Leveraging for Regression

Nigel Duffy*
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA
nigeduff@cse.ucsc.edu

David Helmbold
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA
dph@cse.ucsc.edu

Abstract

In this paper we examine master regression algorithms that leverage base regressors by iteratively calling them on modified samples. The most successful leveraging algorithm for classification is AdaBoost, an algorithm that requires only modest assumptions on the base learning method for its good theoretical bounds. We present three gradient descent leveraging algorithms for regression and prove AdaBoost-style bounds on their sample error using intuitive assumptions on the base learners. We derive bounds on the size of the master functions that lead to PAC-style bounds on the generalization error.

1 Introduction

In this paper we consider the following regression setting. Data is generated IID from a distribution $\mathcal{P}$ on some domain X and labeled according to a function g. A learning algorithm receives a sample $S = \{(x_1, g(x_i)), \ldots, (x_m, g(x_m))\}$ and attempts to return a function f close to g on the domain X. There are many ways to measure the closeness of f to g, for example one may want the expected squared error to be small or one may want f to be uniformly close to g over the entire domain.

In fact, one often considers the case where the data are not labeled perfectly by any function, but rather have random noise added to the labels. In this paper we consider only the noise free case. However, many algorithms with good performance guarantees for the noise free case also work well in practical settings. AdaBoost [13, 2, 19, 12] is one example in the classification setting.

A typical approach for learning is to choose a function class $\mathcal{F}$ and find some $f \in \mathcal{F}$ with small error on the sample. Then if the sample is large enough with respect to the complexity of the class $\mathcal{F}$, f will also provide small error on the domain X with high probability (see e.g. Anthony and Bartlett [1]).

To be useful any algorithm that works in this way must also be computationally efficient, i.e. it must obtain a simple hypothesis, with high accuracy in a short time. There are many learning algorithms that produce simple hypotheses efficiently, but the accuracy of these methods may be less than desirable. Leveraging techniques, such as boosting [23, 10, 13, 11, 14, 8], attempt to take advantage of such algorithms to efficiently obtain a hypothesis with arbitrarily high accuracy. These methods work by repeatedly calling a simple (or base) learning method on modified samples in order to obtain different base hypotheses that are combined into an improved master hypothesis. Of course the complexity of the combined hypothesis will often be much greater than the complexity of the base hypotheses. However, if the improvement in accuracy is large, and the increase in complexity is small, then leveraging can improve generalization.

Leveraging has been examined primarily in the classification setting where AdaBoost [13] and related leveraging techniques [3, 9, 5, 4, 6, 14, 20] have been found to be useful for increasing the accuracy of base classifiers. Such algorithms repeatedly call a base learning algorithm, and construct a linear combination of the hypotheses returned. These techniques have recently been viewed as performing gradient descent on a potential function [4, 6, 20, 14, 9, 18] and this viewpoint has enabled the derivation and analysis of new algorithms in the classification setting [14, 9, 8, 18]. Recent work by Friedman has shown that this gradient descent viewpoint can also be used to construct leveraging algorithms for regression with good empirical performance [15].

We are aware of several other generalizations of gradient descent leveraging to the regression setting [13, 15, 17, 21, 5, 4, 6]. These approaches are discussed in more detail in Section 2.

In analyzing and deriving gradient descent leveraging algorithms, several issues arise. First, a potential function must be chosen such that minimizing this potential implies that the master function performs well with respect to the loss function of interest. This potential should also be amenable to analysis; most of the bounds on such algorithms are proved using an amortized analysis of the potential [13, 10, 9, 11]. This paper examines several potential functions for leveraging in the regression setting and proves performance bounds for the resulting gradient descent algorithms. Second, to derive bounds, assumptions need to be made about the performance of the base learners. In particular, if the base learner does not return useful hypotheses, then the leveraging algorithm cannot be expected to make progress. What constitutes a useful hypothesis will depend on the potential function being minimized. Furthermore, how "usefulness" is measured

*Both authors were supported by NSF grants CCR 9700201 and CCR 9821087.

will have a major impact on the difficulty of proving performance bounds. In this paper, we attempt to use weak assumptions on the base learners, and use measures of "usefulness" that are intuitive. Finally, we desire performance bounds that are of the strongest form possible. In this paper, we prove non-asymptotic bounds on the sample error that hold for every iteration of the leveraging algorithm. These sample error bounds lead to PAC-style generalization bounds showing that with arbitrarily high probability over the random sample, the leveraging algorithm will have arbitrarily low generalization error. To obtain this performance our algorithms need only run for a polynomial number of iterations.

The regression setting has additional considerations not seen in classification problems. In the classification setting the complexity of the combined hypothesis can be bounded in terms of the number of components in the combination and the complexity of the base hypotheses. For regression, however, we must also take the size of the coefficients in the linear combination into account. This means that it may not be best to take a large gradient descent step each iteration, since the step size may induce an overly large coefficient. The choice of step size leads to a tradeoff between the complexity of the combined hypothesis and the number of iterations required to achieve good performance on the training sample. This trade off is discussed further in Section 3.

In the classification setting, the base learner can be forced to return useful hypotheses simply by manipulating the distribution over the sample. This is not the case for regression. In the regression setting, a leveraging algorithm must also modify the sample in some way. However, no useful base learner can perform well with respect to an arbitrarily labeled sample. In Section 3, we illustrate a situation in which the relabeling used by our algorithms is far from arbitrary. In fact, in such situations the relabeling is still consistent with a reasonable hypothesis.

Throughout the paper we use the following notation. The the leveraging algorithm is given a set of m training examples $S = \{(x_1, y_1), \ldots (x_m, y_m)\}$. For a master regression function F, the *residuals* are $r_i = y_i - F(x_i)$ for $1 \leq i \leq m$. Each iteration of the leveraging process the algorithm modifies the sample S to produce $\tilde{S} = \{(x_1, \tilde{y}_1), \ldots, (x_m, \tilde{y}_m)\}$ by changing the target y values to $\tilde{y}$. The algorithm then creates a distribution D over the modified sample $\tilde{S}$ and calls a base regression algorithm on the modified sample with the distribution D. The base regressor produces a function $f \in \mathcal{F}$ with some "edge" ϵ on $\tilde{S}$ under D. (The different algorithms evaluate the amount of "edge" differently). The new master regressor then chooses a coefficient α for the new base function and updates its master regression function to $F + \alpha f$. We often use bold face as abbreviations for vectors over the sample, e.g. $\mathbf{f} = (f(x_1), \ldots, f(x_m))$ and $\mathbf{y} = (y_1, \ldots, y_m)$.

In Section 3 we introduce and discuss three new potentials from which we derive leveraging algorithms for regression. The proofs of these results are deferred to Section 4. The potential functions we examine are two variants of the squared error and an exponential criterion motivated by AdaBoost. Our first algorithm, SQUARELEV.R, uses a uniform distribution on the sample and $\tilde{y}$ labels proportional to the gradient of the potential. An amortized analysis shows that this algorithm effectively reduces the loss on the sample if the base regressor returns a function f whose correlation coefficient with the $\tilde{y}$ values is at least some $\epsilon > 0$.

SQUARELEV.C, our second algorithm, uses the squared error potential, but with confidence rated classifiers as its base regressors. This second procedure places a distribution D on the examples proportional to the absolute value of the gradient of the square loss and $\tilde{y}$ labels equal to the sign of the gradient. We prove that this procedure effectively reduces the loss on the sample when the base classifier produces functions f satisfying

$$\frac{\sum_i D(x_i)\tilde{y}_i f(x_i)}{\sqrt{\sum_i D(x_i)^2 \sum_i f(x_i)^2}} \geq \epsilon \ .$$

Both of these constraints on the base regressor are similar to those assumed by the GeoLev algorithm [9] and analogous to those for AdaBoost [13].

A third algorithm EXPLEV performs gradient descent on the exponential criterion $\sum \exp(sr_i) + \exp(-sr_i) - 2$ where s is a scaling factor. This is a two-sided version of AdaBoost's $\exp(-\text{margin})$ potential. Whereas in the classification setting AdaBoost can be seen as increasing the smallest margin, for regression we want to decrease the magnitude of the residuals. By noting that when $sr_i \gg 0$ or $sr_i \ll 0$, the contribution to our potential is close to $\exp(s|r_i|)$, it seems reasonable that this potential tends to decrease the maximum magnitude of the residuals. The probability $D(x_i)$ used by EXPLEV is proportional to the absolute value of the gradient with respect to $\mathbf{F}$, $|\nabla_i P| = |s\exp(-sr_i) - s\exp(sr_i)|$, and the $\tilde{y}_i$ label is $-\text{sign}(\nabla_i P)$. If the weak regressor returns functions f with edge $\sum_i D(x_i)\tilde{y}_i f(x_i) > \epsilon$ then this procedure rapidly reduces the exponential potential on the sample, and for appropriate s the maximum $|y_i - F(x_i)|$ value is at most η after $O((\ln m)/\eta \ln(\frac{1}{1-\epsilon^2/6}))$ iterations. Therefore, the master regression function approximately interpolates the data [1].

Recall that the master function is $F = \sum_t \alpha_t f_t$ where α_t and f_t are the values computed at iteration t. With additional assumptions on the base regressor we can bound $\sum_t \alpha_t$ and prove (ϵ,δ)-bounds for the generalization error of the master regression function (assuming the sample was drawn IID). For SQUARELEV.R, we require that the standard deviation of $\mathbf{f}$ is not too much smaller than the standard deviation of the $\tilde{y}$ values. For EXPLEV, we truncate large descent steps.

Some contributions of this work are deriving and rigorously analyzing three new leveraging algorithms for regression, and exploring the complexity versus computation trade off that arises in this regression setting.

2 Relation to Other Work

Leveraging techniques work by repeatedly calling a simple (or base) learning method on modified samples to obtain different base rules that are combined into a master rule. In this way leveraging methods attempt to produce a master rule that is better than any of its components. Leveraging methods include ARCing [5, 4, 6], Bagging [3] and Boosting [23, 10, 13, 11, 14, 8]. Leveraging for classification has received considerable attention [13, 3, 10, 2, 19, 12, 25, 20,

18] and it has been observed that many of these algorithms perform an approximate gradient descent of some potential [4, 6, 20, 14, 9, 18]. Given this observation it is possible to derive new leveraging algorithms by choosing a new potential.

The most successful gradient descent leveraging method is Freund and Schapire's Adaboost [13] algorithm for classification. In addition to its empirical success [2, 19, 12], AdaBoost has strong theoretical guarantees [13, 24] with reasonably weak assumptions on the base learners. Here we concentrate on deriving gradient descent leveraging algorithms for regression with similar guarantees.

Although, leveraging for regression has not received nearly as much attention as leveraging for classification, there is some work examining gradient descent leveraging algorithms in the regression context.

The AdaBoost.R algorithm [13] solves the regression problem by reducing it to a classification problem. To fit a set of (x, y) pairs with a regression function, where each $y \in [-1, 1]$, AdaBoost.R converts each (x_i, y_i) regression example into an infinite set of $((x_i, z), \tilde{y}_i)$ pairs, where $z \in [-1, 1]$ and $\tilde{y}_i = \text{sign}(y_i - z)$. The base regressor is given a distribution D over (x_i, z) pairs and must return a function $f(x)$ such that its weighted "error" $\sum_i |\int_{y_i}^{f(x_i)} D(x_i, z)dz|$ is less than 1/2. Although experimental work shows that algorithms related to AdaBoost.R [16, 22] can be effective, it suffers from two drawbacks.

First, it expands each instance in the regression sample into many classification instances. Although the integral above is piecewise linear, the number of different pieces can grow linearly in the number of boosting iterations.

More seriously, the "error" function that the base regressor should be minimizing is not (except for the first iteration) a standard loss function. Furthermore, the loss function changes from iteration to iteration and even differs between examples on the same iteration. Therefore, it is difficult to determine if a particular base regressor is appropriate for AdaBoost.R.

Breiman used a gradient descent approach to the regression problem to prove asymptotic convergence results for his arc-gv algorithm [5, 4, 6]. More recently, Friedman has explored regression using the gradient descent approach [15]. Each iteration, Friedman's master algorithm constructs $\tilde{y}_i$-values for each data-point x_i equal to the (negative) gradient of the loss of its current master hypothesis on x_i. The base learner then finds a function in a class $\mathcal{F}$ minimizing the squared error on this constructed sample. Friedman applies this technique to several loss functions, and has performed experiments demonstrating its usefulness, but does not present analytical bounds.

Friedman's algorithm for the square-loss is closely related to Lee, Bartlett and Williamson's earlier Constructive Algorithm for regression [17]. Bartlett *et al.* prove that the Constructive algorithm is an efficient and effective learning technique when the base learner returns a function in $\mathcal{F}$ approximately minimizing the squared error on the modified sample. These algorithms are very similar to both our SQUARELEV.R and SQUARELEV.C algorithms.

In work parallel to ours, Rätsch *et al.* [21] relate boosting algorithms to barrier methods from linear programming and use that viewpoint to derive new leveraging algorithms. They prove a general asymptotic convergence result for such algorithms applied to a finite base hypothesis class. One of their algorithms ϵ-boost is similar to our EXPLEV algorithm.

AdaBoost and AdaBoost.R only require that the base hypotheses have a slight edge. In contrast, almost all of the work on leveraging for regression assumes that the function returned by the base regressor approximately minimizes the error over its function class. Here, we analyze the effectiveness of gradient descent procedures when the base regressor returns hypotheses that are only slightly correlated with the labels on the sample. In particular, we consider natural potential functions and determine sufficient properties of the base regressor so that the resulting gradient descent procedure produces good master regression functions.

When attempting to derive and analyze leveraging algorithms for regression, several issues arise that do not appear in the classification setting.

In the classification setting, leveraging algorithms are able to extract useful functions from the base learner by manipulating the distribution over the sample, and do not need to modify the sample itself. If the base learner returns functions with small loss (classification error rate on the weighted sample less than 1/2), then several [23, 10, 13, 14, 11, 8] leveraging algorithms can rapidly produce a master function that correctly classifies the entire sample.

A key difference between leveraging for regression and leveraging for classification is the following observation:

> Unlike leveraging classifiers, leveraging regressors cannot always force the base regressor to output a useful function by simply modifying the distribution over the sample.

To see this, consider the regression problem with a continuous loss function L mapping prediction-label pairs to the non-negative reals. Let f be a function having the same modest loss on every instance. Since changing the distribution on the sample does not change the expected loss of f, the base learner can return this same f each iteration. Of course if f consistently underestimates (or overestimates) the y-values in the sample, then the master can shift or scale f and decrease the average loss. However, for many losses (such as the square loss) it is easy to construct (sample,f) pairs where neither shifting nor scaling reduces the loss.

The confidence rated prediction setting of Schapire and Singer [25] (where each $y_i \in \{\pm 1\}$ and $f(x) \in [-1, +1]$) does not have this problem: if the "average loss" $(1 - \sum D(x_i)y_i f(x_i))/2$ is less than 1/2, and the loss on each example $(1 - y_i f(x_i))/2$ is the same, then each $y_i f(x_i) > 0$ and thresholding f gives a perfect classifier on the sample. It is this thresholding property of classification that makes manipulating the distribution sufficient for boosting.

In the PAC setting, it is assumed that the sample is labeled consistently with some hypothesis in a base hypothesis class. When the base learners are PAC weak learners, they return hypotheses with average classification error on the sample less than $1/2$ regardless of the distribution given. Note that random guessing has average classification error 1/2, so it is plausible to assume that the base learner can do slightly better (when the labeling function comes from the

known class). However, if the sample is modified, it is no longer clear that the labels are consistent with any hypothesis in the class, and the weak learner may not have any edge.

In general, no useful function class contains hypotheses consistent with every arbitrary relabeling of a large sample. To justify modification of the samples, therefore, we need to show that there are consistent functions in some reasonable function class. In Section 3 we discuss properties of the base hypothesis class and target function that ensure the relabelings our algorithms use are consistent with a hypothesis in the target class.

We have identified three approaches to modifying the sample for gradient descent leveraging algorithms. Which approach is best depends on the properties of the base regressor and the potential function. The AdaBoost.R algorithm manipulates the loss function as well as the distribution over the sample, but leaves the labels unchanged. Friedman's Gradient Boost algorithm modifies the sample (by setting the labels to the negative gradient) while keeping the distribution constant. A third approach is used by two of the algorithms presented here. Each modified label is the sign of the corresponding component of the (negative) gradient while the distribution is proportional to the magnitudes of the gradient. This third approach uses ± 1 labels, and is suitable for base regressors that solve classification problems.

In the next section we discuss our main results.

3 Algorithms and Main Results

Several issues arise when examining leveraging in the context of regression. First, we must consider the loss function with which we measure the generalization error of the master function – this will motivate the choice of potential function. Second, we must choose a base learner appropriate for the choice of potential. The choice of base learner has two parts: should the base learner be a regression algorithm itself or should it be a classification algorithm? What criteria should the base learner be attempting to optimize?

In this section we explore these issues by examining three algorithms. We prove that each of these algorithms efficiently produces a regression function with low generalization error, measured with respect to an appropriate loss function. These proofs require assumptions on the base learner that seem intuitively reasonable, and we provide some justification for them. However, the base learners must perform well with respect to a modified sample. Although this may not be possible in general, we conclude this section by describing situations for which it is.

The results bounding our algorithms proceed like other amortized analyses of leveraging algorithms [13, 8]. First, we bound the change in potential for a single iteration. Second, this bound is used to derive a bound on the sample error. Third, bounds on the size of the resulting combined hypothesis are used to obtain generalization error bounds. For classification this "size" can be measured just in terms of the number of base hypotheses combined. In particular, for classification we may take a hypothesis class and rescale it without changing its complexity, but this is not the case for classes of regression functions. In the regression setting we must be more careful about the complexity of the combined hypothesis, as the "size" of a linear combination depends on the magnitude of the coefficients in the combination. Therefore we must show that our algorithms are not only computationally efficient but are also efficient with respect to the size of the linear combinations produced. Letting the algorithm take a full gradient step might produce a coefficient that is prohibitively large. Varying the size of the gradient steps taken can lead to a trade off between the time taken to obtain a good regressor and the complexity of that regressor. This trade off is difficult to optimize, and our results do not require such an optimization.

It is interesting to note that the magnitude of the coefficients in the linear combination also appear in the margin analyses of boosting algorithms [24, 8] for classification, so a similar trade off also appears in the classification setting.

In this section we state our main results, PAC-style bounds on the probability that the algorithm fails to obtain a good master function. The proofs are postponed to Section 4. We will assume that the base function class $\mathcal{F}$ is closed under negation, so $f \in \mathcal{F} \Rightarrow -f \in \mathcal{F}$, and that $\mathcal{F}$ contains the zero function.

3.1 Regression with Squared Error

A standard approach to regression involves choosing a regression function f from some class $\mathcal{F}$ that minimizes the squared error on a sample.

Definition 3.1 *[1] Given a real-valued function f and a sample $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ in $(X \times \Re)^m$, the* sample error *of f on S, denoted $\widehat{er}_S(f)$, is*

$$\widehat{er}_S(f) = \frac{1}{m} \sum_{i=1}^{m} (y_i - f(x_i))^2 \ . \tag{1}$$

This is done in the hope that achieving small sample error will imply small error on unseen examples. In fact, this is the case if the complexity of the function class $\mathcal{F}$ is low enough (see e.g. Anthony and Bartlett [1]).

Definition 3.2 *[1] Let $\mathcal{P}$ be a probability distribution on $X \times \Re$. The (generalization)* error *of a function $f : X \to \Re$ with respect to $\mathcal{P}$ is*

$$er_{\mathcal{P}}(f) = \mathbf{E}(y - f(x))^2$$

where the expectation is with respect to a random (x, y) drawn according to $\mathcal{P}$.

Here we describe SQUARELEV.R and SQUARELEV.C, two gradient descent leveraging algorithms for regression that produce hypotheses with low expected squared error. SQUARELEV.R uses a regression algorithm as its base learner while SQUARELEV.C may use a classification algorithm. These algorithms modify the sample and distribution in different ways. Despite this, the convergence bounds we give are similar for both algorithms. We analyze the behavior of SQUARELEV.R in detail and show that, using appropriate base learners, it can efficiently achieve arbitrarily small squared error on the sample. We also show that the resulting hypothesis is simple enough that, with high probability, it achieves small expected squared error on the entire domain. To show this we require a bound on the complexity of the function class $\mathcal{M}$ in which the master functions lie. Since

Input: A sample $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
and a base learning algorithm
Set $D(x_i) = 1/m$
initialize master function F_1 to the zero function. $(t = 1)$
repeat for $t = 2, \ldots$
for $i = 1$ to m
$r_i = y_i - F(x_i)$
end do
for $i = 1$ to m
$\tilde{y}_i = r_i - \frac{1}{m}\sum_i r_i$
end do
$S' = \{(x_1, \tilde{y}_1), \ldots, (x_m, \tilde{y}_m)\}$
Call base learner with distribution D over S',
obtaining hypothesis f
$\epsilon_t = \frac{((\mathbf{r}-\bar{\mathbf{r}})\cdot(\mathbf{f}-\bar{\mathbf{f}}))}{||\mathbf{r}-\bar{\mathbf{r}}||_2||\mathbf{f}-\bar{\mathbf{f}}||_2}$
$\alpha_t = \frac{\epsilon_t||\mathbf{r}-\bar{\mathbf{r}}||_2}{||\mathbf{f}-\bar{\mathbf{f}}||_2}$
$F_t = F_{t-1} + \alpha_t f$

Figure 1: The SQUARELEV.R algorithm.

the master function is a linear combination of base functions, its complexity will depend on the complexity of the base function class $\mathcal{F}$ and the size of the coefficients in the linear combination. Bounding the size of the coefficients in the linear combination is a key step in the analysis of these algorithms.

SQUARELEV.R

Rather than attempting to minimize the mean squared error over the sample directly, SQUARELEV.R uses the variance of the residuals as the potential,

$$P_{var} = ||\mathbf{r} - \bar{\mathbf{r}}||_2^2 \tag{2}$$

where $\mathbf{r}$ is the m-vector of residuals defined by $r_i = y_i - F(x_i)$, and $\bar{\mathbf{r}}$ is the m-vector with all components equal to $\frac{1}{m}\sum_{i=1}^m r_i$. Notice that any $F(x)$ can easily be shifted to $\tilde{F}(x) = F(x) + \frac{1}{m}\sum_{i=1}^m (y_i - F(x_i))$ such that $\widehat{\text{er}}_S(\tilde{F})$ is $\frac{1}{m}$ times the potential of F.

Algorithm SQUARELEV.R (stated formally in Figure 1) gives the base regressor a modified sample $\{(x_i, \tilde{y}_i)\}$, with $\tilde{y}_i = r_i - \bar{r}$ (proportional to the gradient of P_{var} with respect to $\mathbf{F}$), and the uniform distribution over the modified sample. For this algorithm we define the *edge* of the returned base function f as

$$\epsilon = \frac{((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}}))}{||\mathbf{r} - \bar{\mathbf{r}}||_2||\mathbf{f} - \bar{\mathbf{f}}||_2} = \frac{((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}}))}{\sigma_{\mathbf{r}}\sigma_{\mathbf{f}}} \tag{3}$$

where $\sigma_{\mathbf{r}}$ and $\sigma_{\mathbf{f}}$ are the standard deviations of the respective vectors. Note that the edge is just the correlation coefficient between $\mathbf{f}$ and $\mathbf{r}$ (or $\tilde{\mathbf{y}}$). The value α is chosen to minimize the potential of the new master function.

The following theorem shows how the potential (2) decreases each iteration.

Theorem 4.1 *If ϵ is the edge (3) of the base function f in an iteration of* SQUARELEV.R *then the potential P_{var} decreases by a factor of $(1 - \epsilon^2)$ during the iteration.*

Given the relationship between the potential P_{var} and $\widehat{\text{er}}_S(\tilde{F})$, Theorem 4.1 proves that each iteration the mean squared error of $\tilde{F}$ decreases by a multiplicative factor that depends on the edge of the base learner. Using this result and assuming a lower bound ϵ_{min} on the edge of the base learner allows us to prove the following theorem.

Theorem 4.2 *Assume that each $y_i \in \left[-\frac{B}{2}, \frac{B}{2}\right]$. If the edges of the weak hypotheses used by* SQUARELEV.R *are bounded below by $\epsilon_{min} > 0$ then for all $\rho \in (0, 1)$ after*

$$T = \left\lceil \frac{\ln\left(\frac{B^2}{4\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil$$

iterations the function $\tilde{F}_T$ has sample error $\widehat{er}_S(\tilde{F}_T) \leq \rho$. If in addition $\frac{||\mathbf{r}-\bar{\mathbf{r}}||_2}{||\mathbf{f}-\bar{\mathbf{f}}||_2} \leq c$ in every iteration then

$$\sum_{t=1}^{T} \alpha_t \leq c \left\lceil \frac{\ln\left(\frac{B^2}{4\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil .$$

This bounds the number of iterations before $\widehat{\text{er}}_S(\tilde{F}) \leq \rho$ and also the size of the coefficients in the linear combination produced. The bound is independent of the number of examples m because $\widehat{\text{er}}_S(\tilde{F})$ is the potential/m, and the potential decreases by at least a constant factor each iteration.

Using these bounds on T and bounds on the complexity of the master function produced (derived from the bounds on T and $\sum_{t=1}^{T} \alpha_t$) we obtain the following PAC-style result.

Corollary 3.3 *Assume that data is drawn IID from a distribution $\mathcal{P}$ on $X \times \left[\frac{-B}{2}, \frac{B}{2}\right]$, that the base regression functions $f \in \mathcal{F}$ returned by the base learner map to $[-1, +1]$ and satisfy $\epsilon > \epsilon_{min}$, and $\frac{||\mathbf{r}-\bar{\mathbf{r}}||_2}{||\mathbf{f}-\bar{\mathbf{f}}||_2} \leq c$, that the base learner runs in time polynomial in $m, 1/\epsilon$, and that Pdim$(\mathcal{F})$ is finite. Then $\exists m(\rho, \delta)$ polynomial in $1/\rho, 1/\delta$ such that for all $\rho, \delta \in (0, 1)$, with probability $1 - \delta$,* SQUARELEV.R *produces a hypothesis F, in time polynomial in $1/\rho, 1/\delta$, with $er_{\mathcal{P}} < \rho$ when trained on a sample of size $m(\rho, \delta)$.*

This result shows that SQUARELEV.R is an efficient regression algorithm for obtaining functions with arbitrarily small squared error. The result follows from the results in Section 4, which are more precise in nature.

The conditions placed on the base learner are worth examining further. The lower bound on the edge of the base learner ϵ_{min} is similar that used in the GeoLev [9] algorithm and is analogous to that used by AdaBoost. Since the edge of the base learner is simply the correlation coefficient between the residuals and the predictions of the base function, it seems reasonable to assume that this is bounded away from zero. One cannot expect to model the residuals using a function that is totally uncorrelated with them. In addition we require the condition that

$$\frac{||\mathbf{r} - \bar{\mathbf{r}}||_2}{||\mathbf{f} - \bar{\mathbf{f}}||_2} \leq c \text{ , or } \frac{\sigma_{\mathbf{r}}}{\sigma_{\mathbf{f}}} \leq c .$$

This condition requires that the base function does not have much smaller variance over the sample than the residuals.

Given that we are trying to model the residuals using the base function, this seems like a reasonable assumption. In the extreme case where the base function has $\sigma_{\mathbf{f}} = 0$, no progress can be made as $\mathbf{f}$ is constant and the residuals would all be modified by exactly the same amount.

SQUARELEV.C

We define SQUARELEV.C to be the gradient descent leveraging algorithm using the potential

$$P_{sq} = ||\mathbf{y} - \mathbf{F}||_2^2 = ||\mathbf{r}||_2^2 \ . \tag{4}$$

Note that the gradient of P_{sq} w.r.t. $\mathbf{F}$ is $2\mathbf{r}$.

For SQUARELEV.C, we define the edge of the base regressor as:

$$\epsilon = \frac{\sum_{i=1}^m D(x_i)\tilde{y}_i f(x_i)}{||\mathbf{D}||_2||\mathbf{f}||_2} = \frac{(\mathbf{r}\cdot\mathbf{f})}{||\mathbf{r}||_2||\mathbf{f}||_2} \ . \tag{5}$$

SQUARELEV.C mimics SQUARELEV.R with the following exceptions. SQUARELEV.C uses ± 1-valued labels and modifies the distribution each iteration. The modified labels are $\tilde{y}_i = \text{sign}(r_i)$. The distribution $D(x_i)$ sent to the base regressor is recomputed each iteration and is proportional to $|r_i|$ (and thus proportional to the magnitude of the gradient with respect to $\mathbf{F}$). The edge ϵ_t is computed as above. The value $\alpha_t = \frac{\epsilon_t||\mathbf{r}||_2}{||\mathbf{f}||_2} = \frac{(\mathbf{r}\cdot\mathbf{f})}{||\mathbf{f}||_2^2}$.

Note that since SQUARELEV.C uses ± 1-valued labels, it may work well when the base functions are classifiers with range $\{-1, +1\}$. The following theorem provides a progress bound for SQUARELEV.C that is similar to that given for SQUARELEV.R in Theorem 4.1.

Theorem 4.4 *If ϵ is the edge (5) of the base function f in an iteration of* SQUARELEV.C *then the potential P_{sq} decreases by a factor of $(1-\epsilon^2)$ during the iteration.*

The potential and suitable base learners for SQUARELEV.C are closely related to those used by the GeoLev [9] algorithm. In particular, base hypotheses which tend to "abstain" on a large portion of the sample seem appropriate for these algorithms as the edge (5) tends to increase if the base learner effectively trades off abstentions for decreased error.

Due to its similarity to SQUARELEV.R, we omit further analysis of SQUARELEV.C.

3.2 An AdaBoost-like algorithm

An alternative goal for regression is to have almost-uniformly good approximation to the true regression function. One way to achieve this is to obtain a simple function that has small residuals at almost every point in a sufficiently large sample, in other words the goal may be to find a function that interpolates or approximately interpolates the sample. This approach is known as generalization from approximate interpolation [1].

Definition 3.4 *[1] Suppose that $\mathcal{F}$ is a class of functions mapping from a set X to the interval $[0,1]$. Then $\mathcal{F}$* generalizes from approximate interpolation *if for any $\epsilon, \delta, \eta, \gamma \in (0,1)$, there is $m_0(\epsilon,\delta,\eta,\gamma)$ such that for $m \geq m_0(\epsilon,\delta,\eta,\gamma)$, for any probability distribution $\mathcal{P}$ on X and any function $g : X \to [0,1]$, the following holds: with probability at least $1-\delta$, if $x = (x_1, x_2, \ldots, x_m) \in \mathcal{P}^m$, then for any $f \in \mathcal{F}$ satisfying $|f(x_i) - g(x_i)| < \eta$ for $i = 1, 2, \ldots, m$, we have*

$$\mathcal{P}\{x : |f(x) - g(x)| < \eta + \gamma\} > 1 - \rho \ . \tag{6}$$

This property provides a uniform bound on almost all of the input domain and is therefore considerably stronger in nature than a bound on the expected squared error.

The notion of approximate interpolation is closely related to the ϵ-insensitive loss used in support vector machine regression [26].

In this section we examine a third algorithm EXPLEV which uses an exponential potential related to the one used by AdaBoost [13]. The AdaBoost algorithm pushes all examples to have positive margin. In the regression setting, the EXPLEV algorithm pushes the examples to have small residuals. We show that this is possible and, given certain assumptions on the class of base hypotheses, that the residuals are also small on a large portion of unseen examples.

To obtain a uniformly good approximation it is desirable to decrease the magnitude of the largest residual, so one possible potential is $\max_i |r_i|$. However, a gradient descent algorithm for this potential would be difficult to analyze. The EXPLEV algorithm instead uses the two-sided potential

$$P_{exp} = \sum_{i=1}^m \left(e^{sr_i} + e^{-sr_i} - 2\right) \ , \tag{7}$$

where s is a scaling factor. When sr is large P_{exp} behaves like $\exp(s \max_i |r_i|)$. P_{exp} is also non-negative, and zero only when each $F(x_i) = y_i$.

P_{exp} is essentially exponential except for a flat region around 0. The scalar s is chosen so that this flat region corresponds to the region of acceptable approximation. The exponential regions have a similar effect to the exponential potential used by AdaBoost: the example with the largest potential tends to have its potential decreased the most.

The components of the gradient (wrt. $\mathbf{F}$) are

$$\nabla_i P_{exp} = \frac{\partial P_{exp}}{\partial F(x_i)} = -s\exp(sr_i) + s\exp(-sr_i) \ . \tag{8}$$

For EXPLEV we assume that the base hypotheses have range $[-1,+1]$, and that the goal is to find a master hypothesis $F(x) = \sum \alpha_t f_t(x)$ such that $|r_i| = |y_i - F(x_i)| \leq \eta$ for some given η and each of the m examples in the sample. The scaling factor we find most convenient is $s = \ln(m)/\eta$.

Like SQUARELEV.C each iteration the distribution D and the modified labels that EXPLEV gives to the base regressor are

$$D(x_i) = \frac{|\nabla_i P|}{||\nabla P||_1} \tag{9}$$

$$\tilde{y}_i = \text{sign}(r_i) \ . \tag{10}$$

The base regressor could be either a classifier (returning a $\{-1,1\}$-valued f) or a regressor (where the returned f gives values in $[-1,+1]$). In either case, we define the *edge* ϵ of a base hypothesis f as

$$\epsilon = \sum_{i=1}^m D(x_i)\tilde{y}_i f(x_i) \ . \tag{11}$$

Input: A sample $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$,
a base learning algorithm, parameters s and ϵ_{max}
initialize master function F_1 to the zero function. $(t = 1)$
repeat for $t = 2, \ldots$
$P_{exp} = \sum_{i=1}^{m} (e^{sr_i} + e^{-sr_i} - 2)$
for $i = 1$ to m
$r_i = y_i - F(x_i)$
$\nabla_i P_{exp} = -s \exp(sr_i) + s \exp(-sr_i)$
$D_t(x_i) = \frac{|\nabla_i P|}{||\nabla P||_1}$
$\tilde{y}_i = \text{sign}(r_i)$
end do
$S' = \{(x_1, \tilde{y}_1), \ldots, (x_m, \tilde{y}_m)\}$
Call base learner with distribution D over S',
obtaining hypothesis f
$\hat{\epsilon} = \min\left(\sum_{i=1}^{m} D(x_i)\tilde{y}_i f(x_i), \epsilon_{max}\right)$
$\alpha_t = \frac{1}{2s} \ln\left(\frac{sP_{exp} - 2sm + \hat{\epsilon}||\nabla P_{exp}||_1}{sP_{exp} - 2sm - \hat{\epsilon}||\nabla P_{exp}||_1}\right)$
$F_t = F_{t-1} + \alpha_t f$

Figure 2: The EXPLEV algorithm.

This is the same definition of edge used in the confidence rated version of AdaBoost [25]. The main difference between the base learners used here and those used by AdaBoost is that here the base learner must perform well with respect to a relabeled sample. Although, this may not be possible in general it seems reasonable in many situations (see the discussion in Section 3.3).

In addition to the parameter s, EXPLEV also takes a second parameter, ϵ_{max}. This second parameter is used to regularize the algorithm by bounding the size of the steps taken. The algorithm is stated formally in Figure 2.

In the following theorem we bound the progress that EXPLEV makes each iteration. The algorithm sets $\hat{\epsilon}$ to the minimum of ϵ and ϵ_{max}.

Theorem 4.8 *If $m \geq 3$,$P_{exp} \geq m + \frac{1}{m} - 2$ at the start of an iteration of* EXPLEV *and ϵ is the edge of the base function at that iteration then the stepsize $\alpha < \frac{1}{2s} \ln\left(\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}\right) \leq \frac{1}{2s} \ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right)$ and potential P_{exp} decreases by at least a factor of $(1 - \frac{\hat{\epsilon}^2}{6})$ during the iteration.*

Using this bound and assuming a lower bound ϵ_{min} on the edge of the base hypotheses we can prove that EXPLEV obtains a function F such that all of the residuals are small within a number of iterations that is logarithmic in the sample size and linear in $1/\eta$. These results are stated in the following theorem.

Theorem 4.9 *If ϵ_{min} is a lower bound on the edges of the base functions and each $y_i \in \left[-\frac{B}{2}, \frac{B}{2}\right]$ then for all $\eta \in (0, 1)$ algorithm* EXPLEV *with $s = \ln(m)/\eta$ and $\epsilon_{max} \geq \epsilon_{min}$ achieves $|y_i - F(x_i)| < \eta$ within*

$$T = \left\lceil \frac{\ln(m)\left(1 + \frac{B}{\eta}\right)}{\ln\left(\frac{1}{\left(1 - \frac{1}{6}\epsilon_{min}^2\right)}\right)} \right\rceil$$

iterations. Furthermore,

$$\sum_{t=1}^{T} \alpha_t \leq (B + \eta + 1) \frac{\ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\ln \frac{1}{1-\epsilon_{min}^2/6}} .$$

It is worth examining this result in a little more detail. Despite the linear dependence on $1/\eta$ in the bound on T, the η term is negligible in the bound on the α's. As the required accuracy is increased the bound on the length of the total path traversed by the algorithm (the sum of the step sizes) does not change, however, the individual step sizes shrink. The algorithm approximates the steepest descent path more and more closely as the required accuracy increases. This illustrates an interesting tradeoff between the number of iterations and the size of the individual coefficients. To obtain a simple enough hypothesis the coefficients must be kept small at the expense of increased computational cost.

To show that all the residuals on unseen data are also small we require a bound on the complexity of the master function class $\mathcal{M}$ from which F is drawn. Since $\mathcal{M}$ consists of functions which are linear combinations of functions from $\mathcal{F}$, the complexity of $\mathcal{M}$ will depend on the size of the linear combination and the complexity of $\mathcal{F}$.

Once again we can obtain a PAC-style result using these bounds.

Corollary 3.5 *Assume that data is drawn IID from a distribution $\mathcal{P}$ on X with $y = g(x)$ for some function $g : X \to \left[\frac{-B}{2}, \frac{B}{2}\right]$, that the base regression functions $f \in \mathcal{F}$ returned by the base learner map to $[-1, +1]$ and satisfy $\epsilon > \epsilon_{min}$, that the base learner runs in time polynomial in $m, 1/\epsilon$, and that Pdim$(\mathcal{F})$ is finite. Then $\exists m(\rho, \delta, \eta, \gamma)$ a polynomial in $1/\rho, 1/\delta, 1/\eta, 1/\gamma$ such that the following holds for all $\rho, \delta, \eta, \gamma \in (0, 1)$: with probability at least $1 - \delta$, if trained on a sample $x = (x_1, x_2, \ldots, x_m) \in \mathcal{P}^m$, then* EXPLEV *produces a hypothesis F, in time polynomial in $1/\rho, 1/\delta, 1/\eta, 1/\gamma$, satisfying*

$$\mathcal{P}\{x : |F(x) - g(x)| < \eta + \gamma\} > 1 - \rho$$

for all $m > m(\rho, \delta, \eta, \gamma)$.

This result shows that EXPLEV is an efficient regression algorithm for obtaining functions that interpolate a target to arbitrarily high accuracy. The result follows from results in Section 4 which are somewhat more precise.

3.3 Some Re-Labelings are Benign

All of the results discussed in this section require assumptions on the base learners. Although we consider our definitions of the edge to be reasonable and our assumptions lower bounding these edges essential, a substantial weakness remains. The samples we supply to the base learner are modified, and it may appear that we are creating an impossible task. In general, no useful base learner can perform well with respect to arbitrarily relabeled examples. Here we discuss a special case which illustrates that the relabelings we use are not completely general. This special case shows that the relabelings we use can actually be quite benign.

In the PAC literature it is common to assume that the samples are labeled according to a function in the hypothesis

class. Under this assumption many hypothesis classes will allow our algorithms to use benign relabelings. We restrict our attention to base regression functions here, although a somewhat more involved argument may be used for base classifiers.

Theorem 3.6 *Assume that the data are labeled by a linear combination of functions from a class $\mathcal{F}$ that is closed under negation and has finite pseudo-dimension $Pdim(\mathcal{F})$. Then all the modified samples used by* SQUARELEV.R *are consistent with some linear combination of functions in $\mathcal{F}$.*

Proof: Let g be the function labeling the data. The master hypothesis F is a linear combination of functions in $\mathcal{F}$ by definition. The residuals $r(x_i) = g(x_i) - F(x_i)$ are therefore consistent with the function $g - F$, which is a linear combination of functions in $\mathcal{F}$. □

A simple example of such a class is the finite class of monomials of degree up to k with coefficients in $\{-1, 1\}$. The class of linear combinations of these functions has finite pseudo-dimension $k+1$. Therefore, Theorem 3.6 shows that with a base learner using this finite base hypothesis class, SQUARELEV.R creates samples that are consistent with the original target class. A relatively trivial argument can be used to show that, while any residuals are larger than zero, there always exists a monomial with an edge relative to such a relabeling.

Theorem 3.6 is quite general and provides some evidence that the assumptions we place on the base learner are not overly strong.

4 Proofs of Main Results

In this section we prove the main results described in the previous section. These proofs proceed similarly the original proofs for AdaBoost [13]. We begin by using an amortized analysis on the potential to bound the time required to achieve low error on the sample. This is done in two steps, the first bounds the decrease in the potential in a single iteration, the second iterates this bound. We then bound the size of the coefficients of the final hypothesis. Using these bounds we can bound the generalization error using standard results from statistical learning theory [1].

4.1 Performance of SQUARELEV.R

The following theorem shows how the potential (2) decreases each iteration. The value of α in the theorem minimizes the potential of $\mathbf{F} + \alpha\mathbf{f}$.

Theorem 4.1 *If ϵ is the edge (3) of the base function f in an iteration of* SQUARELEV.R *then the potential P_{var} decreases by a factor of $(1 - \epsilon^2)$ during the iteration.*

Proof: Let P_{var}, F, and $\mathbf{r}$ be the potential, master function, and residual vector at the start of the iteration and P'_{var}, $F' = F + \alpha f$, and $\mathbf{r}' = \mathbf{r} - \alpha\mathbf{f}$ be the corresponding quantities at the end of the iteration. Recall that this potential $P_{var} = ||\mathbf{r} - \bar{\mathbf{r}}||_2 = (\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{r} - \bar{\mathbf{r}})$ and this edge $\epsilon = \frac{(\mathbf{r}-\bar{\mathbf{r}})\cdot(\mathbf{f}-\bar{\mathbf{f}})}{||\mathbf{r}-\bar{\mathbf{r}}||_2||\mathbf{f}-\bar{\mathbf{f}}||_2}$.

$$
\begin{aligned}
P'_{var} &= ||\mathbf{r}' - \bar{\mathbf{r}'}||_2^2 \\
&= ||(\mathbf{r} - \bar{\mathbf{r}}) - \alpha(\mathbf{f} - \bar{\mathbf{f}})||_2^2 \\
&= ||\mathbf{r} - \bar{\mathbf{r}}||_2^2 - 2\alpha((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}})) + \alpha^2||\mathbf{f} - \bar{\mathbf{f}}||_2^2 \\
&= P_{var} - 2\alpha((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}})) + \alpha^2||(\mathbf{f} - \bar{\mathbf{f}})||_2^2 \\
&= P_{var} - \frac{((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}}))^2}{||\mathbf{f} - \bar{\mathbf{f}}||_2^2} \\
&\quad \text{(by setting } \alpha = \tfrac{((\mathbf{r}-\bar{\mathbf{r}})\cdot(\mathbf{f}-\bar{\mathbf{f}}))}{||\mathbf{f}-\bar{\mathbf{f}}||_2^2} \text{)} \\
&= P_{var}\left(1 - \frac{((\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{f} - \bar{\mathbf{f}}))^2}{||\mathbf{r} - \bar{\mathbf{r}}||_2^2||\mathbf{f} - \bar{\mathbf{f}}||_2^2}\right) \\
&= P_{var}(1 - \epsilon^2) .
\end{aligned}
$$

□

The next theorem iterates this result to bound the number of iterations before $\widehat{\text{er}}_S(\tilde{F}) \leq \rho$.

Theorem 4.2 *Assume that each $y_i \in \left[-\frac{B}{2}, \frac{B}{2}\right]$. If the edges of the weak hypotheses used by* SQUARELEV.R *are bounded below by $\epsilon_{min} > 0$ then for all $\rho \in (0, 1)$ after*

$$T = \left\lceil \frac{\ln\left(\frac{B^2}{4\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil$$

iterations the function $\tilde{F}_T$ has sample error $\widehat{er}_S(\tilde{F}_T) \leq \rho$. If in addition $\frac{||\mathbf{r}-\bar{\mathbf{r}}||_2}{||\mathbf{f}-\bar{\mathbf{f}}||_2} \leq c$ in every iteration then

$$\sum_{t=1}^{T} \alpha_t \leq c \left\lceil \frac{\ln\left(\frac{B^2}{4\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil .$$

Proof: Let $P_{var,T}$ (and $\mathbf{r}_T$, $\bar{r}_T$, F_T) be the potential (and residuals, average residual, master function) at the end of iteration T. If $P_{var,T} \leq m\rho$ then $\sum_{i=1}^{m}(r_{T,i} - \bar{r}_T)^2 \leq m\rho$ and, for $\tilde{F}(x) = F(x) + \frac{1}{m}\sum_{i=1}^{m}(y_i - F_T(x_i))$, we have that $\widehat{\text{er}}_S(\tilde{F}) \leq \rho$. Furthermore, since the initial potential is at most $m(B/2)^2$ and the potential decreases by at least $(1 - \epsilon_{min}^2)$ each iteration, the potential at the end of iteration T is at most $mB^2(1 - \epsilon_{min}^2)^T/4$. Thus the sample error of $\tilde{F}(x)$ is at most ρ when

$$
\begin{aligned}
mB^2(1 - \epsilon_{min}^2)^T/4 &\leq m\rho \\
T\ln(1 - \epsilon_{min}^2) &\leq \ln\left(\frac{4\rho}{B^2}\right) \\
T &\geq \frac{\ln\left(\frac{B^2}{4\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)}
\end{aligned}
$$

proving the first part of the theorem. On each iteration:

$$\alpha = \epsilon\frac{||\mathbf{r} - \bar{\mathbf{r}}||_2}{||\mathbf{f} - \bar{\mathbf{f}}||_2} \leq c .$$

Multiplying this bound by T gives the second part of the theorem. □

This result shows that SQUARELEV.R takes only polynomial time to obtain a rule with small error on the sample, and low complexity. We now use these facts to derive a bound on the generalization error of the final hypothesis in the following theorem.

Theorem 4.3 *Assume that data is drawn IID from a distribution $\mathcal{P}$ on $X \times \left[-\frac{B}{2}, \frac{B}{2}\right]$ (for some domain X), $\mathcal{F}$ is a class of $[-1,1]$-valued functions with pseudo-dimension Pdim$(\mathcal{F}) = q$, and that each iteration the base regressor returns an $f \in \mathcal{F}$ such that the edge (3) of f is bounded below by ϵ_{min} and $\frac{||\mathbf{r}-\bar{\mathbf{r}}||_2}{||\mathbf{f}-\bar{\mathbf{f}}||_2} \leq c$. Then there exists a constant $A \geq 0$ such that, for all $\rho, \delta \in (0,1)$, if* SQUARELEV.R *is applied to any sample, drawn IID from $\mathcal{P}$ of size at least*

$$m(\rho,\delta) = \left(\frac{AK^4}{\rho^2}\right)\left(\ln\left(\frac{1}{\delta}\right) + \textit{Pdim}(\mathcal{F})\left\lceil\frac{K^4}{\rho^2}\right\rceil \ln\left(\frac{K^8\left\lceil\frac{K^4}{\rho^2}\right\rceil}{\rho^3}\right)\right)$$

where $K = 2\max\left(c\frac{\ln(B^2/2\rho)}{\ln(1/(1-\epsilon_{min}^2))}, B\right)$, *then with probability at least $1-\delta$ the (shifted) master hypothesis $\tilde{F}_T$ after* $T = \left\lceil \frac{\ln\left(\frac{B^2}{2\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil$ *iterations has $er_{\mathcal{P}}(\tilde{F}_T) < \rho$.*

Proof: Fix ρ and δ, and let S be an IID $m \geq m(\rho,\delta)$ sample. After $T = \left\lceil \frac{\ln\left(\frac{B^2}{2\rho}\right)}{\ln\left(\frac{1}{1-\epsilon_{min}^2}\right)} \right\rceil$ iterations the sample error $\widehat{\text{er}}_S(\tilde{F}_T) \leq \frac{\rho}{2}$ from Theorem 4.2. We must bound the probability that the expected squared error is much larger than this. To use the standard theorems we must first scale the y values and final hypothesis so they lie in the interval $[0,1]$. In particular, set $F'(x) = \frac{\tilde{F}(x)}{K} + \frac{1}{2}$ and $S' = \{(x_1, \frac{y_1}{K}+\frac{1}{2}), (x_2, \frac{y_2}{K}+\frac{1}{2}), \ldots, (x_m, \frac{y_m}{K}+\frac{1}{2})\}$. We use $\text{er}_{\mathcal{P}'}(F')$ to denote the expected error $\mathbf{E}((\frac{y}{K}+\frac{1}{2}) - F'(x))^2$. We use $\mathcal{M}$ to denote the original class of master functions $\tilde{F}$ and use $\mathcal{M}'$ for the transformed class of functions to which F' belongs.

The following are equivalent

$$|\text{er}_{\mathcal{P}'}(F') - \widehat{\text{er}_{S'}}(F')| \geq \rho$$

$$\left|\mathbf{E}\left(\frac{y}{K}+\frac{1}{2}-F'(x)\right)^2 - \frac{1}{m}\sum_{i=1}^m\left(\frac{y_i}{K}+\frac{1}{2}-F'(x_i)\right)^2\right| \geq \rho$$

$$\left|\mathbf{E}\left(y-\tilde{F}(x)\right)^2 - \frac{1}{m}\sum_{i=1}^m\left(y_i-\tilde{F}(x_i)\right)^2\right| \geq \rho K^2$$

$$|\text{er}_{\mathcal{P}}(\tilde{F}) - \widehat{\text{er}_S}(\tilde{F})| \geq \rho K^2 \ .$$

So that

$$\begin{aligned}
&\mathcal{P}^m\left\{\exists F \in \mathcal{M} : |\text{er}_{\mathcal{P}}(F) - \widehat{\text{er}_S}(F)| \geq \frac{\rho}{2}\right\} \\
&= \mathcal{P}^m\left\{\exists F' \in \mathcal{M}' : |\text{er}_{\mathcal{P}'}(F') - \widehat{\text{er}_{S'}}(F')| \geq \frac{\rho}{2K^2}\right\} \\
&\leq 4\mathcal{N}_1\left(\frac{\rho}{2^5K^2}, \mathcal{M}', 2m\right)\exp\left(\frac{-\rho^2 m}{2^7K^4}\right) \\
&\leq 4\left(\frac{2^8K^4em}{q\rho}\right)^{q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil}\exp\left(\frac{-\rho^2m}{2^7K^4}\right) \ .
\end{aligned}$$

Where the first inequality follows from Theorem 17.1 in Anthony and Bartlett [1] and the second from Lemma 4.11 and the fact that $\mathcal{N}_1(\epsilon,\mathcal{F},k) < \mathcal{N}_2(\epsilon,\mathcal{F},k)$. This probability will be less than δ if

$$\begin{aligned}
\ln\left(\frac{\delta}{4}\right) &> q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil\ln\left(\frac{2^8K^4em}{\rho q}\right) - \left(\frac{\rho^2 m}{2^7K^4}\right) \\
\frac{\rho^2 m}{2^7K^4} &> \ln\left(\frac{4}{\delta}\right) + q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil\ln\left(\frac{2^8K^4em}{\rho q}\right) \quad (12)
\end{aligned}$$

Since $\ln(a) \leq ab + \ln(1/b) - 1 \ \forall a,b > 0$, we have $ab \geq \ln(a) + \ln(b) + 1$ and so

$$\frac{m\rho^2}{2^8K^4q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil} \geq \ln(m) + \ln\left(\frac{\rho^2}{2^8K^4q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil}\right) + 1$$

$$\frac{m\rho^2}{2^8K^4} \geq q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil\ln\left(\frac{\rho^2}{q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil 2^8K^4}\right) \ . \quad (13)$$

Subtracting 13 from 12 shows that m is large enough if

$$\frac{m\rho^2}{2^8K^4} \geq \ln\left(\frac{4}{\delta}\right) + q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil\ln\left(\frac{2^{16}eK^8\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil}{\rho^3}\right)$$

as required. □

4.2 Performance of SQUARELEV.C

SQUARELEV.C is very similar to SQUARELEV.R, therefore we only derive a single iteration bound on the reduction in potential obtained. This result may be used in the same way as Theorem 4.1 to obtain generalization error results.

Theorem 4.4 *If ϵ is the edge (5) of the base function f in an iteration of* SQUARELEV.C *then the potential P_{sq} decreases by a factor of $(1-\epsilon^2)$ during the iteration.*

Proof: Consider the change in potential for a single iteration, with primes indicating the modified quantities at the end of the iteration. Recall that $\mathbf{r} = \mathbf{y} - \mathbf{F}$, $P_{sq} = ||\mathbf{r}||_2^2$, and $\epsilon = (\mathbf{r}\cdot\mathbf{f})/||\mathbf{r}||_2||\mathbf{f}||_2$.

$$\begin{aligned}
P'_{sq} &= ||\mathbf{y}-\mathbf{F}'||_2^2 \\
&= ||(\mathbf{y}-\mathbf{F}) - \alpha\mathbf{f}||_2^2 \\
&= P_{sq} - 2\alpha(\mathbf{r}\cdot\mathbf{f}) + \alpha^2||\mathbf{f}||_2^2 \\
&= P_{sq} - \frac{(\mathbf{r}\cdot\mathbf{f})^2}{||\mathbf{f}||_2^2} \\
&\qquad \text{using the minimizing } \alpha = \frac{(\mathbf{r}\cdot\mathbf{f})}{||\mathbf{f}||_2^2} \\
&= P_{sq}\left(1 - \frac{(\mathbf{r}\cdot\mathbf{f})}{||\mathbf{r}||_2^2||\mathbf{f}||_2^2}\right) \\
&= P_{sq}(1-\epsilon^2) \ .
\end{aligned}$$

□

4.3 Performance of EXPLEV

We now turn to the results on EXPLEV. We overload the notation and define $P_{exp}(c) = \exp(c) + \exp(-c) - 2$ for scalar c, and use $P_{exp}^{-1}(\cdot)$ for this function's (non-negative) inverse. Since the residuals depend on the master function F, so does the potential $P_{exp}(\mathbf{r}) = \sum_{i=1}^{m} (e^{sr_i} + e^{-sr_i} - 2)$. Recall that $\nabla P_{exp}(\mathbf{r}) = -s\exp(sr_i) + s\exp(-sr_i)$, the gradient of the potential with respect to $\mathbf{F}$.

To start we need to bound the progress obtained by EXPLEV in a single iteration. Our proof bounding this progress uses certain extreme residual vectors, those having a fixed $P_{exp}(\mathbf{r})$ and minimizing $||\nabla P_{exp}(\mathbf{r})||_1$. As shown in Lemma 4.6 these vectors are the $\mathbf{r}^{(c)}$ in following definition.

Definition 4.5 *For $c \geq 0$, let $\mathbf{r}^{(c)} = (P_{exp}^{-1}(c), 0, \ldots, 0)$ and $S_c = \{\mathbf{r} \in \Re^m | P_{exp}(\mathbf{r}) = c\}$.*

Lemma 4.6 $\inf_{\mathbf{r} \in S_c} ||\nabla P_{exp}(\mathbf{r})||_1 = ||\nabla P_{exp}(\mathbf{r}^{(c)})||_1$, *for all $c \geq 0$.*

Proof: The proof appears in the full version of this paper [7]. □

To bound the rate of decrease for this worst case vector of residuals we require the following technical Lemma.

Lemma 4.7 *If $P_{exp}(\mathbf{r}) \geq m + \frac{1}{m} - 2$, $m \geq 3$, and $\epsilon < 1$ then the quantity*

$$\frac{\sqrt{(P_{exp}(\mathbf{r}) + 2m)^2 - \left(||\nabla P_{exp}(\mathbf{r})||_1 \epsilon / s\right)^2} - 2m}{P_{exp}(\mathbf{r})} \leq \left(1 - \frac{1}{6}\epsilon^2\right) .$$

Proof: The proof appears in the full version of this paper [7]. □

We are now ready to prove the main invariant for EXPLEV. The proof of Theorem 4.8 shows that the choice of α used by EXPLEV minimizes an upper bound on the potential, and thus EXPLEV is not exactly maximizing the decrease in potential each iteration.

Theorem 4.8 *If $m \geq 3$, $P_{exp} \geq m + \frac{1}{m} - 2$ at the start of an iteration of* EXPLEV*, and ϵ is the edge of the base function at that iteration then the stepsize $\alpha < \frac{1}{2s} \ln\left(\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}\right) \leq \frac{1}{2s} \ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right)$ and the potential P_{exp} decreases by at least a factor of $(1 - \frac{\hat{\epsilon}^2}{6})$ during the iteration.*

Proof: The proof is structured to motivate the choice of α used by EXPLEV. We define $Q = P_{exp} + 2m$ and relate the potential Q' at the end of an iteration to the potential Q at the beginning of the iteration as follows. Q' is

$$\begin{aligned}
&\sum_i \exp(s(r_i - \alpha f(x_i))) + \exp(-s(r_i - \alpha f(x_i))) \\
&= \sum_i \exp(sr_i)\beta^{-f(x_i)} + \exp(-sr_i)\beta^{f(x_i)} \\
&\quad \text{(setting } \beta = \exp(s\alpha) > 1\text{)} \\
&= \frac{1}{\beta}\sum_i \exp(sr_i)(\beta^2)^{\frac{1-f(x_i)}{2}} + \exp(-sr_i)(\beta^2)^{\frac{1+f(x_i)}{2}} \\
&\leq \frac{1}{\beta}\sum_i \exp(sr_i)\left(1 - (1-\beta^2)\left(\frac{1-f(x_i)}{2}\right)\right) \\
&\quad + \exp(-sr_i)\left(1 - (1-\beta^2)\left(\frac{1+f(x_i)}{2}\right)\right) \qquad (14) \\
&= \frac{1+\beta^2}{2\beta} Q \\
&\quad + \frac{1-\beta^2}{2\beta}\sum_i \exp(sr_i)f(x_i) - \exp(-sr_i)f(x_i) \\
&= \frac{1+\beta^2}{2\beta} Q + \frac{1-\beta^2}{2\beta s}||\nabla P_{exp}||_1 \sum_{i=1}^{m} D(x_i)\tilde{y}_i f(x_i) \\
&= Q\left(\frac{1+\beta^2}{2\beta}\right) + \left(\frac{1-\beta^2}{2\beta s}\right)||\nabla P_{exp}||_1 \epsilon \\
&\leq Q\left(\frac{1+\beta^2}{2\beta}\right) + \left(\frac{1-\beta^2}{2\beta s}\right)||\nabla P_{exp}||_1 \hat{\epsilon} . \qquad (15)
\end{aligned}$$

Now minimizing the right hand side with respect to $\beta > 1$ yields

$$\begin{aligned}
\beta &= \frac{\sqrt{(sQ - ||\nabla P_{exp}||_1\hat{\epsilon})\,(sQ + ||\nabla P_{exp}||_1\hat{\epsilon})}}{(sQ - ||\nabla P_{exp}||_1\hat{\epsilon})} \\
&= \sqrt{\frac{(sQ + ||\nabla P_{exp}||_1\hat{\epsilon})}{(sQ - ||\nabla P_{exp}||_1\hat{\epsilon})}} \text{ so} \qquad (16) \\
\alpha &= \frac{1}{s}\ln\sqrt{\frac{(sQ + ||\nabla P_{exp}||_1\hat{\epsilon})}{(sQ - ||\nabla P_{exp}||_1\hat{\epsilon})}} .
\end{aligned}$$

Since $Q = P_{exp} - 2m$, this is exactly the α used by EXPLEV. Note that $sQ > ||\nabla P_{exp}||_1$, so $\beta < \frac{\sqrt{1+\hat{\epsilon}}}{\sqrt{1-\hat{\epsilon}}}$ and $\alpha < \frac{1}{2s}\ln\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}$ proving the first claim.

We continue by substituting (16) into (15) , simplifying, and subtracting $2m$ from each side, giving

$$P'_{exp} \leq \sqrt{Q^2 - \left(||\nabla P_{exp}||_1\hat{\epsilon}/s\right)^2} - 2m .$$

To obtain the factor by which the potential decreases we divide both sides by P_{exp}.

$$\begin{aligned}
\frac{P'_{exp}}{P_{exp}} &\leq \frac{\sqrt{Q^2 - \left(||\nabla P_{exp}||_1\hat{\epsilon}/s\right)^2} - 2m}{P_{exp}} \\
&= \frac{\sqrt{(P_{exp} + 2m)^2 - \left(||\nabla P_{exp}||_1\hat{\epsilon}/s\right)^2} - 2m}{P_{exp}}
\end{aligned}$$

Now Lemmas 4.6 and 4.7 show that if $P_{exp} \geq m + \frac{1}{m} - 2$ then

$$\frac{P'_{exp}}{P_{exp}} \leq 1 - \frac{\hat{\epsilon}^2}{6} .$$

Thus the potential decreases by a factor of $1 - \hat{\epsilon}^2/6$ per iteration as required. □

We iterate this bound to obtain the following bound on the number of iterations required to achieve small error on the sample, and to bound the size of the α's.

Theorem 4.9 *If ϵ_{min} is a lower bound on the edges of the base functions and each $y_i \in \left[-\frac{B}{2}, \frac{B}{2}\right]$ then for all $\eta \in (0,1)$* EXPLEV *with $s = \ln(m)/\eta$ and $\epsilon_{max} \geq \epsilon_{min}$ achieves $|y_i - F(x_i)| < \eta$ within*

$$T = \left\lceil \frac{\ln(m)\left(1+\frac{B}{\eta}\right)}{\ln\left(\frac{1}{\left(1-\frac{1}{6}\epsilon_{min}^2\right)}\right)} \right\rceil$$

iterations. Furthermore,

$$\sum_{t=1}^{T} \alpha_t \leq (B+\eta+1)\frac{\ln\frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\ln\frac{1}{1-\epsilon_{min}^2/6}} .$$

Proof: If $P_{exp} \leq \exp(s\eta)+\exp(-s\eta)-2$ then each $r_i \leq \eta$ and every $F(x_i)$ is within η of the corresponding y_i. Theorem 4.8 implies that the potential drops by at least a factor of $1-\epsilon_{min}^2/6$ each iteration. Since the initial potential is less than me^{sB}, after T iterations the potential is at most $me^{sB}(1-\epsilon_{min}^2/6)^T$.

Solving $me^{sB}(1-\epsilon_{min}^2/6)^T \leq \exp(s\eta)+\exp(-s\eta)-2$ for T gives that after

$$T = \left\lceil \frac{\ln(m) + sB - \ln(e^{s\eta}+e^{-s\eta}-2)}{\ln(\frac{1}{1-\epsilon_{min}^2/6})} \right\rceil$$

iterations every residual is at most η.

For the second claim, Theorem 4.8 shows that each $\alpha_t \leq \frac{\eta}{\ln(m)}\ln\sqrt{\frac{1+\epsilon_{max}}{1-\epsilon_{max}}}$. Therefore, after T iterations the sum of the α values is at most

$$\left\lceil \frac{(1+B/\eta)\ln(m)}{\ln\frac{1}{1-\epsilon_{min}^2/6}} \right\rceil \frac{\eta}{\ln(m)}\ln\sqrt{\frac{1+\epsilon_{max}}{1-\epsilon_{max}}} \leq \frac{(B+\eta+1)\ln\sqrt{\frac{1+\epsilon_{max}}{1-\epsilon_{max}}}}{\ln\frac{1}{1-\epsilon_{min}^2/6}}$$

as required. □

These results show that EXPLEV will produce a combined function in polynomial time that approximately interpolates the sample to arbitrarily high accuracy. This together with the bound on the α's can be used to obtain the following bound on the generalization error of the final hypothesis produced by EXPLEV.

Theorem 4.10 *Assume that the data is drawn IID from a distribution $\mathcal{P}$ on X with $y = g(x)$ for some function $g : X \to \left[\frac{-B}{2}, \frac{B}{2}\right]$, that the base regression functions $f \in \mathcal{F}$ returned by the base learner map to $[-1,+1]$ and have edges (11) at least ϵ_{min}, and that Pdim$(\mathcal{F}) = q$ is finite. Then $\exists c > 0$ such that the following holds for all $\rho, \delta, \eta, \gamma \in (0,1)$ with probability at least $1-\delta$: if trained on a sample S of size m at least*

$$m(\rho,\delta,\eta,\gamma) = 3W\ln^2(W)$$

then after $T = \left\lceil \frac{(1+B/\eta)\ln(m)}{\ln\left(\frac{1}{1-\epsilon_{min}^2/6}\right)} \right\rceil$ iterations EXPLEV *with parameters $s = \ln(m)/\eta$ and $\epsilon_{max} > \epsilon_{min}$ produces F_T satisfying*

$$\mathcal{P}\{x : |F_T(x) - g(x)| < \eta+\gamma\} > 1-\rho$$

where

$$W = \frac{c}{\rho}\left(\ln\left(\frac{1}{\delta}\right) + \left\lceil\frac{K^4}{\gamma^2}\right\rceil q \ln\left(\frac{K^2}{\gamma q}\right)\ln^2\left(\frac{\left\lceil\frac{K^4}{\gamma^2}\right\rceil q}{\rho\gamma}\ln\left(\frac{K^2}{\gamma q}\right)\right)\right)$$

and $K = \frac{(B+\eta+1)\ln\sqrt{\frac{1+\epsilon_{max}}{1-\epsilon_{max}}}}{\ln\frac{1}{1-\epsilon_{min}^2/6}} \geq ||\alpha||_1$.

Proof: The proof appears in the full version of this paper [7]. □

The following key Lemma relates the complexity of the master function class to the complexity of the base hypothesis class and the size of the coefficients used.

Lemma 4.11 *Suppose $\mathcal{F}$ is a class of $[-1,1]$-valued functions defined on a set X, and the covering number $\mathcal{N}_2(\epsilon,\mathcal{F},m)$ is finite for all $m \in \mathbb{N}$ and $\epsilon > 0$. Suppose in addition that $\mathcal{F} = -\mathcal{F}$ and $\mathcal{F}$ contains the zero function. For $V \geq 1$ define*

$$\mathcal{M} = \left\{ \sum_{i=1}^{N} w_i f_i : N \in \mathbb{N}, f_i \in \mathcal{F}, \sum_{i=1}^{N} |w_i| \leq V \right\} .$$

Then, for $m \geq fat_{\mathcal{M}}(16\epsilon)$,

$$\begin{aligned} fat_{\mathcal{M}}(16\epsilon) &\leq 8\ln\mathcal{N}_2(\epsilon,\mathcal{M},m) \\ &\leq 8\left\lceil\frac{4V^2}{\epsilon^2}\right\rceil \ln\mathcal{N}_2\left(\frac{\epsilon}{2V},\mathcal{F},m\right) \\ &\leq 8\left\lceil\frac{4V^2}{\epsilon^2}\right\rceil Pdim(\mathcal{F})\ln\left(\frac{em4V^2}{\epsilon Pdim(\mathcal{F})}\right) . \end{aligned}$$

Proof: The lemma follows from Theorems 12.10, 14.14 and 12.2 in Anthony and Bartlett [1]. □

5 Conclusions

In this paper we present three leveraging algorithms for the regression setting. We give progress guarantees and generalization bounds that depend on the good behavior of the base regressors. The only regression algorithms that we are aware of with similar bounds are AdaBoost.R [13] and Lee, Bartlett and Williamson's Construct algorithm [1]. The bounds given for AdaBoost.R require the base learner to perform well with respect to a changing loss. The bounds for the Construct algorithm are agnostic in flavor and appear stronger than the bounds we are so far able to show, however, they assume that the base learner returns an almost optimal $f \in \mathcal{F}$. Although, our bounds also rely on assumptions on the base learners, we feel that these assumptions may be more reasonable. In particular, the base regression functions only need to be slightly better than random guessing (in some sense). However, even these assumptions seem strong when the sample fed to the base learner is modified. Perhaps future research will be able to weaken our assumptions and identify conditions ensuring that the constructed samples fall into the base regressors area of competence. In the meantime, it appears that empirical work is needed to determine how well base learners respond to the modified data.

The generalization bounds we derive for SQUARELEV.R and EXPLEV are of very different flavors. In particular, the bound on EXPLEV has a considerably stronger form than that for SQUARELEV.R. The key to obtaining these bounds is bounding $\sum_{t=1}^{T} \alpha_t$. We found it surprising that for EXPLEV, with the appropriate scale factor, the sum of αs depends weakly on the desired accuracy η, while the number of iterations grows as $1/\eta$.

Acknowledgements

This work benefited greatly from discussions with Rob Schapire, Claudio Gentile, Manfred Warmuth and Yoram Singer. We would also like to thank Juergen Forster, Sandra Panizza and Gunnar Rätsch for helpful comments on an earlier version.

References

[1] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[2] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 1998.

[3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] Leo Breiman. Arcing the edge. Technical Report 486, Department of Statistics, University of California, Berkeley, 1997. Available at www.stat.berkeley.edu.

[5] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley, 1997. Available at www.stat.berkeley.edu.

[6] Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517, 1999.

[7] Nigel Duffy and David Helmbold. Leveraging for regression. Technical Report UCSC-CRL-00-11, University of California at Santa Cruz, 2000.

[8] Nigel Duffy and David Helmbold. Potential boosters? In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 258–264. MIT Press, 2000.

[9] Nigel Duffy and David P. Helmbold. A geometric approach to leveraging weak learners. In Paul Fischer and Hans Ulrich Simon, editors, *Computational Learning Theory: 4th European Conference (EuroCOLT '99)*, pages 18–33. Springer-Verlag, March 1999.

[10] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.

[11] Yoav Freund. An adaptive version of the boost-by-majority algorithm. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 102–113. ACM, 1999.

[12] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[13] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford University, 1998.

[15] Jerome H. Friedman. Greedy Function Approximation:A Gradient Boosting Machine. Technical report, Stanford University, 1999.

[16] Drucker H. Improving regressors using boosting techniques. In *Proceedings of the fourteenth International converence on machine learning*, pages 107–115. Morgan-Kaufman, 1997.

[17] Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 369–376. ACM Press, New York, NY, 1995.

[18] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 512–518. MIT Press, 2000.

[19] J. R. Quinlan. Bagging, Boosting and C4.5. In *Proceedings of the Thirteenth National Conference of Artificial Intelligence*, pages 725–730. AAAI Press and the MIT Press, 1996.

[20] Gunnar Rätsch, Takashi Onoda, and Klaus-R. Müller. Soft margins for adaboost. Technical Report NC-TR-1998-021, NeuroCOLT2, 1998.

[21] Gunnar Rätsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus R. Müller. Regularized boosting, barrier methods and regression. In *Proc. 13th Annu. Conf. on Comput. Learning Theory*, 2000.

[22] Greg Ridgeway, David Madigan, and Thomas Richardson. Boosting methodology for regression problems. In D. Heckerman and J. Whittaker, editors, *Proc. Artificial Intelligence and Statistics*, pages 152–161, 1999.

[23] Robert E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. MIT Press, 1992.

[24] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[25] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proc. 11th Annu. Conf. on Comput. learning Theory*, 1998. To appear in Machine Learning.

[26] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.