Accurate Intelligible Models with Pairwise Interactions

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

Giles Hooker
Dept. of Statistical Science
Cornell University
giles.hooker@cornell.edu

ABSTRACT

Standard generalized additive models (GAMs) usually model the dependent variable as a sum of univariate models. Although previous studies have shown that standard GAMs can be interpreted by users, their accuracy is significantly less than more complex models that permit interactions.

In this paper, we suggest adding selected terms of interacting *pairs* of features to standard GAMs. The resulting models, which we call GA^{2}M-models, for *Generalized Additive Models plus Interactions*, consist of univariate terms and a small number of pairwise interaction terms. Since these models only include one- and two-dimensional components, the components of GA^{2}M-models can be visualized and interpreted by users. To explore the huge (quadratic) number of pairs of features, we develop a novel, computationally efficient method called FAST for ranking all possible pairs of features as candidates for inclusion into the model.

In a large-scale empirical study, we show the effectiveness of FAST in ranking candidate pairs of features. In addition, we show the surprising result that GA^{2}M-models have almost the same performance as the best full-complexity models on a number of real datasets. Thus this paper postulates that for many problems, GA^{2}M-models can yield models that are both intelligible and accurate.

Categories and Subject Descriptors

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

Keywords

classification, regression, interaction detection

1. INTRODUCTION

Many machine learning techniques such as boosted or bagged trees, SVMs with RBF kernels, or deep neural nets are powerful classification and regression models for high-dimensional prediction problems. However, due to their complexity, the resulting models are hard to interpret for the user. But in many applications, intelligibility is as important as accuracy [19], and thus building models that users can understand is a crucial requirement.

Generalized additive models (GAMs) are the gold standard for intelligibility when only univariate terms are considered [13, 19]. Standard GAMs have the form

$$g(E[y]) = \sum f_i(x_i), \tag{1}$$

where g is the link function. Standard GAMs are easy to interpret since users can visualize the relationship between the univariate terms of the GAM and the dependent variable through a plot $f_i(x_i)$ *vs.* x_i. However there is unfortunately a significant gap between the performance of the best standard GAMs and full complexity models [19]. In particular, Equation 1 does not model any interactions between features, and it is this limitation that lies at the core of the lack of accuracy of standard GAMs as compared to full complexity models.

EXAMPLE 1. *Consider the function* $F(\boldsymbol{x}) = \log(x_1^2 x_3) + x_2 x_3$. *$F$ has a pairwise interaction* (x_2, x_3), *but no interactions between* (x_1, x_2) *or* (x_1, x_3), *since* $\log(x_1^2 x_3) = 2\log(x_1) + \log(x_3)$, *which is additive.*

Our first contribution in this paper is to build models that are more powerful than GAMs, but are still intelligible. We observe that two-dimensional interactions can still be rendered as heatmaps of $f_{ij}(x_i, x_j)$ on the two-dimensional x_i, x_j-plane, and thus a model that includes only one- and two-dimensional components is still intelligible. Therefore in this paper, we propose building models of the form

$$g(E[y]) = \sum f_i(x_i) + \sum f_{ij}(x_i, x_j); \tag{2}$$

we call the resulting model class *Generalized Additive Models plus Interactions*, or short GA^{2}Ms.

The main challenge in building GA^{2}Ms is the large number of pairs of features to consider. We thus only want to include "true" interactions that pass some statistical test. To this end, we focus on problems with up to thousands of features since for truly high dimensional problems (e.g., millions of features), it is almost intractable to test all possible pairwise interactions (e.g., trillions of feature pairs).

Existing approaches for detecting statistical interactions can be divided into two classes. One class of methods di-

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

ACM 978-1-4503-2174-7/13/08 ...$15.00.

rectly models and compares the interaction effects and additive effects [10, 11, 18, 25]. One drawback of these methods is that spurious interactions may be reported over low-density regions [15]. The second class of methods measures the performance drop in the model if certain interaction is not included; they compare the performance between restricted and unrestricted models, where restricted models are not allowed to model an interaction in question [22]. Although this class of methods does not suffer from the problem of low-density regions, they are computationally extremely expensive even for pairwise interaction detection.

Our second contribution in this paper is to scale the construction of GA^{2}Ms by proposing a novel, extremely efficient method called FAST to measure and rank the strength of the interaction of all pairs of variables. Our experiments show that FAST can efficiently rank all pairwise interactions close to a ground truth ranking.

Our third contribution is an extensive empirical evaluation of GA^{2}M-models. Surprisingly, on many of the datasets included in our study, the performance of GA^{2}M-models is close and sometimes better than the performance of full-complexity models. These results indicate that GA^{2}M-models not only make a significant step in improving accuracy over standard GAMs, but in some cases they actually come all the way to the performance of full-complexity models. The performance may be due to the difficulty of estimating intrinsically high dimensional functions from limited data, suggesting that the bias associated with the GA^{2}M structure is outweighed by a drop in variance. We also demonstrate that the resulting models are intelligible through a case study.

In this paper we make the following contributions:

- We introduce the model class GA^{2}M.
- We introduce our new method FAST for efficient interaction detection. (Section 4)
- We show through an extensive experimental evaluation that (1) GA^{2}Ms have accuracy comparable to full-complexity models; (2) FAST accurately ranks interactions as compared to a gold standard; and (3) FAST is computationally efficient. (Section 5)

We start with a problem definition and a survey of related work in Sections 2 and 3.

2. PROBLEM DEFINITION

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_1^N$ denote a dataset of size N, where $\boldsymbol{x}_i = (x_{i1}, ..., x_{in})$ is a feature vector with n features and y_i is the response. Let $\boldsymbol{x} = (x_1, ..., x_n)$ denote the variables or features in the dataset. For $u \subseteq \{1, ..., n\}$, we denote by x_u the subset of variables whose indices are in u. Similarly x_{-u} will indicate the variables with indices not in u. To simplify notation, we denote $\mathcal{U}^1 = \{\{i\}|1 \leq i \leq n\}$, $\mathcal{U}^2 = \{\{i, j\}|1 \leq i < j \leq n\}$, and $\mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2$, i.e., $\mathcal{U}$ contains all indices for all features and pairs of features.

For any $u \in \mathcal{U}$, let $\mathcal{H}_u$ denote the Hilbert space of Lebesgue measurable functions $f_u(x_u)$, such that $E[f_u] = 0$ and $E[f_u^2] < \infty$, equipped with the inner product $\langle f_u, f'_u \rangle = E[f_u f'_u]$. Let $\mathcal{H}^1 = \sum_{u \in \mathcal{U}^1} \mathcal{H}_u$ denote the Hilbert space of functions that have additive form $F(\boldsymbol{x}) = \sum_{u \in \mathcal{U}^1} f_u(x_u)$ on univariate compnents; we call those components *shape functions* [19]. Similarly let $\mathcal{H} = \sum_{u \in \mathcal{U}} \mathcal{H}_u$ denote the Hilbert space of functions of $\boldsymbol{x} = (x_1, ..., x_n)$ that have additive form $F(\boldsymbol{x}) = \sum_{u \in \mathcal{U}} f_u(x_u)$ on both one- and two-dimensional shape functions. Models described by sums of low-order components are called *generalized additive models (GAMs)*, and in the remainder of the paper, we use GAMs to denote models that only consist of univariate terms.

We want to find the best model $F \in \mathcal{H}$ that minimizes the following objective function:

$$\min_{F \in \mathcal{H}} E[L(y, F(\boldsymbol{x}))], \tag{3}$$

where $L(\cdot, \cdot)$ is a non-negative convex loss function. When L is the squared loss, our problem becomes a regression problem, and if L is logistic loss function, we are dealing with a classification problem.

3. EXISTING APPROACHES

3.1 Fitting Generalized Additive Models

Terms in GAMs can be represented by a variety of functions, including splines [24], regression trees, or tree ensembles [9]. There are two popular methods of fitting GAMs: Backfitting [13] and gradient boosting [10]. When the shape function is spline, fitting GAMs reduces to fitting generalized linear models with different bases, which can be solved by least squares or iteratively reweighted least squares [25].

backfitting? (smoother) non-parametric regression

Spline-based methods become inefficient when modeling higher order interactions because the number of parameters to estimate grows exponentially; tree-based methods are more suitable in this case. Standard additive modeling only involves modeling individual features (also called *feature shaping*). Previous research showed that gradient boosting with ensembles of shallow regression trees is the most accurate method among a number of alternatives [19].

3.2 Interaction Detection

In this section, we briefly review existing approaches to interaction detection. two way ANOVA

ANOVA. An additive model is fit with all pairwise interaction terms [13] and the significance of interaction terms is measured through an analysis of variance (ANOVA) test [25]. The corresponding p-value for each pair can then be computed; however, this requires the computation of the full model, which is prohibitively expensive.

Partial Dependence Function. Friedman and Popescu proposed the following statistic to measure the strength of pairwise interactions,

$$H_{ij}^2 = \frac{\sum_{k=1}^N [\hat{F}_{ij}(x_{ki}, x_{kj}) - \hat{F}_i(x_{ki}) - \hat{F}_j(x_{kj}))]^2}{\sum_{k=1}^N \hat{F}_{ij}^2(x_{ki}, x_{kj})} \tag{4}$$

where $\hat{F}_u(x_u) = E_{x_{-u}}[F(x_u, x_{-u})]$ is the partial dependence function (PDF) [10, 11] and F is a complex multi-dimensional function learned on the dataset. Computing $\hat{F}_u(x_u)$ on the whole dataset is expensive, thus one often specifies a subset of size m on which to compute $\hat{F}_u(x_u)$. The complexity is then $O(m^2)$. However, since partial dependence functions are computed based on uniform sampling, they may detect spurious interactions over low-density regions [15].

GUIDE. GUIDE tests pairwise interactions based on the χ^2 test [18]. An additive model F is fit in $\mathcal{H}^1$ and residuals are obtained. To detect interactions for (x_i, x_j), GUIDE divides the (x_i, x_j)-space into four quadrants by splitting the range of each variable into two halves at the sample median.

Then GUIDE constructs a 2×4 contingency table using the residual signs as rows and the quadrants as columns. The cell values in the table are the number of "+"s and "-"s in each quadrant. These counts permit the computation of a p-value to measure the interaction strength of a pair. While this might be more robust to outliers, in practice it is less powerful than the method we propose.

Grove. Sorokina et al. proposed a grove-based method to detect statistical interactions [22]. To measure the strength of a pair (x_i, x_j), they build both the restricted model $R_{ij}(\boldsymbol{x})$ and unrestricted model $F(\boldsymbol{x})$, where $R_{ij}(\boldsymbol{x})$ is prevented from modeling an interaction (x_i, x_j):

$$R_{ij}(\boldsymbol{x}) = f_{\setminus i}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) + f_{\setminus j}(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n). \quad (5)$$

To correctly estimate interaction strength, such method requires a model to be highly predictive when certain interaction is not allowed to appear, and therefore many learning algorithms are not applicable (e.g., bagged decision trees). To this end, they choose to use Additive Groves [21].

They measure the performance as standardized root mean squared error (RMSE) and quantify the interaction strength I_{ij} by the difference between $R_{ij}(\boldsymbol{x})$ and $F(\boldsymbol{x})$,

$$stRMSE(F(\boldsymbol{x})) = \frac{RMSE(F(\boldsymbol{x}))}{StD(F^*(\boldsymbol{x}))} \quad (6)$$

$$I_{ij} = stRMSE(R_{ij}(\boldsymbol{x})) - stRMSE(F(\boldsymbol{x})) \quad (7)$$

where $Std(F^*(\boldsymbol{x}))$ is calculated as standard deviation of the response values in the training set. The ranking of all pairs can be generated based on the strength I_{ij}.

To handle correlations among features, they use a variant of backward elimination [12] to do feature selection. Although Grove is accurate in practice, building restricted and unrestricted models are computationally expensive and therefore this method is almost infeasible for large high dimensional datasets.

4. OUR APPROACH

For simplicity and without loss of generality, we focus in this exposition on regression problems. Since there are $O(n^2)$ pairwise interactions, it is very hard to detect pairwise interactions when n is large. Therefore we propose a framework using greedy forward stagewise selection strategy to build the most accurate model in $\mathcal{H}$.

Algorithm 1 summarizes our approach called GA^{2}M. We maintain two sets $\mathcal{S}$ and $\mathcal{Z}$, where $\mathcal{S}$ contains the selected pairs so far and $\mathcal{Z}$ is the set of the remaining pairs (Line 1-2). We start with the best additive model F so far in Hilbert space $\mathcal{H}^1 + \sum_{u \in \mathcal{S}} \mathcal{H}_u$ (Line 4) and detect interactions on the residual R (Line 5). Then for each pair in $\mathcal{Z}$, we build an interaction model on the residual R (Line 6-7). We select the best interaction pair and include it in $\mathcal{S}$ (Line 9-10). We then repeat this process until there is no gain in accuracy.

Note that Algorithm 1 will find an overcomplete set $\mathcal{S}$ by the greedy nature of the forward selection strategy. When features are correlated, it is also possible that the algorithm includes false pairs. For example, consider the function in Example 1. If x_1 is highly correlated with x_3, then (x_1, x_2) may look like an interaction pair, and it may be included in $\mathcal{S}$ before we select (x_2, x_3). But since we will refit the model every time we include a new pair, it is expected that F will perfectly model (x_2, x_3) and therefore (x_1, x_2) will become a less important term in F.

For large high-dimensional datasets, however, Algorithm 1 is very expensive for two reasons. First, fitting interaction models for $O(n^2)$ pairs in $\mathcal{Z}$ can be very expensive if the model is non-trivial. Second, every time we add a pair, we need to refit the whole model, which is also very expensive for large datasets. As we will see in Section 4.1 and Section 4.2, we will relax some of the constraints in Algorithm 1 to achieve better scalability while still staying accurate.

Algorithm 1 GA^{2}M Framework

1: $\mathcal{S} \leftarrow \varnothing$
2: $\mathcal{Z} \leftarrow \mathcal{U}^2$
3: **while** not converge **do**
4: $\quad F \leftarrow \arg\min_{F \in \mathcal{H}^1 + \sum_{u \in \mathcal{S}} \mathcal{H}_u} \frac{1}{2} E[(y - F(\boldsymbol{x}))^2]$
5: $\quad R \leftarrow y - F(\boldsymbol{x})$
6: $\quad$ **for all** $u \in \mathcal{Z}$ **do**
7: $\quad\quad F_u \leftarrow E[R|x_u]$
8: $\quad u^* \leftarrow \arg\min_{u \in \mathcal{Z}} \frac{1}{2} E[(R - F_u(x_u))^2]$
9: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{u^*\}$
10: $\quad \mathcal{Z} \leftarrow \mathcal{Z} - \{u^*\}$

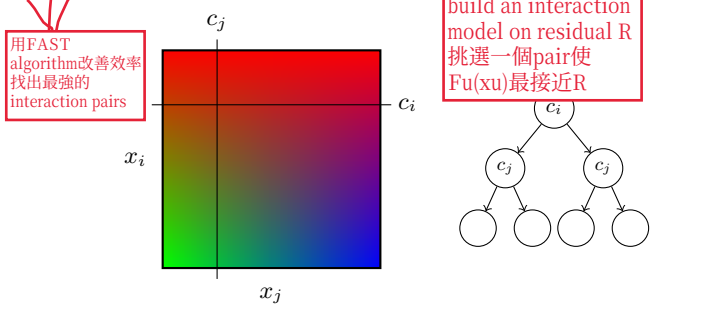


Figure 1: Illustration for searching cuts on input space of x_i and x_j. On the left we show a heat map on the target for different values of x_i and x_j. c_i and c_j are cuts for x_i and x_j, respectively. On the right we show an extremely simple predictor of modeling pairwise interaction.

4.1 Fast Interaction Detection

Consider the conceptual additive model in Equation 2, given a pair of variables (x_i, x_j) we wish to measure how much benefit we can get if we model $f_{ij}(x_i, x_j)$ instead of $f_i(x_i) + f_j(x_j)$. Since we start with shaping individual features and always detect interactions on the residual, $f_i(x_i) + f_j(x_j)$ are presumably modeled and therefore we only need to look at the residual sum of squares (RSS) for the interaction model f_{ij}. The intuition is that when (x_i, x_j) is a strong interaction, modeling f_{ij} can significantly reduce the RSS. However, we do not wish to fully build f_{ij} since this is a very expensive operation; instead we are looking for a cheap substitute.

4.1.1 Overview

Our idea is to build an extremely simple model for f_{ij} using cuts on the input space of x_i and x_j, as illustrated in Figure 1. The simplest model we can build is to place one cut on each variable, i.e., we place one c_i and one cut

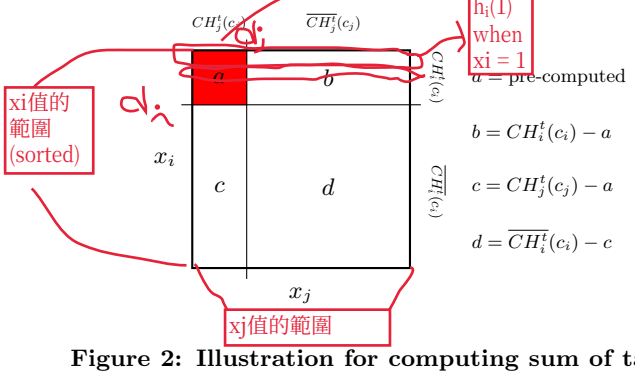


Figure 2: Illustration for computing sum of targets for each quadrant. Given that the value of red quadrant is known, we can easily recover values in other quadrant using marginal cumulative histograms.

c_j on x_i and x_j, respectively. Those cuts are parallel to the axes. The interaction predictor T_{ij} is constructed by taking the mean of all points in each quadrant. We search for all possible (c_i, c_j) and pick the best T_{ij} with the lowest RSS, which is assigned as weight for (x_i, x_j) to measure the strength of interaction.

4.1.2 Constructing Predictors

Naïve implementation of FAST is straightforward, but careless implementation has very high complexity since we need to repeatedly build a lot of T_{ij} for different cuts. The key insight for faster version of FAST is that we do not need to scan through the dataset each time to compute T_{ij} and compute its RSS. We show that by using very simple bookkeeping data structures, we can greatly reduce the complexity.

Let $\text{dom}(x_i) = \{v_i^1, ..., v_i^{d_i}\}$ be a sorted set of possible values for variable x_i, where $d_i = |\text{dom}(x_i)|$. Define $H_i^t(v)$ as the sum of targets when $x_i = v$, and define $H_i^w(v)$ as the sum of weights (or counts) when $x_i = v$. Intuitively, these are the standard histograms when constructing regression trees. Similarly, we define $CH_i^t(v)$ and $CH_i^w(v)$ as the cumulative histogram for sum of targets and sum of weights, respectively, i.e., $CH_i^t(v) = \sum_{u \le v} H_i^t(u)$ and $CH_i^w(v) = \sum_{u \le v} H_i^w(u)$. Accordingly, define $\overline{CH_i^t}(v) = \sum_{u > v} H_i^t(u) = CH_i^t(v_i^{d_i}) - CH_i^t(v)$ and define $\overline{CH_i^w}(v) = \sum_{u > v} H_i^w(u) = CH_i^w(v_i^{d_i}) - CH_i^w(v)$. Furthermore, define $H_{ij}^t(u, v)$ and $H_{ij}^w(u, v)$ as the sum of targets and the sum of weights, respectively, when $(x_i, x_j) = (u, v)$.

Consider again the input space for (x_i, x_j), we need a quick way to compute the sum of targets and sum of weights for each quadrant. Figure 2 shows an example for computing sum of targets on each quadrant. Given the above notations, we already know the marginal cumulative histograms for x_i and x_j, but unfortunately using these marginal values only can not recover values on four quadrants. Thus, we have to compute value for one quadrant.

We show that it is very easy and efficient to compute all possible values for the red quadrant given any cuts (c_i, c_j) using dynamic programming. Once that quadrant is known, we can easily recover values in other quadrant using marginal cumulative histograms. We store those values into lookup tables. Let $L^t(c_i, c_j) = [a, b, c, d]$ be the lookup table for sum

Algorithm 2 ConstructLookupTable

```
1:  sum ← 0
2:  for q = 1 to d_j do
3:      sum ← sum + H_ij^t(v_i^1, v_j^q)
4:      a[1][q] ← sum
5:      L(v_i^1, v_j^q) ← ComputeValues(CH_i^t, CH_j^t, a[1][q])
6:  for p = 2 to d_i do
7:      sum ← 0
8:      for q = 1 to d_j do
9:          sum ← sum + H_ij^t(v_i^p, v_j^q)
10:         a[p][q] ← sum + a[p − 1][q]
11:         L(v_i^p, v_j^q) ← ComputeValues(CH_i^t, CH_j^t, a[p][q])
```

of targets on cuts (c_i, c_j), and denote $L^w(c_i, c_j) = [a, b, c, d]$ as the lookup table for sum of weights on cuts (c_i, c_j).

Algorithm 2 describes how to compute the lookup table L^t. We focus on computing quadrant a and other quadrants can be easily computed, which is handled by subroutine $ComputeValues$. Given H_{ij}^t, we first compute as for the first row of L^t (Line 3-5). Let $a[p][q]$ denote the value for cuts (p, q). Note $a[p][q] = a[p-1][q] + \sum_{k \le q} H_{ij}^t(v_i^p, v_j^k)$. Thus we can efficiently compute the rest of the lookup table row by row (Line 6-11).

Once we have L^t and L^w, given any cuts (c_i, c_j), we can easily construct T_{ij}. For example, we can set the leftmost leaf value in T_{ij} as $L^t(c_i, c_j).a / L^w(c_i, c_j).a$. It is easy to see that with those bookkeeping data structures, we can reduce the complexity of building predictors to $O(1)$.

4.1.3 Calculating RSS

In this section, we show that calculating RSS for T_{ij} can be very efficient. Consider the definition of RSS. Let $T_{ij}.r$ denote the prediction value on region r, where $r \in \{a, b, c, d\}$.

$$RSS = \sum_{k=1}^{N} (y_k - T_{ij}(\boldsymbol{x}_k))^2 \tag{8}$$

$$= \left(\sum_{k=1}^{N} y_k^2 - 2 \sum_r T_{ij}.r L^t.r + \sum_r (T_{ij}.r)^2 L^w.r \right) \tag{9}$$

In practical implementation, we only need to care about $\sum_r (T_{ij}.r)^2 L^w.r - 2 \sum_r T_{ij}.r L^t.r$ since we are only interested in relative ordering of RSS, and it is easy to see the complexity of computing RSS for T_{ij} is $O(1)$.

4.1.4 Complexity Analysis

For each pair (x_i, x_j), computing the histograms and cumulative histograms needs to scan through the data and therefore its complexity is $O(N)$. Constructing the lookup tables takes $O(d_i d_j + N)$ time. Thus, the time complexity of FAST is $O(d_i d_j + N)$ for one pair (x_i, x_j). Besides, Since we need to store d_i-by-d_j matrices for each pair, the space complexity is $O(d_i d_j)$.

For continuous features, $d_i d_j$ can be quite large. However, we can discretize the features into b equi-frequency bins. Such feature discretizing usually does not hurt the performance of regression tree [17]. As we will see in Section 5, FAST is not sensitive to a wide range of bs. Therefore, the complexity can be reduced to $O(b^2 + N)$ per pair when we

discretize features into b bins. For small bs ($b \leq 256$), we can quickly process each pair.

4.2 Two-stage Construction

With FAST, we can quickly rank of all pairs in $\mathcal{Z}$, the remaining pair set, and add the best interaction to the model. However, refitting the whole model after each pair is added can be very expensive for large high-dimensional datasets. Therefore, we propose a two-stage construction approach.

1. In Stage 1, build the best additive model F in $\mathcal{H}^1$ using only one-dimensional components.

2. In Stage 2, fix the one-dimensional functions, and build models for pairwise interactions on residuals.

4.2.1 Implementation Details

To scale up to large datasets and many features, we discretize the features into 256 equi-frequency bins for continuous features.[1] We find such feature discretization rarely hurts the performance but substantially reduces the running time and memory footprint since we can use one byte to store a feature value. Besides, discretizing the features removes the sorting requirement for continuous features when searching for the best cuts in the space.

Previous research showed that feature shaping using gradient boosting [10] with shallow regression tree ensembles can achieve the best accuracy [19]. We follow similar approach (i.e., gradient boosting with shallow tree-like ensembles) in this work. However, a regression tree is not the ideal learning method for each component for two reasons. First, while regression trees are good as a generic shape functions for any x_u, shaping a single feature is equivalent to cutting on a line, but line cutting can be made more efficient than regression tree. Second, using regression tree to shape pairwise functions can be problematic. Recall that in Stage 1, we obtain the best additive model after gradient boosting converges. This means adding more cuts to any one feature does not reduce the error, and equivalently, any cut on a single feature is random. Therefore, when we begin to shape pairwise interactions, the root test in a regression tree that is constructed greedily top-down is random.

Similar to [19], to effectively shape pairwise interactions, we build shallow tree-like models on the residuals as illustrated in Figure 3. We enumerate all possible cuts c_i on x_i. Given this cut, we greedily search the best cut c_j^1 in the region above c_i and similarly greedily search the best cut c_j^2 in the region below c_i. Note we can reuse the lookup table L^t and L^w we developed for FAST for fast search of those three cuts. Figure 3 shows an example of computing the leaf values given c_i, c_j^1 and c_j^2. Similarly, we can quickly compute the RSS given any combination of 3 cuts once the leaf values are available, just as we did in Section 4.1.4, and therefore it is very fast to search for the best combination of cuts in this space. Similarly, we search for the best combination of 3 cuts with 1 cut on x_j and 2 cuts on x_i and pick the better model with lower RSS. It is easy to see the complexity is $O(N + b^2)$, where b is the number of bins for each feature and $b = 256$ in our case.

[1]Note that this is not the number of bins used in FAST, the interaction detection process. Here we use 256 bins for feature/pair shaping.

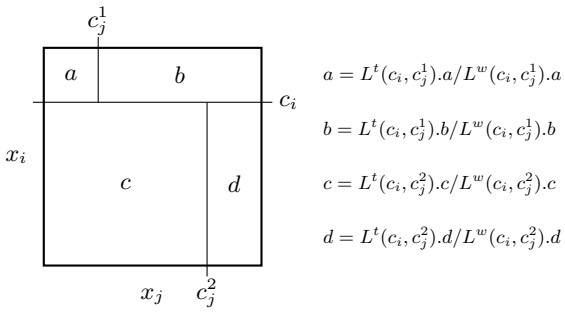


Figure 3: Illustration for computing shape function for pairwise interaction.

Dataset	Size	Attributes	%Pos
Delta	7192	6	-
CompAct	8192	22	-
Pole	15000	49	-
CalHousing	20640	9	-
MSLR10k	1200192	137	-
Spambase	4601	58	39.40
Gisette	6000	5001	50.00
Magic	19020	11	64.84
Letter	20000	17	49.70
Physics	50000	79	49.72

Table 1: Datasets.

4.2.2 Further Relaxation

For large datasets, even refitting the model on selected pairs can be very expensive. Therefore, we propose to use the ranking of FAST right after Stage 1, to select the top-K pairs to $\mathcal{S}$, and fit a model using the pairs in $\mathcal{S}$ on the residual R, where K is chosen according to computing power.

4.2.3 Diagnostics

Models that combine both accuracy and intelligibility are important. Usually $\mathcal{S}$ will still be an overcomplete set. For intelligibility, once we have learned the best model in $\mathcal{H}$, we would like to rank all terms (one- and two-dimensional components) so that we can focus on the most important features, or pairwise interactions. Therefore, we need to assign weights for each term. We use $\sqrt{E[f_u^2]}$, the standard deviation of f_u (since $E[f_u] = 0$), as the weight for term u. Note this is a natural generalization of the weights in the linear models; this is easy to see since $f_i(x_i) = w_i x_i$, $\sqrt{E[f_i^2]}$ is equivalent to $|w_i|$ if features are normalized so that $E[x_i^2] = 1$.

feature selection 想法：結合lasso?

5. EXPERIMENTS

In this section we report experimental results on both synthetic and real datasets. The results in Section 5.1 show GA^{2}M learns models that are nearly as accurate as full-complexity random forest models while using terms that depend only on single features and pairwise interactions and thus are intelligible. The results in Section 5.2 demonstrate that FAST finds the most important interactions of $O(n^2)$ feature pairs to include in the model. Section 5.3 compares the computational cost of FAST and GA^{2}M to competing methods. Section 5.4 briefly discusses several important de-

Model	Delta	CompAct	Pole	CalHousing	MSLR10k	Mean
Linear Regression	0.58±0.01	7.92±0.47	30.41±0.24	7.28±0.80	0.76±0.00	1.52±0.79
GAM	0.57±0.02	2.74±0.04	21.62±0.38	5.76±0.55	0.75±0.00	1.00±0.00
GA^{2}M Rand	-	-	11.37±0.38	-	0.73±0.00	-
GA^{2}M Coef	-	-	11.61±0.43	-	0.73±0.00	-
GA^{2}M Order	-	-	10.81±0.29	-	0.74±0.00	-
GA^{2}M FAST	**0.55±0.02**	**2.53±0.02**	**10.59±0.35**	**5.00±0.91**	**0.73±0.00**	**0.84±0.20**
Random Forests	0.53±0.19	2.45±0.08	11.38±1.03	4.90±0.81	0.71±0.00	0.83±0.17

Table 2: RMSE for regression datasets. Each cell contains the mean RMSE ± one standard deviation. Average normalized score is shown in the last column, calculated as relative improvement over GAM.

Model	Spambase	Gisette	Magic	Letter	Physics	Mean
Logistic Regression	6.22±0.93	15.78±3.28	17.11±0.08	27.54±0.27	30.02±0.37	1.79±1.25
GAM	5.09±0.64	3.95±0.65	14.85±0.28	17.84±0.20	28.83±0.24	1.00±0.00
GA^{2}M Rand	5.04±0.52	3.53±0.61	-	-	28.82±0.25	-
GA^{2}M Coef	4.89±0.54	3.43±0.55	-	-	28.74±0.37	-
GA^{2}M Order	4.93±0.65	3.08±0.55	-	-	28.76±0.34	-
GA^{2}M FAST	**4.78±0.70**	**2.91±0.38**	**13.88±0.32**	**8.62±0.31**	**28.20±0.18**	**0.81±0.21**
Random Forests	4.76±0.70	3.25±0.47	12.45±0.64	6.16±0.22	28.48±0.40	0.79±0.26

Table 3: Error rate for classification datasets. Each cell contains the error rate ± one standard deviation. Average normalized score is shown in the last column, calculated as relative improvement over GAM.

sign choices made for FAST and GA^{2}M. Finally, Section 5.5 concludes with a case study.

5.1 Model Accuracy on Real Datasets

We run experiments on ten real datasets to show the accuracy that GA^{2}M can achieve with models that depend only on 1-d features and pairwise feature interactions.

5.1.1 Datasets

Table 1 summarizes the 10 datasets. Five are regression problems: "Delta" is the task of controlling the ailerons of an F16 aircraft [1]. "CompAct" is from the Delve repository and describes the state of multiuser computers [2]. "Pole" describes a telecommunication problem [23]. "CalHousing" describes how housing prices depend on census variables [16]. "MSLR10k" is a learning-to-rank dataset but we treat relevance as regression targets [3]. The other five datasets are binary classification problems: The "Spambase", "Magic" and "Letter" datasets are from the UCI repository [4]. "Gisette" is from the NIPS feature selection challenge [5]. "Physics" is from the KDD Cup 2004 [6].

The features in all datasets are discretized into 256 equi-frequency bins. For each model we include at most 1000 feature pairs; we include all feature pairs in the six problems with least dimension, and the top 1000 feature pairs found by FAST on the "Pole", "MSLR10k", "Spambase", "Gisette", and "Physics" datasets. Although it is possible that higher accuracy might be obtained by including more or fewer feature pairs, search for the optimal number of pairs is expensive and GA^{2}M is reasonably robust to excess feature pairs. However, it is too expensive to include all feature pairs on problems with many features. We use 8 bins for FAST in all experiments.

5.1.2 Results

We compare GA^{2}M to linear/logistic regression, feature shaping (GAMs) without interactions, and full-complexity random forests. For regression problems we report root mean squared error (RMSE) and for classification problems we report 0/1 loss. To compare results across different datasets, we normalize results by the error of GAMs on each dataset. For all experiments, we train on 80% of the data and hold aside 20% of the data as test sets.

In addition to FAST, we also consider three baseline methods on five high dimensional datasets, i.e., GA^{2}M Rand, GA^{2}M Coef and GA^{2}M Order. GA^{2}M Rand means we add same number of random pairs to GAM. GA^{2}M Order and GA^{2}M Coef use the weights of 1-d features in GAM to propose pairs; GA^{2}M Order generates pairs by the order of 1-d features and GA^{2}M Coef generates pairs by the product of weights of 1-d features.

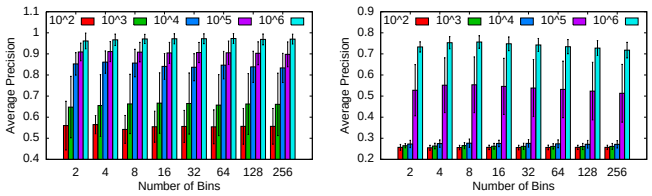
The regression and classification results are presented in Table 2 and Table 3. As expected, the improvement over linear models from shaping individual features (GAMs) is substantial: on average feature shaping reduces RMSE 34% on the regression problems, and reduces 0/1 loss 44% on the classification problems. What is surprising, however, is that by adding shaped pairwise interactions to the models, GA^{2}M FAST substantially closes the accuracy gap between unintelligible full-complexity models such as random forests and GAMs. On some datasets, GA^{2}M FAST even outperforms the best random forest model. Also, none of the baseline methods perform comparably GA^{2}M FAST.

5.2 Detecting Feature Interactions with FAST

In this section we evaluate how accurately FAST detects feature interactions on synthetic problems.

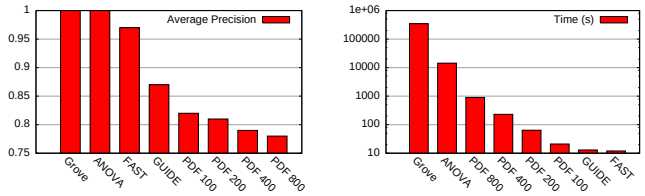
5.2.1 Sensitivity to the Number of Bins

To evaluate sensitivity of FAST we use the synthetic function generator in [10] to generate random functions. Because these are synthetic function, we know the ground truth interacting pairs and use average precision (area under the precision-recall curve evaluated at true points) as the eval-



(a) 10 features. (b) 100 features.

Figure 4: Sensitivity of FAST to the number of bins.



(a) (b)

Figure 5: Precision/Cost on synthetic function.

uation metric. We vary $b = 2, 4, ..., 256$ and the dataset size $N = 10^2, 10^3, ..., 10^6$. For each fixed N, we generate datasets with n features and k higher order interactions x_u, where $|u| = \lfloor 1.5 + r \rfloor$ and r is drawn from an exponential distribution with mean $\lambda = 1$. We experiment with two cases: 10 features with 25 higher order interactions and 100 features with 1000 higher order interactions.

Figure 4 shows the mean average precision and variance for 100 trials at each setting. As expected, average precision increases as dataset size increases, and decreases as the number of features increases from 10 (left graph) to 100 (right graph). When there are only 10 features and as many as 10^6 samples, FAST ranks *all* true interactions above all non-interacting pairs (average precision = 1) in most cases, but as the sample size decreases or the problem difficulty increases average precision drops below 1. In the graph on the right with 100 features there are 4950 feature pairs, and FAST needs large sample sizes (10^6 or greater) to achieve average precision above 0.7, and as expected performs poorly when there are fewer samples than pairs of features.

On these test problems the optimal number of bins appears to be about $b = 8$, with average precision falling slightly for number of bins larger and smaller than 8. This is a classic bias-variance tradeoff: smaller b reduces the chances of overfitting but at the risk of failing to model some kinds of interactions, while large b allows more complex interactions to be modeled but at the risk of allowing some false interactions to be confused with weak true interactions.

5.2.2 Accuracy

The previous section showed that FAST accurately detects feature interactions when the number of samples is much larger than the number of feature pairs, but that accuracy drops as the number of feature pairs grows comparable to and then larger than the number of samples. In this section we compare the accuracy of FAST to the interaction detection methods discussed in Section 3.2. For ANOVA, we use R package `mgcv` to compute p-values under a Wald test [25]. For PDF, we use `RuleFit` package and we choose $m = 100, 200, 400, 800$, where m is the sample size that trades off efficiency and accuracy [7]. Grove is available in `TreeExtra` package [8].

Here we conduct experiments on synthetic data generated by the following function [14, 22].

$$F(\boldsymbol{x}) = \pi^{x_1 x_2}\sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7 \quad (10)$$

Variables x_4, x_5, x_8, x_{10} are uniformly distributed in $[0.6, 1]$ and the other variables are uniformly distributed in $[0, 1]$. We generate $10,000$ points for these experiments. Figure 5(a) shows the average precision of the methods. On this problem, the Grove and ANOVA methods are accurate and rank all 11 true pairs in the top of the list. FAST is almost as good and correctly ranks the top ten pairs. The other methods are significantly less accurate than Grove, ANOVA, and FAST.

To understand why FAST does not pick up the 11^{th} pair, we plot heat maps of the residuals of selected pairs in Figure 6. (x_1, x_2) and (x_2, x_7) are two of the correctly ranked true pairs, (x_1, x_7) is a false pair ranked below the true pairs FAST detects correctly but above the true pair it misses, and (x_8, x_{10}) is the true pair FAST misses and ranks below this false pair. The heat maps show strong interactions are easy to distinguish, but some false interactions such as (x_1, x_7) can have signal as strong as that of weak true interactions such as (x_8, x_{10}). In fact, Sorokina et al. found that x_8 is a weak feature, and do not consider pairs that use x_8 as interactions on $5,000$ samples [22], so we are near the threshold of detectability of (x_8, x_{10}) going from $5,000$ to $10,000$ samples.

5.2.3 Feature Correlation and Spurious Pairs

If features are correlated, spurious interactions may be detected because it is difficult to tell the difference between a true interaction between x_1 and x_2 and the spurious interaction between x_1 and x_3 when x_3 is strongly correlated with x_2; any interaction detection method such as FAST that examines pairs in isolation will have this problem. With GA^{2}M, however, it is fine to include some false positive pairs because GA^{2}M is able to post-filter false positive pairs by looking at the term weights of shaped interactions in the final model.

To demonstrate this, we use the synthetic function in Equation 10, but make x_6 correlated to x_1. We generate 2 datasets, one with $\rho(x_1, x_6) = 0.5$ and the other with $\rho(x_1, x_6) = 0.95$, where ρ is the correlation coefficient. We run FAST on residuals after feature shaping. We give the top 20 pairs found by FAST to GA^{2}M, which then uses gradient boosting to shape those pairwise interactions. Figure 7 illustrates how the weights of selected pairwise interactions evolve after each step of gradient boosting. Although the pair (x_2, x_6) can be incorrectly introduced by FAST because of the high correlation between x_1 and x_6, the weight on this false pair decreases quickly as boosting proceeds, indicating that this pair is spurious. This not only allows the model trained on the pairs to remain accurate in the face of spurious pairs, but also reduces the weight (and ranking) given to this shaped term so that intelligibility is not be hurt by the spurious term.

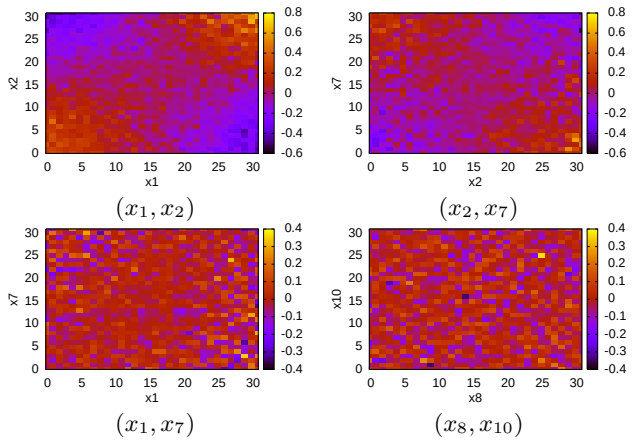
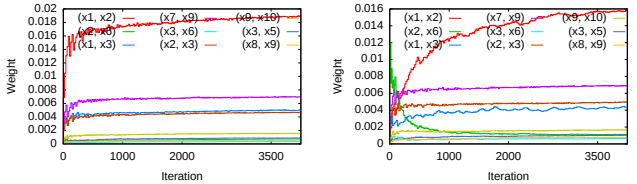


Figure 6: True/Spurious heat maps. Features are discretized into 32 bins for visualization.



(a) $\rho(x_1, x_6) = 0.5$ (b) $\rho(x_1, x_6) = 0.95$

Figure 7: Weights for pairwise interaction terms in the model.

5.3 Scalability

Figure 5(b) illustrates the running time of different methods on $10,000$ samples from Equation 10. Model building time is included. FAST takes about 10 seconds to rank all possible pairs while the two other accurate methods, ANOVA and Grove, are 3-4 orders of magnitude slower. Grove, which is probably the most accurate interaction detection method currently available, takes almost a week to run once on this data. This shows the advantage of FAST; it is very fast with high accuracy. On this problem FAST takes less than 1 second to rank all pairs and the majority of time is devoted to building the additive model.

Figure 8 shows the running time of FAST per pair on real datasets. It is clear that on real datasets, FAST is both accurate and efficient.

5.4 Design Choices

An alternate to interaction detection that we considered was to build ensembles of trees on residuals after shaping the individual features and then look at tree statistics to find combinations of features that co-occur in paths more often than their independent rate warrants. By using 1-step look-ahead at the root we also hoped to partially mitigate the myopia of greedy feature installation to make interactions more likely to be detected. Unfortunately, features with high "co-occurence counts" did not correlate well with true interactions on synthetic test problems, and the best tree-based methods we could devise did not detect interactions as well as FAST, and were considerably more expensive.

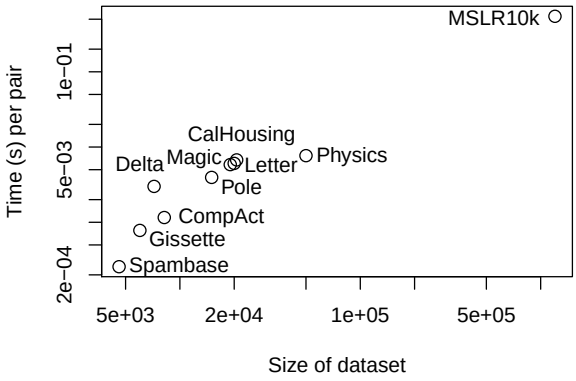


Figure 8: Computational cost on real datasets.

5.5 Case Study: Learning to Rank

Learning-to-rank is an important research topic in the data mining, machine learning and information retrieval communities. In this section, we train intelligible models with shaped one-dimensional features and pairwise interactions on the "MSLR10k" dataset. A complete description of features can be found in [3]. We show the top 10 most important individual features and their shape functions in first two rows of Figure 9. The number above each plot is the weight for the corresponding term in the model. Interestingly, we found BM25 [20], usually considered as a powerful feature for ranking, ranked 70^{th} (BM25_url) in the list after shaping. Other features such as IDF (inverse document frequency) enjoy much higher weight in the learned model.

The last two rows of Figure 9 show the 10 most important pairwise interactions and their term strengths. Each of them shows a clear interaction that could not be modeled by additive terms. The non-linear shaping of the individual features in the top plots and the pairwise interactions in the bottom plots are intelligible to experts and feature engineers, but would be well hidden in full-complexity models.

6. CONCLUSIONS

We present a framework called GA^{2}M for building intelligible models with pairwise interactions. Adding pairwise interactions to traditional GAMs retains intelligibility, while substantially increasing model accuracy. To scale up pairwise interaction detection, we propose a novel method called FAST that efficiently measures the strength of all potential pairwise interactions.

Acknowledgements. We thank the anonymous reviewers for their valuable comments, and we thank Nick Craswell of Microsoft Bing for insightful discussions. This research has been supported by the NSF under Grants IIS-0911036 and IIS-1012593. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

7. REFERENCES

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

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

[3] `http://research.microsoft.com/en-us/projects/mslr/`.

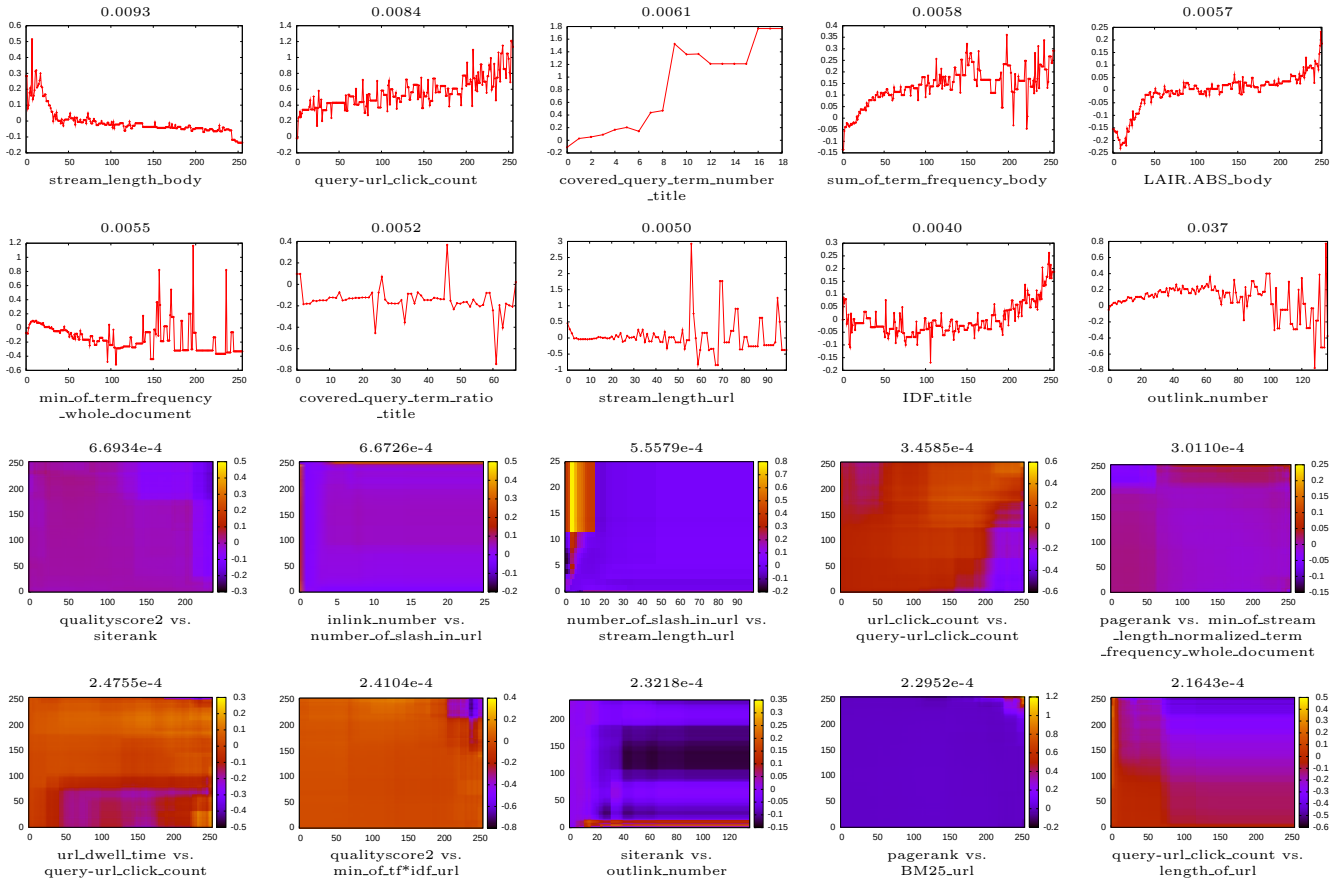


Figure 9: Shapes of features and pairwise interactions for the "MSLR10k" dataset with weights. Top two rows show top 10 strongest features. Next two rows show top 10 strongest interactions.

[4] `http://archive.ics.uci.edu/ml/`.

[5] `http://www.nipsfsc.ecs.soton.ac.uk/`.

[6] `http://osmot.cs.cornell.edu/kddcup/`.

[7] `http://www-stat.stanford.edu/~jhf/R-RuleFit.html`.

[8] `http://additivegroves.net`.

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

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

[11] J. Friedman and B. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, pages 916–954, 2008.

[12] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

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

[14] G. Hooker. Discovering additive structure in black box functions. In *KDD*, 2004.

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

[16] R. Kelley Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[17] P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 2007.

[18] W. Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386, 2002.

[19] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *KDD*, 2012.

[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008.

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

[22] D. Sorokina, R. Caruana, M. Riedewald, and D. Fink. Detecting statistical interactions with additive groves of trees. In *ICML*, 2008.

[23] S. M. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

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

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