A Gradient-Based Boosting Algorithm for Regression Problems

RICHARD S. ZEMEL, *Department of Computer Science, University of Toronto, Toronto, ON M5S 1A4, CANADA.*
E-mail: zemel@cs.toronto.edu

TONIANN PITASSI, *Department of Computer Science, University of Toronto, Toronto, ON M5S 1A4, CANADA.*
E-mail: toni@cs.toronto.edu

Abstract

Adaptive boosting, in which several weak learners trained sequentially are combined to boost the overall algorithm performance, is one of the most significant current developments in machine learning. Recently adaptive boosting methods for classification problems have been derived as gradient descent algorithms. This formulation justifies key elements and parameters in the methods, all chosen to optimize a single common objective function. The same formulation has yet to be developed for adaptive boosting of regression problems. We propose a novel objective function for regression problems that leads to a simple boosting algorithm. We prove that this method reduces training error, and compare its performance to other regression methods.

Keywords: adaptive boosting algorithms, regression problems, machine learning

1 Introduction

Boosting is a technique originally developed in computational learning theory (Schapire, 1990). Its aim is to "boost" the small advantage that a *hypothesis* produced by a *weak* learner can achieve over random guessing, by using the weak learning procedure several times on a sequence of carefully constructed distributions. Later boosting methods, notably AdaBoost (Freund & Schapire, 1997), created simple yet powerful algorithms which proved to be easy to implement and yielded excellent results in practice (e.g., Breiman, 1998; Schwenk & Bengio, 1998).

The significance of boosting extends beyond the learning community. It has important links to computational complexity (Impagliazzo, 1995; Klivans & Servedio, 1999), such as hardness amplification methods that have been used to prove inapproximability of certain boolean functions. There also turns out to be a close connection between boosting, game theory, and linear programming (Freund & Schapire, 1996; Grove & Schuurmans, 1998).

Two crucial elements of the adaptive boosting algorithms are the way in which a new distribution is constructed for the learning procedure to produce the next hypothesis in the sequence, and the way in which the (just-better-than-chance) hypotheses are combined to produce a highly accurate output. Both of these involve a set of parameters, whose values appeared to be determined in an ad hoc manner. Recently adaptive boosting algorithms have been derived as gradient descent algorithms (Schapire

& Singer, 1998; Friedman et al., 1999; Mason et al., 1999). These formulations justify the parameter values as all serving to optimize a single common objective function. To date these optimization formulations of adaptive boosting have been developed for *classification* problems,but not for *regression* problems.We propose a new boosting algorithm for regression problems, in which a single common objective function determines the cost function used to produce the hypotheses, and the other parameters in the algorithm. In our method, standard algorithms may be used to produce hypotheses, and the algorithm is simple to implement and run. Finally, we provide a simple proof guaranteeing convergence on training error.

In the remainder of this paper, we first describe the basic adaptive boosting algorithm (for classification problems), and existing boosting methods for regression. We then present our novel method for boosting regression problems. Following the theorem and proof of error convergence, we describe empirical results comparing our method to others on some standard problems.

2 A Brief Summary of Boosting Methods

Adaptive boosting methods such as AdaBoost (Freund & Schapire, 1997) are simple modular algorithms that operate as follows. Let $g : \mathbf{X} \rightarrow Y$ be the function that we are trying to learn, where the label set Y is finite, typically binary-valued. The algorithm uses a learning procedure, which has access to n training examples, $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, drawn randomly from $\mathbf{X} \times Y$ according to distribution $\mathcal{D}$. After some amount of time, the learning procedure must output a hypothesis $f : \mathbf{X} \rightarrow Y$. The error of the hypothesis f is the expected value of a loss function on $f(\mathbf{x}), g(\mathbf{x})$ where $\mathbf{x}$ is chosen according to $\mathcal{D}$. A *strong* learning procedure is a procedure that given $\epsilon, \delta > 0$ and access to random examples, outputs with probability $1 - \delta$ a hypothesis with error at most ϵ, and where the running time is polynomial in $1/\epsilon$, $1/\delta$ and the number of examples. A *weak* learning procedure satisfies the same conditions, but where ϵ need only be better than 1/2.

Schapire (1990) showed that given any weak learning procedure, denoted **WeakLearn**, it is possible to "boost" it, or efficiently transform it into a strong learning procedure. The AdaBoost algorithm achieves this by calling **WeakLearn** multiple times, in a sequence of T stages, each time presenting it with a different distribution over $\mathbf{X}$ and finally combining all of the hypotheses. At stage t, the algorithm maintains a set of weights w_t^i over the training examples and a distribution $\mathcal{D}_t$ is computed by normalizing these weights. The initial distribution $\mathcal{D}_1$ is typically uniform, so $w_1^i = \frac{1}{n}$ for all $i \leq n$. The algorithm loops through these two steps:

1. At stage t, the distribution $\mathcal{D}_t$ is given to **WeakLearn**, which generates a hypothesis f_t. The error rate ϵ_t of f_t w.r.t. $\mathcal{D}_t$ is:

$$\epsilon_t = \sum_{i : f_t(\mathbf{x}^i) \neq y^i} w_t^i \Big/ \sum_{i=1}^{n} w_t^i$$

2. The new training distribution is obtained from the new weights:

$$w_{t+1}^i = w_t^i * (\epsilon_t/(1-\epsilon_t))^{1-|f_t(\mathbf{x}^i)-y^i|}$$

Intuitively, the new training distribution is chosen to focus the next hypothesis on the parts of the input space that are mis-classified by the existing hypotheses. Thus, w^{t+1} should increase whenever f_t errs on example i, but the exact choice of w^{t+1} did not appear to be derived to optimize any particular function, but rather just given and then shown to work.

After T stages, a test example $\mathbf{x}$ will be classified by a combined hypothesis which is a weighted majority of the outputs of the T hypotheses: $\widehat{y} = \mathbf{sgn}(\sum_{t=1}^{T} c_t f_t(\mathbf{x}))$. Each *combination coefficient* $c_t = \mathcal{L}((\infty - \epsilon_{\sqcup})/\epsilon_{\sqcup})$ takes into account the accuracy of hypothesis f_t with respect to its distributions. Again here, while it is clear that c_t should increase as ϵ_t decreases, the rationale for these particular c_t values are somewhat of a mystery.

The original boosting algorithm described above applies to binary-valued classification problems. Adaptive boosting methods have been extended to apply to multiple-valued classification problems, and regression problems. Intuitively, regression problems (i.e., with real-valued targets) present special difficulties because hypotheses may not just be right or wrong, but can be a little wrong or very wrong. Previous regression boosting methods overcome this difficulty using a variety of techniques. One method converts the regression problem into a series of binary classification problems (Freund & Schapire, 1997); this approach permits the application of some of the analysis developed for the classification methods but is unwieldy in practice. Avnimelech & Intrator (1999) use a threshold to evaluate a response as correct or incorrect, and attempt to minimize the number of incorrect cases. This produces a strong dependence on the choice of threshold. Drucker (1997) avoids this conversion to classification-style problems by using a continuous loss function for training new hypotheses and reweighting the patterns. This method scales the errors of each hypothesis based on its maximal error, which can lead to big changes in combination coefficients due to a singular large error.

3 An Objective Function for Boosting Regression Problems

None of these boosting algorithms for regression take advantage of the recent work showing how adaptive boosting for classification can be derived as an optimization algorithm (Schapire & Singer, 1998; Friedman et al., 1999; Mason et al., 1999). These methods allow the parameters c_t and w_t to be derived from an overall objective function, which uses the training set to estimate the expected error of the combined hypotheses. The same objective is used in the **WeakLearn** procedure, as the new hypothesis is the step in function space in the direction of steepest descent of this objective.

We adopt this approach to derive a gradient-based boosting algorithm for regression problems. The objective function is:

$$J_T = \frac{1}{n}\sum_{i=1}^{n} \exp\left(\sum_{t=1}^{T} c_t [(f_t(\mathbf{x}^i) - y^i)^2 - \tau]\right) \quad (3.1)$$

In this formulation, each hypothesis is trained to minimize the squared error of a weighted training distribution. The constant τ is a threshold used to demarcate correct from incorrect responses. This threshold is the single parameter of this algorithm that must be chosen in a problem-dependent manner.

Adopting this objective makes the costs on each example multiplicative in the costs of the individual hypotheses. This key feature allows a boosting method to be formulated in an optimization framework; two attractive properties follow: (1) the objective function used to optimize a hypothesis is formed simply by re-weighting the training distribution; and (2) the new weight of a training example only depends on its old weight and the error of the latest hypothesis on that example.
In our algorithm, training hypothesis T to minimize J_T, the cost after T stages of boosting, amounts to minimizing the cost of that hypothesis on the weighted training distribution:

$$\begin{aligned} J_T =& \frac{1}{n}\sum_{i=1}^{n} \exp(\sum_{t=1}^{T-1} c_t[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) \exp(c_T[(f_T(\mathbf{x}^i) - y^i)^2 - \tau]) \\ =& \sum_{i=1}^{n} w_T^i \exp(c_T[(f_T(\mathbf{x}^i) - y^i)^2 - \tau]) \end{aligned}$$

The training distribution for the next hypothesis is then formed by normalizing these updated weights.
In the standard AdaBoost algorithm, the combination coefficient c_t can be analytically determined by solving $\frac{\partial J_t}{\partial c_t} = 0$ for c_t. Unfortunately, one cannot analytically determine the combination coefficient c_t in our algorithm, but a simple line search can be used to find value of c_t that minimizes the cost J_t. Finally, minimizing J with respect to y produces a simple linear combination rule for the estimate: $\widehat{y} = \sum_t c_t f_t(\mathbf{x}) / \sum_t c_t$.
The algorithm can be summarized as follows:

New Boosting Algorithm

1. Input:
   - training set examples $(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n)$ with $y \in \Re$;
   - **WeakLearn**: a learning procedure that produces a hypothesis $f_t(\mathbf{x})$ whose accuracy on the training set is judged according to J
2. Choose initial distribution $p_1(\mathbf{x}^i) = p_1^i = w_1^i = \frac{1}{n}$
3. Iterate:
   - Call **WeakLearn** – minimize J_t with distribution $\mathbf{p}_t$
     (accept iff $\epsilon_t = \sum_i p_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau] < 1$)
   - Set c_t to minimize J_t (using line search)
   - Update training distribution

$$\begin{aligned} w_{t+1}^i =& w_t^i * \exp(c_t[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) \\ p_{t+1}^i =& w_{t+1}^i / \sum_j w_{t+1}^j \end{aligned}$$

4. Estimate output y on input $\mathbf{x}$:

$$\widehat{y} = \sum_t c_t f_t(\mathbf{x}) / \sum_t c_t$$

4 Proof of Convergence

We currently can prove the standard boosting property for our algorithm in the case where all combination coefficients $c_t = 1$. Let $p_t^i = w_t^i / \sum_{j=1}^n w_t^j$, and ϵ_t denote the error that hypothesis t makes on its distribution,

$$\epsilon_t = \sum_{i=1}^{n} p_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]$$

Theorem: *Assume that for all $t \leq T$, hypothesis t makes error ϵ_t on its distribution. If the combined output $\widehat{y}$ is considered to be in error iff $(\widehat{y} - y)^2 > \tau$, then the output of the boosting algorithm (after T stages) will have error at most ϵ, where*

$$\epsilon = P[(\widehat{y^i} - y^i)^2 > \tau] \leq \prod_{t=1}^{T} \epsilon_t$$

.

Proof: We follow the approach used in the AdaBoost proof (Freund & Schapire, 1997). We show that the sum of the weights at stage T is bounded above by the product of the ϵ_t's, while at the same time, for each input i that is incorrect, its corresponding weight w_T^i at stage T is significant.

$$\sum_{i=1}^{n} w_{T+1}^i = \sum_i w_T^i \exp[(f_T(\mathbf{x}^i) - y^i)^2 - \tau] = \epsilon_T \sum_i w_T^i = \prod_{t=1}^{T} \epsilon_t$$

The second equality holds because $\epsilon_t = (\sum_i w_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) / \sum_i w_t^i$. Now let $\widehat{y^i} = \sum_t f_t(\mathbf{x}^i)/T$. Then the weight of example i at time t is:

$$\begin{aligned} w_t^i &= \exp(\sum_t [(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) = \exp(\sum_t [((f_t(\mathbf{x}^i) - \widehat{y^i}) + (\widehat{y^i} - y^i))^2 - \tau]) \\ &= \exp(T[\mathbf{Var}(f(\mathbf{x}^i)) + (\widehat{y^i} - y^i)^2 - \tau]) \geq \exp(T[(\widehat{y^i} - y^i)^2 - \tau]) \end{aligned}$$

The last inequality above holds because $\mathbf{Var}(f(\mathbf{x}^i)) = 1/T \sum_t (f_t(\mathbf{x}^i) - \widehat{y^i})^2$.
Now consider an example input i such that the final answer is an error. Then we have $(\widehat{y^i} - y^i)^2 > \tau \quad \Rightarrow \quad w_t^i \geq 1$ If ϵ is the total error rate of the combination output, then $\sum_i w_t^i \geq \epsilon$. Thus, as desired,

$$\epsilon \leq \sum_i w_{T+1}^i = \prod_{t=1}^{T} \epsilon_t$$

Note that as in the binary AdaBoost theorem, there are no assumptions made here about ϵ_t, the error rate of individual hypotheses. Also, if all $\epsilon_t < 1 = \Delta$, where $\Delta < 1$, then $\epsilon < \Delta^T$.

5 Empirical Tests of Algorithm

We report results comparing the performance of our new algorithm with two other algorithms. The first is the sequential residual-fitting algorithm (Hastie & Tibshirani, 1990), which produces hypotheses sequentially to reduce *residual* error. Here hypothesis T is trained not to produce the target output y^i on a given case i, but instead to fit the current residual, $\widetilde{y}$, where $\widetilde{y}^i = y^i - \sum_{t=1}^{T-1} c_t f_t(\mathbf{x})$. We used a least-squares loss function; Friedman (1999) explored the performance of this algorithm under various loss criteria. Note that this algorithm is similar to ours in that it develops a series of hypotheses all based on optimizing a common objective. It deviates from standard boosting, however, in that the distribution of examples is not used to control the generation of hypotheses, and each hypothesis is not trained to learn the same underlying function. Karakoulas and Shawe-Taylor (1999) have proposed a more standard boosting method based on residual-fitting.

The second algorithm used for comparison is a version of the mixture-of-experts algorithm (Jacobs et al., 1991). Here the hypotheses (or experts) are trained simultaneously. The standard mixture-of-experts uses a gating network to combine the hypotheses, and these combination coefficients also depend on the input. We set the weights from the input into the gating network to zero, and only adapted the biases to these units in order to make this model comparable to the others. This algorithm provides a good alternative to the greedy stage-wise methods, in that the experts are trained simultaneously to collectively fit the data.

We have selected two test problems on which to evaluate these algorithms. The first was proposed by Friedman (1991). F1 is a nonlinear prediction problem which has 10 independent input variables that are uniform in $[0, 1]$.

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - .5)^2 + 10x_4 + 5x_5 + n$$

where n is a random variable drawn from a mean-zero, unit-variance normal distribution. In this problem, only five input variables (x_1 to x_5) are needed to predict the correct output y, but the predictor must try to distinguish those input variables that have no prediction ability (x_6 to x_{10}) from those that do. We rescaled the target values y to be in $[0, 3]$. We used 400 training examples, and 100 validation and 100 test examples. The second test problem is the Boston Housing problem, a standard regression problem from the UCI machine learning database. Here there are 506 examples with the dependent variable being the price of housing in the Boston area, and twelve continuous input variables. We scaled the input variables to be in $[0, 1]$, and the outputs to be in $[0, 3]$. We used 400 of the examples for training, 50 for validation, and the remainder to test generalization.

In these experiments, we used neural networks as the hypotheses and back-propagation as the learning procedure to train them. Each network had a layer of 3 $tanh()$ units between the input units and a single linear output. For all three algorithms, we used early stopping with a validation set in order to reduce over-fitting in the hypotheses. Figure 1 shows average results over 20 runs of each algorithm on the two problems, with each run involving a new assignment of examples into the three sets. For the results shown in the figure, $\tau = 0.1$ in our algorithm. The results were consistent for different values of τ. All three algorithms steadily reduce the training error. The critical comparison considers their performance on the test examples. We measure test error as the normalized mean squared error (NMSE), the MSE divided by the variance across the data. As can be seen in the figure, the residual-fitting algorithm initially out-performs our boosting algorithm. Yet with more stages of boosting, the test error of the residual-fitting algorithm began to *increase*, which means that the combined

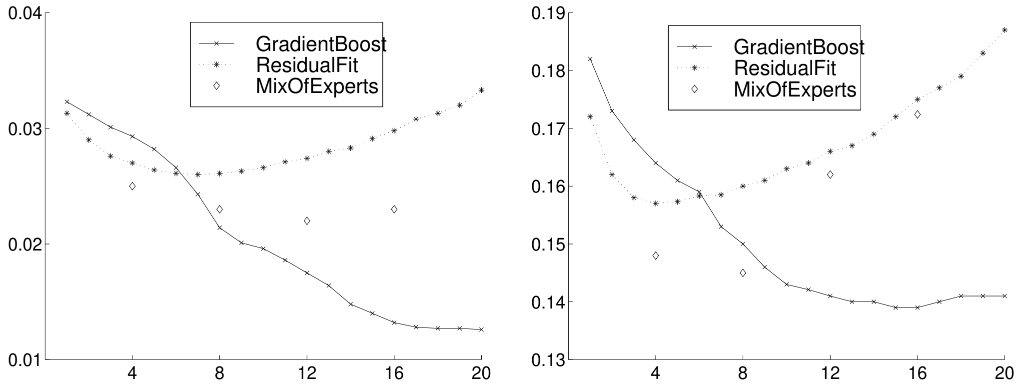

FIG. 1: Generalization results for our gradient-based boosting algorithm, compared to the residual-fitting and mixture-of-experts algorithms. Left: Test problem F1; Right: Boston housing data. Normalized mean-squared error is plotted against the number of stages of boosting (or number of experts for the mixture-of-experts).

hypotheses were over-fitting the training examples. This was not the case for our algorithm, as the test error steadily decreased until it reached a plateau (although some over-fitting was observed after many stages). The mixture-of-experts algorithm generally out-performed the sequential training algorithms for a small number of experts, but its performance was also not as good as the gradient-based boosting after many stages. The results followed this same pattern for different numbers of hidden units in the neural networks.

6 Conclusion

We have proposed a new boosting algorithm for regression problems. This algorithm shares properties with existing boosting methods for regression, but its collection of key features make it unique. A single common objective function is both used by the weak learning procedure to produce hypotheses, and determines the other parameters in the algorithm. A simple proof guarantees convergence on training error. The distribution of examples is used to control the generation of hypotheses, and each hypothesis is trained to learn the same underlying function. Standard algorithms, such as back-propagation, may be used to produce the hypotheses, training to minimize a weighted cost function based on mean-squared error. And finally, a threshold is used to determine if a response is correct or not.

Empirical investigations have suggested that this algorithm is less prone to over-fitting than another greedy stage-wise approach, the residual-fitting algorithm. Both algorithms, however, will over-fit after a sufficient number of stages. We are currently exploring a regularization technique known as shrinkage, where the combination coefficients are scaled by a parameter less than one. This method has been found to improve generalization in gradient boosting applications to classification (Friedman, 1999). Our empirical studies also suggest that the greedy stage-wise approach can out-perform a simultaneous training method, like the mixture of experts. Generally one would expect a simultaneous approach to be less prone to local minima problems than a greedy one, but an important advantage of the boosting methods is that the number of experts need not be specified in advance.

We are currently exploring an extension of our gradient boosting method, one which brings it a step closer to the full mixture-of-experts algorithm. The combination coefficient c_t associated with hypothesis t can be input-dependent – a learner can return not only $f_t(\mathbf{x}^i)$ but also $k_t(\mathbf{x}^i) \in [0,1]$, a measure of confidence in its prediction. Other sequential regression algorithms do not allow this form of specialization, although it is mentioned in (Freund & Schapire, 1997). This elaboration makes the weak learning task harder, but may extend the range of applicability of the algorithm: letting each learner focus on a subset of its weighted training distribution permits a divide-and-conquer approach to function approximation.

Acknowledgements

This work was funded by ONR N00014-98-1-0509 to RZ; NSF grants CCR-9457782 and CCR-9820831, and US-Israel BSF Grant 95-00238 to TP.

References

[1] Avnimelech, R. & Intrator, N. (1999). Boosting regression estimators. *Neural Computation, 11*, 499-520.

[2] Breiman, L. (1996). Bagging predictors. *Machine Learning, 26*, 123-140.

[3] Drucker, H. (1997). Improving regressors using boosting techniques. In *14th International Conference on Machine Learning*. Morgan Kaufmann.

[4] Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156.

[5] Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and System Sci., 55*, 119-139.

[6] Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics, 19*, 1-82.

[7] Friedman, J. H. (1999). Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University.

[8] Friedman, J. H., Hastie, T., & Tibshirani, R. (1999). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, To appear.

[9] Grove, A. J. & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intellige nce.*

[10] Hastie, T. & Tibshirani, R. (1990). *Generalized Additive Models.* Chapman and Hall.

[11] Impagliazzo, R. (1995). Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, 538-545.

[12] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79-87.

[13] Karakoulas, G., & Shawe-Taylor, J. (1999). Towards a strategy for boosting regressors. In *Advances in Large Margin Classifiers*, Smola, Bartlett, Schölkopf & Schuurmans (Eds.).

[14] Klivans, A. R. & Servedio, R. A. (1999). Boosting and hard-core sets. In *40th Annual Symposium on Foundations of Computer Science*, 624-633.

[15] Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Boosting algorithms as gradient descent in function space. In *Advances in Neural Information Processing Systems 11*. Morgan Kaufmann.

[16] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5*, 197-227.

[17] Schapire, R. E. & Singer, Y. (1998). Improved boosting algorithms using confidence-rated precitions. In *Proceedings of the 11th COLT Conference.*

[18] Schwenk, H. & Bengio, Y. (1998). Training methods for adaptive boosting of neural networks for character recognition. In *NIPS 10*. Morgan Kaufmann, 647-653.