u is determined based on its values at a finite number of points.)

For any calibrated regression graph $\bar{G} \in \bar{\mathcal{G}}$, we can write the quality of a leaf split in terms of covariance and X_ℓ,

Lemma 5. *Let $h : \mathcal{X} \to \{0,1\}$ be a binary function and $\bar{G} \in \bar{\mathcal{G}}$ be a calibrated regression graph. The split of leaf ℓ into (calibrated) leaves by the predicate $h : \mathcal{X} \to \{0,1\}$ reduces $\epsilon(\bar{G})$ (equivalently $\xi(\bar{G})$) by $P[x \in \mathcal{X}_\ell](cov(f(X_\ell), h(X_\ell)))^2/var(h(X_\ell))$.*

Proof. In the proof of Lemma 1, eq. (4), we argue that, for data-calibrated graphs, merging two leaves ℓ_i and ℓ_j increases $\epsilon(G, \mathcal{Z})$ by,

$$\frac{w_{\ell_i} w_{\ell_j}}{w_{\ell_i} + w_{\ell_j}} (p_{\ell_i} - p_{\ell_j})^2.$$

By the same argument, merging two leaves in a *calibrated* graph increases $\epsilon(G)$ by,

$$\frac{\mathbf{P}[X \in \mathcal{X}_{\ell_i}]\mathbf{P}[X \in \mathcal{X}_{\ell_j}]}{\mathbf{P}[X \in \mathcal{X}_{\ell_i}] + \mathbf{P}[X \in \mathcal{X}_{\ell_j}]} \left(\mathbf{E}[Y_{\ell_i}] - \mathbf{E}[Y_{\ell_j}]\right)^2. \tag{7}$$

One can also see this by imagining the data set size going to infinity, so that $\epsilon(G, \mathcal{Z}) \to \epsilon(G)$, $w_\ell \to \mathbf{P}[X \in \mathcal{X}_\ell]$, and $p_\ell \to \mathbf{E}[Y_\ell]$. Similarly, splitting a leaf ℓ into two leaves ℓ_i and ℓ_j *decreases* $\epsilon(G)$ by the same amount above, because a split is exactly the opposite of a merge.

Suppose that h splits the leaf ℓ into ℓ_0 where $h(X) = 0$ and ℓ_1 where $h(X) = 1$. Then, by (6), since h is binary,

$$\begin{aligned} \text{cov}(h(X_\ell), f(X_\ell)) &= \mathbf{P}[h(X_\ell) = 1]\mathbf{P}[h(X_\ell) = 0](\mathbf{E}[f(X_{\ell_1})] - \mathbf{E}[f(X_{\ell_0})]) \\ &= \frac{\mathbf{P}[X \in \mathcal{X}_{\ell_0}]\mathbf{P}[X \in \mathcal{X}_{\ell_1}]}{\mathbf{P}[X \in \mathcal{X}_\ell]^2}(\mathbf{E}[Y_{\ell_1}] - \mathbf{E}[Y_{\ell_0}]). \end{aligned}$$

Also, $\text{var}(h(X_\ell)) = \mathbf{P}[X \in \mathcal{X}_{\ell_0}]\mathbf{P}[X \in \mathcal{X}_{\ell_1}]/\mathbf{P}[X \in \mathcal{X}_\ell]^2$. Combining these with (7) and $\mathbf{P}[X \in X_\ell] = \mathbf{P}[X \in \mathcal{X}_{\ell_0}] + \mathbf{P}[X \in \mathcal{X}_{\ell_1}]$ gives the lemma. ☐

We are now ready to prove Lemma 4.

Proof of Lemma 4. By Lemma 5, the reduction in $\epsilon(\bar{G})$ due to a split $h_\ell : \mathcal{X} \to \{0,1\}$ on leaf ℓ is at least $4P[X \in \mathcal{X}_\ell](\text{cov}(f(X_\ell), h_\ell(X_\ell)))^2$ since the variance of a $\{0,1\}$ random variable is at most $1/4$. By Lemma 3, for each leaf ℓ, there is an axis-parallel split $h_\ell : \mathcal{X} \to \{0,1\}$ with $\text{cov}(f(X_\ell), h_\ell(X_\ell)) \geq \text{var}(f(X_\ell))/V$. This follows from the fact that the random variables (X_ℓ, Y_ℓ) are also distributed according to a GAM with parameter at most V. Hence, there is a way to split each leaf so that the total reduction in $\epsilon(\bar{G})$ is at least,

$$\begin{aligned}\xi(\bar{G}) - \xi(\bar{H}) &\geq \sum_\ell 4w_\ell(\mathrm{cov}(f(X_\ell), h_\ell(X_\ell)))^2 \\ &\geq \frac{4}{V^2}\sum_\ell w_\ell(\mathrm{var}(f(X_\ell)))^2 \\ &\geq \frac{4}{V^2}\left(\sum_\ell w_\ell \mathrm{var}(f(X_\ell))\right)^2 \quad \text{(by convexity of } x^2\text{)}\end{aligned}$$

Finally, since $\bar{G}$ is calibrated, we have,

$$\xi(\bar{G}) = \sum_\ell w_\ell \mathbf{E}\left[(f(X) - p_\ell)^2 | X \in \ell\right] = \sum_\ell w_\ell \mathrm{var}(f(X_\ell)). \qquad \square$$

4.3 Proof of Theorem 2

Let $q(\mathcal{Z})$ be,

$$q(\mathcal{Z}) = \max_{G \in \mathcal{G}, |G| \leq 3n^{3/7}} |\epsilon(G) - \epsilon(G, \mathcal{Z})| \tag{8}$$

We can now state the progress rate in terms of $q(\mathcal{Z})$.

Lemma 6. *Let G^t be the regression graph after t steps. Then, for $t \leq n^{3/7}$,*

$$\epsilon(G^t, \mathcal{Z}) \leq \epsilon(f) + q(\mathcal{Z}) + (V/2)\sqrt{(2/t)^{2/3} + 2q(\mathcal{Z})}. \tag{9}$$

Proof. Suppose not. Since $\epsilon(G^i, \mathcal{Z})$ is nonincreasing in i, this means that for all $1 \leq i \leq t$,

$$\epsilon(G^i, \mathcal{Z}) \geq \epsilon(f) + q(\mathcal{Z}) + (V/2)\sqrt{b + 2q(\mathcal{Z})}, \tag{10}$$

where $b = (2/t)^{2/3}$. We first argue that this means that $\Delta^i \geq b/|G^i|$, for each i. To see this, let $\bar{G}^i$ be the calibrated version of regression graph G^i. Since $\epsilon(\bar{G}^i) \geq \epsilon(\bar{G}^i, \mathcal{Z}) - q(\mathcal{Z}) \geq \epsilon(G^i, \mathcal{Z}) - q(\mathcal{Z})$, we have,

$$\xi(\bar{G}^i) = \epsilon(\bar{G}^i) - \epsilon(f) \geq \epsilon(G^i, \mathcal{Z}) - q(\mathcal{Z}) - \epsilon(f) \geq (V/2)\sqrt{b + 2q(\mathcal{Z})}.$$

Let $\bar{H}^i$ be the graph formed by splitting each leaf of $\bar{G}^i$ by an axis-parallel split as guaranteed in Lemma 4, so

$$\epsilon(\bar{H}^i) \leq \epsilon(\bar{G}^i) - (2\xi(\bar{G}^i)/V)^2 \leq \epsilon(\bar{G}^i) - b - 2q(\mathcal{Z}) \leq \epsilon(G^i) - b - 2q(\mathcal{Z}).$$

Since $|\bar{H}^i| \leq 3|G^i| \leq 3n^{3/7}$, we have that both $|\epsilon(\bar{H}^i) - \epsilon(\bar{H}^i, \mathcal{Z})|, |\epsilon(G^i) - \epsilon(G^i, \mathcal{Z})| \leq q(\mathcal{Z})$. Together with the above displayed equation, this implies that $\epsilon(\bar{H}^i, \mathcal{Z}) \leq \epsilon(G^i, \mathcal{Z}) - b$. Thus, the total reduction due to at most $|G^i|$ splits is b, there must be one axis-parallel split that gives a reduction of at least $b/|G^i|$. Moreover, the algorithm chooses the axis-parallel split to maximize Δ^i, so $\Delta^i \geq b/|G^i|$.

On the other hand, we now argue that $|G^i| \leq 5b^{-1/2}$, for each $1 \leq i \leq t$. If not, then for some i, $|G^i| \leq 5b^{-1/2} < |G^{i+1}|$, we did a split on graph G^i to get G^{i+1} with $|G^{i+1}| = |G^i| + 1$. In other words, we did a split but *no* merges. Now, by Lemma 1, there would have been some merger of adjacent nodes in G^{i+1} that increases empirical error by at most $8/|G^{i+1}|^3 < 8/(5b^{-1/2})^3 = 8b^{3/2}/125 < b^{3/2}/15$. However, since this increase is less than $\Delta^i/3 \geq b/(3|G^i|) \geq b/(3 \cdot 5b^{-1/2}) = b^{3/2}/15$, we would have done a merger on step i, and we have a contradiction. Thus $|G^i| \leq 5b^{-1/2}$ for all $1 \leq i \leq t$.

Since $\Delta^i \geq b/|G^i| \geq b^{3/2}/5$, for all i, we have that the decrease $\epsilon(G^i, \mathcal{Z}) - \epsilon(G^{i+1}, \mathcal{Z}) = \Delta^i - \nabla^i \geq (2/3)b^{3/2}/5$. By our choice of $b = (2/t)^{2/3}$, this reduction in empirical error is at least $(2/3)(2/t)/5 \geq 1/(4t)$. Hence, if equation (10) held for each $1 \leq i \leq t$, then we would have a total reduction in empirical error of $1/4$. However, $\epsilon(G^1, \mathcal{Z}) \leq 1/4$, so we cannot have this large a reduction in empirical error for t steps. Hence, equation (10) must fail to hold for some $i \leq t$ and must also fail for $i = t$. □

Theorem 2 now follows simply from the previous Lemma and the bounds on $q(\mathcal{Z})$.

Proof of Theorem 2. The algorithm generates a graph G of size at most $|G| \leq n^{3/7}$. By Theorem 3 and the fact that $q(\mathcal{Z})$ is a maximum over graphs of size $3n^{3/7}$, we have that with probability $1 - \delta$,

$$q(\mathcal{Z}) \leq 3.8\sqrt{\frac{3n^{3/7}\ln(9dn^{10/7}/\delta)}{n}} = 3.8\sqrt{\frac{3}{n^{4/7}}\frac{10}{7}\ln(n(9d/\delta)^{7/10})} \leq 8n^{-2/7}\sqrt{\ln(5nd/\delta)}.$$

By the previous Lemma, after $n^{3/7}$ steps,

$$\begin{aligned}
\epsilon(G) &\leq \epsilon(f) + q(\mathcal{Z}) + (V/2)\sqrt{(2n^{-3/7})^{2/3} + 2q(\mathcal{Z})} \\
&\leq \epsilon(f) + q(\mathcal{Z}) + V\sqrt{n^{-2/7} + q(\mathcal{Z})} \\
&\leq \epsilon(f) + (1+V)\sqrt{n^{-2/7} + q(\mathcal{Z})} \qquad (\text{since } q(\mathcal{Z}) \leq 1) \\
&\leq \epsilon(f) + (1+V)\sqrt{9n^{-2/7}\sqrt{\ln(5nd/\delta)}}.
\end{aligned}$$

This completes the proof of the theorem. □

5 Correlation boosting

In this section we generalize the previous algorithm and analysis, and adopt the conventions commonly used in PAC learning, which nicely capture the polynomial dependence on accuracy, sample size, and runtime, as well as the curse of dimensionality. Keeping $\mathcal{Y} \subseteq [0,1]$, we consider a general set $\mathcal{X}$ and a general family $\mathcal{F}$ of functions $f : \mathcal{X} \to [0,1]$. In this section, we suppose that the learning algorithm is given an *example oracle* $\mathcal{Z}$

which supplies an unlimited number of examples distributed i.i.d. like (X, Y). This can be thought of as a button: when the algorithm wants an example (X^i, Y^i), it pushes the $\mathcal{Z}$ button and, in unit time, receives examples distributed with $f(x) = \mathbf{E}[Y|X = x] \in \mathcal{F}$. The limitation on quantity is solely based on its runtime — the number of examples it uses is of course no larger than its runtime. Following the definitions of Kearns and Schapire (1994), we use the following definition for computationally efficient learnability for a regression problem (analogous to PAC learning for classification).

Definition 1. *Algorithm A* efficiently learns *family $\mathcal{F}$ of functions $f : \mathcal{X} \to [0, 1]$ if, given inputs $\epsilon, \delta > 0$, and an example oracle, where $f(x) = E[Y|X = x] \in \mathcal{F}$, with probability $1 - \delta$ it outputs $h : \mathcal{X} \to [0, 1]$ such that $\epsilon(h) \leq \epsilon$ and uses runtime (and hence number of examples) polynomial in $1/\epsilon$ and $1/\delta$.*

Put another way, the error rate $\epsilon(h)$ of the algorithm's output is inverse polynomial in its runtime and hence in the number of examples it uses, i.e., given n examples, it is at most $c_1 n^{-c_2}$ for $c_1, c_2 > 0$. However, it will be more convenient here to operate in the PAC-style of bounding the number of examples and runtime required to achieve error ϵ with probability $1 - \delta$.

For example, we have shown that the class of d-dimensional GAMs of parameter V is efficiently learnable. To capture the curse of dimensionality, a sequence of sets $X_d \subseteq [0, 1]^d$ and families of functions $\mathcal{F}_d$ for $d = 1, 2, \ldots$, algorithms work for every d, and efficiency requires a polynomial dependence on d as well. For notational simplicity, we have not included this aspect, though it could easily be added to the model, and we have shown that learning GAMs is efficient in this sense as well.

In analogy with a γ-*weak learner* in classification (that achieves accuracy $\Pr[h(X) = Y] \geq 1/2 + \gamma$), we define a ρ-*weak correlator*. In order to guarantee any positive correlation, we need some dependence on $\mathrm{var}(f(X))$. If $\mathrm{var}(f(X)) = 0$, no correlation is possible, and in general, more data may be required and smaller correlation may be guaranteed when $\mathrm{var}(f(X))$ is small.

Definition 2. *For $\rho : [0, 1] \to [0, 1]$ a ρ-*weak correlator *for a family of functions $\mathcal{F}$ is an algorithm that takes as input a parameter ϵ and an example oracle and outputs $h : \mathcal{X} \to \mathbb{R}$. For any $\epsilon > 0$ and distribution over (X, Y) such that $f(x) = E[Y|X = x] \in \mathcal{F}$ and var$(f(X)) \geq \epsilon$, with probability at least 3/4 over examples from the sample oracle, it outputs a hypothesis $h : \mathcal{X} \to \mathbb{R}$ such that cor$(h(X), f(X)) \geq \rho(\epsilon)$.*

A weak correlator is *efficient* if the runtime and $1/\rho(\epsilon)$ are smaller than some polynomial in $1/\epsilon$. For the class of functions $\mathcal{F}$ that are GAMs with parameter V, Lemma 3 implies that there is an axis-parallel split that has $\mathrm{cor}(h(X), f(X)) \geq \mathrm{cov}(h(X), f(X)) \geq \sigma^2/V$. Hence, the algorithm that finds the most correlated split is, ignoring sampling error, a $\rho(\epsilon) = \epsilon/V$ weak-correlator. One may be tempted to more simply define a V-weak-correlator as one for which $\text{cor}(h(X), f(X)) \geq \text{var}(f(X))/V$. However, as the example in Section 5.1 illustrates, there are natural examples where the guaranteed correlation is only polynomial in $\text{var}(f(X))$.

The main theorem of this section is the following.

Theorem 4. *There is a boosting algorithm and a polynomial $q(\cdot)$ such that, given as input a ρ-weak correlator for $\mathcal{F}$, $\epsilon, \delta > 0$, and examples distributed with $f(x) = \mathbf{E}[Y|X = x] \in \mathcal{F}$, with probability $1-\delta$, outputs h such that $\epsilon(h) \leq \epsilon(f) + \epsilon$. The algorithm makes less than $q(1/(\delta\epsilon\rho(\epsilon)))$ calls to the weak correlator (with parameter $\epsilon/2$), and uses at most an additional $q(1/\delta\epsilon\rho(\epsilon)))$ runtime and examples.*

In order to use the regression graph algorithm, we need binary split predicates, but the weak correlator outputs $h : \mathcal{X} \to \mathbb{R}$, a real-valued function. The following lemma is helpful for this:

Lemma 7. *Let $S \in [0, 1]$ be a random variable and $T \in \mathbb{R}$ be a positively correlated random variable. Then there exists some threshold $\theta \in \mathbb{R}$ such that the indicator random variable $\mathbf{I}[T \geq \theta]$ has correlation near $cor(S, T)$,*

$$cor(S, \mathbf{I}[T \geq \theta]) \geq \frac{1}{3}\left(\log \frac{3}{cor(S,T)\,var(S)}\right)^{-1/2} cor(S, T).$$

Most importantly, the only dependence on T is through $\text{cor}(S, T)$ and not $\text{var}(T)$ or bounds on T. We prove Lemma 7 in Appendix B.

For simplicity, KM, MM, and Kalai and Servedio (2003) model their problems assuming that they can compute $\mathbf{E}[Y_\ell]$ and $\mathbf{P}[X \in \mathcal{X}_\ell]$ exactly. In other words, they have an idealized model with an infinite data sets on which they can compute these quantities exactly. This simplifying assumption makes their analysis more understandable, at the cost of less concrete bounds and algorithms. As they correctly argue, this assumption does not qualitatively change their results to within a polynomial factor, because each of these quantities can be estimated to sufficient additive accuracy τ, with probability $1-\delta$, with a number of additional examples that is polynomial in $1/\tau$ and $\log 1/\delta$. In the GAM analysis, for completeness, we have gone through the painstaking process of proving a generalization bound and analyzing the realistic algorithm that cannot compute these quantities exactly. In this section, we begin by following the precedent of KM and the others by analyzing a hypothetical algorithm for which $p_\ell = \mathbf{E}[Y_\ell]$ and $w_\ell = \mathbf{P}[X \in \mathcal{X}_\ell]$ and we have an *oracle* that can compute these quantities for any potential leaf in unit time. We also assume that we have an *idealized* ρ-weak correlator, which is an idealized algorithm that takes random variables (X, Y) as input, and outputs an $h : \mathcal{X} \to \mathbb{R}$ such that $\text{cor}(h(X), f(X)) \geq \rho(\text{var}(f(X)))$. Note that the idealized weak correlator takes the

```
Input: $T > 0$, an *idealized* weak correlator for $\mathcal{C}$, and an oracle that computes
$w_\ell = \mathbf{P}[X \in \mathcal{X}_\ell]$ and $p_\ell = \mathbf{E}[Y_\ell]$ for any potential leaf $\ell$.
Begin with the trivial one-node graph ($\mathcal{X}_\ell = \mathcal{X}$). For each leaf $\ell$ created by alg.:

  • Compute $w_\ell := \mathbf{P}[X \in \mathcal{X}_\ell]$ and $p_\ell := \mathbf{E}[Y_\ell]$ using the oracle.

  • Run the idealized weak correlator on $(X_\ell, Y_\ell)$ to get $h_\ell : \mathcal{X} \to \mathbb{R}$. Choose $\theta_\ell$
    so as to maximize the decrease in $\epsilon(G)$ (equivalently in $\sum_\ell p_\ell(1-p_\ell)$) due to
    the best split of leaf $\ell$ using predicate $h_\ell(x) < \theta_\ell$, and let $\Delta_\ell$ be this decrease.

For $t := 1, 2, \ldots, T$ :

  1. Split: Let $\Delta^t := \max_\ell \Delta_\ell$, and split the corresponding maximizing $\ell$.

  2. Merge(s):

     (a) Let $\nabla^t := 0$.

     (b) Sort leaves so that $p_{\ell_1} \le p_{\ell_2} \le \ldots$. For each $i$, let $\nabla_i^t$ be the increase in
         $\epsilon(G, \mathcal{Z})$ that *would result* from merging leaves $\ell_i$ and $\ell_{i+1}$.

     (c) If $\nabla^t + \min_i \nabla_i^t \le \Delta^t/3$ then: for the minimizing $i$, merge $\ell_i$ and $\ell_{i+1}$,
         set $\nabla^t := \nabla^t + \nabla_i^t$, and GOTO (b).
```

Figure 4: The idealized correlation boosting algorithm.

distribution itself as input, and it succeeds with probability 1 rather than 3/4. We later justify why these idealizations suffice to obtain Theorem 4.

Lemma 8. *Given an idealized ρ-weak correlator and an oracle for exactly computing $p_\ell = \mathbf{E}[Y_\ell]$ and $w_\ell = \mathbf{P}[X \in \mathcal{X}_\ell]$, if $f(x) = \mathbf{E}[Y|X = x] \in \mathcal{C}$ then the idealized boosting algorithm of Figure 4 outputs a graph with error at most $\epsilon(f) + \epsilon$ after*

$$t \ge \left(\frac{6\sqrt{\ln(6/\rho(\epsilon/2)\epsilon)}}{\rho(\epsilon/2)\sqrt{\epsilon}} \right)^3$$

iterations. The idealized weak correlator is called with parameter $\epsilon/2$.

Proof. The proof is straightforward given the existing lemmas. Say at step t, $\xi(G) > \epsilon$ and there are L_t leaves. As shown in Lemma 4, $\xi(G) = \sum_\ell w_\ell \text{var}(f(X_\ell))$. Then the sum of $w_\ell \text{var}(f(X_\ell))$ over leaves with $\text{var}(f(X_\ell)) \le \epsilon/2$ is at most $\epsilon/2$. Hence, the sum over leaves with $\text{var}(f(X_\ell)) > \epsilon/2$ is at least $\epsilon/2$, and there must be some leaf ℓ with $\text{var}(f(X_\ell)) > \epsilon/2$ and $w_\ell \text{var}(f(X_\ell)) \ge \epsilon/(2L_t)$. Consider this leaf ℓ. By definition, the (ρ, σ)-weak correlator must have output a $h_\ell : \mathcal{X} \to \mathbb{R}$ such that $\text{cor}(h_\ell(X_\ell), f(X_\ell)) \ge$

$\rho(\epsilon/2)$. For succinctness, write $\rho = \rho(\epsilon/2)$. By Lemma 7, the binary split $P_\ell(x) = \mathbf{I}[h_\ell(x) \le \theta_\ell]$ achieves $\mathrm{cor}(P_\ell(X_\ell), f(X_\ell)) \ge \rho/(3\sqrt{\ln(6/\rho\epsilon)})$. By Lemma 5, this split reduces $\epsilon(G)$ by

$$\Delta^t \ge w_\ell \frac{\mathrm{cov}(f(X_\ell), P_\ell(X_\ell))^2}{\mathrm{var}(P_\ell(X_\ell))} = w_\ell var f(X_\ell) \mathrm{cor}(f(X_\ell), P_\ell(X_\ell))^2 \ge \frac{\epsilon\rho^2}{18 L_t \ln(6/\rho\epsilon)}.$$

By Lemma 1, there is a merge that increases $\epsilon(G)$ by at most $8/L_t^3$. Hence if,

$$\frac{1}{3}\frac{\epsilon\rho^2}{18 L_t \ln(6/\rho\epsilon)} \ge \frac{8}{L_t^3} \iff L_t \ge \frac{12}{\rho\sqrt{\epsilon}}\sqrt{3\ln(6/\rho\epsilon)},$$

then we would certainly merge. Thus, as long as $\xi(G) > \epsilon$, there will never be more than $L_t \le 12\rho^{-1}\epsilon^{-1/2}\sqrt{3\ln(6/\rho\epsilon)}$ leaves. Since the net reduction in each step is at least $(2/3)\Delta^t \ge \epsilon\rho^2/(27 L_t \ln(6/\rho\sigma))$, and $\epsilon(G) \in [0, 1/4]$, then after

$$\frac{1}{4}\frac{27\ln(6/\rho\sigma)}{\epsilon\rho^2}\frac{12}{\rho\sqrt{\epsilon}}\sqrt{3\ln(6/\rho\epsilon)} \le \left(\frac{6\sqrt{\ln(6/\rho\sigma)}}{\rho\sqrt{\epsilon}}\right)^3$$

steps, $\epsilon(G) \le \epsilon$. □

Finally, it suffices to run the above the algorithm given only estimates of w_ℓ and p_ℓ that are accurate to an additive, say, $\tau = \epsilon^4\rho^4/10$, to be safe. This will guarantee that we will get the same reduction in $\epsilon(G)$ per step, up to a constant factor. The number of such estimates that are required is at most $O(T)$. Estimating w_ℓ for all of these is easy, and can be done to accuracy τ using $O(\log(T/\delta)/\tau^2)$ samples, with error probability, say, $\delta/10$. To be safe, one could even use fresh data to estimate each of the quantities. Estimating p_ℓ is more difficult, especially when w_ℓ is small. However, if the weak learner outputs a split with w_ℓ very small for one of the leaves, say $\le 2\tau$, we do not need to estimate p_ℓ since we know this split cannot reduce $\epsilon(G)$ significantly. Furthermore, for w_ℓ larger than τ, we can estimate each p_ℓ using $O(\log(T/\delta)/\tau^4)$ fresh examples, with error probability $\delta/10$. Furthermore, we can use a non-idealized weak correlator that has failure probability $1/4$. We can run each one $O(\log(T/\delta))$ times and take the best split as measured by $O(\log(T/\delta)/\tau^4)$ examples, so that with probability at least $1 - \delta/10$, we find a sufficiently good split for each leaf. Finally, there is no guarantee about what happens when we run the weak-correlator with parameter $\epsilon/2$ but on a leaf with $\mathrm{var}(f(X_\ell)) < \epsilon/2$. Within the definition, the algorithm might not even terminate. Hence, we can run each weak correlator for a specified number of steps (or all in parallel) and abort any that are taking too long. This analysis can, at length, be formalized to prove Theorem 4, and the parameters and algorithm could of course be improved. However, the above analysis suffices to quickly see that the polynomial guarantees apply.

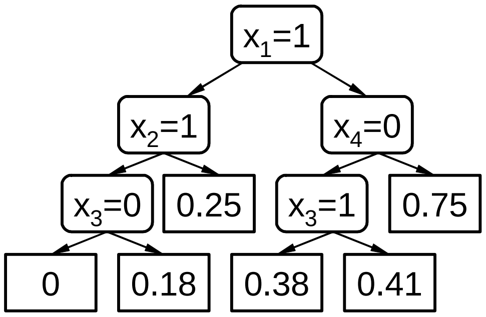

Figure 5: An orderly regression tree. The arrows to the right correspond to 1 labeled edges, which are followed when the predicate holds, and the arrows to the left are followed when the predicate fails. The leaves are increasing in value from left to right. Also notice that the function represented is not monotonic in x_3.

5.1 Orderly regression trees

In this section, we illustrate the potential applicability of the boosting theorem by demonstrating that another interesting class of functions is efficiently learnable. Let $\mathcal{X} = \{0,1\}^d$, $\mathcal{Y} = \{0,1\}$, and take a regression tree with axis parallel splits, i.e., of the $\mathbf{I}[x_i = 0]$ or $\mathbf{I}[x_i = 1]$. We define an orderly regression tree (ORT) to be a regression tree whose leaves can be arranged in nondecreasing order of p_ℓ. An ORT is one where, for every node v, and every pair of leaves ℓ_0 in the $P_v(x) = 0$ subtree and ℓ_1 in the $P_v(x) = 1$ subtree, $p_{\ell_0} \le p_{\ell_1}$. An example ORT is shown in Figure 5. Despite considerable effort, there are no known learning algorithms for decision trees that do not suffer from the curse of dimensionality. Since regression trees are a generalization of decision trees, they are also unlikely to be efficiently learnable. On the other hand, ORTs are learnable, as we argue. Moreover they generalize decision lists and the probabilistic decision lists of Kearns and Schapire (1994).

Observe that orderly regression trees can be non-monotonic in various attributes. That is, changing x_i from 0 to 1 may increase $f(x)$, for some x (with $x_i = 0$), and decrease $f(x)$ for other x (with $x_i = 0$). This property is not observed in any type of GAMs nor in most common learnable families of functions (that are efficiently learnable in d over $\mathcal{X} = \{0,1\}^d$), with the notable exception of the (noiseless) parity functions. Another interesting property is that, with d attributes and of size $O(d)$, there may be no attribute that has correlation with $f(X)$ larger than $O(2^{-d})$, even when $f(X)$ has considerable variance. Thus the regression graph algorithm of Figure 3 is not an efficient learner of this class of functions.

We now argue that ORTs are efficiently learnable by giving an idealized weak cor-

```
Input: variance guarantee $\epsilon$.

  • Let $\mathcal{I} := \emptyset$.

  • While there exist $(j,b) \in [d] \times \{0,1\}$ such that $\mathbf{P}[X_j = b | X \notin \mathcal{I}] \in \left(0, \frac{\epsilon}{8d}\right]$, let
    $\mathcal{I} := \mathcal{I} \cup \{x \in \mathcal{X} | x_j = b\}$.

  • For $a, b \in \{0,1\}, i \in [d]$, let $h_{a,b,i}(x) = \begin{cases} a & \text{if } x \in \mathcal{I} \\ b & \text{if } x \notin \mathcal{I} \text{ and } x_i = 1 \\ 1-b & \text{if } x \notin \mathcal{I} \text{ and } x_i = 0 \end{cases}$

  • Return $h_{a,b,i}(x)$ where $a$, $b$, and $i$ are chosen to maximize $\text{cov}(h_{a,b,i}(X), Y)$.
```

Figure 6: An idealized weak correlator for ORTs. The set $\mathcal{I}$ can be represented by a list of pairs (j, b).

relator for them, which can easily (but tediously) be converted to a weak correlator.

Lemma 9. *The algorithm in Figure 6 is an idealized $\epsilon^2/(5d)$-weak correlator for the class of functions representable by ORTs.*

Proof. Suppose $\text{var}(f(X)) \geq \epsilon$. We will show that $\text{cov}(h_{a,b,i}(X), f(X)) \geq \epsilon^2/(20d)$ for the function $h_{a,b,i}$ returned by the algorithm, which suffices since $\text{var}(h_{a,b,i})$ and $\text{var}(f)$ are both at most 1/4. By (5), for $\hat{X}$ distributed independently but identically to X,

$$\text{var}(f(X)) = \frac{1}{2}\mathbf{E}\left[(f(X) - f(\hat{X}))^2\right] \leq \frac{1}{2}\mathbf{E}[|f(X) - f(\hat{X})|] = \mathbf{E}[\max\{f(X) - f(\hat{X}), 0\}].$$

Consider the ORT representing f. After conditioning on $X \notin \mathcal{I}$, there may be several predicates in the tree that have become *obsolete*, where a predicate is obsolete if all data that reaches that node (conditioned on $X \notin \mathcal{I}$) has one predicate value. However, we can descend down the tree through all obsolete predicates (taking the probability 1 path) until we reach node with a non-obsolete predicate. Consider this node and let $\mathcal{L}, \mathcal{R} \subset \mathcal{X}$ represent the data that falls into the left and right subtrees, respectively.

$$\begin{aligned}
\text{var}(f(X)) &\leq \mathbf{E}[\max\{f(X) - f(\hat{X}), 0\}] \\
&\leq \mathbf{P}[X, \hat{X} \notin \mathcal{I}]\mathbf{E}[\max\{f(X) - f(\hat{X}), 0\} \mid X, \hat{X} \notin \mathcal{I}] + 1 - \mathbf{P}[X, \hat{X} \notin \mathcal{I}] \\
&\leq \mathbf{E}[\max\{f(X) - f(\hat{X}), 0\} \mid X, \hat{X} \notin \mathcal{I}] + 2\mathbf{P}[X \in \mathcal{I}] \\
&\leq \mathbf{E}[\max\{f(X) - f(\hat{X}), 0\} \mid X \in \mathcal{R}\backslash\mathcal{I}, \hat{X} \in \mathcal{L}\backslash\mathcal{I}] + 2\mathbf{P}[X \in \mathcal{I}] \qquad (11) \\
&= \mathbf{E}[f(X) - f(\hat{X}) \mid X \in \mathcal{R}\backslash\mathcal{I}, \hat{X} \in \mathcal{L}\backslash\mathcal{I}] + 2\mathbf{P}[X \in \mathcal{I}] \\
&= \mathbf{E}[f(X) | X \in \mathcal{R}\backslash\mathcal{I}] - \mathbf{E}[f(X) | X \in \mathcal{L}\backslash\mathcal{I}] + 2\mathbf{P}[X \in \mathcal{I}]. \qquad (12)
\end{aligned}$$

In the above, (11) follows from the previous line because conditioning on the expectation $\mathbf{E}[\max\{f(X)-f(\hat{X}),0\} \mid X,\hat{X} \notin \mathcal{I}]$ can be written as a convex combination of four expectations, depending on whether either of X and $\hat{X}$ are in $\mathcal{L}\backslash\mathcal{I}$ or $\mathcal{R}\backslash\mathcal{I}$ and the one in (11) is certainly the largest of the four, by the properties of an ORT and the fact that $\mathcal{L}$ and $\mathcal{R}$ represent the first non-obsolete split. Without loss of generality, say $\mathbf{P}[X \in \mathcal{R} | X \notin \mathcal{I}] \geq 1/2$. Then consider the function $h(x) = \mathbf{I}\left[x \in \mathcal{R} \cup \mathcal{I}\right]$. By (6) and basic properties of conditional expectation,

$$\begin{aligned}
\operatorname{cov}(h(X), f(X)) &= \operatorname{var}(h(X))(\mathbf{E}[f(X) \mid h(X)=1] - \mathbf{E}[f(X) \mid h(X)=0]) \\
&\geq \operatorname{var}(h(X))(\mathbf{E}[f(X) \mid X \in \mathcal{R}\cup\mathcal{I}] - \mathbf{E}[f(X) \mid X \in \mathcal{L}\backslash\mathcal{I}]) \\
&\geq \operatorname{var}(h(X))(\mathbf{E}[f(X) | X \in \mathcal{R}\backslash\mathcal{I}] - \mathbf{E}[f(X) | X \in \mathcal{L}\backslash\mathcal{I}] - 2\mathbf{P}[X \in \mathcal{I}]) \\
&\geq \operatorname{var}(h(X))(\operatorname{var}(f(X)) - 4\mathbf{P}[X \in \mathcal{I}]). \qquad (13)
\end{aligned}$$

The last line follows from (12). Because each of the at most d additions to $\mathcal{I}$ remove at most a $1-\epsilon/8d$ fraction of the remaining set $\mathcal{X}\backslash\mathcal{I}$, $\mathbf{P}[X \notin \mathcal{I}] \geq (1-\epsilon/8d)^d \geq 1-\epsilon/8$, and $\mathbf{P}[X \in \mathcal{I}] \leq \epsilon/8$. Also, we have that $\epsilon \leq 1/4$ and,

$$\operatorname{var}(h(X)) = \mathbf{P}[h(X)=0]\mathbf{P}[h(X)=1] \geq \frac{\epsilon}{8d}\mathbf{P}[X \notin \mathcal{I}](1-\frac{\epsilon}{8d}\mathbf{P}[X \notin \mathcal{I}]) \geq \frac{\epsilon}{8d}\frac{31}{32}\frac{31}{32} \geq \frac{\epsilon}{9d}.$$

Combining these with (13) gives the $\operatorname{cov}(h(X), f(X), \geq)\epsilon^2/18d$. Since the idealized algorithm returns the best $h_{a,b,i}$, it will return an $h_{a,b,i}$ with covariance at least this large. □

6 Conclusions

We have generalized the natural boosting model and decision graph learning algorithms of KM and MM to the regression setting. We have demonstrated their power by showing that they efficiently learns GAMs. We have also extended weak-learning/strong-learning boosting theorems of classification to the regression setting. In this case, it not as clear what the definition of weak learning should be. We propose a bi-criteria definition based on correlation and variance, and illustrate its utility through two classes of functions. It is well-known that, disregarding computation time, classification boosting over $\mathcal{X} = \{0,1\}^n$ using weak learners $X_i = 1$ or $X_i = 0$ is capable of learning exactly the set of linear threshold functions (halfspaces). However, the extension of this to the real-valued setting reveals that a much richer class of functions is learnable. In particular, it includes families of functions that are not monotonic in the base attributes X_i, such as ORTs.

The algorithm we analyze is discrete and tree-like (and most likely suboptimal), like the original boosting algorithm of Schapire (1990). One would hope that further work will lead to improved boosting algorithms and analyses that are most likely less discrete and more similar to AdaBoost.

References

Baum, E. (1991). A polynomial time algorithm that learns two hidden unit nets. *Neural Computation* 2, 510–522.

Blum, A., Frieze, A., Kannan, R., and Vempala, S. (1997). A polynomial time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22, 35–52.

Blum, A., Furst, M., Kearns, M., and Lipton, R.J. (1994). Cryptographic primitives based on hard learing problems. *Lecture Notes in Computer Science: Advances in cryptology—CRYPTO '93*, 773, 278–291.

Blum, A., Kalai, A., and Wasserman, H. (2003). Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50, 506–519.

Breiman, L., Friedman, J, Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*, Wadsworth International Group.

Buhlmann, P., and Yu, B. (2003). Boosting with the L2 Loss: Regression and Classification. *J. Amer. Statist. Assoc.* 98, 324–340.

Bylander, T. (1993) Polynomial learnability of linear threshold approximations. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning*, 297–302.

Hastie, T.J., and Tibshirani, R.J. (1990). *Generalized Additive Models,* London: Chapman and Hall.

Horowitz, J., and Mammen, E. (2004). Nonparametric Estimation of a Generalized Additive Model with an Unknown Link Function. *Annals of Statistics*, 32, 24122443.

Kalai, A. (2004). Learning Monotonic Linear Functions. In *Lecture Notes in Computer Science: Proceedings of the 17th Annual Conference on Learning Theory*, 3120, 487–501.

Kalai, A., and Servedio, R. (2003). Boosting in the presence of Noise. In *Proceedings of the thirty-fifth ACM symposium on theory of computing,* 195–205. To appear in *Journal of Computer and System Sciences.*

Kearns, M., and Mansour, Y. (1999). On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58, 109–128.

Kearns, M., and Schapire, R. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and Systems Sciences*, 48, 464–497.

Kearns, M., and Valiant, L. (1988). Learning boolean formulae or finite automata is as hard as factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory.

Kohavi, R. (1995). Wrappers for Performance Enhancement and Oblivious Decision Graphs. Ph.D. dissertation, Comput. Sci. Depart., Stanford Univ., Stanford, CA, 1995.

Mansour, Y., and McAllester, D. (2002). Boosting using branching programs. *Journal of Computer and System Sciences*, 64, 103–112.

Oliver, J. (1993). Decision graphs – an extension of decision trees. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, 334–350.

Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, 84–93.

Royden, H. (1988). *Real Analysis*, 3rd edition, New York: Macmillan.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, 5, 197–227.

A Learning parity with noise

In the noiseless parity model, $\mathcal{X} = \{0,1\}^d$, $\mathcal{Y} = \{0,1\}$ and there is an unknown subset $S \subseteq [d]$ ($[d] = \{1,2,\ldots,d\}$) that we must identify from examples (X^i, Y^i) drawn independently from a distribution that is uniform over $\mathcal{X}$ and $Y = \sum_{i \in S} X_i \pmod 2$. Identifying S in the can be done easily and efficiently from $O(d)$ noiseless examples by Gaussian elimination.

In the η-noisy parity model, $Y \in \{0,1\}$ is corrupted (flipping 0 to 1 and 1 to 0) with probability η, independently for all X, so $f(x) = \eta + (1-\eta)\left(\sum_{i \in S} x_i \pmod 2\right)$. Even for small constant η, the fastest currently known algorithm for this problem has very poor runtime dependence on d, namely $2^{O(d/\log d)}$ (Blum et al., 2003). This problem over the uniform distribution on $\mathcal{X}$ has been conjectured to be computationally intractable in d, for all constant $\eta \in (0, 1/2)$, and has been proposed as the basis of cryptographic systems (Blum et al., 1994; Regev, 2005). An efficient algorithm for learning parity with η noise would be one that, for all d and $S \subseteq [d]$, outputs S with probability greater than 3/4, given poly(d) independent examples from an η-noisy parity model and runtime that is poly(d) as well. The *hardness of learning parity with noise assumption* is that, for every constant $\eta \in (0, 1/2)$, there is no efficient algorithm for learning parity with η noise.

A *non-monotonic* GAM is a model in which $f(x) = u\left(\sum v_i(x_i)\right)$ for u which is not necessarily monotonic. The noisy parity model can be represented as a non-monotonic GAM, where $v_i(x_i) = x_i$ for $i \in S$ and 0 for $i \notin S$, and $u(m) = \eta$ for even integers m and $1 - \eta$ for odd integers m.

Observation 2. *There is no algorithm for estimating non-monotonic GAMs that runs in time polynomial in d, under the hardness of learning parity with noise assumption.*

The parity with η noise model can be also stated as a GAM with $\eta \in (0, 1/2)$, $\mathcal{X} = \{0,1\}^d$, $\mathcal{Y} = \{0,1\}$, and $f(x) = u\left(\sum v_i(x_i)\right)$, where (non-monotonic) $u : \mathbb{Z} \to [0,1]$, and $v_i : \{0,1\} \to \{0,1\}$ are defined as follows, for some hidden set $S \subseteq [d]$,

$$u(m) = \eta + (1-\eta)(m \bmod 2), \quad v_i(b) = \begin{cases} b & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases} \tag{14}$$

(While u is only defined on the integers, it can be extended to take real inputs.) The distribution over X is assumed to be uniform.

To argue this, it suffices to argue that an efficient algorithm for learning non-monotonic GAMs implies an efficient algorithm for learning parity with noise. Then this means that one could predict output $h : \mathcal{X} \to [0,1]$ with error $\xi(h) = \mathbf{E}[(h(X) - f(X))^2] \le (1/2 - \eta)^2/2$.

More precisely, to output the hidden set S, it suffices to be able to distinguish the case of Y being a parity with noise from the case of $f(X) = 1/2$ for all X, truly random noise. One could identify the set S by, for each i, removing the attribute X_i and feeding data of the form (X_{-i}, Y) (where $X_{-i} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_d)$) into the algorithm. Note that, if $i \in S$, then Y is completely independent of X_{-i}, while if $i \notin S$, then it will remain a parity with noise model. Moreover, it is clear that any good estimator for non-monotonic GAMs would be able to distinguish these two cases.

As mentioned, the second and more bothersome assumption is that $f(X) = E[Y|X]$ is a GAM. Ideally, it would be nice to predict *as well as the best GAM*. Namely, one would like to allow $f(x)$ to be arbitrary, and simply attempt to output an $h : \mathcal{X} \to [0,1]$ that achieves error approaching $\epsilon(f^*_{\alpha\beta})$, where $f^*_{\alpha\beta} : \mathcal{X} \to [0,1]$ is the GAM (with monotonic u) of lowest error and derivatives bounded by α and β. One would like a convergence rate that is inverse polynomial in n and d. It is not difficult to see, as we explain in Appendix A, that this too would imply an algorithm for learning parity with noise:

Observation 3. *There is no algorithm for predicting as well as the best GAM that runs in time polynomial in d, under the hardness of learning parity with noise assumption.*

This follows for similar reasons. Although the parity with noise problem is not a monotonic GAM, there is a monotonic GAM that has sufficiently less than 1/4, i.e., better than the trivial estimate of $g(x) = 1/2$. For example, take the function $h(x) = u(\sum f_i(x_i))$ for u such that $u(m) = 0$ for $m \leq 2\lfloor |S|/4 \rfloor$, and 1 for $m \geq 2\lfloor |S|/4 \rfloor + 1$, and v_i as in (14). It can be shown to have error noticeably less than 1/4. Hence, if one can efficiently predict as well as the best GAM, then one could distinguish between a parity with noise model and random noise and thus efficiently solve the parity with noise learning problem.

B Proof of Lemma 7

In this section, we use the notation $\sigma_A = \sqrt{\mathrm{var}(A)}$, $\sigma_{AB} = \mathrm{cov}(A, B)$, and $\rho_{AB} = \mathrm{cor}(A, B)$ for random variables $A, B \in \mathbb{R}$. Let $T_\theta = \mathbf{I}[T \geq \theta]$ for any $\theta \in \mathbb{R}$. Scaling and shifting T does not affect its correlation with S, so WLOG suppose $\mathbf{E}[T] = 1/2$. We also scale T so that $\sigma_T^2 = \sigma_{ST}/2$, WLOG. We do this so that an "important" part of the distribution of T is in the range $[-1, 1]$. The proof, like that of Lemma 3, follows the probabilistic method. Imagine choosing a random threshold θ uniformly from $[-1, 1]$. Then, $2\mathbf{E}_\theta[T_\theta] = \int_{-1}^{1} \mathbf{I}[T \geq \theta]\, d\theta = 1 + T_1$ where,

$$T_1 = \begin{cases} 1 & \text{if } T \geq 1 \\ T & \text{if } T \in (-1, 1) \\ -1 & \text{if } T \leq -1 \end{cases}$$

Now, by linearity of expectation and covariance, $2\mathbf{E}_\theta[\sigma_{ST_\theta}] = \sigma_{ST_1}$. Notice that for *any* random variable $A \in \mathbb{R}$, since $S \in [0, 1]$, $|S - \mu_S| \leq 1$ for $\mu_S = \mathbf{E}[S]$ and hence $\sigma_{SA} = \mathbf{E}[(S - \mu_S)A] \leq \mathbf{E}[|A|]$. We can now lower bound the expected covariance,

$$2\mathbf{E}_\theta[\sigma_{ST_\theta}] = \sigma_{ST_1} = \sigma_{ST} - \sigma_{S(T-T_1)} \geq \sigma_{ST} - \mathbf{E}\left[|T - T_1|\right].$$

It is not difficult to see that $|T - T_1| \leq T^2$, because $|T - T_1| \leq |T|$ and $1 \leq |T|$ whenever $|T - T_1| \neq 0$. Now the point of the assumption that $\sigma_T^2 = \mathbf{E}[T^2] = \sigma_{ST}/2$ is clear,

$$2\mathbf{E}_\theta[\sigma_{ST_\theta}] \geq \sigma_{ST} - \mathbf{E}\left[T^2\right] \geq \frac{\sigma_{ST}}{2}. \tag{15}$$

To lower-bound ρ_{ST_θ}, we now upper-bound σ_{T_θ}, in expectation. Note that since T_θ is binary, $\sigma_{T_\theta} = \sqrt{\mathbf{P}[T < \theta]\mathbf{P}[T \geq \theta]}$ and $\sigma_{T_\theta} \leq 1/2$. Hence,

$$2\mathbf{E}_\theta[\sigma_{T_\theta}] = \int_{-1}^{1} \sigma_{T_\theta} d\theta \leq \int_{-1}^{-\sigma_T} \sqrt{\mathbf{P}[T < \theta]} d\theta + \int_{-\sigma_T}^{\sigma_T} \frac{1}{2} d\theta + \int_{\sigma_T}^{1} \sqrt{\mathbf{P}[T \geq \theta]} d\theta.$$

Using the fact that $\sqrt{a} + \sqrt{b} \leq \sqrt{2(a+b)}$ for $a, b \geq 0$,

$$2\mathbf{E}_\theta[\sigma_{T_\theta}] \leq \sigma_T + \int_{\sigma_T}^{1} \sqrt{2(\mathbf{P}[T < -\theta] + \mathbf{P}[T \geq \theta])} d\theta \leq \sigma_T + \int_{\sigma_T}^{1} \sqrt{2\mathbf{P}[|T| \geq \theta]} d\theta.$$

By the Cauchy-Schwartz inequality,

$$\int_{\sigma_T}^1 \sqrt{\mathbf{P}[|T| \geq \theta]} d\theta = \int_{\sigma_T}^1 \sqrt{\mathbf{P}[|T| \geq \theta]}\sqrt{\theta} \cdot \frac{1}{\sqrt{\theta}} dt \leq \sqrt{\int_{\sigma_T}^1 \mathbf{P}[|T| \geq \theta]\theta d\theta \int_{\sigma_T}^1 \frac{1}{\theta} d\theta}.$$

Also, we have the identity,

$$\mathbf{E}[T^2] = \int_0^\infty \mathbf{P}[T^2 \geq t]dt = \int_0^\infty \mathbf{P}[T^2 \geq \theta^2]2\theta d\theta = 2\int_0^\infty \mathbf{P}[|T| \geq \theta]\theta d\theta.$$

In the above we have made the change of variable $\theta^2 = t$. Putting the last three equations together gives,

$$2\mathbf{E}_\theta[\sigma_{T_\theta}] \leq \sigma_T + \sqrt{2\int_{\sigma_T}^1 \mathbf{P}[|T| \geq \theta]\theta d\theta \int_{\sigma_T}^1 \frac{1}{\theta} d\theta} \leq \sigma_T + \sigma_T\sqrt{\log(1/\sigma_T)}. \tag{16}$$

Putting (15) and (16) together gives,

$$\mathbf{E}_\theta[\sigma_{ST_\theta}] \geq \frac{\sigma_{ST}}{2\sigma_T(1+\sqrt{\log(1/\sigma_T)})}\mathbf{E}_\theta[\sigma_{T_\theta}].$$

Hence there must be some θ for which the above holds instantaneously,

$$\begin{aligned} \sigma_{ST_\theta} &\geq \frac{\sigma_{ST}}{2\sigma_T(1+\sqrt{\log(1/\sigma_T)})}\sigma_{T_\theta} \\ \rho_{ST_\theta} = \frac{\sigma_{ST_\theta}}{\sigma_{T_\theta}\sigma_S} &\geq \frac{\sigma_{ST}}{2\sigma_T(1+\sqrt{\log(1/\sigma_T)})\sigma_S} = \frac{\rho_{ST}}{2(1+\sqrt{\log(1/\sigma_T)})} \end{aligned}$$

By our choice of $\sigma_T^2 = \sigma_{ST}/2$, we have $\sigma_T = \rho_{ST}\sigma_S/2$. Again using $\sqrt{a}+\sqrt{b} \leq \sqrt{2(a+b)}$, we have,

$$1+\sqrt{\log(1/\sigma_T)} \leq \sqrt{2+2\log(2/\rho_{ST}\sigma_S)} = \sqrt{2\log(2e/\rho_{ST}\sigma_S)} \leq \sqrt{2\log(e/\rho_{ST}\sigma_S^2)}.$$

Using the fact that $e \leq 3$ and $2\sqrt{2} \leq 3$, we are done. □

C Generalization bounds for regression graphs

It suffices to consider regression graphs with split predicates of the form $x_i < \theta$ or $x_i \leq \theta$ (predicates of the form $x_i > \theta$ or $x_i \geq \theta$ can easily be swapped to this form). Next, notice that to prove Theorem 3, it suffices to consider regression graphs with split predicates of the form $x_i < \theta$ only. For, whenever there is some regression graph with the violating condition $|\epsilon(G, \mathcal{Z}) - \epsilon(G)| > \tau$ for some $\tau \geq 0$, we can modify each split of the form $x_i \leq \theta$ to $x_i < \theta + \epsilon$ for sufficiently small ϵ, while maintaining the violating condition. Hereafter, we assume the graph only has "<" splits.

Take a fixed regression graph $G \in \mathcal{G}$ that is chosen prior to the sample $\mathcal{Z}$. In this case, the training error is an unbiased estimate of the generalization error, formed by averaging n independent $[0,1]$ random variables. Hence, by Chernoff bounds (see, e.g., Alon and Spencer **?**), for any $\delta \in (0,1]$,

$$\mathbf{P}_{\mathcal{Z}}\left[|\epsilon(G,\mathcal{Z}) - \epsilon(G)| > \sqrt{\frac{\ln(2/\delta)}{2n}}\right] \le \delta. \tag{17}$$

However, we cannot simply take a union bound over all regression graphs, because there are infinitely many of them.

Instead, define the set of size s regression graphs $\mathcal{G}_s = \{G \in \mathcal{G} \mid |G| = s\}$. Define $\mathcal{G}'_s \subset \mathcal{G}_s$ to be the set of size-s regression graphs where all leaf prediction values are in $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$. Define $\mathcal{G}''_s \subset \mathcal{G}'_s$ to be the set of all regression graphs in $\mathcal{G}'_s$ where all internal nodes have splits of the form $x_i < \theta$ for $\theta = x_i^j$ for some $i \in [d], j \in [n]$. (The set $\mathcal{G}''_s$ depends on the sample $\mathcal{Z}$ and would be more accurately written $\mathcal{G}''_{s,\mathcal{Z}}$). We will first argue that with probability $1-\delta/2$, all $G \in \mathcal{G}''_s$ have true error close to their training error. We will then argue that this implies a similar bound, with probability $1-\delta$, for all regression graphs.

Lemma 10. *With high probability, over a sample $\mathcal{Z}$ of size n, every regression graph $G \in \mathcal{G}''_s$ has close training and generalization errors. In particular,*

$$\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G}''_s \text{ s.t. } |\epsilon(G,\mathcal{Z}) - \epsilon(G)| > \sqrt{\frac{s\ln(6nds/\delta)}{n}}\right] \le \frac{\delta}{2}.$$

Proof. Let us describe each regression graph $G \in \mathcal{G}''_s$ by s 4-tuples, one for each node. Each 4-tuple will be of the form (i,j,l,r), where $i \in [d], j \in [n], l, r \in [s]$. For an internal node, l and r ($l \ne r$) represent the indices of the left and right children, and the split is $x_i < x_i^j$. For a leaf, $i = l = r = 1$ and j/n is the leaf prediction value. This gives us an upper bound of $(dns^2)^s$ on the number of regression graph descriptions of $G \in \mathcal{G}''_s$.

Fix a regression graph description. We now argue that, with high probability, the corresponding regression graph has close training and generalization errors. Let $S \subseteq [n]$ correspond to the indices of the examples that were referred to in splits in the description, so $|S| < s$. Let $\mathcal{Z}_S = \langle (x^i, y^i) \rangle_{i \in S}$ be the subsequence of data that are in S and $Z_{S^c} = \langle (x^i, y^i) \rangle_{i \in [n] \setminus S}$ be the remaining data. Imagine choosing $\mathcal{Z}$ by first choosing $\mathcal{Z}_S$ and later $\mathcal{Z}_{S^c}$. After choosing $\mathcal{Z}_S$, the regression graph G is completely determined. Moreover, the remaining $n-|S|$ examples in $\mathcal{Z}_{S^c}$ give independent, unbiased estimates of the generalization error of G. Since y^i and the predictions of G are in $[0,1]$, the training error on any single example is in $[0,1]$. As in (17), Chernoff bounds imply that, for any $\delta' \in (0,1]$ (we will fix δ' shortly), with probability at least $1-\delta'$, $|\epsilon(G,\mathcal{Z}_{S^c}) - \epsilon(G)| \le \sqrt{\ln(2/\delta')/(2|S^c|)}$. Hence the training error of any nice regression graph G on all of $\mathcal{Z}$

can be bounded by,

$$\begin{aligned}
|\epsilon(G,\mathcal{Z})-\epsilon(G)| &= \left|\frac{|S|}{n}(\epsilon(G,\mathcal{Z}_S)-\epsilon(G))+\frac{|S^c|}{n}(\epsilon(G,\mathcal{Z}_{S^c})-\epsilon(G))\right| \\
&\le \frac{|S|}{n}+\frac{|S^c|}{n}|\epsilon(G,\mathcal{Z}_{S^c})-\epsilon(G)| \\
&\le \frac{|S|}{n}+\frac{|S^c|}{n}\sqrt{\frac{\ln(2/\delta')}{2|S^c|}} \quad \text{(with probability } 1-\delta'\text{)} \\
&\le \sqrt{\frac{|S|}{n}+\frac{|S^c|}{n}\frac{\ln(2/\delta')}{2|S^c|}} \quad \text{(by Jensen's inequality on concave function } f(x)=\sqrt{x}\text{)} \\
&= \sqrt{\frac{2|S|+\ln(2/\delta')}{2n}}.
\end{aligned}$$

Choosing $\delta'=\delta/(2(nds^2)^s)$, by the union bound on the $(nds^2)^s$ descriptions of simple regression graphs, and using $|S|\le s$, we have that with probability $1-\delta/2$, all $G\in\mathcal{G}''_s$ have,

$$|\epsilon(G,\mathcal{Z})-\epsilon(G)|\le\sqrt{\frac{2s+\ln(4(nds^2)^s/\delta)}{2n}}\le\sqrt{\frac{2s\big(\ln(e)+\ln(2nds/\delta)\big)}{2n}}. \qquad \square$$

To proove Theorem 3, we show that one can "round" every regression graph $G\in\mathcal{G}$ to a similar $G'\in\mathcal{G}'_s$ and then to another similar $G''\in\mathcal{G}''_s$, without changing the training or generalization errors significantly. Then we can apply the above lemma to argue that the training and generalization errors of all graphs are close. To do this, we must show that such rounding can be done uniformly, meaning that with probability $1-\delta/2$, simultaneously every graph $G\in\mathcal{G}$ can be rounded to a similar $G''\in\mathcal{G}''_s$ without significant change training or generalization errors.

The rounding procedure involves, first, rounding the leaf prediction values to the nearest multiple of $\frac{1}{n}$ and, second, rounding the splits to be at sample points. For the former, it is easy to show that the predictions $G(x)$ and $G'(x)$ are close for all x. For the latter, we need to argue that, on each coordinate $i\in[d]$, the samples $\mathcal{Z}$ are a close representation of the distribution over the ith coordinate. In particular, imagine ordering the samples in $\mathcal{Z}$ by their ith coordinate, and taking two consecutive examples (examples for which there is no $j\in[n]$ with x_i^j in between the two). The following lemma states that with high probability, the measure of the interval between the two is low.

Lemma 11. *With probability at least $1-\delta/2$, for all $i\in[d]$ and all intervals (open, closed, or half-open) $I\subset\mathbb{R}$, at least one of the following two conditions hold:*

$$\textit{(a) } \exists j\in[n] \textit{ s.t. } x_i^j\in I \quad \textit{or} \quad \textit{(b) } \mu(\{(x,y):x_i\in I\})\le\frac{\ln(2dn^2/\delta)}{n-2}.$$

Proof. Suppose there is an $i \in [d]$ and $I \subset \mathbb{R}$ for which neither of the conditions hold. WLOG we can assume that I is of the form $I = (a, b)$, (a, ∞), or $(-\infty, b)$ with $a = x_i^{j_1}$, and $b = x_i^{j_2}$ for some $j_1, j_2 \in [n]$. This is because we could extend any violating interval to be of this form without satisfying either condition (a) or (b).

There are at most $\binom{n}{2} + 2n \leq n^2$ (WLOG $n \geq 3$) intervals of the form, for $j_1, j_2 \in [n]$. Fix any one of them. The chance that neither (a) nor (b) hold can be bounded by the fact that, given that (b) does not hold, the chance that a given independent sample $j \in [n] \setminus \{j_1, j_2\}$ does not have $x_i^j \in I$ is at most $1 - \ln(2dn^2/\delta)/(n-2)$. Hence the chance that none of the $n-2$ remaining independent samples have $x_i^j \in I$ is at most,

$$\left(1 - \frac{\log(2dn^2/\delta)}{n-2}\right)^{n-2} \leq \left(e^{-\frac{\log(2dn^2/\delta)}{n-2}}\right)^{n-2} = \frac{\delta}{2dn^2}.$$

By the union bound, the chance that any of the dn^2 intervals of the above form (varying $i \in [d]$ as well) have neither (a) nor (b) holding is at most $\delta/2$. $\square$

Lemma 12. *For any $s, n \geq 1$,*

$$\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G}_s \text{ s.t. } |\epsilon(G, \mathcal{Z}) - \epsilon(G)| \geq 2\sqrt{\frac{6s\ln(6nds/\delta)}{5n}}\right] \leq \delta$$

Proof. Every size-s regression graph G can be "rounded" to a size-s regression graph $G' \in \mathcal{G}'_s$ with leaf predictions values in $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$, by rounding each leaf prediction values to the nearest multiple of $1/n$. Such a prediction rounding will change the prediction $G(x)$ on any example by at most $|G'(x) - G(x)| \leq 1/(2n)$. Hence, for any $x \in \mathcal{X}$, $y \in [0, 1]$,

$$\left|(G'(x) - y)^2 - (G(x) - y)^2\right| = |G'(x) - y + G(x) - y|\,|G'(x) - G(x)| \leq 2\,|G'(x) - G(x)|$$

This implies that $|\epsilon(G') - \epsilon(G)|, |\epsilon(G', \mathcal{Z}) - \epsilon(G, \mathcal{Z})| \leq 1/n$ and hence,

$$|\epsilon(G, \mathcal{Z}) - \epsilon(G)| \leq |\epsilon(G', \mathcal{Z}) - \epsilon(G')| + \frac{2}{n} \tag{18}$$

We can round up a split $x_i < \theta$ to the split $x_i < \theta'$ for $\theta' = \min_{j \in [n]}\{x_i^j : x_i^j \geq \theta\}$, or $\theta' = \infty$ if this min does not exist (equivalently, we can remove the test and one child). This rounding has the property that it has not changed the prediction on any samples in $\mathcal{Z}$, and has changed prediction only on examples for which $x_i \in [\theta, \theta')$. Notice that $x_i^j \notin [\theta, \theta')$ for all $j \in [n]$, otherwise we would not have rounded all the way up to θ'. By Lemma 11, we see that with probability $1 - \delta/2$, this means that rounding a single such split will only affect the predictions on a $\mu\{(x, y) : x_i \in [\theta, \theta')\} \leq \log(2dn^2/\delta)/(n-2)$ fraction of the distribution μ.

Imagine performing this split rounding on all (less than s) splits in any G' to get $G'' \in \mathcal{G}''_s$. Lemma 11 together with the union bound implies that, with probability $1 - \delta/2$, the predictions change on at most a $\mu(\{(x,y) : G''(x) \neq G'(x)\}) < s\log(2dn^2/\delta)/(n-2)$ fraction of μ. This implies that,

$$\mathbf{P}_{\mathcal{Z}}\left[\exists G' \in \mathcal{G}'_s \text{ s.t. } |\epsilon(G'') - \epsilon(G')| > \frac{s\ln(2dn^2/\delta)}{n-2}\right] \le \frac{\delta}{2} \tag{19}$$

Combining the fact that $\epsilon(G', \mathcal{Z}) = \epsilon(G'', \mathcal{Z})$ with (18), (19), and Lemma 10, for

$$A = \sqrt{\frac{s\ln(6nds/\delta)}{n}} + \frac{2}{n} + \frac{s\ln(2dn^2/\delta)}{n-2} \le \sqrt{\frac{s\ln(6nds/\delta)}{n}} + \frac{2(1+s\ln(2dn/\delta))}{n-2} \tag{20}$$

we have $\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G}\ |\epsilon(G,\mathcal{Z}) - \epsilon(G)| \ge A\right] \le \delta$. To simplify further, noting that $1 + \ln(2) \le \ln(6)$, we can write $A \le \sqrt{B} + 2B$, for $B = s\ln(6nds/\delta)/(n-2)$. Also, one can verify that $\max\{\sqrt{B}+2B, 1\} \le 2\sqrt{B}$. Thus $\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G} \text{ s.t. } |\epsilon(G,\mathcal{Z}) - \epsilon(G)| \ge 2\sqrt{B}\right] \le \delta$. Also notice that the Lemma holds trivially for $n \le 11$ because $|\epsilon(G,\mathcal{Z}) - \epsilon(G)| \le 1$. For $n \ge 12$, we have $n - 2 \ge (5/6)n$, in which case,

$$2\sqrt{B} =\le 2\sqrt{\frac{s\ln(6nds/\delta)}{(5/6)n}} = 2\sqrt{\frac{6}{5}\frac{s\ln(6nds/\delta)}{n}}.$$ □

The proof of Theorem 3 follows simply from Lemma 12.

Proof of Theorem 3. By Lemma 12, for any $n, s \ge 1$,

$$\mathbf{P}_{\mathcal{Z}}\left[\exists \text{ s.t. } G \in \mathcal{G}_s |\epsilon(G,\mathcal{Z}) - \epsilon(G)| > 2\sqrt{\frac{6s\ln(12nds^3/\delta)}{5n}}\right] \le \frac{\delta}{2s^2}.$$

Finally, because $\sqrt[3]{12} \le 3$ and $2\sqrt{18/5} \le 4$,

$$2\sqrt{\frac{6s\ln(12nds^3/\delta)}{5n}} = 2\sqrt{\frac{18s\ln\sqrt[3]{12nds^3/\delta}}{5n}} \le 3.8\sqrt{\frac{s\ln(3nds/\delta)}{n}}.$$

By the union bound over $s = 1, 2, \ldots$,

$$\mathbf{P}_{\mathcal{Z}}\left[\exists G \in \mathcal{G} \text{ s.t. } |\epsilon(G,\mathcal{Z}) - \epsilon(G)| > 3.8\sqrt{\frac{|G|\ln(3nd|G|/\delta)}{n}}\right] \le \frac{\delta}{2}\sum_{s=1}^{\infty}\frac{1}{s^2} = \frac{\delta}{2}\frac{\pi^2}{6}.$$ □