Intelligible Models for Classification and Regression

Yin Lou
Dept. of Computer Science
Cornell University
yinlou@cs.cornell.edu

Rich Caruana
Microsoft Research
Microsoft Corporation
rcaruana@microsoft.com

Johannes Gehrke
Dept. of Computer Science
Cornell University
johannes@cs.cornell.edu

ABSTRACT

Complex models for regression and classification have high accuracy, but are unfortunately no longer interpretable by users. We study the performance of generalized additive models (GAMs), which combine single-feature models called *shape functions* through a linear function. Since the shape functions can be arbitrarily complex, GAMs are more accurate than simple linear models. But since they do not contain any interactions between features, they can be easily interpreted by users.

We present the first large-scale empirical comparison of existing methods for learning GAMs. Our study includes existing spline and tree-based methods for shape functions and penalized least squares, gradient boosting, and backfitting for learning GAMs. We also present a new method based on tree ensembles with an adaptive number of leaves that consistently outperforms previous work. We complement our experimental results with a bias-variance analysis that explains how different shape models influence the additive model. Our experiments show that shallow bagged trees with gradient boosting distinguish itself as the best method on low- to medium-dimensional datasets.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Learning—*Induction*

Keywords

intelligible models, classification, regression

1. INTRODUCTION

Everything should be made as simple as possible, but not simpler.
— Albert Einstein.

Classification and regression are two of the most important data mining tasks. Currently, the most accurate methods on many datasets are complex models such as boosted trees, SVMs, or deep neural nets. However, in many applications *what* is learned is just as important as the accuracy of the predictions. Unfortunately, the high accuracy of complex models comes at the expense of interpretability; e.g., even the contribution of individual features to the predictions of a complex model are difficult to understand.

KDD'12, August 12–16, 2012, Beijing, China.

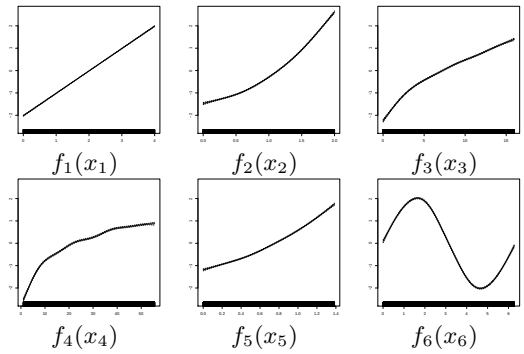


Figure 1: Shape Functions for Synthetic Dataset in Example 1.

The goal of this work is to construct accurate models that are interpretible. By interpretibility we mean that users can understand the contribution of individual features in the model; e.g., we want models that can quantify the impact of each predictor. This desiderata permits arbitrary complex relationships between individual features and the target, but excludes models with complex interactions between features. Thus in this paper we fit models of the form:

$$g(y) = f_1(x_1) + ... + f_n(x_n), \tag{1}$$

which are known as *generalized additive models* in the literature [15, 22]. The function $g(\cdot)$ is called the *link function* and we call the f_is *shape functions*. If the link function is the identity, Equation 1 describes an additive model (e.g., a regression model); if the link function is the logistic function, Equation 1 describes a generalized additive model (e.g., a classification model).

EXAMPLE 1. *Assume we are given a dataset with 10,000 points generated from the model* $y = x_1 + x_2^2 + \sqrt{x_3} + \log(x_4) + \exp(x_5) + 2\sin(x_6) + \epsilon$, *where* $\epsilon \sim \mathcal{N}(0, 1)$. *After fitting an additive model to the data of the form shown in Equation 1, we can visualize the contribution of* x_i*s as shown in Figure 1: Because predictions are a linear function of the* $f_i(x_i)$*, scatterplots of* $f_i(x_i)$ *on the y-axis vs.* x_i *on the x-axis allow us to visualize the shape function that relates the* $f_i(x_i)$ *to the* x_i*, thus we can easily understand the contribution of* x_i *to the prediction.*

Because the data in Example 1 was drawn from a model with no interactions between features, a model of the form in Equation 1 is able to fit the data perfectly (modulo noise). However, data are not always so simple in practice. As a second example, consider a real dataset where there may be interactions between features.

Model	Form	Intelligibility	Accuracy
Linear Model	$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$	+++	+
Generalized Linear Model	$g(y) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$	+++	+
Additive Model	$y = f_1(x_1) + ... + f_n(x_n)$	++	++
Generalized Additive Model	$g(y) = f_1(x_1) + ... + f_n(x_n)$	++	++
Full Complexity Model	$y = f(x_1, ..., x_n)$	+	+++

Table 1: From Linear to Additive Models.

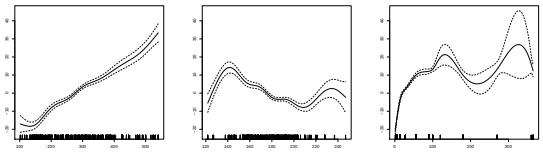


Figure 2: Shape Functions for Concrete Dataset in Example 2.

EXAMPLE 2. *The "Concrete" dataset relates the compressive strength of concrete to its age and ingredients. The dataset contains 1030 points with eight numerical features. We again fit an additive model of the form in Equation 1. Figure 2 shows scatterplots of the shape functions learned for three of the eight features. As we can see from the figure, the compressibility of concrete depends nearly linearly on the Cement feature, but it is a complex non-linear function of the Water and Age features; we say that the model has* shaped *these features. A linear model without the ability to shape features would have worse fit because it cannot capture these non-linearities. Moreover, an attempt to interpret the contribution of features by examining the slopes of a simple linear model would be misleading; the additive model yields much better fit to the data while still remaining intelligible.*[1]

As we saw in the examples, additive models explicitly decompose a complex function into one-dimensional components, its shape functions. Note, however, that the shape functions themselves may be non-linear: Each feature x_i can have a complex non-linear shape $f_i(x_i)$, and thus the accuracy of additive models can be significantly higher than the accuracy of simple linear models. Table 1 summarizes the differences between models of different complexity that we consider in this paper. Linear models, and generalized linear models (GLMs) are the most intelligible, but often the least accurate. Additive models, and generalized additive models (GAMs) are more accurate than GLMs on many data sets because they capture non-linear relationships between (individual) features and the response, but retain much of the intelligibility of linear models. Full complexity models such as ensembles of trees are more accurate on many datasets because they model both non-linearity and interaction, but are so complex that it is nearly impossible to interpret them.

In this paper we present the results of (to the best of our knowledge) the largest experimental study of GAMs. We consider shape functions based on splines [14, 22] and boosted stumps [13], as well as novel shape functions based on bagged and boosted ensembles of trees that choose the number of leaves adaptively. We experiment with (iteratively re-weighted) least squares, gradient boosting, and backfitting to both iteratively refine the shape functions and construct the linear model of the shaped features. We apply these methods to six classification and six regression tasks. For comparison, we fit simple linear models as a baseline. We also fit unrestricted ensembles of trees as full complexity models to get an idea of what accuracy is achievable.

[1]See Section 4 for the fit of different models to this dataset.

Model	Regression	Classification	Mean
Linear/Logistic	1.68	1.22	1.45
P-LS/P-IRLS	1.00	1.00	1.00
BST-SP	1.03	1.00	1.02
BF-SP	1.00	1.00	1.00
BST-bagTR2	0.96	0.96	0.96
BST-bagTR3	0.97	0.94	0.96
BST-bagTR4	0.99	0.95	0.97
BST-bagTRX	0.95	0.94	0.95
Random Forest	0.88	0.80	0.84

Table 2: Preview of Empirical Results.

Table 2 summarizes the key findings of our study. Entries in the table are the average accuracies on the regression and classification datasets, normalized by the accuracy of Penalized (Iteratively Reweighted) Least Squares with Splines (P-LS/P-IRLS). As expected, the accuracy of GAMs falls between that of linear/logistic regression without feature shaping and full-complexity models such as random forests. However, surprisingly, the best GAM models have accuracy much closer to the full-complexity models than to the linear models. Our results show that bagged trees with 2-4 leaves as shape functions in combination with gradient boosting as learning method (Methods BST-bag-TR2 to BST-bag-TR4) outperform all other methods on most datasets. Our novel method of adaptively selecting the right number of leaves (Method BST-bagTRX) is almost always even better, and thus we recommend it as the method of choice. On average, this method reduces loss by about 5% over previous GAM models, a significant improvement in practice.

The rest of the paper is structured as follows. Section 2 presents algorithms for fitting generalized additive models with various shape functions and learning methods. Section 3 describes our experimental setup, Section 4 presents the results and their interpretation, followed by a discussion in Section 5 and an overview of related work in Section 6. We conclude in Section 7.

2. METHODOLOGY

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_1^N$ denote a training dataset of size N, where $\boldsymbol{x}_i = (x_{i1}, ..., x_{in})$ is a feature vector with n features and y_i is the target. In this paper, we consider both regression problems where $y_i \in \mathbb{R}$ and binary classification problems where $y_i \in \{1, -1\}$. Given a model F, let $F(\boldsymbol{x}_i)$ denote the prediction of the model for data point $\boldsymbol{x}_i$. Our goal in both classification and regression is to minimize the expected value of some loss function $L(y, F(\boldsymbol{x}))$.

We are working with generalized additive models of the form in Equation 1. To train such models we have to select (i) the shape functions for individual features and (ii) the learning method used to train the overall model. We discuss these two choices next.

2.1 Shape Functions

In our study we consider two classes of shape functions: regression splines and trees or ensembles of trees. Note that all shape functions relate a single attribute to the target.

Regression Splines. We consider regression splines of degree d of the form $y = \sum_{k=1}^{d} \beta_k b_k(x)$.

Trees and Ensembles of Trees. We also use binary trees and ensembles of binary trees with largest variance reduction as split selection method. We control tree complexity by either fixing the number of leaves or by disallowing leaves that have fewer than an α-fraction of the number of training examples.

We consider the following ensemble variants:

- Single Tree. We use a single regression tree as a shape function.
- Bagged Trees. We use the well-known technique of bagging to reducing variance [6].
- Boosted Trees. We use gradient boosting, where each successive tree tries to predict the overall residual from all preceding trees [12].
- Boosted Bagged Trees. We use a bagged ensemble in each step of stochastic gradient boosting [13], resulting in a boosted ensemble of bagged trees.

2.2 Generalized Additive Models

We consider three different methods for fitting additive models in our study: Least squares fitting for learning regression spline shape functions, and gradient boosting and backfitting for learning tree and tree ensemble shape functions. We review them here briefly for completeness although we would like to emphasize that these methods are not a contribution of this paper.

2.2.1 Least Squares

Fitting a spline reduces to learning the weights $\beta_k(x)$ for the basis functions $b_k(x)$. Learning the weights can be reduced to fitting a linear model $y = \boldsymbol{X}\boldsymbol{\beta}$, where $\boldsymbol{X}_i = [b_1(x_{i1}), ..., b_k(x_{in})]$; the coefficients of the linear model can be computed exactly using the least squares method [22]. To control smoothness, there is a "wiggliness" penalty: we minimize $\|y - \boldsymbol{X}\boldsymbol{\beta}\| + \lambda \sum_i \int [f_i''(x_i)]^2 dx$ with the smoothing parameter λ. Large values of λ lead to a straight line for f_i while low values of λ allow the spline to fit closely to the data. We use thin plate regression splines from the R package "mgcv" [22] that automatically selects the best values for the parameters of the splines [21]. We call this method penalized least-squares (P-LS) in our experiments.

The fitting of an additive logistic regression model using splines is similarly reduced to fitting a logistic regression with a different basis, which can be solved using penalized-iteratively reweighted least squares (P-IRLS) [22].

2.2.2 Gradient Boosting

We use standard gradient boosting [12, 13] with one difference: Since we want to learn shape functions for all features, in each iteration of boosting we cycle sequentially through all features. For completeness, we include pseudo-code in Algorithms 1 and 2. In Algorithm 1, we first set all shape functions to zero (Line 1). Then we loop over M iterations (Line 2) and over all features (Line 3) and then calculate the residuals (Line 4). We then learn then one-dimensional function to predict the residuals (Line 5) and add it to the shape function (Line 6).

2.2.3 Backfitting

A popular algorithm for learning additive models is the backfitting algorithm [15]. The algorithm starts with an initial guess of all shape functions (such as setting them all to zero). The first shape function f_1 is then learned using the training set with the goal to predict y. Then we learn the second shape function f_2 on the residuals $y - f_1(x_1)$, i.e., using training set $\{(x_{i2}, y - f_1(x_{i1}))\}_1^N$. The third shape function is trained on the residuals $y - f_1(x_1) - f_2(x_2)$, and so on. After we have trained n shape functions, the first shape function is *discarded* and retrained on the residuals of the other $n-1$ shape functions. Note that backfitting is a form of the "Gauss-Seidel" algorithm and its convergence is usually guaranteed [15]. Its pseudocode looks identical to Algorithm 1 except that Line 6 is replaced by $f_j \leftarrow S$.

To fit an additive logistic regression model, we can use a generalized version of the backfitting algorithm called the "Local Scoring Algorithm" [15], which is a general method for fitting generalized additive models. We form the response

$$\tilde{y}_i = F(\boldsymbol{x}_i) + \frac{\mathbf{1}(y_i = 1) - p(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i)(1 - p(\boldsymbol{x}_i))},$$

where $p(\boldsymbol{x}_i) = \frac{1}{1+\exp(-F(\boldsymbol{x}_i))}$. We then apply the weighted backfitting algorithm to the response $\tilde{y}_i$ with observation weights $p(\boldsymbol{x}_i)$ $(1 - p(\boldsymbol{x}_i))$ [15].

Algorithm 1 Gradient Boosting for Regression

1: $f_j \leftarrow 0$
2: **for** $m = 1$ to M **do**
3: **for** $j = 1$ to n **do**
4: $\mathcal{R} \leftarrow \{x_{ij}, y_i - \sum_k f_k\}_1^N$
5: Learn shaping function $S : x_j \rightarrow y$ using $\mathcal{R}$ as training dataset
6: $f_j \leftarrow f_j + S$

Algorithm 2 Gradient Boosting for Classification

1: $f_j \leftarrow 0$
2: **for** $m = 1$ to M **do**
3: **for** $j = 1$ to n **do**
4: $\tilde{y}_i \leftarrow \frac{2y_i}{1+\exp(2y_i F(\boldsymbol{x}_i))}, i = 1, ..., N$
5: Learn $\{R_{km}\}_1^K \leftarrow$, a tree with K leaf nodes using $(\{x_{ij}, \tilde{y}_i\}_1^N)$ as training dataset
6: $\gamma_{km} = \frac{\sum_{x_{ij} \in R_{km}} \tilde{y}_i}{\sum_{x_{ij} \in R_{km}} |\tilde{y}_i|(2-|\tilde{y}_i|)}, k = 1, ..., K$
7: $f_j \leftarrow f_j + \sum_{k=1}^K \gamma_{km} \mathbf{1}(x_{ij} \in R_{km})$

3. EXPERIMENTAL SETUP

In this section we describe the experimental design.

3.1 Datasets

We selected datasets of low-to-medium dimensionality with at least 1000 points. Table 3 summarizes the characteristics of the 12 datasets. One of the regression datasets is a synthetic problem used to illustrate feature shaping (but we do not use the results on this dataset when comparing the accuracy of the methods).

The "Concrete," "Wine," and "Music" regression datasets are from the UCI repository [1]; "Delta" is the task of controlling the ailerons of a F16 aircraft [2]; "CompAct" is a regression dataset from the Delve repository that describes the state of multiuser computers [3]. The synthetic dataset was described in Example 1.

The "Spambase," "Insurance," "Magic," "Letter" and "Adult" classification datasets are from the UCI repository. "Adult" contains nominal attributes that we transformed to boolean attributes

Dataset	Size	Attributes	%Pos
Concrete	1030	9	-
Wine	4898	12	-
Delta	7192	6	-
CompAct	8192	22	-
Music	50000	90	-
Synthetic	10000	6	-
Spambase	4601	58	39.40
Insurance	9823	86	5.97
Magic	19020	11	64.84
Letter	20000	17	49.70
Adult	46033	9/43	16.62
Physics	50000	79	49.72

Table 3: Datasets.

Shape Function	Least Squares	Gradient Boosting	Backfitting
Splines	P-LS/P-IRLS	BST-SP	BF-SP
Single Tree	N/A	BST-TRx	BF-TR
Bagged Trees	N/A	BST-bagTRx	BF-bagTR
Boosted Trees	N/A	BST-TRx	BF-bstTRx
Boosted Bagged Trees	N/A	BST-bagTRx	BF-bbTRx

Table 4: Notation for learning methods and shape functions.

(one boolean per value). "Letter" has been converted to a binary problem by using A-M as positives and the rest as negatives. The "Physics" dataset is from the KDD Cup 2004 [4].

3.2 Methods

Recall from Section 2 that we have two different types of shape functions and three different methods of learning generalized additive models; see Table 4 for an overview of these methods and their names. While penalized least squares for regression (P-LS) and penalized iteratively re-weighted least squares for classification (P-IRLS) only work with splines, gradient boosting and backfitting can be applied to both splines and ensembles of trees.

In gradient boosting, we vary the number of leaves in the bagged or boosted trees: $2, 3, 4, 8, 12$ to 16 (indicated by appending this number to the method names). Trained models will contain M such trees for each shape function after M iterations. In backfitting, we re-build the shape function for each feature from scratch in each round, so the shape function needs to have enough expressive power to capture a complex function. Thus we control the complexity of the tree not by the number of leaves, but by adaptively choosing a parameter α that stops splitting nodes smaller than an α fraction of the size of the training data; we vary $\alpha \in \{0.00125, 0.025, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$. A summary of the combinations of shape functions and learning methods can be found in Table 4.

Beyond the parameters that we already discussed, P-LS and P-IRLS have a parameter λ, which is estimated using generalized cross validation as discussed in Section 2. We do not fix the number of iterations for gradient boosting and backfitting but instead run these methods until convergence as follows: We divide the training set into five partitions. We then set aside one of the partitions as a validation set, train the model on the remaining four partitions, and use the validation set to check for convergence. We repeat this process five times and then compute M, the average number of iterations until convergence across the five iterations. We then retrain the model using the whole training set for M iterations. We follow a similar procedure for backfitting where we pick the best α for each partition and average them to train the final model using the whole training dataset.

3.3 Metrics

For regression problems, we report the root mean squared error (RMSE) for linear regression (no feature shaping), additive models with shaping with splines or trees (penalized least squares, gradient boosting and backfitting), and unrestricted full-complexity models (random forest regression trees and Additive Groves [5, 19]).

For classification problems, we report the error rates for logistic regression, generalized additive models with splines or trees (penalized iteratively re-weighted least squares, gradient boosting and backfitting), and full-complexity unrestricted models (random forests [8]).[2]

In all experiments we use 100 trees for bagging. We do not notice significant improvements by using more iterations of bagging. For Additive Groves, the number of trees is automatically selected by the algorithm on the validation set. For P-LS and P-IRLS, we use an R package called "mgcv" [22]. We perform 5-fold cross validation for all experiments.[3]

4. RESULTS

The regression and classification results are presented in Table 5 and Table 6, respectively. We report means and standard deviations on the 5-fold cross validation test-sets. To facilitate comparison across multiple datasets, we compute normalized scores that average the performance of each method across the datasets, normalized by the accuracy of P-LS/P-IRLS on each dataset.

Table 5 and Table 6 are laid out as follows: The top of each table shows results for linear/logistic regression (no feature shaping) and the traditional spline-based GAM models P-LS/P-IRLS, BST-SP, and BF-SP. The middle of the tables present results for new methods that do feature shaping with trees instead of splines such as boosted size-limited trees (e.g., BST-TR3), boosted-bagged size-limited trees (e.g., BST-bagTR3), backfitting of boosted trees (e.g., BF-bstTR3), and backfitting of boosted-bagged trees (e.g., BF-bbTR3). The bottom of each table presents results for unrestricted full-complexity models such as random forests and additive groves. Our goal is to devise more powerful GAM models that are as close in accuracy as possible to the full-complexity models, while preserving the intelligibility of linear models.

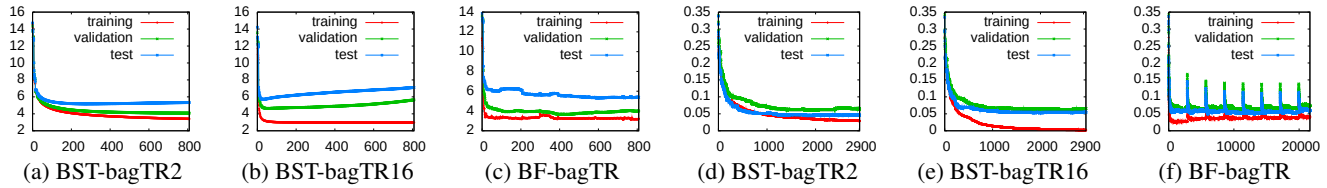
Several clear patterns emerge in both tables.

(1) There is a large gap in accuracy between linear methods that do not do feature shaping (linear or logistic regression) and most methods that perform feature shaping. For example, on average the spline-based P-LS GAM model has 60% lower normalized RMSE than vanilla linear regression. Similarly, on average, P-IRLS is about 20% more accurate than logistic regression.

(2) The new tree-based shaping methods are more accurate than the spline-based methods as long as model complexity (and variance — see Section 5.1) is controlled. In both tables, the most accurate tree-based GAM models use boosted-bagged trees that are size-limited to 2-4 leaves.

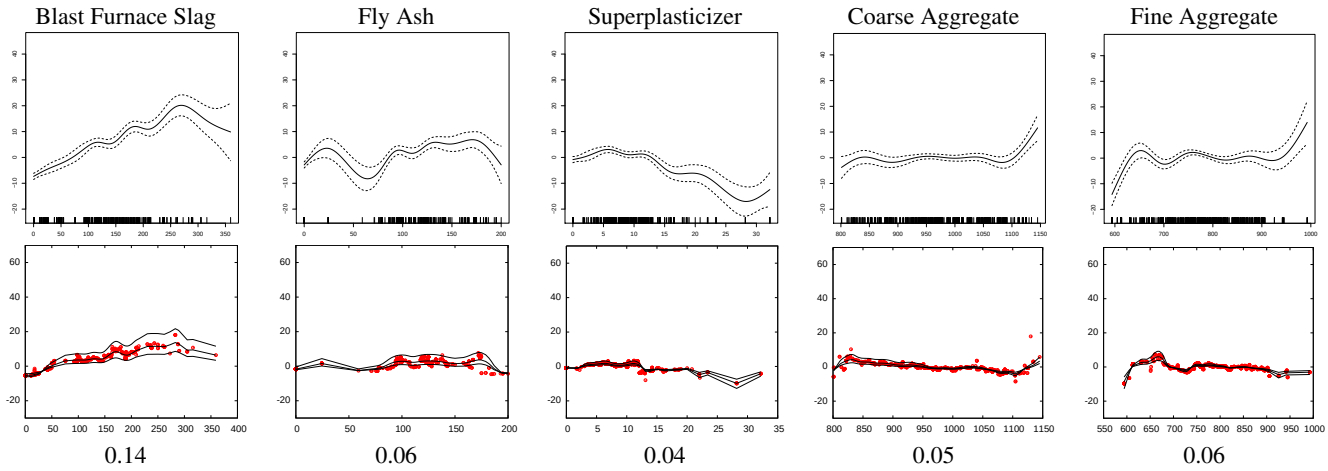
(3) Unrestricted full-complexity models such as random forests and additive groves are more accurate than any of the GAM models because they are able to model feature interactions, which linear models of shaped features cannot capture. Our goal is to push the

[2]Random forests is a very competitive full complexity model [10].
[3]We use 5-fold instead of 10-fold cross validation because some of the experiments are very expensive.



(a) BST-bagTR2 (b) BST-bagTR16 (c) BF-bagTR (d) BST-bagTR2 (e) BST-bagTR16 (f) BF-bagTR

Figure 3: Training curves for gradient boosting and backfitting. Figure (a), (b) and (c) show the behavior of BST-bagTR2, BST-bagTR16 and BF-bagTR on the "Concrete" regression problem, respectively. Figure (d), (e) and (f) illustrate behavior of BST-bagTR2, BST-bagTR16 and BF-bagTR on the "Spambase" classification, respectively.



0.14 0.06 0.04 0.05 0.06

Figure 4: Shapes of features for the "Concrete" dataset produced by P-LS (top) and BST-bagTR3 (bottom).

accuracy of linear shaped models as close as possible to the accuracy of these unrestricted full-complexity models.

Looking more closely at the results for models that shape features with trees, the most accurate model on average is BST-bagTR2 for regression, and BST-bagTR3 for classification. Models that use more leaves are consistently less accurate than comparable models with 2-4 leaves. It is critical to control tree complexity when boosting trees for feature shaping. Moreover, the most accurate methods used bagging inside of boosting to reduce variance. (More on model variance in Section 5.1.) Finally, on the regression problems, methods based on gradient boosting of residuals slightly edged out the methods based on backfitting, though the difference is not statistically significant. On the classification problems, however, where backfitting is performed on pseudo-residuals, there were stability problems that caused some runs to diverge or fail to terminate. Overall, tree-based shaping methods based on gradient-boosting appear to be preferable to tree-based methods based on backfitting because the gradient boosting methods may be a little more accurate, are often faster, and on some problems are more robust.

Although tree-based feature shaping yields significant improvements in accuracy for GAMs, on most problems they are not able to close the gap with unrestricted full-complexity models such as random forests. For example, all linear methods have much worse RMSE on the wine regression problem than the unrestricted random forest model. On problems where feature interaction is important, linear models without interaction terms must be less accurate.

4.1 Model Selection

There is a risk when comparing many parameterizations of a new method against a small number of baseline methods, that the new method will appear to be better because selecting the best model on the test set leads to overfitting to the test sets. To avoid this, the table includes results for a method called "BST-bagTRX" that uses the cross-validation validation sets (*not* the CV test sets) to pick the best parameters from the BST-bagTRx models for each dataset. This method is not biased by looking at results on test sets, is fully automatic and thus does not depend on human judgement, and is able to select different parameters for each problem. The results in Table 5 and Table 6 suggest that BST-bagTRX is more accurate than any single fixed parameterization. Looking at the models selected by BST-bagTRX, we see that BST-bagTRX usually picks models with 2, 3 or 4 leaves, and that the model it selects often is the one with the best test-set performance. On both the regression and classification datasets, BF-bagTRX is significantly more accurate than any of the models that use splines for feature shaping.

5. DISCUSSION

5.1 Bias-Variance Analysis

The results in Tables 5 and 6 show that adding feature shaping to linear models significantly improves accuracy on problems of small-medium dimensionality, and feature shaping with tree-based models significantly improves accuracy compared to feature shaping with splines. But why are tree-based methods more accurate for feature shaping than spline-based methods? In this section we show that splines tend to underfit, i.e., have very low variance at the expense of higher bias, but tree-based shaping models can have both low bias and low variance if tree complexity is controlled.

To show why spline models do not perform as well as tree models, and why controlling complexity is so critical with trees, we perform a bias-variance analysis on the regression datasets.[4] As in previous experiments, we randomly select 20% of the points as test

[4]We do not perform bias-variance analysis on the classification

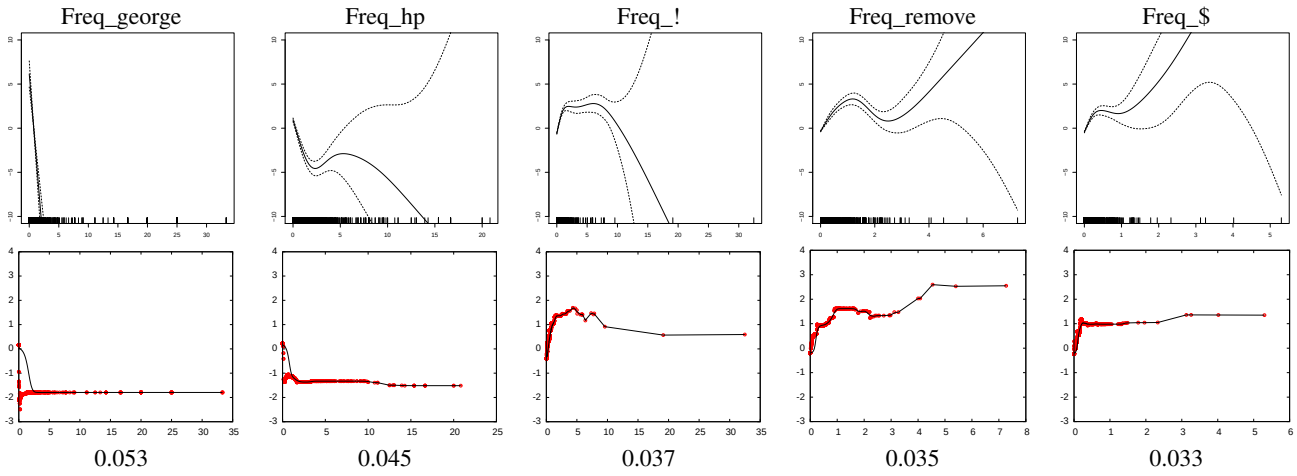


Figure 5: Shapes of features for the "Spambase" dataset produced by P-IRLS (top) and BST-bagTR3 (bottom).

sets. We then draw L samples of size $M = 0.64N$ points from the remaining points to keep the training sample size the same as with 5-fold cross validation in previous experiments. We use $L = 10$ trials. The bias-variance decomposition is calculated as follows:

$$Expected\ Loss = (bias)^2 + variance + noise$$

Define the average prediction on L samples for each point $(\boldsymbol{x}_i, y_i)$ in test set as $\bar{y}_i = \frac{1}{L}\sum_{l=1}^{L} \hat{y}_i^l$, where $\hat{y}_i^l$ is the predicted value for $\boldsymbol{x}_i$ using sample l. The squared bias $(bias)^2 = \frac{1}{N'}\sum_{i=1}^{N'}[\bar{y}_i - y_i]^2$, where y_i is the known target in the test set and $N' = 0.2N$ is the size of test set. The variance is calculated as $variance = \frac{1}{N'}\sum_{i=1}^{N'}\frac{1}{L}\sum_{l=1}^{L}[\hat{y}_i^l - \bar{y}_i]^2$.

The bias-variance results for the six regression datasets are shown in Figure 6. We can see that methods based on regression splines have very low variance, but sometimes at the expense of increased bias, while the best tree-based methods consistently have lower bias combined with low-enough variance to yield better overall RMSE. If tree complexity is not carefully controlled, however, variance explodes and hurts total RMSE. As expected, adding bagging inside boosting further reduces variance, making tree-based feature shaping methods based on gradient-boosting of residuals with internal bagging the most accurate method overall. (We do not expect bagging would help regression splines because the variance of regression splines is so low to begin with.) But even bagging will not prevent overfitting if the trees are too complex. Figure 3 shows training curves on the train, validation and test sets for gradient boosting with bagging and backfitting with bagging on a regression and classification problem. BST-bagTR2 is more resistant to overfitting than BST-bagTR16 which easily overfits.

The training curves for backfitting are not monotonic, and have distinct peaks on the classification problem. Each peak corresponds to a new backfitting iteration when pseudo residuals are updated. In our experience, backfitting on classification problems is consistently inferior to other methods, in part because it is harder for the local scoring algorithm to find a "good" set of pseudo residuals, which ultimately leads to instability and poor fit. Interestingly, in the bias-variance analysis of backfitting, both bias and variance often increase as the trees become larger, and the worst performing models on the five non-synthetic datasets are backfit models with large trees. We suspect backfitting can get stuck in inferior local minima when shaping with larger trees, hurting both bias and variance, which may explain the instability and convergence problems observed with backfitting on some classification problems.

problems because the bias-variance decomposition for classification is not as well defined.

5.2 Underfitting, Intelligibility, and Fidelity

One of the main reasons to use GAMs (linear models of non-linearly shaped features) is intelligibility. Figure 1 in Section 1 showed shape models for features in the synthetic dataset. In this section we show shape models learned for features from real regression and classification problems.

Figure 4 shows feature shape plots for the "Concrete" regression problem. Figure 5 show shape plots for the "Spambase" clasification problem. In each figure, the top row of plots are from the P-LS spline method, and the bottom row of plots are from BST-bagTR3. Confidence intervals for the least squres method can be computed analytically. Confidence intervals for BST-bagTR3 are generated by running BST-bagTR3 multiple times on bootstrap samples. As expected, the spline-based approach produces smooth plots. The cost of this smoothness, however, is poorer fit that results in higher RMSE or lower accuracy. Moreover, not all phenomena are smooth. Freezing and boiling occur at distinct temperatures, the onset of instability occurs abruptly as fluid flow increases relative to Reynolds Number, and many human decision making processes such as loans, admission to school, or administering medical procedures use discrete thresholds.

On the "Concrete" dataset, all features in Figure 4 are clearly non-linear. On this dataset P-LS is sacrificing accuracy for smoothness — the tree-based fitting methods have significantly lower RMSE than P-LS. The smoother P-LS models may appear more appealing and easier to interpret, but there is structure in the BST-bagTR3 models that is less apparent or missing in the P-LS plots that might be informative or important. As just one example, the P-LS and BST-bagTR3 models do not agree on the slopes of parts of the models for the Coarse and Fine Aggregate features.

On the "Spambase" dataset, the shape functions are nonlinear with sharp turns. Again, the BST-bagTR3 model is significantly more accurate than P-IRLS. Interestingly, the spline models for features Freq_hp, Freq_!, Freq_remove and Freq_$ show strong positive or negative slope in the right-hand side of the shape plots where data is sparse (albeit with very wide confidence intervals) while the BST-bagTR3 shape plots appear to be better behaved.

Below each shape plot in Figures 4 and 5 is the weight of each shape term in the linear model. These weights tell users how im-

Model	Concrete	Wine	Delta	CompAct	Music	Synthetic	Mean
Linear Regression	10.43±0.49	7.55±0.13	5.68±0.14	9.72±0.55	9.61±0.09	1.01±0.00	1.68±0.98
P-LS	5.67±0.41	7.25±0.21	5.67±0.16	2.81±0.13	9.27±0.07	0.04±0.00	1.00±0.00
BST-SP	5.79±0.37	7.27±0.18	5.68±0.18	3.19±0.37	9.29±0.08	0.04±0.00	1.03±0.06
BF-SP	5.66±0.42	7.25±0.21	5.67±0.17	2.77±0.06	9.27±0.08	0.04±0.00	1.00±0.01
BST-TR2	5.19±0.39	7.17±0.10	5.75±0.18	2.68±0.33	9.55±0.08	0.11±0.00	0.98±0.08
BST-TR3	5.13±0.37	7.20±0.16	5.82±0.19	3.18±0.45	9.77±0.07	0.05±0.01	1.02±0.11
BST-TR4	5.24±0.39	7.24±0.15	5.83±0.21	3.70±0.52	9.88±0.09	0.07±0.01	1.07±0.15
BST-TR8	5.57±0.61	7.35±0.17	5.97±0.22	5.07±0.54	10.03±0.10	0.19±0.02	1.19±0.33
BST-TR12	5.92±0.63	7.39±0.13	6.03±0.19	6.59±0.71	10.15±0.07	0.26±0.02	1.31±0.54
BST-TR16	6.08±0.37	7.41±0.23	6.09±0.19	7.07±1.01	10.23±0.08	0.33±0.03	1.36±0.60
BST-bagTR2	5.06±0.39	7.05±0.11	5.67±0.20	**2.59±0.34**	**9.42±0.08**	0.07±0.00	0.96±0.08
BST-bagTR3	4.93±0.41	7.01±0.10	5.67±0.20	2.82±0.35	9.45±0.07	**0.03±0.00**	0.97±0.09
BST-bagTR4	4.99±0.43	7.01±0.12	5.70±0.20	2.95±0.35	9.46±0.08	**0.03±0.00**	0.99±0.09
BST-bagTR8	5.04±0.43	7.04±0.13	5.79±0.18	3.40±0.34	9.48±0.08	0.06±0.00	1.02±0.13
BST-bagTR12	5.11±0.44	7.07±0.15	5.85±0.18	3.76±0.33	9.50±0.07	0.07±0.00	1.05±0.16
BST-bagTR16	5.18±0.49	7.10±0.18	5.91±0.20	4.16±0.39	9.52±0.09	0.09±0.00	1.09±0.21
BST-bagTRX	**4.89±0.37**	**7.00±0.10**	5.65±0.20	**2.59±0.34**	**9.42±0.08**	**0.03±0.00**	**0.95±0.09**
BF-TR	5.80±0.60	7.19±0.09	5.67±0.21	2.81±0.25	9.88±0.08	0.06±0.03	1.02±0.07
BF-bagTR	5.10±0.49	7.02±0.13	**5.61±0.21**	2.69±0.31	9.43±0.07	0.04±0.00	0.97±0.07
BF-bstTR2	5.11±0.37	7.14±0.11	5.73±0.20	2.66±0.35	9.62±0.07	0.13±0.01	0.98±0.08
BF-bstTR3	5.21±0.38	7.29±0.19	5.84±0.21	4.38±0.24	10.77±0.10	0.04±0.01	1.14±0.24
BF-bstTR4	5.49±0.72	7.44±0.20	5.94±0.21	4.97±0.73	11.24±0.07	0.06±0.03	1.21±0.33
BF-bstTR8	6.74±0.76	7.93±0.32	6.08±0.24	9.18±0.77	12.08±0.07	0.04±0.01	1.59±0.87
BF-bstTR12	7.13±0.68	8.10±0.27	6.15±0.24	11.20±0.72	12.31±0.15	0.08±0.03	1.76±1.16
BF-bstTR16	7.22±0.73	8.33±0.35	6.18±0.24	11.41±0.29	12.59±0.10	0.11±0.08	1.79±1.17
BF-bbTR2	5.13±0.41	7.05±0.12	5.66±0.19	2.59±0.37	9.50±0.07	0.12±0.00	0.97±0.08
BF-bbTR3	5.15±0.44	7.07±0.17	5.74±0.20	2.85±0.33	9.80±0.07	0.04±0.00	0.99±0.09
BF-bbTR4	6.20±0.86	7.12±0.22	5.80±0.23	3.01±0.23	9.83±0.08	0.04±0.00	1.05±0.09
BF-bbTR8	6.33±0.46	7.30±0.21	5.95±0.23	3.72±0.84	9.86±0.11	0.04±0.00	1.11±0.16
BF-bbTR12	6.52±0.56	7.52±0.30	6.01±0.20	4.32±0.94	9.89±0.06	0.04±0.00	1.17±0.23
BF-bbTR16	6.37±0.48	7.63±0.26	6.07±0.24	4.85±0.76	9.91±0.07	0.05±0.01	1.21±0.29
Random Forests	4.98±0.44	6.05±0.23	5.34±0.13	2.45±0.09	9.70±0.07	0.55±0.00	0.88±0.06
Additive Groves	4.25±0.47	6.21±0.20	5.35±0.14	2.23±0.15	9.03±0.05	0.02±0.00	0.86±0.10

Table 5: RMSE for regression datasets. Each cell contains the mean RMSE ± one standard deviation. Average normalized score on five datasets (excludes synthetic) is shown in the last column, where the score is calculated as relative improvement over P-LS.

portant each term is to the model. Terms can be sorted by weight, and if necessary terms with low weight can be removed from the model and the retained features reshaped.

In both figures there is coarse similarity between the feature shape plots learned by the spline and tree-based methods, but in many plots the tree-based methods appear to have caught structure that is missing or more difficult to see in the spline plots. The spline models may be more appealing to the eye, but they are clearly less accurate and appear to miss some of details of the shape functions.

5.3 Computational Cost

In our experiments, P-LS and P-IRLS are very fast on small datasets, but on the larger datasets they are slower than the BST-TRx. Due to the extra cost of bagging, BST-bagTRx, BF-bagTR and BF-bbTRx are much slower than P-LS/P-IRLS or BST-TRx. The slowest method we tested is backfitting, which is expensive because at each iteration the previous shape functions are discarded and a new fit for each feature must be learned. Gradient boosting converges faster because in each iteration the algorithm adds a patch to the existing pool of predictors, thus building on previous efforts rather than discarding them.

Gradient boosting is easier to parallelize [17] than backfitting (Gauss-Seidel). The Jacobi method is sometimes used as an alternative to Gauss-Seidel because it is easier to parallelize, however, in our experience, Jacobi-based backfitting converges to suboptimal solutions that can be much worse.

5.4 Limitations and Extensions

The experiments in this paper are on datasets of low-to-medium dimensionality (less than 100 dimensions). Our next step is to scale the algorithm to datasets with more dimensions (and more training points). Even linear models lose intelligibility when there are hundreds of terms in the model. To help retain intelligibility in high dimensional spaces, we have begun developing an extension to BST-bagTRX that incorporates feature selection in the feature shaping process to retain only those features that, after shaping, make the largest contributions to the model. We do not present results for feature selection in this paper because of space limitations, and because it is important to focus first on the foundational issue of what algorithm(s) train the best models.

In this paper we focus exclusively on shape functions of individual features; feature interaction is not allowed. Because of this, the models will not be able to achieve the same accuracy as unrestricted full-complexity models on many datasets. The addition of a few carefully selected interaction terms would further close this gap [16]. Because 3-D plots can be visualized, we may be able to allow pairwise interactions in our models while preserving some of the intelligibility.

Model	Spambase	Insurance	Magic	Letter	Adult	Physics	Mean
Logistic Regression	7.67±1.03	6.11±0.29	20.99±0.46	27.54±0.27	16.04±0.46	29.24±0.36	1.22±0.23
P-IRLS	6.43±0.77	6.11±0.30	14.53±0.41	17.47±0.24	15.00±0.28	29.04±0.49	1.00±0.00
BST-SP	6.24±0.65	6.07±0.31	14.54±0.31	17.61±0.23	15.02±0.25	28.98±0.43	1.00±0.03
BF-SP	6.37±0.29	6.11±0.29	14.58±0.32	17.52±0.17	15.01±0.28	28.98±0.46	1.00±0.03
BST-TR2	5.22±0.77	5.97±0.38	14.63±0.36	17.40±0.22	14.90±0.26	29.58±0.53	0.97±0.08
BST-TR3	5.09±0.79	5.97±0.38	14.54±0.14	17.29±0.25	14.58±0.33	28.81±0.52	0.95±0.08
BST-TR4	5.11±0.70	5.97±0.38	14.60±0.25	17.44±0.26	14.65±0.35	28.72±0.48	0.96±0.08
BST-TR8	5.39±1.06	5.97±0.38	14.64±0.23	17.44±0.27	14.61±0.34	28.77±0.55	0.96±0.08
BST-TR12	5.61±0.76	5.97±0.38	14.57±0.41	17.45±0.24	14.57±0.36	28.63±0.60	0.97±0.08
BST-TR16	5.93±0.96	5.97±0.38	14.83±0.38	17.47±0.23	14.62±0.32	28.63±0.51	0.98±0.05
BST-bagTR2	5.00±0.65	5.97±0.38	14.47±0.20	17.25±0.22	14.95±0.35	29.32±0.67	0.96±0.09
BST-bagTR3	4.89±1.01	5.97±0.38	14.39±0.13	17.22±0.24	**14.57±0.29**	28.65±0.47	0.94±0.09
BST-bagTR4	4.98±1.07	5.98±0.35	14.40±0.28	17.31±0.23	14.63±0.30	29.05±0.50	0.95±0.09
BST-bagTR8	5.22±1.05	5.99±0.36	14.43±0.33	17.42±0.15	14.68±0.35	28.73±0.64	0.96±0.08
BST-bagTR12	5.48±1.09	6.00±0.36	14.44±0.35	17.45±0.19	14.67±0.39	29.06±0.77	0.97±0.07
BST-bagTR16	5.52±1.01	5.99±0.36	14.45±0.29	17.47±0.23	14.69±0.34	28.77±0.65	0.97±0.07
BST-bagTRX	**4.78±0.82**	**5.95±0.37**	**14.31±0.21**	**17.21±0.23**	14.58±0.28	**28.62±0.49**	**0.94±0.09**
BF-TR	6.41±0.37	6.34±0.27	16.81±0.35	17.36±0.26	14.96±0.28	31.64±0.57	1.05±0.07
BF-bagTR	5.63±0.47	6.34±0.22	15.09±0.48	17.41±0.23	14.95±0.34	29.51±0.46	0.99±0.06
BF-bstTR2	5.39±0.68	6.28±0.18	14.43±0.37	17.44±0.35	14.87±0.21	29.70±0.66	0.98±0.35
BF-bstTR3	6.85±1.48	6.31±0.54	15.11±0.24	17.53±0.18	14.64±0.32	29.90±0.34	1.02±0.06
BF-bstTR4	7.63±0.85	6.40±0.48	15.47±0.26	17.46±0.29	14.66±0.27	29.67±0.80	1.05±0.09
BF-bstTR8	10.20±1.30	6.52±0.54	16.26±0.36	17.47±0.25	14.60±0.36	30.32±0.41	1.13±0.24
BF-bstTR12	12.39±1.04	6.53±0.54	16.95±0.40	17.50±0.25	14.76±0.32	31.08±0.43	1.21±0.35
BF-bstTR16	13.11±1.32	6.55±0.58	17.68±0.56	17.52±0.24	14.79±0.33	31.97±0.37	1.24±0.40
BF-bbTR2	5.48±0.59	6.20±0.26	15.26±0.43	17.86±0.30	14.90±0.31	29.36±0.56	0.99±0.08
BF-bbTR3	5.83±0.76	6.42±0.24	14.64±0.18	17.43±0.34	14.77±0.34	28.64±0.56	0.99±0.06
BF-bbTR4	6.13±0.90	6.48±0.20	14.68±0.24	17.43±0.26	14.74±0.35	28.64±0.50	1.00±0.07
BF-bbTR8	6.48±0.97	6.59±0.26	14.79±0.20	17.51±0.27	14.64±0.33	28.65±0.50	1.01±0.07
BF-bbTR12	7.35±1.24	6.56±0.21	14.90±0.22	17.53±0.18	14.58±0.33	28.88±0.29	1.04±0.10
BF-bbTR16	7.72±1.36	6.56±0.20	15.02±0.40	17.52±0.24	14.58±0.28	29.16±0.38	1.05±0.11
Random Forests	4.48±0.64	5.97±0.41	11.99±0.50	6.23±0.27	14.85±0.25	28.55±0.56	0.80±0.23

Table 6: Error rate for classification datasets. Each cell contains the classification error ± one standard deviation. Averaged normalized score on all datasets is shown in the last column, where the score is calculated as relative improvement over P-IRLS.

Our empirical results suggest that bagging small trees of only 2-4 leaves yields the best accuracy. These are very small trees trained for one feature at a time and thus they divide the number line into 2-4 subintervals. We could imagine replacing bagged decision trees with some type of dynamic programming algorithm that directly works on (possibly smoothed) subintervals of the number line.

6. RELATED WORK

Generalized additive models were introduced by the statistics community [14, 15, 22] and have been extended to include LASSO feature selection [18] and to incorporate interaction terms [16]. Binder and Tutz performed a comprehensive comparison of methods for fitting GAMs with regression splines [7]. They compared backfitting, boosting, and penalized iteratively re-weighted least squares on simulated datasets. Our work differs from theirs in that we examine both regression splines and regression trees, most of our experiments are with real datasets, we look at both regression and classification, and we introduce a new method that is more accurate than splines.

Methods have been proposed for fitting GAMs with arbitrary link functions where the link function also is unknown and must be fitted. ACE [9] is probably the most well-known method for fitting these kind of GAMs. We do not evaluate ACE in this work because learned link functions can be complex, making it difficult to interpret the feature shape models. We focus on the identity and logit link functions because these are the link functions appropriate for regression and classification.

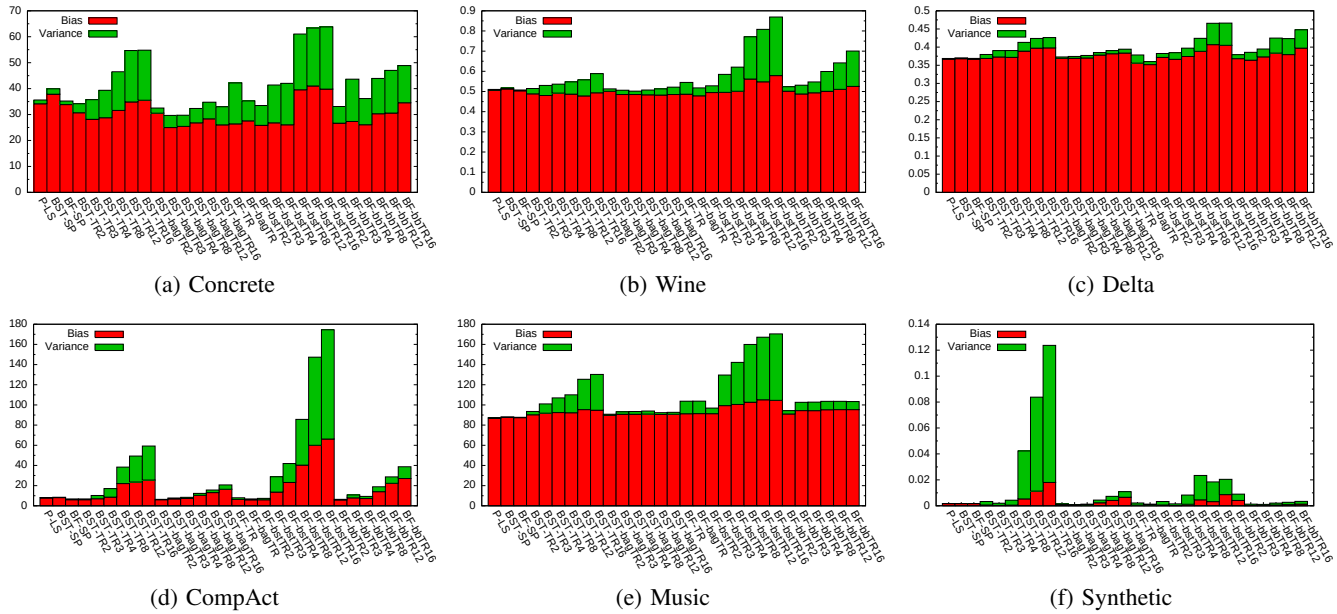
Forman *et al.* proposed feature shaping for linear SVM classifiers [11]. Their focus is on estimating the posterior probability $P(y = 1|\boldsymbol{x}_i = v)$.

Recently there have been efforts to scale GAMs. [17] uses MapReduce to parallelize gradient boosting and large tree construction. [20] parallelizes growing regression trees via gradient boosting using a master-worker paradigm where data are partitioned among workers. The algorithm carefully orchestrates overlap between communication and computation to achieve good performance.

7. CONCLUSIONS

We present a comprehensive empirical study of algorithms for fitting generalized additive models (GAMs) with spline and tree-based shape functions. Our bias-variance analysis shows that spline-based methods tend to underfit and thus may miss important non-smooth structure in the shape models. As expected, the bias-variance analysis also shows that tree-based methods are prone to overfitting and require careful regularization. We also introduce a new GAM method based on gradient boosting of size-limited bagged trees that yields significantly more accuracy than previous algorithms on both regression and classification problems while retaining the intelligibility of GAM models.

Acknowledgments. We thank the anonymous reviewers for their



(a) Concrete (b) Wine (c) Delta

(d) CompAct (e) Music (f) Synthetic

Figure 6: Bias-variance analysis for the six regression problems (bias = red at bottom of bars; variance = green at top of bars).

valuable comments. This research has been supported by the NSF under Grants IIS-0911036 and IIS-1012593 and by a gift from NEC. Any opinions, findings, conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsors.

8. REFERENCES

[1] http://archive.ics.uci.edu/ml/.

[2] http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html.

[3] http://www.cs.toronto.edu/~delve/data/datasets.html.

[4] http://osmot.cs.cornell.edu/kddcup/.

[5] http://additivegroves.net.

[6] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.

[7] H. Binder and G. Tutz. A comparison of methods for the fitting of generalized additive models. *Statistics and Computing*, 18(1):87–99, 2008.

[8] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] L. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, pages 580–598, 1985.

[10] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, 2006.

[11] G. Forman, M. Scholz, and S. Rajaram. Feature shaping for linear svm classifiers. In *KDD*, 2009.

[12] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[13] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 2002.

[14] T. Hastie and R. Tibshirani. Generalized additive models (with discussion). *Statistical Science*, 1:297–318, 1986.

[15] T. Hastie and R. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990.

[16] G. Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2007.

[17] B. Panda, J. Herbach, S. Basu, and R. Bayardo. Planet: massively parallel learning of tree ensembles with mapreduce. *PVLDB*, 2009.

[18] P. Ravikumar, H. Liu, J. Lafferty, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):1009–1030, 2009.

[19] D. Sorokina, R. Caruana, and M. Riedewald. Additive groves of regression trees. In *ECML*, 2007.

[20] S. Tyree, K. Weinberger, K. Agrawal, and J. Paykin. Parallel boosted regression trees for web search ranking. In *WWW*, 2011.

[21] S. Wood. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):95–114, 2003.

[22] S. Wood. *Generalized additive models: an introduction with R*. CRC Press, 2006.