Machine Learning, 47, 153–200, 2002

Boosting Methods for Regression*

NIGEL DUFFY nigeduff@cse.ucsc.edu
DAVID HELMBOLD dph@cse.ucsc.edu
Computer Science Department, University of California, Santa Cruz, Santa Cruz, CA 95064, USA

Editor: Jyrki Kivinen

Abstract. In this paper we examine ensemble methods for regression that leverage or "boost" 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 strong theoretical guarantees. We present several gradient descent leveraging algorithms for regression and prove AdaBoost-style bounds on their sample errors using intuitive assumptions on the base learners. We bound the complexity of the regression functions produced in order to derive PAC-style bounds on their generalization errors. Experiments validate our theoretical results.

Keywords: learning, boosting, arcing, ensemble methods, regression, gradient descent

1. Introduction

In this paper we consider the following regression setting. Data is generated IID from an unknown distribution $\mathcal{P}$ on some domain X and labeled according to an unknown 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 sample may not be labeled perfectly by any function, but rather has random noise modifying 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 (Freund & Schapire, 1997; Bauer & Kohavi, 1999; Quinlan, 1996; Freund & Schapire, 1996) is one example in the classification setting, although its performance does degrade as the amount of noise increases.

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. If the sample is large enough with respect to the complexity of the class $\mathcal{F}$, function f will also have small error on the domain X with high probability (see e.g. Anthony & Bartlett, 1999).

Computationally efficient algorithms using this approach must obtain a simple and accurate hypothesis in a short time. There are many learning algorithms that efficiently

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

produce simple hypotheses, but the accuracy of these methods may be less than desirable. Leveraging techniques, such as boosting (Duffy & Helmbold, 2000; Freund, 1995; Freund, 1999; Freund & Schapire, 1997; Friendman, Hastie, & Tibshirani, 2000; Schapire, 1992) and Arcing (Breiman, 1998, 1999) 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 compared to the increase in complexity, then leveraging can improve not only the training error, but also the generalization performance.

Leveraging has been examined primarily in the classification setting where AdaBoost (Freund & Schapire, 1997) and related leveraging techniques (Breiman, 1996, 1998, 1999; Duffy & Helmbold, 1999; Friedman, Hastie, & Tibshirani, 2000; Rätsch, Onoda, & Müller, 2001) are useful for increasing the accuracy of base classifiers. These algorithms construct a linear combination of the hypotheses returned by the base algorithm, and have recently been viewed as performing gradient descent on a potential function (Breiman, 1999; Rätsch, Onoda, & Müller, 2001; Friedman, Hastie, & Tibshirani, 2000; Duffy & Helmbold, 1999; Mason et al., 2000). This viewpoint has enabled the derivation and analysis of new algorithms in the classification setting (Friedman, Hastie, & Tibshirani, 2000; Duffy & Helmbold, 1999, 2000; Mason et al., 2000). 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 (Friedman, 1999a).

We are aware of several other applications of gradient descent leveraging to the regression setting (Freund & Schapire, 1997; Friedman, 1999a; Lee, Bartlett, & Williamson, 1995; Rätsch, Onoda, & Müller, 2000; Breiman, 1998, 1999). These approaches are discussed in more detail in Section 2.

Several issues arise when analyzing and deriving gradient descent leveraging algorithms. First, the potential function must be chosen carefully; minimizing this potential must imply 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 (Freund & Schapire, 1997; Freund, 1995; Duffy & Helmbold, 1999; Freund, 1999). This paper examines several potential functions for leveraging in the regression setting and proves performance bounds for the resulting gradient descent algorithms. Second, assumptions need to be made about the performance of the base learners in order to derive bounds. 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 has a major impact on the difficulty of proving performance bounds. In this paper, we attempt to use weak assumptions on the base learners and intuitive measures of "usefulness." 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 two main contributions of this work are the first direct application of AdaBoost-style analysis to the regression setting and the development of a general methodology leading to three principled leveraging algorithms for regression. We also present several important observations about how the regression setting differs from classification. For example, one of our algorithms exhibits an interesting tradeoff between its computation time and the size of the coefficients multiplying the base hypotheses. Note that unlike classification, the standard generalization bounds for regression depend on the size of these coefficients. This tradeoff is discussed further in Section 3.

One key difference between classification and regression is that base learners in the classification setting can be forced to return useful hypotheses simply by manipulating the distribution over the sample. This is not true for regression, where leveraging algorithms must also modify the sample in some other way (see Remark 2.1). In particular, our algorithms modify the labels in the sample. However, no useful base learner can perform well with respect to an arbitrarily labeled sample. In Section 2.3, we show that the relabeling method used by our algorithms is not arbitrary, often creating samples that are consistent with the original target class. Experiments described in Section 5 show that, on real world data sets, the base learners maintain a significant edge for a large number of iterations, despite the relabelings used by our algorithms.

Throughout the paper we use the following notation. The leveraging algorithm is given a set S of m training examples, $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Each iteration of the leveraging process the algorithm modifies the sample S to produce $\tilde{S} = \{(x_1, \tilde{y}_1), \ldots, (x_m, \tilde{y}_m)\}$ with the same x's but possibly different $\tilde{y}$ values. The leveraging 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 positive coefficient[1] α for the new base function and updates its master regression function to $F + \alpha f$. For a master regression function F, the *residuals* are $r_i = y_i - F(x_i)$ for $1 \leq i \leq m$. We often use bold face as abbreviations for vectors over the sample, e.g. $\mathbf{f} = (f(x_1), \ldots, f(x_m))$, $\mathbf{y} = (y_1, \ldots, y_m)$, and $\mathbf{r} = (r_1, \ldots, r_m)$.

We use three different potentials to derive leveraging algorithms for regression. Two of these potential functions are variants of the squared error and the third is 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 bounded away from 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

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

Both of these constraints on the base regressor are similar to those assumed by the GeoLev algorithm (Duffy & Helmbold, 1999) and analogous to those for AdaBoost (Freund & Schapire, 1997).

A third algorithm ExpLev performs gradient descent on the following exponential criterion: $P(\mathbf{r}) = \sum_i \exp(sr_i) + \exp(-sr_i) - 2$ where s is a scaling factor. This is a two-sided version of AdaBoost's exp(−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 of $P(\mathbf{r})$ 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 (Anthony & Bartlett, 1999). Better bounds can be proved when ExpLev is run in stages, with the maximum residual reduced by a constant factor each stage. Other variations on ExpLev, including one appropriate for base regressors, are also discussed.

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)-style 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. We truncate large descent steps for ExpLev and its staged version.

The remainder of the paper is organized as follows. The following section gives a brief overview of related work and contrasts the classification and regression settings. In Section 3 we give our methodology and describe our algorithms and their performance bounds. Section 4 contains the formal statements and proofs of our main results. We describe experiments validating our assumptions and theoretical bounds on the sample error in Section 5 and concluding remarks appear in Section 6.

2. Relation to other work

2.1. *Classification*

Leveraging techniques are ensemble methods that 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 the base components. Leveraging methods include ARCing (Breiman, 1998, 1999), Bagging (Breiman, 1996), and Boosting (Schapire, 1992; Freund, 1995; Freund & Schapire, 1997; Freund, 1999; Friedman, Hastie, & Tibshirani, 2000; Duffy & Helmbold, 2000). Leveraging for classification has received considerable attention (Freund & Schapire, 1997; Breiman, 1996; Freund, 1995; Bauer & Kohavi, 1999; Quinlan, 1996; Freund & Schapire, 1996; Schapire & Singer, 1999; Rätsch, Onoda, & Müller, 2001; Mason et al., 2000) and it has been observed that many of these algorithms perform an approximate gradient descent of some potential (Breiman, 1999; Rätsch, Onoda, & Müller, 2001; Friedman, Hastie, & Tibshirani, 2000; Duffy & Helmbold, 1999; Mason et al., 2000). 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 (Freund & Schapire, 1997) algorithm for classification. In addition to its empirical success (Bauer & Kohavi, 1999; Quinlan, 1996; Freund & Schapire, 1996), AdaBoost has strong theoretical guarantees (Freund & Schapire, 1997; Schapire et al., 1998) with reasonably weak assumptions on the base learners. Since we concentrate on deriving gradient descent leveraging algorithms for regression with similar guarantees, we first review some of the relevant concepts for classification.

In the (strong) PAC setting proposed by Valiant (1984), the data is drawn IID from an unknown distribution $\mathcal{P}$ on a domain X and labeled according to a concept from some known class. The learning algorithm is given accuracy and confidence requirements, and must output an accurate enough hypothesis (with respect to the same distribution $\mathcal{P}$) with at least the given confidence, regardless of the particular target concept and distribution $\mathcal{P}$. Efficient PAC learning algorithms are required to run in time polynomial in the parameters.

Weak PAC learning is like strong PAC learning, but the algorithm need only work for some particular accuracy and confidence, rather than all accuracies and confidences. Kearns and Valiant (1994) introduced weak PAC learning and asked if weak and strong PAC learning are equivalent. This question was answered in the affirmative by Schapire in his thesis (Schapire, 1992). See Kearns and Vazirani's book (Kearns & Vazirani, 1994) for a more detailed treatment of PAC learning.

An algorithm has the *PAC boosting property* if it can convert a weak PAC learner into a strong PAC learner. Freund and Schapire (1997) show that AdaBoost has the PAC boosting property. Of the many leveraging algorithms for classification, only a few are known to have the PAC boosting property (Duffy & Helmbold, 2000; Freund, 1995; 1999; Freund & Schapire, 1997; Schapire, 1992; Schapire & Singer, 1999).

Requiring a weak PAC learner to perform well with respect to an arbitrary distribution allows it to be manipulated by modifying the distribution $\mathcal{P}$. By careful modification of the distribution a PAC boosting algorithm can convert a weak PAC learner into a strong PAC learner. The PAC model balances this requirement that the algorithm perform well with respect to an arbitrary distribution, with an assumption that the labels are consistent with a concept from some known class. This consistency assumption is weakened in the agnostic setting, where the algorithm has to obtain a hypothesis that performs competitively with the best concept from a fixed class.[2]

The PAC boosting results rely on some strong assumptions: that the data is labeled consistently with a concept from some known concept class and that a weak PAC learner for

the problem is available. When these assumptions hold, algorithms with the PAC boosting property will efficiently leverage any weak PAC learner to obtain a hypothesis that is arbitrarily good. These assumptions may not hold for real world problems. However, PAC boosting algorithms often perform well in practice (Bauer & Kohavi, 1999; Quinlan, 1996; Freund & Schapire, 1996). To understand this one may take a more pragmatic view of the PAC boosting property. A boosting algorithm *leverages the properties of the base learner to which it is applied*. Pragmatically, this means that if the base learner behaves well on the given problem then the boosting algorithm will be able to efficiently leverage this performance. In practice, many of the bounds are of the form: if the base learner behaves reasonably well, then the master hypothesis will improve by a bounded amount. We take this more pragmatic view and use the term *leveraging algorithm* for an algorithm that can efficiently and effectively improve the performance of a base learner. Our theoretical results follow the PAC model, making broad intuitive assumptions about base learners to show general results for our leveraging algorithms. Although these assumptions may not hold in all cases, our bounds give guarantees whenever the base learner behaves appropriately.

2.2. *Regression*

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 (Freund & Schapire, 1997) attacks 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 (H, 1997; Ridgeway, Madigan, & Richardson, 1999; Bertoni, Campadelli, & Parodi, 1997) 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.

Recently, Friedman has explored regression using the gradient descent approach (Friedman, 1999a). 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 (Lee, Bartlett, & Williamson, 1995) which in turn is an extension of results of Jones (1992) and Barron (1993). Lee, Bartlett, and Williamson 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 independent work, Rätsch, Onoda, and Müller (2000) 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 when applied to a finite base hypothesis class. Although arrived at from a different direction, their ϵ-boost algorithm is very similar to our EXPITERLEV 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.

2.3. Sample relabeling

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 (Schapire, 1992; Freund, 1995; Freund & Schapire, 1997; Friedman, Hastie, & Tibshirani, 2000; Freund, 1999; Duffy & Helmbold, 2000) 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 given by the following observation:

Remark 2.1. 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 (1999) (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.

We know of three approaches to modifying the sample for gradient descent leveraging algorithms in the regression setting. 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 (Friedman, 1999b) 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 thus can use base regressors that solve classification problems. When using a base classifier, the master algorithm has the base method output a classifier that tends to separate those sample points on which the current master function is too high from those where it is too low.

Recall that the PAC setting assumes 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. As random guessing has average classification error 1/2, it is plausible 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. Our methods relabel the sample according to the residuals—the modified labels are either the signs of the residuals or the residuals themselves. To justify this modification of the sample, we now show that for many function classes this relabeling is benign.

Remark 2.2. Assume that the sample is labeled by a linear combination of functions from a class $\mathcal{F}$ of regression functions that is closed under negation. If the base regressor returns functions that are in $\mathcal{F}$ (or linear combinations of functions in $\mathcal{F}$) then any modified sample where the $\tilde{y}$ values are the residuals is also consistent with some linear combination of functions in $\mathcal{F}$.

Proof: Let g be the function labeling the data. We assume that the master hypothesis F is a linear combination of base hypotheses, and thus is a linear combination of functions in $\mathcal{F}$. 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 $\mathcal{F}$. □

This remark holds only when the $\tilde{y}$ values are the residuals $\mathbf{r}$, however, several of our algorithms use $\tilde{y}$ values that are the sign of the residuals and classes of base functions

mapping to $\{-1, 1\}$. In this case, the modified sample will not generally be consistent with any linear combination of functions from the base class. However, for all of our algorithms the edge of a base function f is positive if $\mathbf{r} \cdot \mathbf{f} > 0$. If the sample is labeled consistently with a linear combination of base functions, then while any residuals are larger than zero, there is always a function $f \in \mathcal{F}$ with positive edge. This can be seen by examining the positive quantity $(\mathbf{g} - \mathbf{F}) \cdot (\mathbf{g} - \mathbf{F})$. By writing the second $\mathbf{g} - \mathbf{F}$ as a sum of base functions, it is clear that for at least one base function f, $(\mathbf{g} - \mathbf{F}) \cdot \mathbf{f} > 0$. So even when the $\tilde{y}$'s are the signs of the residuals, there is still a function with an edge available to the base learner.

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, Remark 2.2 shows that with a base learner using this finite base hypothesis class, our relabeling methods create samples that are consistent with a polynomial and there will always be a monomial with a positive edge.

Although this argument justifies the assumption that the base learners can produce functions having positive edges on the relabeled sample, it does not guarantee that these edges remain bounded away from zero. However, our experimental results in Section 5 indicate that the edges often remain sufficiently large for the master regressor to make good progress.

3. Methodology and algorithms

In this section we describe our methodology and apply it to generate three different leveraging algorithms.

Our methodology begins by establishing the criteria for successful generalization, such as minimizing the expected squared error or maximizing the probability that the prediction is within an error tolerance of the true value. Once these criteria are known, we can then determine what behavior on the sample leads to the desired generalization performance. Anthony and Bartlett's book (Anthony & Bartlett, 1999) is a useful reference for theorems bounding the generalization error in terms of the performance on a sample.

The key step in our methodology is the construction of a "potential" that measures how close the current performance on the sample is to that desired. It is natural for the potential to be a non-negative function of the residual vector, $\mathbf{r}$ (recall that $r_i = y_i - F(x_i)$, the amount by which the master function must be increased in order to perfectly predict the label y_i), and go to zero as $\mathbf{r}$ goes to the zero vector. It will soon become clear that the potential must be chosen carefully so that it has several other properties.

The master algorithm attempts to reduce the residuals by adding in a multiple of the base hypothesis at each iteration. In essence, the master algorithm is performing an approximate gradient descent on the potential under the constraint that each descent step must be in the direction of the current base hypothesis (see Duffy & Helmbold, 1999 for further geometric intuition). Therefore, the current base hypothesis must be correlated with the direction of steepest descent—the negative gradient of the potential, with respect to the master function, on the sample.

The master algorithm must manipulate the sample so that any hypothesis with an "edge" is correlated with the negative gradient of the potential. It must also find a step size such that each iteration the potential decreases by a multiplicative factor depending on the amount of

"edge" in the base hypothesis. The way the modified sample is constructed, the potential function, the way the edge is measured, and the choice of step size must all be compatible in order to obtain this multiplicative decrease. Although any measure of "edge" can be used, we will use natural "edges" that are reasonable measures of success for a base regression algorithm.

The generalization bounds we are aware of require that the master function not only approach perfection on the sample, but also be relatively simple. The exponential decrease in the potential implies that relatively few iterations are needed before the performance on the sample is acceptable. However, the generalization bounds also require that the step sizes (i.e. coefficients multiplying the base hypotheses) are also small enough. Therefore we must show that our algorithms are not only computationally efficient but are also efficient with respect to the coefficients in the linear combinations produced. Letting the algorithm take a full gradient step might produce a coefficient that is prohibitively large and varying the size of the gradient steps taken can lead to a trade off between the number of iterations 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.

This amortized analysis of leveraging algorithms is reminiscent of that used in the classification setting (see Freund & Schapire, 1997; Duffy & Helmbold, 2000 for examples). First, the change in potential for a single iteration is bounded, leading to a bound on the potential after several iterations. Second, the bound on the potential is used to obtain 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 the "size" of the master hypothesis is usually measured by the number of base hypotheses, without worrying about the size of the coefficients. Note that the magnitude of the coefficients in the linear combination does appear in the margin analyses of boosting algorithms (Schapire et al., 1998; Duffy & Helmbold, 2000) for classification. This hints that the classification setting might have an iteration/complexity tradeoff like the regression results here.

To prove our bounds on the generalization error we make considerable use of results from statistical learning theory. These results rely on several measures of the complexity of a class of functions, i.e. the fat shattering dimension $fat(\ )$, the pseudo-dimension $Pdim(\ )$ and the (one- and two-norm) covering numbers $\mathcal{N}_1(\ ), \mathcal{N}_2(\ )$ (see Anthony & Bartlett, 1999 for a thorough discussion of these quantities). We use these quantities without definition throughout the paper and note that they can be thought of as generalizations of the VC-dimension.

In the remainder of this section we will state our three master algorithms and (informally) give their PAC-style bounds. The formal theorem statements and proofs appear in 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

In this subsection we assume that the generalization criterion is to minimize the expected squared error. Uniform convergence results imply that if the master function is in some relatively simple class and has small empirical squared error, then it will have small generalization error as well (see e.g. the treatment in Anthony & Bartlett, 1999).

Definition 3.1. 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

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

Definition 3.2 (Anthony & Bartlett, 1999). 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 $\mathcal{P}$.

We now 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 is suitable for use with a classification algorithm. These algorithms modify the sample and distribution in different ways. Despite this, the convergence bounds and techniques are similar for both algorithms. We present SQUARELEV.R in detail, and show (in Section 4.1) that it efficiently achieves arbitrarily small squared error on the sample when using an appropriate base learner. We also show that the resulting hypothesis is simple enough that, with high probability, it achieves small expected squared error on the entire domain. This requires a bound on the complexity of the function class $\mathcal{M}$ in which the master functions lie. Since the master function is a linear combination of base functions, the complexity of $\mathcal{M}$ depends on the complexity of the base function class $\mathcal{F}$ and the size of the coefficients in the linear combination. As mentioned before, bounding the size of the coefficients in the linear combination is a key step in the analysis of these algorithms.

SQUARELEV.R**.** The potential associated with SQUARELEV.R is the variance of the residuals,

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

where $\mathbf{r}$ is the m-vector of residuals defined by $r_i = y_i - F(x_i)$, $\bar{r} = \frac{1}{m}\sum_{i=1}^{m} r_i$, and $\bar{\mathbf{r}}$ is the m-vector with all components equal to $\bar{r}$. 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 $\widehat{\mathrm{er}}_S(\tilde{F})$ 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 negative 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

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

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

Figure 1. The SQUARELEV.R algorithm.

where $\sigma_{\mathbf{r}}$ and $\sigma_{\mathbf{f}}$ are the standard deviations of the respective vectors and $\bar{\mathbf{f}}$ is the m-vector with all components equal to $\frac{1}{m}\sum_{i=1}^m f(x_i)$. This edge is just the correlation coefficient between $\mathbf{f}$ and $\mathbf{r}$ (or equivalently, $\tilde{\mathbf{y}}$). Note that this edge measure is self-normalized in the sense that the edges of f and any positive multiple of f are the same. Any base function with a positive edge can be used to reduce the potential of the master function. In particular, the value α is chosen to minimize the potential of the new master function.

Theorem 4.1 from the next section shows that each iteration the potential decreases by a factor of $(1 - \epsilon^2)$, where ϵ is the edge (3) of the base function f. Given the relationship between the potential P_{var} and $\widehat{\text{er}}_S(\tilde{F})$, this implies that each iteration the mean squared error of $\tilde{F}$ decreases by a multiplicative factor that depends on the edge of the base learner. Assuming a lower bound ϵ_{min} on the edge of the base learner will allow us to prove that, for any $\rho \in \Re_+$, after

$$O\left(\frac{\ln(1/\rho)}{\epsilon_{min}^2}\right)$$

iterations the sample error of $\tilde{F}$ is at most ρ. Furthermore, we can also show that the sum of the coefficients is bounded by a similar quantity, assuming that the variance of the base function at each iteration is not too much less than the variance of the corresponding residual.

Note that the number of iterations before $\widehat{\text{er}}_S(\tilde{F}) \leq \rho$ and the bound on the size of the coefficients are independent of the number of examples m. This is because $\widehat{\text{er}}_S(\tilde{F})$ equals the potential/m, and the potential decreases by at least a constant factor each iteration.

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

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

Corollary 3.3 shows that SQUARELEV.R is an efficient regression algorithm for obtaining functions with arbitrarily small squared error. It follows from Theorem 4.3 which is somewhat more precise.

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 to that used in the GeoLev (Duffy & Helmbold, 1999) 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 it is larger than zero. Note that the edge (or correlation) is likely to go to zero as the number of iterations go to infinity. However, the proof of the above result requires only that the base hypotheses have edge at least ϵ_{min} for finitely many iterations. If the edge as a function of the iteration t is $\Omega(1/\sqrt{t})$ then a suitable ϵ_{min} can be found.

In addition we require the condition that

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

This condition requires that the base function does not have much smaller variance over the sample than the residuals. 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. Recall that the range of f is $[-1, 1]$, this assumption also requires that the range of f on the sample is not much smaller than its entire range. The base function f could have poor generalization error if it has much smaller variation on the sample than on the entire domain and so its addition may adversely affect the generalization error of the combined function F.

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

SQUARELEV.C mimics SQUARELEV.R with the following modifications. The modified labels are ± 1-valued: $\tilde{y}_i = \text{sign}(r_i)$. 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 $\mathbf{F}$).

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

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

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 $\{-1, +1\}$. Theorem 4.2 shows that SQUARELEV.C decreases its potential by a factor of $(1 - \epsilon^2)$ each iteration.

The potential and suitable base learners for SQUARELEV.C are closely related to those used by the GeoLev (Duffy & Helmbold, 1999) 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.

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. Anthony and Bartlett (1999) call this approach "generalization from approximate interpolation."

Definition 3.4 (Anthony & Bartlett, 1999). 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 $\rho, \delta \in (0, 1)$, $\eta, \gamma \in \Re_+$, there is $m_0(\rho, \delta, \eta, \gamma)$ such that for $m \geq m_0(\rho, \delta, \eta, \gamma)$, 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

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

Basically, a function class generalizes from approximate interpolation if there is a sample size such that whenever a function in the class is within η on every point in a sample of that size, then it will be (with high confidence) within $\eta + \gamma$ on (almost) the whole domain. Since this property provides a uniform bound on almost all of the input domain, it is 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 (Vapnik, 1998).

In this section we introduce the EXPLEV algorithm, which uses an exponential potential related to the one used by AdaBoost (Freund & Schapire, 1997). 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, the discontinuous derivatives make it difficult to analyze descent algorithms using this potential. The EXPLEV algorithm instead uses the two-sided potential

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

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

For a single example, P_{exp} increases exponentially with the magnitude of the residual while having a relatively flat (quadratic rather than exponential) 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

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

These are negative when $r_i = y_i - F_i$ is positive (increasing F_i decreases r_i and decreases the potential), and positive when r_i is negative.

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 each $|r_i| = |y_i - F(x_i)| \leq \eta$ for some given η. Here we find the scaling factor $s = \ln(m)/\eta$ most convenient.

Like SQUARELEV.C, each iteration the distribution D and the modified labels that EXPLEV gives to the base regressor follow from the potential. In particular,

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

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

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

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

This is the same definition of edge used in the confidence rated version of AdaBoost (Schapire & Singer, 1999). 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 2.3).

The ϵ defined in (11) is not "self normalizing;" scaling f also scales the edge. We could normalize it by dividing through by $\|\mathbf{f}\|_\infty$, but this is equivalent to our assumption that each $f(x_i) \in [-1, 1]$. Although the two-norm was used for normalization in the algorithms for minimizing the squared error, the infinity norm is more convenient for minimizing the maximum residual. Duffy and Helmbold (1999) discuss the choice of norms for the edge in the context of classification.

In addition to the desired residual bound η, 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.

Note that ExpLev sets $\hat{\epsilon}$ to the minimum of the ϵ defined in (11) and the parameter ϵ_{max}. Theorem 4.10 shows that ExpLev decreases its potential by at least a factor of $(1 - \frac{\hat{\epsilon}^2}{6})$ each iteration (unless the potential is already very small).

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

Input: A sample $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$,
a base learning algorithm, initial function F_1,
parameters η and ϵ_{max}.
$s = \ln(m)/\eta$; $t = 1$
for $i = 1$ to m
$r_i = y_i - F_t(x_i)$
while $\max_i |r_i| > \eta$ and making sufficient progress do
$t = t + 1$
for $i = 1$ to m
$\nabla_i P_{exp} = -s \exp(sr_i) + s \exp(-sr_i)$
$D_t(x_i) = \dfrac{|\nabla_i P_{exp}|}{||\nabla P_{exp}||_1}$
$\tilde{y}_i = \text{sign}(r_i)$
$S' = \{(x_1, \tilde{y}_1), \ldots, (x_m, \tilde{y}_m)\}$
Call base learner with distribution D_t over S',
obtaining hypothesis f
$\hat{\epsilon} = \min\left(\sum_{i=1}^m D(x_i)\tilde{y}_i f(x_i), \epsilon_{max}\right)$
$\alpha_t = \dfrac{1}{2s} \ln\left(\dfrac{sP_{exp} + 2sm + \hat{\epsilon}||\nabla P_{exp}||_1}{sP_{exp} + 2sm - \hat{\epsilon}||\nabla P_{exp}||_1}\right)$
$F_t = F_{t-1} + \alpha_t f$
for $i = 1$ to m
$r_i = y_i - F_t(x_i)$
return $(F_t, \max_i |r_i|)$

Figure 2. The ExpLev algorithm. Recall that $P_{exp} = \sum_{i=1}^m (e^{sr_i} + e^{-sr_i} - 2)$.

number of iterations that is logarithmic in the sample size m and linear in $1/\eta$. In particular, Theorem 4.11 shows that if each $y_i \in [-B, B]$ then $|y_i - F(x_i)| < \eta$ within

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

iterations, and

$$\sum_{t=1}^{T} \alpha_t \leq (2\eta + B)\frac{\ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3}.$$

It is worth examining this result in a little more detail. Despite the linear dependence on $1/\eta$ in the bound on T, the bound on the sum of the α values actually decreases (slightly) as η is reduced. As the required accuracy is increased the individual step sizes shrink and the bound on the length of the total path traversed by the algorithm (the sum of the step sizes) actually drops slightly. The η parameter causes the algorithm to approximate 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 coefficients (small coefficients tend to give better generalization error). Note the individual α values chosen by the algorithm depend linearly on η.

Although we can prove a PAC-style result for EXPLEV, we obtain better bounds using one of the modifications discussed below.

Modifications to EXPLEV. There are several modifications to EXPLEV that have the potential of improving its performance. We discuss three possible modifications here.

From the proof of Theorem 4.10 one can see that our choice of α is convenient for the analysis, but is generally not the step size that minimizes the potential. Therefore an obvious modification to EXPLEV performs a line search to find the best step size rather than using the explicit value convenient for our proofs. (A similar line search is suggested by Schapire and Singer for their confidence rated prediction version of AdaBoost (Schapire & Singer, 1999)).

If the base learning algorithm performs regression rather than classification, then the following variant of EXPLEV might be preferable. This variant sets $\tilde{y}_i = s\exp(sr_i) - s\exp(-sr_i)$ and keeps the distribution uniform over the modified sample. When the edge is defined[3] to be $\frac{\sum_{i=1}^{m} \tilde{y}_i f(x_i)}{\|\tilde{\mathbf{y}}\|_1 \|\mathbf{f}\|_\infty}$, then the same analysis used for EXPLEV also applies to this variant (with only a minor modification to the algebra in the proof of Theorem 4.10).

Our third variant runs EXPLEV in stages with different η parameters. Recall that our time bound for EXPLEV depends linearly on η when a single η parameter is used. We can obtain a faster algorithm by using a sequence of exponentially decreasing η parameters, decreasing η by a factor of z after each stage. In what follows we examine the case where the target sample approximation is reduced by a constant factor each stage and call this algorithm EXPITERLEV (presented formally in figure 3).

Theorem 4.12 shows that if the y values are in $[-B, B]$ then (with z set to 2) after $T = \lceil \frac{18\ln(m)}{\epsilon_{min}^2} \rceil \lceil \ln(B/\eta') \rceil$ calls of the base learning algorithm the error on the sample points

Input: A sample $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$,
    a base learning algorithm, parameters η', ϵ_{max}, and z.
initialize $\tau = 1$ and master function F_τ to the zero function
set B and $r_{\max}$ to $\max_i |y_i|$
while $r_{\max} > \eta'$ and making sufficient progress do
    set $\eta_\tau = B/z^\tau$
    Call ExpLev with S, base learner, F_τ, parameters $\eta = \eta_\tau$, ϵ_{max}
        obtaining new master function $F_{\tau+1}$ and maximum residual $r_{\max}$
    $\tau = \tau + 1$
return (F_τ)

Figure 3. The ExpIterLev algorithm.

is at most η'. Furthermore, the same theorem shows that the sum of the coefficients in the master function is at most $\frac{24B}{\epsilon_{min}^2} \ln(\frac{1+\epsilon_{max}}{1-\epsilon_{max}})$.

Since the master function class $\mathcal{M}$ consists of functions which are linear combinations of functions from $\mathcal{F}$, this gives us a bound on the complexity of $\mathcal{M}$ in terms of the complexity of the class $\mathcal{F}$ from which the base regression functions are drawn. Using this bound on the complexity of $\mathcal{M}$ we can obtain the following PAC-style result from Theorem 4.13.

Corollary 3.5. *Assume that data is drawn IID from a distribution $\mathcal{P}$ on X with $y = g(x)$ for some function $g : X \to [-B, B]$, 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 $\exists m(\rho, \delta, \eta, \gamma, \epsilon_{min})$ a polynomial in $1/\rho$, $1/\delta$, $1/\eta$, $1/\gamma$ such that the following holds for all $\rho, \delta, \epsilon_{min} \in (0, 1)$, $\eta, \gamma \in \Re_+$: with probability at least $1 - \delta$, if trained on a sample $x = (x_1, x_2, \ldots, x_m) \in \mathcal{P}^m$, then* ExpIterLev *(with parameters $s = \ln(m)/\eta$ and $\epsilon_{max} \geq \epsilon_{min}$) produces a hypothesis F, in time polynomial in $1/\rho$, $1/\delta$, $1/\eta$, $1/\gamma$, ϵ_{min}, satisfying*

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

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

This shows that ExpIterLev is an efficient regression algorithm for obtaining functions that interpolate a target to arbitrarily high accuracy. A similar result can be proved for the unmodified ExpLev algorithm using the weaker results of Theorem 4.11 instead of Theorem 4.12.

4. Proofs of main results

In this section we prove the main results described in the previous section. These proofs proceed similarly to the original proofs for AdaBoost (Freund & Schapire, 1997). 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 (Anthony & Bartlett, 1999).

4.1. Performance of SQUARELEV.R

The following theorem shows how the potential (2) decreases each iteration. The value of α used in the proof 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 $\epsilon = \frac{(\mathbf{r}-\bar{\mathbf{r}})\cdot(\mathbf{f}-\bar{\mathbf{f}})}{\|\mathbf{r}-\bar{\mathbf{r}}\|_2\|\mathbf{f}-\bar{\mathbf{f}}\|_2}$.

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

□

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 $y_i \in [-\frac{B}{2}, \frac{B}{2}]$. If the edges of the weak hypotheses used by* SQUARELEV.R *are bounded below by $\epsilon_{min} > 0$ then for all $\rho > 0$ after*

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

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

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

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 $\tilde{F}_T(x) = F_T(x) + \frac{1}{m}\sum_{i=1}^{m}(y_i - F_T(x_i))$, we have that $\widehat{\mathrm{er}}_S(\tilde{F}_T) \leq \rho$. The initial potential is at most $m(B/2)^2$ as $P_{var} \leq \sum_{i=1}^{m} r_i^2$ and the initial r_i's are bounded by $B/2$. Since 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 $\tilde{F}_T(x)$ is at most ρ when

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

Using the fact that $\ln(\frac{1}{1-\epsilon_{min}^2}) > \epsilon_{min}^2$ proves 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 the bound on T gives the second part of the theorem. □

The previous theorem shows that SQUARELEV.R takes only polynomial time to obtain a function having both small error on the sample and low complexity. We now derive a bound on the generalization error of the master regression function.

Theorem 4.3. *Assume that data is drawn IID from a distribution $\mathcal{P}$ on $X \times [-\frac{B}{2}, \frac{B}{2}]$ (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 $\delta \in (0, 1)$ and $\rho > 0$, if* SQUARELEV.R *is applied to a random sample (drawn IID from $\mathcal{P}$) of size at least*

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

where $K = 2\max(c\frac{\ln(B^2/2\rho)}{\epsilon_{min}^2}, B)$, then with probability at least $1 - \delta$ the (shifted) master hypothesis $\tilde{F}_T$ after $T = \lceil\frac{\ln(\frac{B^2}{2\rho})}{\epsilon_{min}^2}\rceil$ 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 $T = \lceil\frac{\ln(\frac{B^2}{2\rho})}{\epsilon_{min}^2}\rceil$ iterations the sample error $\widehat{\mathrm{er}}_S(\tilde{F}_T) \leq \frac{\rho}{2}$ 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 $\mathrm{er}_{\mathcal{P}'}(F')$ 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 for any $F' \in \mathcal{M}'$ and the corresponding $\tilde{F} \in \mathcal{M}$:

$$
\begin{aligned}
|\mathrm{er}_{\mathcal{P}'}(F') - \widehat{\mathrm{er}_{S'}}(F')| &\geq \rho \\
\left| \mathbf{E}\left(\frac{y}{K} + \frac{1}{2} - F'(x)\right)^2 - \frac{1}{m}\sum_{i=1}^{m}\left(\frac{y_i}{K} + \frac{1}{2} - F'(x_i)\right)^2 \right| &\geq \rho \\
\left| \mathbf{E}\,(y - \tilde{F}(x))^2 - \frac{1}{m}\sum_{i=1}^{m}(y_i - \tilde{F}(x_i))^2 \right| &\geq \rho K^2 \\
|\mathrm{er}_{\mathcal{P}}(\tilde{F}) - \widehat{\mathrm{er}_{S}}(\tilde{F})| &\geq \rho K^2.
\end{aligned}
$$

So that

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

where the first inequality follows from Theorem 17.1 in Anthony and Bartlett (1999) and the second from Lemma 6 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

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

We will now remove the m from the right-hand-side of (12). Since $\ln(a) \leq ab + \ln(1/b) - 1\ \forall a, b > 0$, we have $ab \geq \ln(a) + \ln(b) + 1$ and so

$$
\begin{aligned}
\frac{m\rho^2}{2^8K^4q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil} &\geq \ln(m) + \ln\left(\frac{\rho^2}{2^8K^4q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil}\right) + 1 \\
\frac{m\rho^2}{2^8K^4} &\geq q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil\ln\left(\frac{m\rho^2}{q\left\lceil\frac{2^{12}K^4}{\rho^2}\right\rceil 2^8K^4}\right). \qquad (13)
\end{aligned}
$$

Subtracting (13) from (12) shows that the sample size 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

The analysis of SQUARELEV.C follows the same outline as that given for SQUARELEV.R. Therefore we quickly derive the main 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 $\epsilon = (\mathbf{r}\cdot\mathbf{f})/\|\mathbf{r}\|_2\|\mathbf{f}\|_2$.

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

□

Theorem 4.5. *Assume that each $y_i \in [\frac{-B}{2}, \frac{B}{2}]$. If the edges of the weak hypotheses used by* SQUARELEV.C *are bounded below by $\epsilon_{min} > 0$ then for all $\rho > 0$ after*

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

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

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

Proof: If $P_{sq} \leq m\rho$ then $\widehat{\text{er}}_S \leq \rho$. Furthermore, since the initial potential is at most $m\left(\frac{B}{2}\right)^2$ 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 F_T is at most ρ when

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

proving the first part of the theorem. For the second part, on each iteration:

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

Multiplying this bound by T gives the second part. □

Theorem 4.6. *Assume that data is drawn IID from a distribution $\mathcal{P}$ on $X \times [-\frac{B}{2}, \frac{B}{2}]$ (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 (5) of f is bounded below by ϵ_{min} and $\frac{\|\mathbf{r}\|_2}{\|\mathbf{f}\|_2} \leq c$. Then there exists a constant $A \geq 0$ such that, for all $\delta \in (0,1)$ and $\rho > 0$, if* SQUARELEV.C *is applied to a random sample (drawn IID from $\mathcal{P}$) of size at least*

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

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

Proof: The proof is identical to that of Theorem 4.3, with the assumption

$$\frac{\|\mathbf{r}-\bar{\mathbf{r}}\|_2}{\|\mathbf{f}-\bar{\mathbf{f}}\|_2} \leq c$$

replaced by

$$\frac{\|\mathbf{r}\|_2}{\|\mathbf{f}\|_2} \leq c.$$

□

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 $P_{exp}(\mathbf{r}) = \sum_{i=1}^{m} \left(e^{sr_i} + e^{-sr_i} - 2\right)$. Recall that $\nabla P_{exp}(\mathbf{r}) = -s\exp(sr_i) + s\exp(-sr_i)$, the gradient of the potential with respect to $\mathbf{F}$, and the "edge" is defined by

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

where we assume f takes values in $[-1, 1]$. If this is not the case, then the edge ϵ can be normalized by $\|\mathbf{f}\|_\infty$. Throughout the remainder of the section we assume that f takes values in $[-1, 1]$.

Again we start by bounding 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 4.8 these vectors are the $\mathbf{r}^{(c)}$ in the following definition.

Definition 4.7. 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 4.8. *For all $c \geq 0$, we have that* $\inf_{\mathbf{r}\in S_c} \|\nabla P_{exp}(\mathbf{r})\|_1 = \|\nabla P_{exp}(\mathbf{r}^{(c)})\|_1$.

Proof: See Appendix. □

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

Lemma 4.9. *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: See Appendix. □

We are now ready to prove the main invariant for ExpLev. The proof of Theorem 4.10 shows that the choice of α used by ExpLev minimizes an upper bound on the potential, and thus ExpLev may not be minimizing the new potential each iteration (Section 3.2 discusses replacing the explicit choice of α with a line search). Note that the following theorem does not rely on ExpLev's choice of scale factor s.

Theorem 4.10. *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 step size $\alpha < \frac{1}{2s}\ln(\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}) \leq \frac{1}{2s}\ln(\frac{1+\epsilon_{max}}{1-\epsilon_{max}})$ 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.

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

Inequality (14) uses the linear approximation of a^b for $b \in [0, 1]$. Minimizing the right hand side of (15) 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

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

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

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

Now Lemmas 4.8 and 4.9 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. The following Theorem relies on the scale factor s being set to $\ln(m)/\eta$.

Theorem 4.11. *If $m \geq 3$, ϵ_{min} is a lower bound on the edges of the base functions, and each initial residual $y_i - F_1(x_i) \in [-B, B]$, then for all parameter settings $\eta > 0$ and $\epsilon_{max} \geq \epsilon_{min}$,* ExpLev *achieves $|y_i - F_T(x_i)| < \eta$ within*

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

iterations. Furthermore,

$$\sum_{t=1}^{T} \alpha_t \leq \frac{(B + 2\eta) \ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3}.$$

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 4.10 implies that the potential drops by at least a factor of $1 - \epsilon_{min}^2/6$ each iteration until $P_{exp} \leq \exp(s\eta) + \exp(-s\eta) - 2$. 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

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

iterations every residual is at most η. Using the fact that $s=\frac{\ln(m)}{\eta}$ and simplifying we obtain

$$\begin{aligned}\left\lceil \frac{\ln(m)+sB-\ln(m+1/m-2)}{\ln\left(\frac{1}{1-\epsilon_{min}^2/6}\right)} \right\rceil &\leq \left\lceil \frac{\ln\left(\frac{m^2}{m^2-2m+1}\right)+sB}{\epsilon_{min}^2/6} \right\rceil \\ &\leq \left\lceil \frac{2\ln(3/2)+sB}{\epsilon_{min}^2/6} \right\rceil \\ &\leq \left\lceil \frac{1+sB}{\epsilon_{min}^2/6} \right\rceil.\end{aligned}$$

For the second claim, Theorem 4.10 shows that each $\alpha_t \leq \frac{\eta}{2\ln(m)}\ln(\frac{1+\epsilon_{max}}{1-\epsilon_{max}})$. Therefore, after T iterations the sum of the α values is at most

$$\begin{aligned}&\left\lceil \frac{\frac{B}{\eta}\ln(m)+1}{\epsilon_{min}^2/6} \right\rceil \frac{\eta}{2\ln(m)}\ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right) \\ &\quad\leq \left(\frac{B}{\epsilon_{min}^2/3}+\frac{\eta}{\ln(m)\epsilon_{min}^2/3}+\frac{\eta}{2\ln(m)}\right)\ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right) \\ &\quad\leq \frac{2\eta+B}{\epsilon_{min}^2/3}\ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right)\end{aligned}$$

as required. □

Theorem 4.11 shows that EXPLEV produces a combined function in polynomial time that approximately interpolates the sample to arbitrarily high accuracy.

In fact we can obtain better results by running the EXPLEV algorithm in stages, improving the target sample approximation each stage. EXPITERLEV does this as discussed in Section 3.2, and has the following bound based on Theorem 4.11. Although the theorem is stated for general parameter settings, setting $z=2$ and $\epsilon_{max}=1/2$ are natural choices.

Theorem 4.12. *If $m \geq 3$, ϵ_{min} is a lower bound on the edges of the base functions, and each $y_i \in [-B, B]$ then* EXPITERLEV *with parameters $\eta' > 0$, $\epsilon_{max} \geq \epsilon_{min}$, and $z > 1$ has each $|y_i - F_\tau(x_i)| < \eta'$ within*

$$T = \left\lceil \frac{\ln(m)z+1}{\epsilon_{min}^2/6} \right\rceil \left\lceil \frac{\ln(B/\eta')}{\ln(z)} \right\rceil$$

calls of the base learning algorithm. Furthermore,

$$\sum_{t=1}^{T} \alpha_t \leq \frac{(z+2)B}{(z-1)} \frac{\ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right)}{\epsilon_{min}^2/3}$$

Proof: Since the residuals decrease by at least a factor of $1/z$ each call to EXPLEV, at most $\tau_{\max} = \lceil \frac{\ln(B/\eta')}{\ln(z)} \rceil$ calls to EXPLEV are needed.

At the start of the jth call to EXPLEV the residuals are in $[-B/z^{j-1}, B/z^{j-1}]$ and the η parameter is B/z^j. From Theorem 4.11 we see that at most

$$\left\lceil \frac{\ln(m)z + 1}{\epsilon_{min}^2/3} \right\rceil$$

calls to the base learner are made during each call of EXPLEV. Multiplying this by the bound on the number of calls to EXPLEV gives the desired result.

Theorem 4.11 shows that, during the j^{th} call to EXPLEV, the sum of the α values increases by at most

$$\frac{\left(\frac{B}{z^{j-1}} + 2\frac{B}{z^j}\right) \ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3}$$

Therefore after $\tau_{\max}$ calls to EXPLEV, the sum of the α values is at most

$$\begin{aligned}\sum_{j=1}^{\tau_{\max}} \frac{\left(\frac{B}{z^{j-1}} + 2\frac{B}{z^j}\right) \ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3} &= \frac{\left(B + 2\frac{B}{z}\right) \ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3} \sum_{j=0}^{\tau_{\max}-1} \frac{1}{z^j} \\ &< \frac{z}{z-1} \frac{\left(B + 2\frac{B}{z}\right) \ln \frac{1+\epsilon_{max}}{1-\epsilon_{max}}}{\epsilon_{min}^2/3}\end{aligned}$$

as desired. □

The next theorem bounds the generalization error of the final hypothesis produced by EXPITERLEV. It shows that, with high confidence, the master function returns a hypothesis that is close to the true function on almost all of the domain. As before, it might help to think of $z = 2$, and $\epsilon_{max} = 1/2$. In addition, the error tolerance of the final master function is broken into two parts, η and γ. The value η is the maximum error on the sample while γ is the additional acceptable error on unseen instances. One might naturally set each of these to half of the allowable error. Ignoring the extra logarithmic factors, the sample size depends on: the confidence as $\ln(1/\delta)$, the error region of the master function as $1/\rho$, and the error tolerance as $1/\gamma^2$. In addition, the sample size depends on the range of the target function as B^4 (through K). Note that the number of iterations depends logarithmically on both m (the number of examples) and B/η (the range divided by the error tolerance on the sample).

Theorem 4.13. *Assume that the data is drawn IID from a distribution $\mathcal{P}$ on X with $y = g(x)$ for some function $g : X \to [-B, B]$, 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 $\rho, \delta \in (0, 1)$ and all $\eta, \gamma > 0$ with probability at least $1 - \delta$:* ExpIterLev *with parameters η, $\epsilon_{max} \geq \epsilon_{min}$, and $z > 1$ returns an F_τ satisfying*

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

after $T = \lceil \frac{\ln(m)z+1}{\epsilon_{min}^2/6} \rceil \lceil \frac{\ln(B/\eta)}{\ln(z)} \rceil$ *leveraging iterations if trained on a sample S of size m at least*

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

where

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

and $K = \dfrac{(z+2)B}{(z-1)} \dfrac{\ln\left(\frac{1+\epsilon_{max}}{1-\epsilon_{max}}\right)}{\epsilon_{min}^2/3} \geq \sum_{t=1}^{T} \alpha_t$

Proof: Let $F_\tau = F_T = \sum_{t=1}^{T} \alpha_t f_t$ be the master function returned by ExpIterLev after T leveraging iterations (over τ stages). By Theorem 4.12, the master function F_T has residuals at most η and $\sum_{t=1}^{T} \alpha_t \leq K$, so the master function F_T lies in the class $\mathcal{M} = \{\sum_{i=i}^{N} w_i f_i : f_i \in \mathcal{F}, \sum_{i=1}^{N} |w_i| \leq K\}$.

In order to apply the standard generalization bounds, we need to re-scale the functions so that they map X to $[0, 1]$. Let $F' = \frac{F_T}{2K} + \frac{1}{2}$ and $g' = \frac{g}{2K} + \frac{1}{2}$. Furthermore, for each $h \in \mathcal{M}$ let $h'(x) = \frac{h(x)}{2K} + \frac{1}{2}$ and $\mathcal{M}' = \{h' : h \in \mathcal{M}\}$. (Note that the constraints on ϵ_{max} and z guarantee that $K > B$).

The functions in $\mathcal{M}'$ map X to $[0, 1]$ and

$$|h'(x) - g'(x)| < \eta + \gamma \Leftrightarrow |h(x) - g(x)| < 2\eta K + 2\gamma K$$

so

$$\begin{aligned} &\mathcal{P}^m\{S : (\exists h' \in \mathcal{M}')(\forall x \in S)|h'(x) - g'(x)| < \eta + \gamma\} \\ &\quad = \mathcal{P}^m\{S : (\exists h \in \mathcal{M})(\forall x \in S)|h(x) - g(x)| < 2\eta K + 2\gamma K\}. \end{aligned}$$

Therefore if $\mathcal{M}'$ generalizes from approximate interpolation with sample size $m_{\mathcal{M}'}(\rho, \delta, \eta, \gamma)$ then $\mathcal{M}$ generalizes from approximate interpolation with sample size

$$m_{\mathcal{M}}(\rho, \delta, \eta, \gamma) = m_{\mathcal{M}'}\left(\rho, \delta, \frac{\eta}{2K}, \frac{\gamma}{2K}\right). \tag{17}$$

Theorem 21.14 in Anthony and Bartlett (1999) shows that if $\mathcal{M}'$ has finite fat-shattering dimension then $\exists c > 0$ such that a sufficient sample size for generalization from approximate interpolation is

$$\frac{c}{\rho}\left(\ln\left(\frac{1}{\delta}\right) + \text{fat}_{\mathcal{M}'}\left(\frac{\gamma}{8}\right)\ln^2\left(\frac{\text{fat}_{\mathcal{M}'}\left(\frac{\gamma}{8}\right)}{\gamma\rho}\right)\right). \tag{18}$$

We can bound $\text{fat}_{\mathcal{M}'}$ in terms of K and q (recall that q is a bound on the pseudo-dimension of the base function class) using Lemma A.2 to obtain

$$\text{fat}_{\mathcal{M}'}\left(\frac{\gamma}{8}\right) \le 8\left\lceil\frac{2^{14}4K^2}{\gamma^2}\right\rceil q\ln\left(\frac{2^7 em 4K^2}{\gamma q}\right) \le 8\left\lceil\frac{2^{16}K^2}{\gamma^2}\right\rceil q\ \ln\left(\frac{2^9 emK^2}{\gamma}\right).$$

Let $c_1 = 8\lceil\frac{2^{16}K^2}{\gamma^2}\rceil q$ and $c_2 = \frac{2^9 eK^2}{\gamma}$, then $\exists c > 1$ such that a sufficient size for generalization from approximate interpolation for $\mathcal{M}'$ is any m satisfying

$$m \ge \frac{c}{\rho}\left(\ln\left(\frac{1}{\delta}\right) + c_1\ln(mc_2)\ln^2\left(\frac{c_1\ln(mc_2)}{\gamma\rho}\right)\right). \tag{19}$$

Since $m \ge c_1 > c_2$,

$$\begin{aligned}
\ln(mc_2)\ln^2\left(\frac{c_1\ln(mc_2)}{\gamma\rho}\right) &< 2\ln(m)\ln^2\left(\frac{2c_1\ln(m)}{\gamma\rho}\right)\\
&< 2\ln(m)\left(\ln(\ln(m)) + \ln\left(\frac{2c_1}{\gamma\rho}\right)\right)^2\\
&< 2\ln(m)\ln^2(\ln(m))\left(\ln\left(\frac{2ec_1}{\gamma\rho}\right)\right)^2.
\end{aligned}$$

Plugging this inequality back into (19) gives that a sufficient sample size for generalization is any m satisfying

$$\frac{m}{\ln(m)\ln^2(\ln(m))} \ge \frac{c}{\rho}\left(\ln\left(\frac{1}{\delta}\right) + 2c_1\ln^2\left(\frac{2ec_1}{\gamma\rho}\right)\right).$$

Note that if $m \ge 3$ and $m \ge 3w\ln^2(w)$, then $\frac{m}{\ln(m)\ln^2(\ln(m))} \ge w$. Therefore, a sufficient sample size for generalization from approximate interpolation for $\mathcal{M}'$ is

$$m'_{\mathcal{M}}(\rho, \delta, \eta, \gamma) > 3w'\ln^2(w').$$

where

$$w' = \frac{c}{\rho}\left(\ln\left(\frac{1}{\delta}\right) + 2c_1\ln^2\left(\frac{2ec_1}{\gamma\rho}\right)\right).$$

Recall that w' depends on γ and K (and thus η and ϵ_{min}) through c_1. Replacing γ with $\gamma/2K$ as in (17) shows that there is a $\tilde{c}$ such that a sufficient sample size for the original class $\mathcal{M}$ to generalize from approximate interpolation is

$$m_{\mathcal{M}}(\rho, \delta, \eta, \gamma) = m_{\mathcal{M}'}\left(\rho, \delta, \frac{\eta}{2K}, \frac{\gamma}{2K}\right) \geq 3w \ln^2(w)$$

where

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

as desired. □

The proof of Theorem 4.13 can also be used to prove a similar result for ExpLev, with the bounds of Theorem 4.11 replacing those of Theorem 4.12. Note that we have made little effort to optimize the constants in the generalization bound, as we use several general purpose bounds with intrinsic large constants in the proof.

5. Experimental results

We performed a modest set of experiments to validate our theoretical results. We tested all four algorithms on six standard data sets: Boston housing, servo, auto-mpg and abalone data sets from UCI (Blake & Merz, 1998), the Friedman 1 data set from Breiman et al. (1984), and data generated from the SINC function.[4] We especially wanted to test the assumption that simple base learners can achieve an edge on the modified samples created by the algorithms (see Section 2.3). The experiments indicate that even simple base learners generally achieve a reasonable edge on the modified samples, even after thousands of iterations.

In addition to examining the performance of the base learners, the experiments also validate several aspects of the analysis. They show that (for most data sets) the algorithms exponentially decrease the potential and maximum residuals. As expected, ExpLev and ExpIterLev tend to be more effective than the SqrLev algorithms in reducing the maximum residual. However, we were surprised to find that SquareLev.C is often *more* effective in reducing the residual on *small* samples. Overall, the behavior of the algorithms is, at least qualitatively, as predicted by the theoretical analysis.

5.1. Experimental procedure

We evaluated all four algorithms on four data sets from the UCI (Blake & Merz, 1998) repository and two artificial data sets, the one dimensional SINC function and the Friedman 1 data (Breiman et al., 1984). Some properties of these data sets are summarized in Table 1.

With the algorithms capable of using a base classifier (ExpLev, ExpIterLev and SquareLev.C) we used decision stumps minimizing the classification error as the base learner. With SquareLev.R we used regression stumps minimizing the squared loss.

Table 1. Table of parameters of data sets.

Data Set	Size	Features	Initial s
SINC	500	1	150
Friedman 1	400	10	20
Abalone	2090	8	20
auto-mpg	196	7	10
servo	84	4	80
Boston housing	253	13	5

Each iteration the coefficients for the base rules were found using a line search to minimize the resulting potential.

For EXPLEV, the performance of the algorithm is very sensitive to the initial choice of s. For these experiments, we chose s to be as large as possible given the precision of the machine (see Table 1).

Several practical issues arise when applying EXPITERLEV, in particular, the choice of rescaling parameter z is not obvious. We implemented a modified version of EXPITERLEV that directly manipulates the scale factor s used by EXPLEV. This version keeps the potential between the sample size m and m^2, increasing the scale factor s so that the potential becomes m^2 whenever it becomes too small. This ad-hoc range is somewhat arbitrary, and we made no effort to tune it.

The experiments focus on the sample performance rather than the generalization error. The in-sample performance is sufficient for most of the properties we are interested in verifying. Examining the generalization error requires more sophisticated experiments with large numbers of runs to obtain statistically significant results. Furthermore, the general purpose techniques we use to bound the generalization error involve large constants that will make even qualitative empirical validation difficult.

5.2. Results

The main weakness of our theoretical results is the assumption that the base learner can consistently return hypotheses with a useful edge, even when the data are re-labeled by the master algorithm. Fortunately, our experiments appear to indicate that this assumption may often hold in practice. On all but one of the data sets we examine, the edge remains significant over thousands of iterations of the algorithm. The edges on all data sets for the four algorithms are plotted in figure 4. The edges start high and rapidly decay to a moderate value, usually around 0.05. Even after several thousand iterations, the edges are still significant, and it is plausible that the edges remain significant for many more iterations. The one exception is the difficult servo data set. There it appears that our simple decision stump base learners are not sufficiently powerful. The spikes in the plot for EXPITERLEV reflect changes in the scaling factor.

In addition to examining the edges of the base hypotheses we examined the decrease in potential and maximum residual yielded by each of the algorithms. The logarithm of the

potential on each data set for each algorithm is plotted in figure 5 and the logarithm of the maximum residual on each training set is plotted in figure 6. Note that the saw-tooth behavior of EXPITERLEV's potential is due to the changes in the scale factor. After the initial rapid learning, the log of the potential and the log of the maximum residuals decrease roughly linearly for many thousands of iterations. This supports our theoretical results in Theorems 4.1, 4.4, and 4.10 which prove that the potentials decrease exponentially given a consistent edge.

We were surprised to see that the SqrLev algorithms often obtained better maximum residuals than EXPLEV. However, this appears to happen primarily on smaller data sets. To examine this phenomenon in more detail we tested SQUARELEV.C and EXPLEV on the SINC and Friedman 1 artificial data sets while varying the sample size. The results are plotted in figures 7 and 8. The plots illustrate that, while SQUARELEV.C may obtain smaller maximum residuals on very small data sets, its ability to reduce the maximum residual quickly degrades as the size of the training set increases.

These figures raise another interesting issue. The maximum residuals for EXPLEV often have an interesting "swan's neck" appearance (figure 8(b)). We believe that this slow-fast-slow rate of decrease is due to the choice of scaling factor, and progress is most rapid when the scaling factor is appropriate for the current magnitude of the residuals. This reinforces the motivation behind EXPITERLEV, which adaptively changes the scaling factor. Although our ad-hoc tuning for EXPITERLEV is not always better than EXPLEV, it does appear to be competitive on all of the data sets and considerably better on several.

Although the progress bounds for the SqrLev algorithms are tight, we make approximations when bounding EXPLEV's performance. The ratio between the observed decrease in potential for EXPLEV and the analytical bound on its decrease is plotted for the Friedman data set in figure 9. This figure shows that the analytic bound is initially quite loose, but it appears to tighten over time, with the largest value for the ratio being 0.992. This implies that there may be room for modest improvement in our bounds on EXPLEV.

Another source of looseness in the EXPLEV bounds on the maximum residual comes from converting the bound on the potential to a bound on the maximum residual in Theorem 4.11. In figure 10 we plot the ratio between the actual maximum residual and the bound on the maximum residual based on the current potential for the Friedman 1 data set. Although the bound is pessimistic, it remains reasonable. It is worth noting that the scale factor s is what makes this bound tight. A large s factor increases the proportion of the potential placed on the example with the largest residual.

For some of the data sets we examined the EXPLEV potential falls below the sample size m. For example, after approximately 500 iterations EXPLEV achieves potential less than $9 \approx \ln(m)$ on the SINC data set. Recall that the progress bound in Theorem 4.10 only applies when the total potential is larger than m. If the total potential is less than m then the magnitude of each residual is small, at most $(\ln m)/s$. A second order Taylor expansion of the EXPLEV potential, $e^{sr} + e^{-sr} - 2$, around $sr = 0$ shows that the potential approaches the SQUARELEV.C potential. In fact, it is possible to show a progress bound (for small enough residuals) that is arbitrarily close to the corresponding bound for SQUARELEV.C. We see this behavior on very small samples. Figure 11 plots the maximum residuals for EXPLEV, SQUARELEV.C and EXPITERLEV on a 100 example subset of the Friedman 1 data.

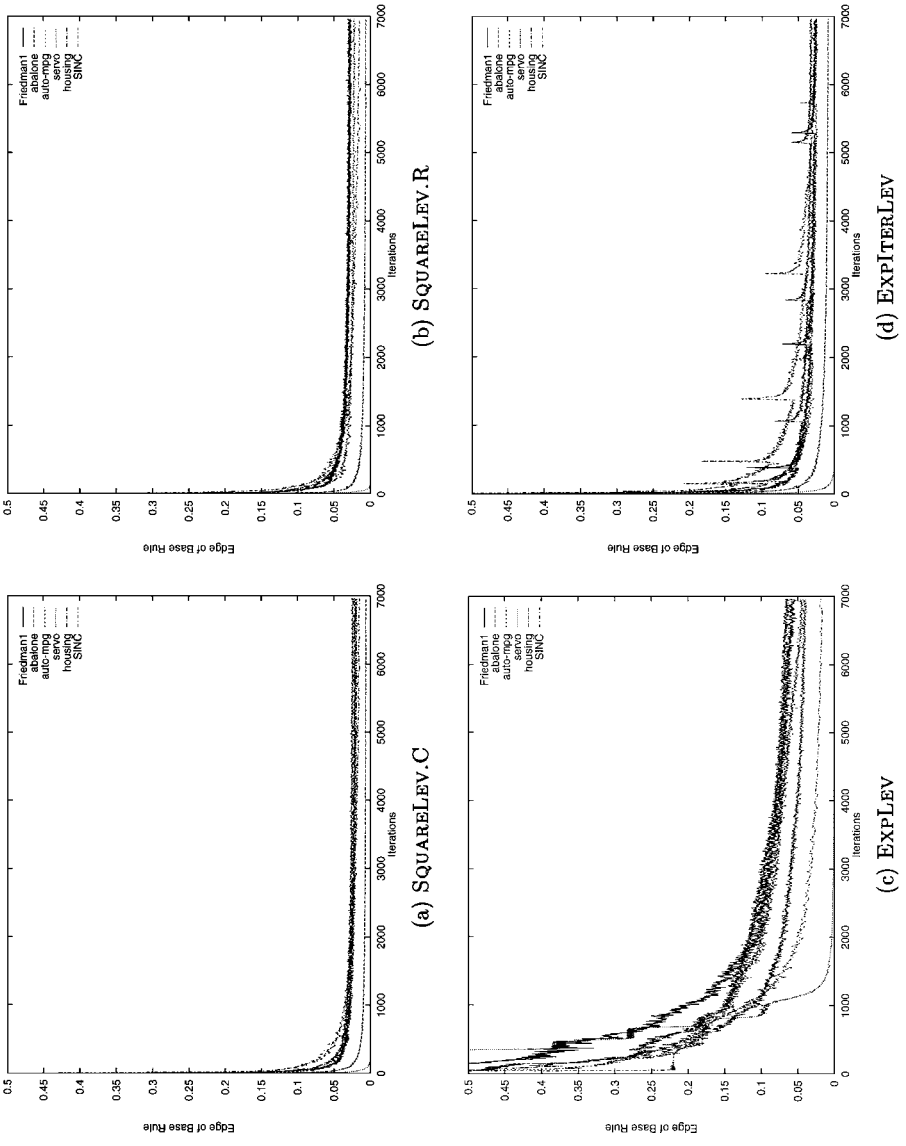


Figure 4. The edges obtained by each algorithm.

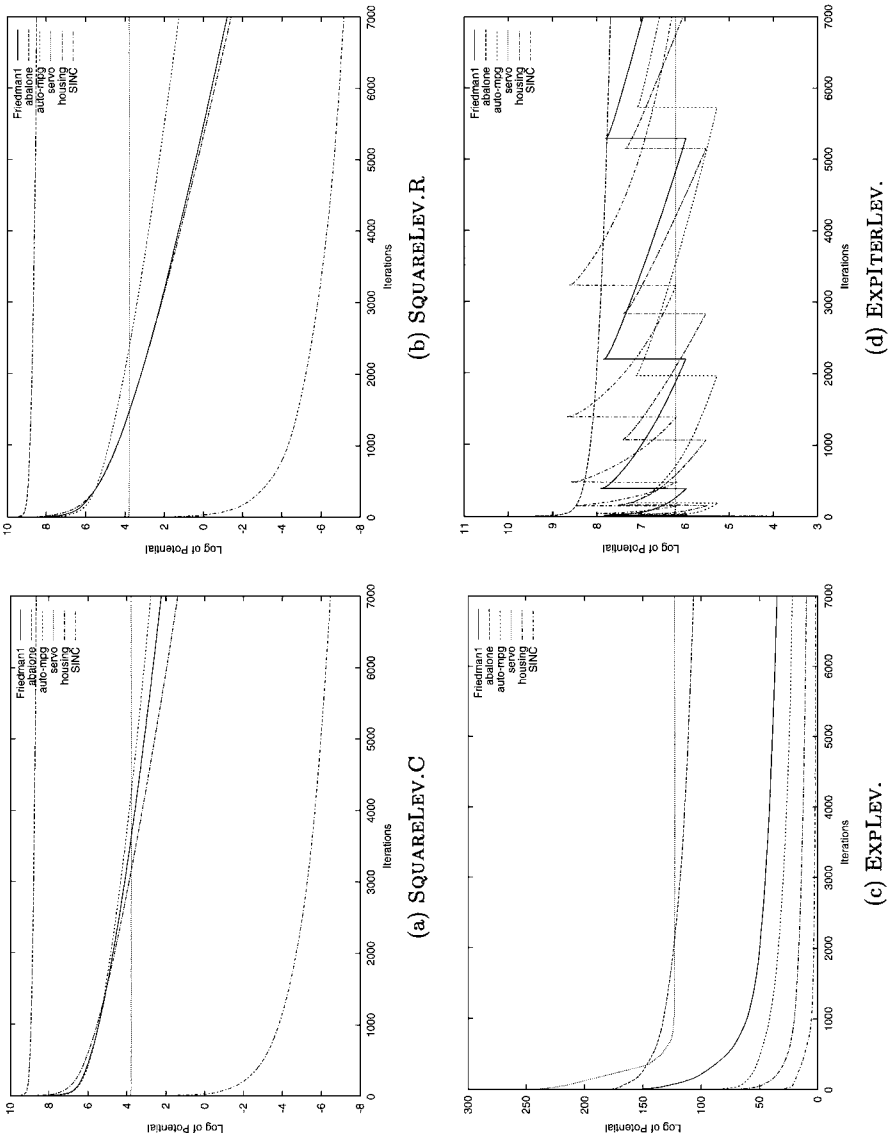


Figure 5. The natural logarithm of the potential achieved by each algorithm.

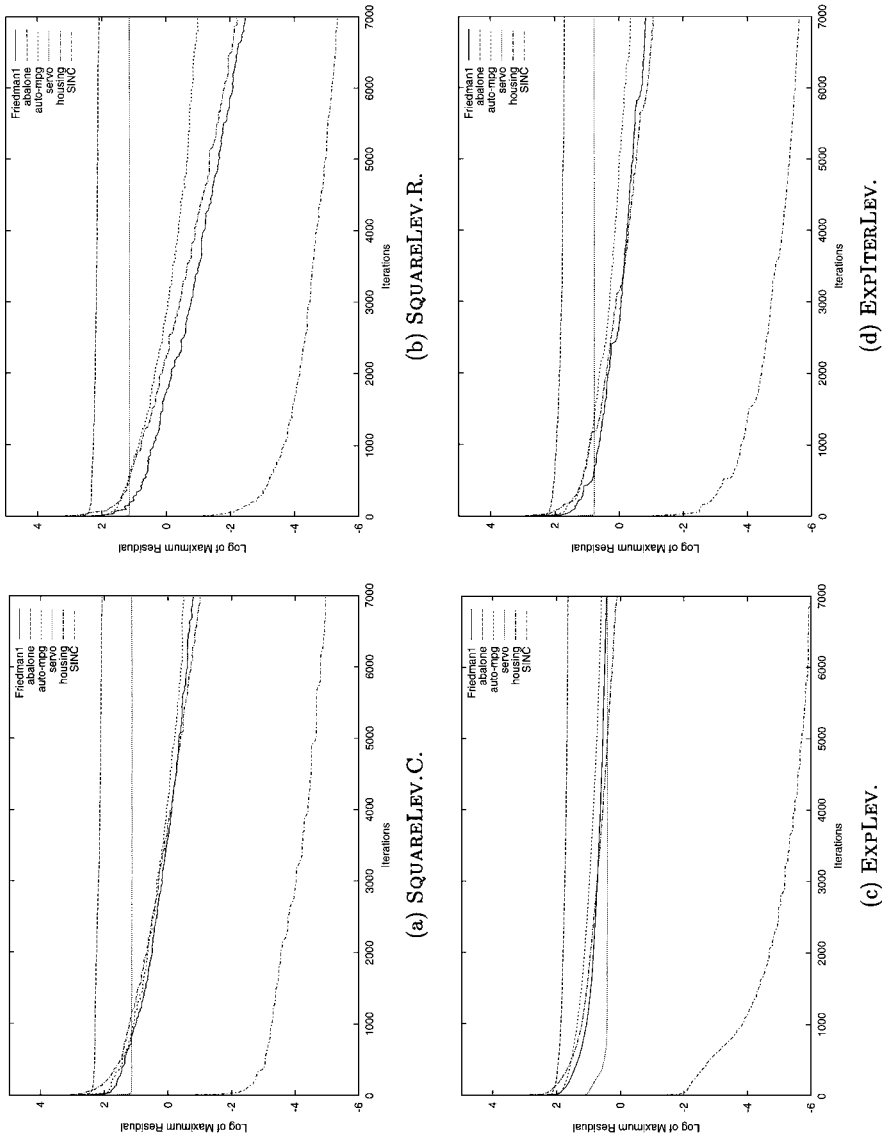
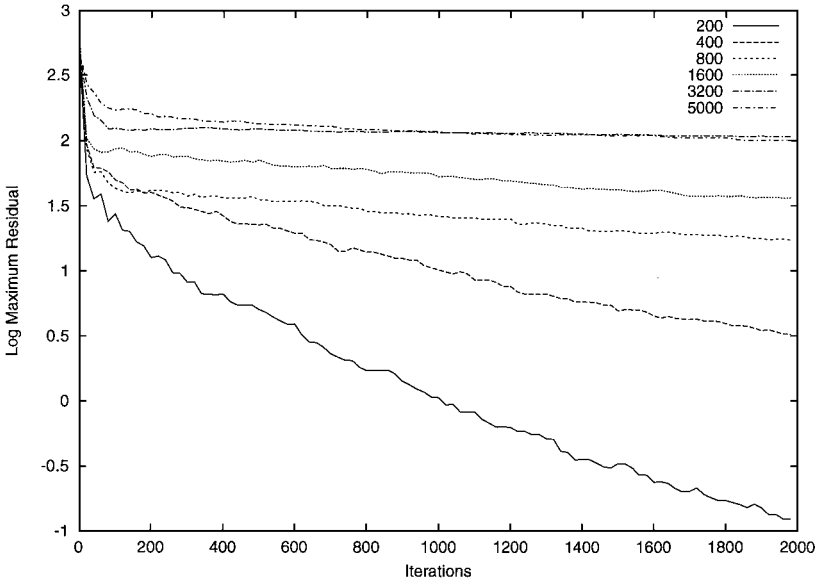
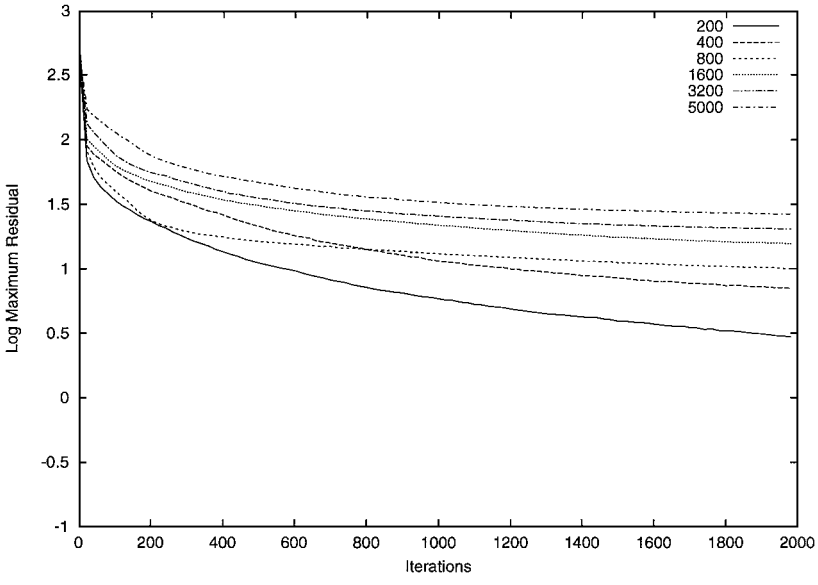


Figure 6. The natural logarithm of the maximum residual achieved by each algorithm.

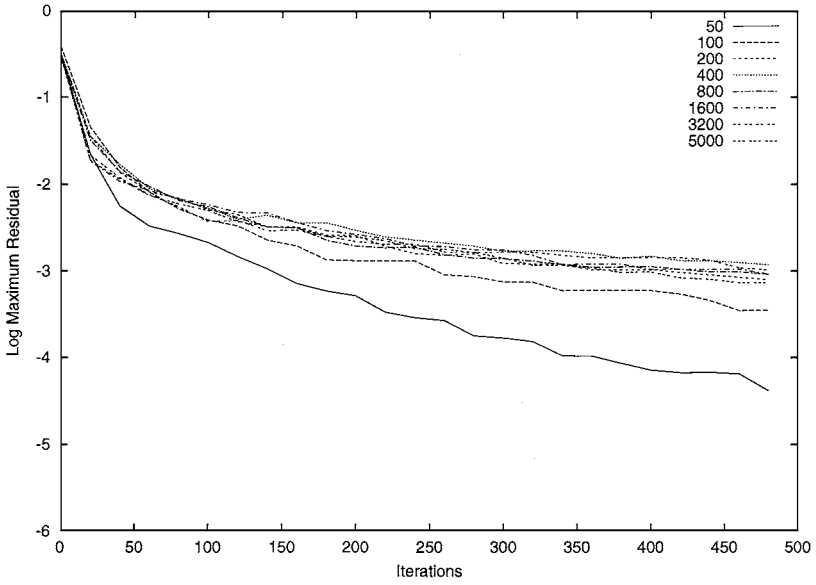


(a) SQUARELEV.C

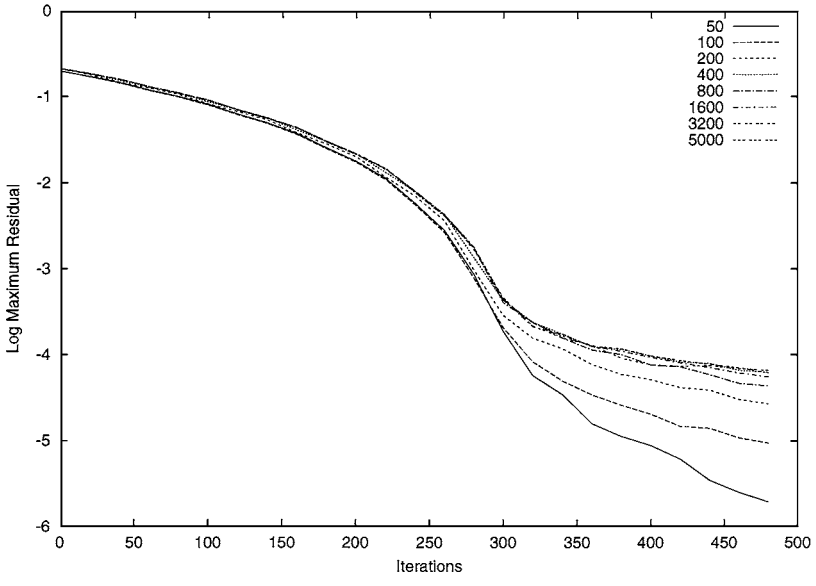


(b) EXPLEV

Figure 7. The natural log of the maximum residual on the Friedman 1 data set with increasing training set size.



(a) SQUARELEV.C



(b) EXPLEV

Figure 8. The natural log of the maximum residual on the SINC data set with increasing training set size.

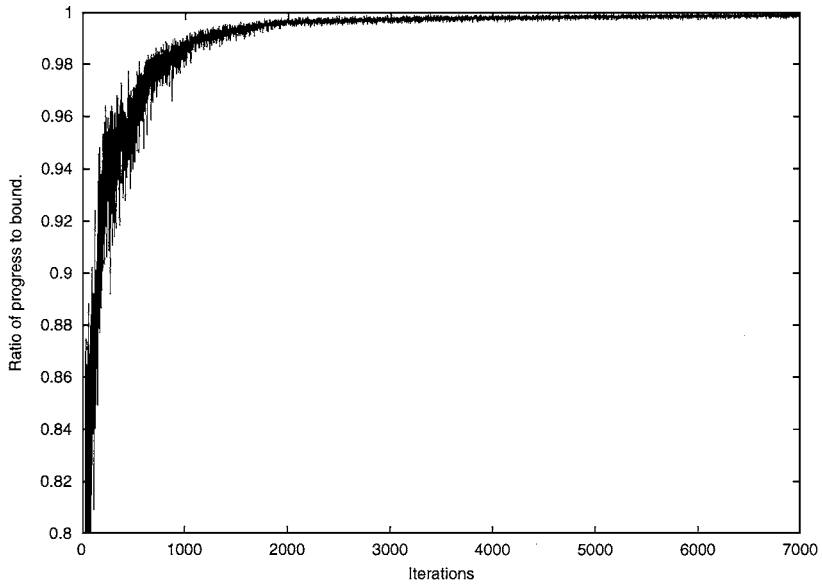


Figure 9. The ratio between progress bound and actual progress on the Friedman 1 data set.

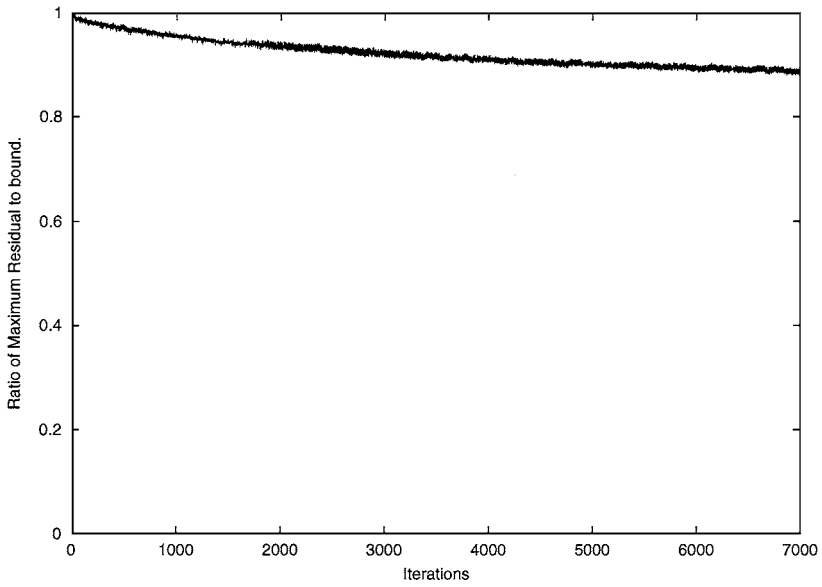


Figure 10. The ratio between the maximum residual achieved by EXPLEV and the bound given by the current potential i.e. $\ln(P_{exp})/s$ for EXPLEV on the Friedman 1 data set.

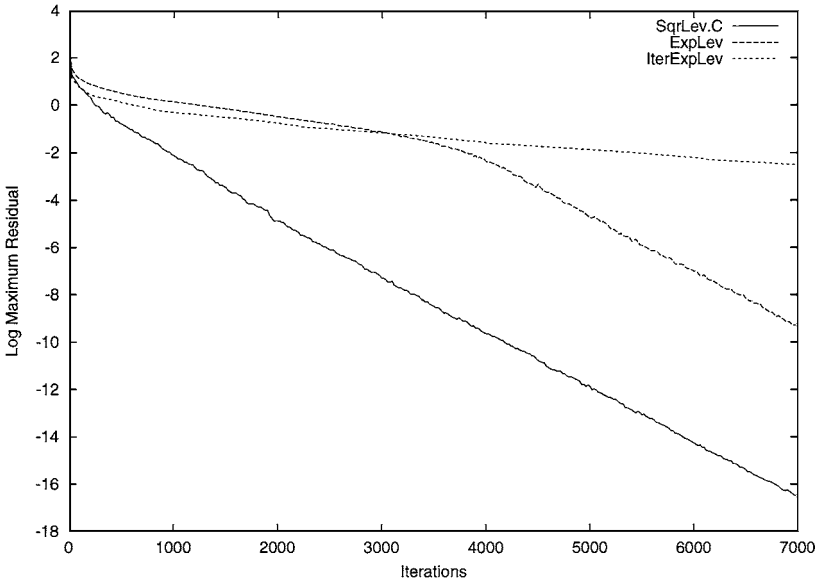


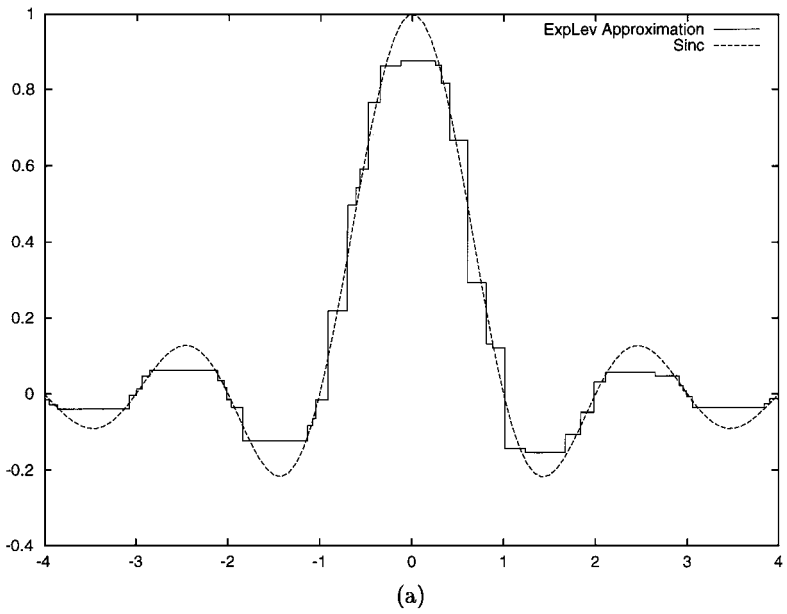
Figure 11. The natural log of the maximum residuals for a small version of the Friedman 1 data set for EXPLEV,EXPITERLEV and SQUARELEV.C.

In this figure we observe that the maximum residual curves for EXPITERLEV and EXPLEV are parallel for the initial iterations. However, once the residuals become small enough, EXPLEV begins performing like SQUARELEV.C instead i.e. their maximum residual curves become parallel. This happens around iteration 4000 at which point the potential is approximately 66. It appears therefore, that EXPLEV will continue to improve its master hypothesis even when the conditions of Theorem 4.10 are not satisfied.

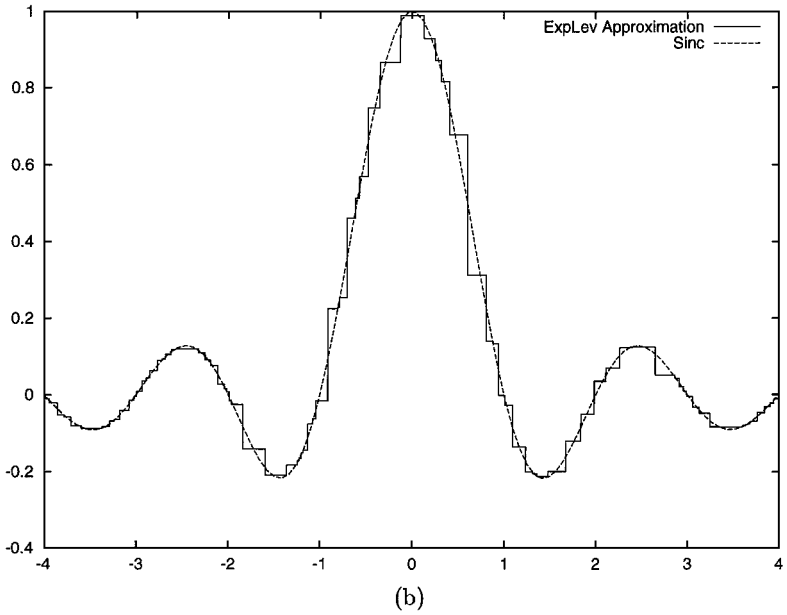
Finally, figure 12(a) shows how closely the Sinc function is approximated after 100 iterations on a training set of 100 examples, using EXPLEV with decision stumps, while figure 12(b) shows the same experiment after 1000 iterations. These figures illustrate how well the algorithm can approximate the target even on a small dataset.

6. 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 (Freund & Schapire, 1997) and Lee, Bartlett, and Williamson's Construct algorithm (Anthony & Bartlett, 1999). 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,



(a)



(b)

Figure 12. The approximation to the Sinc function produced by EXPLEV on 100 examples after (a) 100 rounds and (b) 1000 rounds using decision stumps.

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.

Our experiments validate our theoretical results highlighting several interesting features of the algorithms. In particular they show how the algorithms behave as the size of the training set changes. They also demonstrate that EXPLEV continues to work well outside the regions for which the theoretical bounds hold. Furthermore, these experiments suggest that the base learners continue to have a significant edge despite the relabelings imposed by our algorithms.

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 α values depends weakly on the desired accuracy η, while the number of iterations grows as $1/\eta$. EXPITERLEV has a similar behavior, the number of iterations grows as $z/\ln z$ while the sum of the α values shrinks as $(z+2)/(z-1)$.

Apppendix A: Technical Lemmas

Lemma 4.8. *For all $c \geq 0$, we have that* $\inf_{\mathbf{r} \in S_c} \|\nabla P_{exp}(\mathbf{r})\|_1 = \|\nabla P_{exp}(\mathbf{r}^{(c)})\|_1$.

Proof: Assume to the contrary that some $\mathbf{r} \in S_c$ has $\|\nabla P_{exp}(\mathbf{r})\|_1 < \|\nabla P_{exp}(\mathbf{r}^{(c)})\|_1$. If $c = 0$ then $S_c = \{\mathbf{r}^{(c)}\}$ so we may assume without loss of generality that $c > 0$ and $r_1 > 0$. If $\mathbf{r}$ has no other non-zero component, then $\|\nabla P_{exp}(\mathbf{r})\|_1 = \|\nabla P_{exp}(\mathbf{r}^{(c)})\|_1$. If $\mathbf{r}$ has another non-zero component (say r_i) then we construct $\mathbf{r}'$ from $\mathbf{r}$ by setting $r'_i = 0, r'_1$ so that $\mathbf{r}' \in S_c$, and preserving the other components of $\mathbf{r}$. We now show that $\|\nabla P_{exp}(\mathbf{r}')\|_1 < \|\nabla P_{exp}(\mathbf{r})\|_1$. Repeating the construction yields $\mathbf{r}^{(c)}$ and gives the contradiction.

To simplify the notation, let $a = \exp(s|r_1|)$, $b = \exp(s|r_i|)$, and $z = a + b + 1/a + 1/b - 2$. Note that z is an increasing function of a and b since $a > 1, b > 1$. Set $d = \exp(s|r'_1|) > 1$, and since $P_{exp}(\mathbf{r}) = P_{exp}(\mathbf{r}')$,

$$P_{exp}(r'_1) + P_{exp}(0) = P_{exp}(r_1) + P_{exp}(r_i) \tag{20}$$

$$d + \frac{1}{d} - 2 + 0 = a + \frac{1}{a} + b + \frac{1}{b} - 4 \tag{21}$$

$$d + 1/d = z. \tag{22}$$

Solving for d and taking the root greater than 1 gives

$$d = \frac{1}{2}z + \frac{1}{2}\sqrt{z^2 - 4}\,. \tag{23}$$

We now consider the norms of the gradients. Note that for all $\mathbf{v} \in S_c$,

$$\|\nabla P_{exp}(\mathbf{v})\|_1 = \sum_{i=1}^{m} |-se^{sv_i} + se^{-sv_i}| \tag{24}$$

$$= s \sum_{i=1}^{m} \left(e^{|sv_i|} + e^{-|sv_i|} - 2e^{-|sv_i|}\right) \tag{25}$$

$$= s \left(c + 2m - 2 \sum_{i=1}^{m} e^{-|sv_i|}\right) . \tag{26}$$

Therefore,

$$\begin{aligned} &\frac{\|\nabla P_{exp}(\mathbf{r}')\|_1 - \|\nabla P_{exp}(\mathbf{r})\|_1}{s} \\ &= \left(\frac{\|\nabla P_{exp}(\mathbf{r}')\|_1}{s} - c - 2m\right) - \left(\frac{\|\nabla P_{exp}(\mathbf{r})\|_1}{s} - c - 2m\right) \\ &= \left(-\frac{2}{d} - 2\right) - \left(-\frac{2}{a} - \frac{2}{b}\right). \end{aligned}$$

Lemma A.1 shows that the above quantity is negative for $a, b > 1$ and $\|\nabla P_{exp}(\mathbf{r}')\|_1 < \|\nabla P_{exp}(\mathbf{r})\|_1$ as desired. □

Lemma A.1. *For $a, b > 1$, the quantity*

$$\frac{-1}{\left(a + b + \frac{1}{a} + \frac{1}{b} - 2 + \sqrt{\left(a + b + \frac{1}{a} + \frac{1}{b} - 2\right)^2 - 4}\right)} - 2 + \frac{2}{a} + \frac{2}{b} \tag{27}$$

is negative.

Proof: This quantity can be rewritten as

$$\frac{-(a+b)((ab+1)(ab-b-a) + 2ab - (ab-b-a)\sqrt{x})}{(a^2b + ab^2 + b + a - 2ab + \sqrt{x})} \tag{28}$$

where

$$x = (ab+1)(a+b)((ab+1)(a+b) - 4ab) \tag{29}$$

Since $a, b > 1$, the denominator is positive so we concentrate on the numerator. We begin by noting that the numerator is negative if $ab \geq (a+b)$ so we focus on case $ab < (a+b)$. In this case $-(ab-b-a)\sqrt{x}$ is positive and we now proceed to upper bound x. Starting

with the third factor,

$$\begin{aligned}
&(ab+1)(a+b) - 4ab < (ab-1)^2 \\
&\quad \Leftrightarrow\ a^2b + ab^2 - 4ab + a + b < a^2b^2 - 2ab + 1 \\
&\quad \Leftrightarrow\ a^2b + ab^2 + a + b - 2ab - a^2b^2 - 1 < 0 \\
&\quad \Leftrightarrow\ -(b-1)(a-1)(ab+1) < 0
\end{aligned} \tag{30}$$

which is true since $a, b > 1$.

Since we are assuming that $ab < (a+b)$ then either $a < 2$ or $b < 2$, without loss of generality we assume that $a \leq b$ and hence $a < 2$. With this assumption we have

$$\begin{aligned}
-a^3 + a - a^2b + b &< 0 \\
(ab+1)(a+b) - a(a+b)^2 &< 0 \\
(ab+1)(a+b) &< a(a+b)^2 \,.
\end{aligned} \tag{31}$$

Combining the left side of (30) with (31) shows that the square root term in (28) is bounded by

$$-(a+b)(ab-a-b)(ab-1)\sqrt{a}\,.$$

We now lower bound the first term in the numerator of (28).

$$\begin{aligned}
ab(a-1)(b-1) > 0 &\Leftrightarrow a^2b^2 - ab^2 - a^2b + ab > 0 \\
&\Leftrightarrow a^2b^2 - ab^2 - a^2b + 3ab - b - a > 2ab - a - b \\
&\Leftrightarrow (ab+1)(ab-b-a) + 2ab > 2ab - a - b
\end{aligned}$$

So the numerator in (28) is upper bounded by

$$-(a+b)((2ab-a-b) + (ab-b-a)(ab-1)\sqrt{a})$$

We can simplify this by removing a factor of $(a+b)$ and replacing a by q^2, to obtain

$$-(q-1)(-q^4b + q^4b^2 - q^3b + q^3b^2 + q^2 - 2q^2b + b)\,.$$

We can remove a factor of $(q-1)$ to obtain

$$-(-q^4b + q^4b^2 - q^3b + q^3b^2 + q^2 - 2q^2b + b). \tag{32}$$

and it is then straightforward to show that what remains is negative for $b = q^2\ \forall q > 1$ and $q = 1\ \forall b > 1$. The proof is now completed by showing that this is decreasing for all $b > q^2$ and so is negative for all $q > 1, b > q^2$ as required. The derivative of (32) is

$$-2(q^4 - q^3)b + q^4 + q^3 + 2q^2 - 1 \tag{33}$$

which is negative. This completes the proof. □

Lemma 4.9. *If* $P_{exp}(\mathbf{r}) \geq m + \frac{1}{m} - 2$, $m \geq 3$, *and* $\epsilon < 1$ *then the quantity*

$$g(\mathbf{r}) = \frac{\sqrt{(P_{exp}(\mathbf{r}) + 2m)^2 - \left(\|\nabla P_{exp}(\mathbf{r})\|_1 \epsilon / s\right)^2} - 2m}{P_{exp}(\mathbf{r})} \leq \left(1 - \frac{1}{6}\epsilon^2\right) . \tag{34}$$

Proof: Fix any $c \geq m + \frac{1}{m} - 2$, and consider $g(\mathbf{r})$ restricted to $\mathbf{r} \in S_c$. Once the potential is fixed, $g(\mathbf{r})$ is decreasing in $\|\nabla P_{exp}(\mathbf{r})\|_1$. Therefore it suffices to show the inequality for a vector in S_c that minimizes $\|\nabla P_{exp}(\mathbf{r})\|_1$. Lemma 4.8 shows that $\mathbf{r}^{(c)} = (r_1^{(c)}, 0, \ldots, 0)$ is one such vector.

The constraint on c implies that $\exp(s r_1^{(c)}) \geq m$. Set $a = \exp(s r_1^{(c)})/m$, so: $a \geq 1$, $P_{exp}(\mathbf{r}^{(c)}) = am + \frac{1}{am} - 2$, and $\|\nabla P_{exp}(\mathbf{r}^{(c)})\|_1 = am + 1/(am)$. We use a linear over-approximation of $\sqrt{x - y}$ around x to bound $g(\mathbf{r}^{(c)})$ as follows:

$$\begin{aligned}
g(\mathbf{r}^{(c)}) &= \frac{\sqrt{\left(am + \frac{1}{am} - 2 + 2m\right)^2 - \left(am - \frac{1}{am}\right)^2 \epsilon^2} - 2m}{am + \frac{1}{am} - 2} \\
&\leq \frac{am + \frac{1}{am} - 2 + 2m - \frac{\left(am - \frac{1}{am}\right)^2 \epsilon^2}{2\left(am + \frac{1}{am} - 2 + 2m\right)} - 2m}{am + \frac{1}{am} - 2} \\
&= 1 - \frac{\left(am - \frac{1}{am}\right)^2 \epsilon^2}{2(am + \frac{1}{am} - 2 + 2m)(am + \frac{1}{am} - 2)} \\
&= 1 - \frac{((am)^2 - 1)^2 \epsilon^2}{2((am)^2 + 1 - 2am + 2am^2)(am - 1)^2} \\
&= 1 - \frac{(am + 1)^2 \epsilon^2}{2((am)^2 + 2am^2 - 2am + 1)} \doteq h(a, m) .
\end{aligned}$$

We now show that $h(a, m)$ is decreasing in a.

$$\frac{\partial h(a, m)}{\partial a} = -\frac{(am^2 - 2am - m + 2)m(am + 1)\epsilon^2}{(a^2 m^2 + 2am^2 - 2am + 1)^2} ,$$

which is negative whenever $a \geq 1$ and $m \geq 3$. Therefore, $h(a, m)$ is maximized over $a \in [1, \infty)$ when $a = 1$, and

$$g\left(\mathbf{r}^{(c)}\right) \leq 1 - \frac{(m + 1)^2 \epsilon^2}{2(3m^2 - 2m + 1)} .$$

Taking the derivative of this bound with respect to m gives

$$\frac{2\epsilon^2 (2m - 1)(m + 1)}{(3m^2 - 2m + 1)^2}$$

which is positive when $m \geq 3$. Therefore

$$g\left(\mathbf{r}^{(c)}\right) \leq \lim_{m\to\infty}\left(1 - \frac{(m+1)^2\epsilon^2}{2(3m^2-2m+1)}\right) = 1 - \frac{\epsilon^2}{6}$$

as desired. □

The following 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 A.2. *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 natural numbers m and $\epsilon > 0$. Suppose in addition that $\mathcal{F} = -\mathcal{F}$ and $\mathcal{F}$ contains the zero function. For $V \geq 1$ define*

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

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 (1999). □

Acknowledgments

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.

Notes

1. In some circumstances a "bad" regressor can be "negated" to obtain a function with a positive edge. Previous boosting work for classification dealt with this by allowing the master algorithm to choose a negative coefficient α. Here we assume that the base regressor performs the negation so that the master's coefficients are always positive.
2. The Constructive algorithm of Lee, Bartlett, and Williamson (1995) is a regression algorithm for this agnostic framework.
3. Recall that the $\|\mathbf{f}\|_\infty$ normalization is equivalent to the assumption that $f(x) \in [-1, 1]$.
4. $\mathrm{SINC}(x) = \sin(\pi x)/\pi x$

References

Anthony, M., & Bartlett, P. L. (1999). *Neural network learning: Theoretical foundations*. Cambridge: Cambridge University Press.

Barron, A. R. (1993). Universal approximation bounds for superposition of a sigmoidal function. *IEEE Trans. on Information Theory, 39*, 930–945.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning, 36: 1/2*, 105–39.

Bertoni, A., Campadelli, P., & Parodi, M. (1997). A boosting algorithm for regression. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds.), *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks* (pp. 343–348). Berlin: Springer. Vol. V of LNCS.

Blake, C. E. K., & Merz, C. (1998). UCI repository of machine learning databases.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24:2*, 123–140.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics, 26:3*, 801–849.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation, 11*, 1493–1517.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth International Group.

Duffy, N., & Helmbold, D. (2000). Potential boosters? In S. Solla, T. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 258–264) Cambridge, MA: MIT Press.

Duffy, N. & Helmbold, D. (1999). A geometric approach to leveraging weak learners. In P. Fischer, & H. U. Simon (Eds.), *Computational learning theory: 4th European Conference (EuroCOLT '99)* (pp. 18–33). Berlin: Springer-Verlag.

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

Freund, Y. (1999). An adaptive version of the boost by majority algorithm. In *Proc. 12th Annu. Conf. on Comput. Learning Theory* (pp. 102–113). New York, NY: ACM Press.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new Boosting algorithm. In *Proc. 13th International Conference on Machine Learning* (pp. 148–156). San Matco, CA: Morgan Kaufmann.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55:1*, 119–139.

Friedman, J. H. (1999a). Greedy function approximation: A gradient boosting machine. Technical Report, Department of Statistics, Sequoia Hall, Stanford University, Stanford California 94305.

Friedman, J. H. (1999b). Stochastic gradient boosting. Technical Report, Department of Statistics, Sequoia Hall, Stanford University, Stanford California 94305.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics, 28:2*, 337–374.

Drucker, H. (1997). Improving regressors using boosting techniques. In *Proceedings of the Fourteenth International Converence on Machine Learning* (pp. 107–115). San Matco, CA: Morgan-Kaufman.

Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics, 20*, 608–613.

Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM, 41:1*, 67–95.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory*. Cambridge, MA: The MIT Press.

Lee, W. S., Bartlett, P. L., & Williamson, R. C. (1995). On efficient agnostic learning of linear combinations of basis functions. In *Proc. 8th Annu. Conf. on Comput. Learning Theory* (pp. 369–376). New York, NY: ACM Press.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. In S. Solla, T. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 512–518). Cambridge, MA: MIT Press.

Quinlan, J. R. (1996). Bagging, Boosting and C4.5. In *Proceedings of the Thirteenth National Conference of Artificial Intelligence* (pp. 725–730). Cambridge, MA: AAAI Press MIT Press.

Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost'. *Machine Learning, 42:3*, 287–320.

Rätsch, G., Warmuth, M., Mika, S., Onoda, T., Lemm, S., & Müller, K. R. (2000). Barrier Boosting. In *Proc. 13th*

Annu. Conference on Comput. Learning Theory (pp. 170–179). San Francisco: Morgan Kaufmann.
Ridgeway, G., Madigan, D., & Richardson, T. (1999). Boosting methodology for regression pproblems. In D. Heckerman, & J. Whittaker (Eds.), *Proc. Artificial Intelligence and Statistics* (pp. 152–161).
Schapire, R. E. (1992). *The design and analysis of efficient learning algorithms*. Cambridge, MA: MIT Press.
Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics, 26:5*, 1651–1686.
Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning, 37:3*, 297–336.
Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM, 27:11*, 1134–1142.
Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Received September 29, 2000
Revised May 18, 2001
Accepted August 10, 2001
Final manuscript October 16, 2001