Rule-Enhanced Penalized Regression by Column Generation using Rectangular Maximum Agreement

Jonathan Eckstein [1] **Noam Goldberg** [2] **Ai Kagawa** [3]

Abstract

We describe a procedure enhancing L_1-penalized regression by adding dynamically generated rules describing multidimensional "box" sets. Our rule-adding procedure is based on the classical column generation method for high-dimensional linear programming. The pricing problem for our column generation procedure reduces to the $\mathcal{NP}$-hard rectangular maximum agreement (RMA) problem of finding a box that best discriminates between two weighted datasets. We solve this problem exactly using a parallel branch-and-bound procedure. The resulting rule-enhanced regression method is computation-intensive, but has promising prediction performance.

1. Motivation and Overview

This paper considers the general learning problem in which we have m observation vectors $X_1, \ldots, X_m \in \mathbb{R}^n$, with matching response values $y_1, \ldots, y_m \in \mathbb{R}$. Each response y_i is a possibly noisy evaluation of an unknown function $f : \mathbb{R}^n \to \mathbb{R}$ at X_i, that is, $y_i = f(X_i) + e_i$, where $e_i \in \mathbb{R}$ represents the noise or measurement error. The goal is to estimate f by some $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ such that $\hat{f}(X_i)$ is a good fit for y_i, that is, $|\hat{f}(X_i) - y_i|$ tends to be small. The estimate $\hat{f}$ may then be used to predict the response value y corresponding to a newly encountered observation $x \in \mathbb{R}^n$ through the prediction $\hat{y} = \hat{f}(x)$. A classical linear regression model is one simple example of the many possible techniques one might employ for constructing $\hat{f}$. The classical regression approach to this problem is to posit a particular functional form for $\hat{f}(x)$ (for example, an affine function of x) and then use an optimization procedure to estimate the parameters in this functional form.

Here, we are interested in cases in which a concise candidate functional form for $\hat{f}$ is not readily apparent, and we wish to estimate $\hat{f}$ by searching over a very high-dimensional space of parameters. For example, Breiman (2001) proposed the method of random forests, which constructs $\hat{f}$ by training regression trees on multiple random subsamples of the data, and then averaging the resulting predictors. Another proposal is the RuleFit algorithm (Friedman & Popescu, 2008), which enhances L_1-regularized regression by generating box-based rules to use as additional explanatory variables. Given $a, b \in \mathbb{R}^n$ with $a \leq b$, the rule function $r_{(a,b)} : \mathbb{R}^n \to \{0, 1\}$ is given by

$$r_{(a,b)}(x) = I\left(\wedge_{j\in\{1,\ldots,n\}}(a_j \leq x_j \leq b_j)\right), \qquad (1)$$

that is $r_{(a,b)}(x) = 1$ if $a \leq x \leq b$ (componentwise) and $r_{(a,b)}(x) = 0$ otherwise. RuleFit generates rules through a two-phase procedure: first, it determines a regression tree ensemble, and then decomposes these trees into rules and determines the regression model coefficients (including for the rules).

The approach of Dembczyński et al. (2008a) generates rules more directly (without having to rely on an initial ensemble of decision trees) within gradient boosting (Friedman, 2001) for non-regularized regression. In this scheme, a greedy procedure generates the rules within a gradient descent method runs that for a predetermined number of iterations. Aho et al. (2012) extended the RuleFit method to solve more general multi-target regression problems. For the special case of single-target regression, however, their experiments suggest that random forests and RuleFit outperform several other methods, including their own extended implementation and the algorithm of Dembczyński et al. (2008a). Compared with random forests and other popular learning approaches such as kernel-based methods and neural networks, rule-based approaches have the advantage of generally being considered more accessible and easier to interpret by domain experts. Rule-based methods also have a considerable history in classification settings, as in for example Weiss & Indurkhya (1993), Cohen & Singer

[1]Management Science and Information Systems, Rutgers University, Piscataway, NJ, USA [2]Department of Management, Bar-Ilan University, Ramat Gan, Israel [3]Doctoral Program in Operations Research, Rutgers University, Piscataway, NJ, USA. Correspondence to: Jonathan Eckstein <jeckstei@business.rutgers.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

(1999), and Dembczyński et al. (2008b).

Here, we propose an iterative optimization-based regression procedure called REPR (Rule-Enhanced Penalized Regression). Its output models resemble those of RuleFit, but our methodology draws more heavily on exact optimization techniques from the field of mathematical programming. While it is quite computationally intensive, its prediction performance appears promising. As in RuleFit, we start with a linear regression model (in this case, with L_1-penalized coefficients to promote sparsity), and enhance it by synthesizing rules of the form (1). We incrementally adjoin such rules to our (penalized) linear regression model as if they were new observation variables. Unlike RuleFit, we control the generation of new rules using the classical mathematical programming technique of column generation. Our employment of column generation roughly resembles its use in the LPBoost ensemble classification method of Demiriz et al. (2002).

Column generation involves cyclical alternation between optimization of a *restricted master* problem (in our case a linear or convex quadratic program) and a *pricing problem* that finds the most promising new variables to adjoin to the formulation. In our case, the pricing problem is equivalent to an $\mathcal{NP}$-hard combinatorial problem we call Rectangular Maximum Agreement (RMA), which generalizes the Maximum Mononial Agreement (MMA) problem as formulated and solved by Eckstein & Goldberg (2012). We solve the RMA problem by a similar branch-and-bound method procedure, implemented using parallel computing techniques.

To make our notation below more concise, we let X denote the matrix whose rows are $X_1^\top, \ldots, X_m^\top$, and also let $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$. We may then express a problem instance by the pair (X, y). We also let x_{ij} denote the $(i,j)^{\text{th}}$ element of this matrix, that is, the value of variable j in observation i.

2. A Penalized Regression Model with Rules

Let K be a set of pairs $(a, b) \in \mathbb{R}^n \times \mathbb{R}^n$ with $a \leq b$, constituting a catalog of all the possible rules of the form (1) that we wish to be available to our regression model. The set K will typically be extremely large: restricting each a_j and b_j to values that appear as x_{ij} for some i, which is sufficient to describe all possible distinct behaviors of rules of the form (1) on the dataset X, there are still $\prod_{j=1}^n \ell_j(\ell_j+1)/2 \geq 3^n$ possible choices for (a, b), where $\ell_j = |\bigcup_{i=1}^m \{x_{ij}\}|$ is the number of distinct values for x_{ij}.

The predictors $\hat{f}$ that our method constructs are of the form

$$\hat{f}(x) = \beta_0 + \sum_{j=1}^{n} \beta_j x_j + \sum_{k \in K} \gamma_k r_k(x) \tag{2}$$

for some $\beta_0, \beta_1, \ldots, \beta_n, (\gamma_k)_{k \in K} \in \mathbb{R}$. Finding an $\hat{f}$ of this form is a matter of linear regression, but with the regression coefficients in a space with the potentially very high dimension of $1 + n + |K|$. As is now customary in regression models in which the number of explanatory variables potentially outnumbers the number of observations, we employ a LASSO-class model in which all explanatory variables except the constant term have L_1 penalties. Letting $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}^{|K|}$, let $f_{\beta_0,\beta,\gamma}(\,\cdot\,)$ denote the predictor function in (2). We then propose to estimate β_0, β, γ by solving

$$\min_{\beta_0,\beta,\gamma} \left\{ \sum_{i=1}^{m} |f_{\beta_0,\beta,\gamma}(X_i) - y_i|^p + C \left\|\beta\right\|_1 + E \left\|\gamma\right\|_1 \right\}, \tag{3}$$

where $p \in \{1, 2\}$ and $C, E \geq 0$ are scalar parameters. For $p = 2$ and $C = E > 0$, this model is essentially the classic LASSO as originally proposed by Tibshirani (1996).

To put (3) into a more convenient form for our purposes, we split the regression coefficient vectors into positive and negative parts, so $\beta = \beta^+ - \beta^-$ and $\gamma = \gamma^+ - \gamma^-$, with $\beta^+, \beta^- \in \mathbb{R}^n_+$ and $\gamma^+, \gamma^- \in \mathbb{R}^{|K|}_+$. Introducing one more vector of variables $\epsilon \in \mathbb{R}^m$, the model shown as (4) in Figure 1 is equivalent to (3). The model is constructed so that $\epsilon_i = |f_{\beta_0,\beta,\gamma}(X_i) - y_i|$ for $i = 1, \ldots, m$. If $p = 1$, the model is a linear program, and if $p = 2$ it is a convex, linearly constrained quadratic program. In either case, there are $2m$ constraints (other than nonnegativity), but the number of variables is $1 + m + 2n + 2\,|K|$.

Because of this potentially unwieldy number of variables, we propose to solve (4) by using the classical technique of column generation, which dates back to Ford & Fulkerson (1958) and Gilmore & Gomory (1961); see for example Section 7.3 of Griva et al. (2009) for a recent textbook treatment. In brief, column generation cycles between solving two optimization problems, the restricted master problem and the pricing problem. In our case, the restricted master problem is the same as (4), but with K replaced by some (presumably far smaller) $K' \subseteq K$. We initially choose $K' = \emptyset$. Solving the restricted master problem yields optimal Lagrange multipliers $\nu \in \mathbb{R}^m_+$ and $\mu \in \mathbb{R}^m_+$ (for the constraints other than simple nonnegativity). For each rule $k \in K$, these Lagrange multipliers yield respective reduced costs $\mathrm{rc}[\gamma_k^+]$, $\mathrm{rc}[\gamma_k^-]$ for the variables γ_k^+, γ_k^- that are in the master problem, but not the restricted master. One then solves the pricing problem, whose job is to identify the smallest of these reduced costs. The reduced cost $\mathrm{rc}[v]$ of a variable v indicates the rate of change of the objective function as one increases v away from 0. If the smallest reduced

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m} \epsilon_i^p + C \sum_{j=1}^{n} (\beta_j^+ + \beta_j^-) + E \sum_{k \in K} (\gamma_k^+ + \gamma_k^-) \\
\text{s.t.} \quad & \beta_0 + X_i^\top (\beta^+ - \beta^-) + \sum_{k \in K} r_k(X_i)(\gamma_k^+ - \gamma_k^-) - \epsilon_i \;\le\; y_i, \qquad i = 1, \ldots, m \\
& -\beta_0 - X_i^\top (\beta^+ - \beta^-) - \sum_{k \in K} r_k(X_i)(\gamma_k^+ - \gamma_k^-) - \epsilon_i \;\le\; -y_i, \qquad i = 1, \ldots, m \\
& \epsilon \ge 0 \qquad \beta^+, \beta^- \ge 0 \qquad \gamma^-, \gamma^+ \ge 0
\end{aligned}
\tag{4}
$$

Figure 1. Formulation of the REPR master problem, with scalar parameters $p \in \{1, 2\}$, $C \ge 0$, and $E \ge 0$. The decision variables are $\epsilon \in \mathbb{R}^m$, $\beta_0 \in \mathbb{R}$, $\beta^+, \beta^- \in \mathbb{R}^n$, and $\gamma^+, \gamma^- \in \mathbb{R}^{|K|}$. The input data are $X = [X_1 \cdots X_m]^\top \in \mathbb{R}^{m \times n}$ and $y \in \mathbb{R}^m$.

cost is nonnegative, then clearly all the reduced costs are nonnegative, which means that the current restricted master problem yields an optimal solution to the master problem by setting $\gamma_k^+ = \gamma_k^- = 0$ for all $k \in K \backslash K'$, and the process terminates. If the smallest reduced cost is negative, we adjoin elements to K', including at least one corresponding to a variable γ_k^+ or γ_k^- with a negative reduced cost, and we repeat the process, re-solving the expanded restricted master problem.

In our case, the reduced costs take the form

$$
\begin{aligned}
\mathrm{rc}[\gamma_k^+] &= E - \sum_{i=1}^{m} r_k(x_i)\nu_i + \sum_{i=1}^{m} r_k(x_i)\mu_i \\
\mathrm{rc}[\gamma_k^-] &= E + \sum_{i=1}^{m} r_k(x_i)\nu_i - \sum_{i=1}^{m} r_k(x_i)\mu_i
\end{aligned}
$$

and hence we have for each $k \in K$ that

$$
\min\left\{\mathrm{rc}[\gamma_k^+], \mathrm{rc}[\gamma_k^-]\right\} = E - \left| \sum_{i=1}^{m} r_k(x_i)(\nu_i - \mu_i) \right|. \tag{5}
$$

Therefore, the pricing problem may be solved by maximizing the second term on the right-hand side of (5), that is, finding

$$
z^* = \max_{k \in K} \left| \sum_{i=1}^{m} r_k(x_i)(\nu_i - \mu_i) \right|, \tag{6}
$$

and the stopping condition for the column generation procedure is $z^* \le E$. This problem turns out to be equivalent to the RMA problem, whose formulation and solution we now describe.

3. The RMA Problem

3.1. Formulation and Input Data

Suppose we have m observations and n explanatory variables, expressed using a matrix $X \in \mathbb{R}^{m \times n}$ as above. Each observation $i \in \{1, \ldots, m\}$ is assigned a nonnegative weight $w_i \in \mathbb{R}_+$. For any set $S \subseteq \{1, \ldots, m\}$, let $w(S) = \sum_{i \in S} w_i$. We also assume we are given a partition of the observations into two subsets, a "positive" subset $\Omega^+ \subset \{1, \ldots, m\}$ and a "negative" subset $\Omega^- = \{1, \ldots, m\} \backslash \Omega^+$.

Given two vectors $a, b \in \mathbb{R}^n$, let $B(a, b)$ denote the "box" $\{x \in \mathbb{Z}^n \mid a \le x \le b\}$. Given the input data X, the *coverage* $\mathrm{Cvr}_X(a, b)$ of $B(a, b)$ consists of the indices of the observations from X falling within $B(a, b)$, that is,

$$
\mathrm{Cvr}_X(a, b) = \{i \in \{1, \ldots, m\} \mid a \le X_i \le b\}.
$$

The rectangular maximum agreement (RMA) problem is

$$
\begin{aligned}
\max \quad & \left| w\big(\Omega^+ \cap \mathrm{Cvr}_X(a, b)\big) - w\big(\Omega^- \cap \mathrm{Cvr}_X(a, b)\big) \right| \\
\text{s.t.} \quad & a, b \in \mathbb{R}^n,
\end{aligned}
\tag{7}
$$

with decision variables $a, b \in \mathbb{R}^n$. Essentially implicit in this formulation is the constraint that $a \le b$, since if $a \not\le b$ then $\mathrm{Cvr}_X(a, b) = \emptyset$ and the objective value is 0. The previously mentioned MMA problem is the special case of RMA in which all the observations are binary, $X \in \{0, 1\}^{m \times n}$. Since the MMA problem is $\mathcal{NP}$-hard (Eckstein & Goldberg, 2012), so is RMA.

If we take K to be the set of all possible boxes on $\mathbb{R}^n$, the pricing problem (6) may be reduced to RMA by setting

$$
(\forall\, i = 1, \ldots, m): \quad w_i = |\nu_i - \mu_i| \tag{8}
$$

$$
\Omega^+ = \{i \in \{1, \ldots, m\} \mid \nu_i \ge \mu_i\}, \tag{9}
$$

and thus $\Omega^- = \{i \in \{1, \ldots, m\} \mid \nu_i < \mu_i\}$.

3.2. Preprocessing and Restriction to $\mathbb{N}$

Any RMA problem instance may be converted to an equivalent instance in which all the observation data are integer. Essentially, for each coordinate $j = 1, \ldots, n$, one may simply record the distinct values of x_{ij} and replace each x_{ij} with its ordinal position among these values. Algorithm 1, with its parameter δ set to 0, performs exactly this procedure, outputing a equivalent data matrix $\overline{X} \in \mathbb{N}^{m \times n}$ and a vector $\ell \in \mathbb{N}^n$ whose j^{th} element is $\ell_j = \left|\bigcup_{i=1}^{m} \{x_{ij}\}\right|$

Algorithm 1 Preprocessing discretization algorithm

Input: $X \in \mathbb{R}^{m \times n}$, $\delta \geq 0$
Output: $\overline{X} \in \mathbb{N}^{m \times n}$, $\ell \in \mathbb{N}^n$
ProcessData
for $j = 1$ **to** n **do**
  $\ell_j \leftarrow 0$
  Sort $x_{1j}, \ldots, x_{mj}$ and set $(k_1, \ldots, k_m)$ such that $x_{k_1 j} \leq x_{k_2 j} \leq \cdots \leq x_{k_m j}$
  $\bar{x}_{k_1, j} \leftarrow 0$
  for $i = 1$ **to** $m - 1$ **do**
    if $x_{k_{i+1} j} - x_{k_i j} > \delta \cdot (x_{k_m j} - x_{k_1 j})$ **then**
      $\ell_j \leftarrow \ell_j + 1$
    end if
    $\bar{x}_{k_{i+1} j} \leftarrow \ell_j$
  end for
  $\ell_j \leftarrow \ell_j + 1$
end for
return $(\overline{X}, \ell)$

as defined in the previous section. Algorithm 1's output values $\bar{x}_{ij}$ for attribute j vary between 0 and $\ell_j - 1$.

The number of distinct values ℓ_j of each explanatory variable j directly influences the difficulty of RMA instances. To obtain easier instances, Algorithm 1 can combine its "integerization" process with some binning of nearby values. Essentially, if the parameter δ is positive, the algorithm bins together consecutive values x_{ij} that are within relative tolerance δ, resulting in a smaller number of distinct values ℓ_j for each explanatory variable j.

Some datasets contain both categorical and numerical data. In addition to Algorithm 1, we also convert each k-way categorical attribute into $k - 1$ binary attributes.

Within the context of our REPR regression method, we set the RMA weight vector and data partition as in (8)-(9), integerize the data X using Algorithm 1 with some (small) parameter value δ, solve the RMA problem, and then translate the resulting boxes back to the original, pre-integerized coordinate system. We perform this translation by expanding box boundaries to lie halfway between the boundaries of the clusters of points grouped by Algorithm 1, except when the lower boundary of the box has the lowest possible value or the upper boundary has the largest possible value. In these cases, we expand the box boundaries to $-\infty$ or $+\infty$, respectively. More precisely, for each observation variable j and $v \in \{0, \ldots, \ell_j - 1\}$, let $x^{\min}_{j,v}$ be the smallest value of x_{ij} assigned to the integer value v by Algorithm 1, and $x^{\max}_{j,v}$ be the largest. If $\hat{a}, \hat{b} \in \mathbb{N}^n$, $\hat{a} \leq \hat{b}$ describe an integerized box arising from the solution of the preprocessed RMA problem, we choose the corresponding box boundaries $a, b \in \mathbb{R}^n$ in the original coordinate system to be given by, for $j = 1, \ldots, n$,

$$a_j = \begin{cases} -\infty, & \text{if } \hat{a}_j = 0 \\ \frac{1}{2}(x^{\max}_{j,\hat{a}_j - 1} + x^{\min}_{j,\hat{a}_j}), & \text{otherwise} \end{cases}$$

$$b_j = \begin{cases} +\infty, & \text{if } \hat{b}_j = \ell_j - 1 \\ \frac{1}{2}(x^{\max}_{j,\hat{b}_j} + x^{\min}_{j,\hat{b}_j + 1}), & \text{otherwise.} \end{cases}$$

Overall, our procedure is equivalent to solving the pricing problem (6) over some set of boxes $K = \mathcal{K}_\delta(X)$. For $\delta = 0$, the resulting set of boxes $\mathcal{K}_0(X)$ is such that the corresponding set of rules $\{r_k \mid k \in \mathcal{K}_0(X)\}$ comprises every box-based rule distinguishable on the dataset X. For small positive values of δ, the set of boxes $\mathcal{K}_\delta(X)$ excludes those corresponding to rules that "cut" between very closely spaced observations.

3.3. Branch-and-Bound Subproblems

In this and the following two subsections, we describe the key elements of our branch-and-bound procedure for solving the RMA problem, assuming that the data X have already been preprocessed as above. For brevity, we omit some details which will instead be covered in a forthcoming publication. For general background on branch-and-bound algorithms, Morrison et al. (2016) provide a recent survey with numerous citations.

Branch-and-bound methods search a tree of *subproblems*, each describing some subset of the search space. In our RMA method, each subproblem P is characterized by four vectors $\underline{a}(P), \overline{a}(P), \underline{b}(P), \overline{b}(P) \in \mathbb{N}^n$, and represents search space subset consisting of vector pairs (a, b) for which $\underline{a}(P) \leq a \leq \overline{a}(P)$ and $\underline{b}(P) \leq b \leq \overline{b}(P)$. Any valid subproblem conforms to $\underline{a}(P) \leq \overline{a}(P)$, $\underline{b}(P) \leq \overline{b}(P)$, $\underline{a}(P) \leq \underline{b}(P)$, and $\overline{a}(P) \leq \overline{b}(P)$. The root problem R of the branch-and-bound tree is $R = (\mathbf{0}, \ell - \mathbf{1}, \mathbf{0}, \ell - \mathbf{1})$, where where $\ell \in \mathbb{N}^n$ is as output from Algorithm 1, and $\mathbf{0}$ and $\mathbf{1}$ respectively denote the vectors $(0, 0, \ldots, 0) \in \mathbb{N}^n$ and $(1, 1, \ldots, 1) \in \mathbb{N}^n$.

3.4. Inseparability and the Bounding Function

In branch-and-bound methods, the bounding function provides an upper bound (when maximizing) on the best possible objective value in the region of the search space corresponding to a subproblem. Our bounding function is based on an extension of the notion of *inseparability* developed by Eckstein & Goldberg (2012). Consider any subproblem $P = (\underline{a}, \overline{a}, \underline{b}, \overline{b})$ and two observations i and i'. If $x_{ij} = x_{i'j}$ or $\overline{a}_j \leq x_{ij}, x_{i'j} \leq \underline{b}_j$ for each $j = 1, \ldots, n$, then $x_i, x_{i'} \in \mathbb{N}^n$ are *inseparable* with respect to $\overline{a}, \underline{b} \in \mathbb{N}^n$, in the sense that any box $B(a, b)$ with $a \leq \overline{a}$ and $b \geq \underline{b}$ must either cover both of $x_i, x_{i'}$ or neither of them.

Inseparability with respect to $\overline{a}, \underline{b}$ is an equivalence relation,

$$\beta(\underline{a}, \overline{a}, \underline{b}, \overline{b}) = \max \left\{ \begin{array}{c} \sum\limits_{C \in \mathcal{E}(\overline{a}, \underline{b})} \left[w(C \cap \mathrm{Cvr}_X(\underline{a}, \overline{b}) \cap \Omega^+) - w(C \cap \mathrm{Cvr}_X(\underline{a}, \overline{b}) \cap \Omega^-) \right]_+, \\ \sum\limits_{C \in \mathcal{E}(\overline{a}, \underline{b})} \left[w(C \cap \mathrm{Cvr}_X(\underline{a}, \overline{b}) \cap \Omega^-) - w(C \cap \mathrm{Cvr}_X(\underline{a}, \overline{b}) \cap \Omega^+) \right]_+ \end{array} \right\}. \tag{10}$$

Figure 2. The RMA bounding function.

and we denote the equivalence classes it induces among the observation indices $1, \ldots, m$ by $\mathcal{E}(\overline{a}, \underline{b})$. That is, observation indices i and i' are in the same equivalence class of $\mathcal{E}(\overline{a}, \underline{b})$ if x_i and $x_{i'}$ are inseparable with respect to $\overline{a}, \underline{b}$.

Our bounding function $\beta(\underline{a}, \overline{a}, \underline{b}, \overline{b})$ for each subproblem $P = (\underline{a}, \overline{a}, \underline{b}, \overline{b})$ is shown in (10) in Figure 2. The reasoning behind this bound is that each possible box in the set specified by $(\underline{a}, \overline{a}, \underline{b}, \overline{b})$ must either cover or not cover the entirety of each $C \in \mathcal{E}(\overline{a}, \underline{b})$. The first argument to the "max" operation reflects the situation that every equivalence class C with a positive net weight is covered, and no classes with negative net weight are covered; this is the best possible situation if the box ends up covering a higher weight of positive observations than of negative. The second "max" argument reflects the opposite situation, the best possible case in which the box covers a greater weight of negative observations than of positive ones.

3.5. Branching

The branching scheme of a branch-and-bound algorithm divides subproblems into smaller ones in order to improve their bounds. In our case, branching a subproblem $P = (\underline{a}, \overline{a}, \underline{b}, \overline{b})$ involves choosing an explanatory variable $j \in \{1, \ldots, n\}$ and a *cutpoint* $v \in \{\underline{a}_j, \ldots, \overline{b}_j - 1\} \in \mathbb{N}^n$.

There are three possible cases, the first of which is when $\underline{b}_j < \overline{a}_j$ and $v \in \{\underline{b}_j, \ldots, \overline{a}_j - 1\}$. In this case, our scheme creates three children based on the disjunction that either $b_j \leq v - 1$ (the box lies below v), $a_j \leq v \leq b_j$ (the box straddles v), or $a_j \geq v + 1$ (the box lies above v). The next case is that $v \in \{\underline{a}_j, \ldots, \min\{\overline{a}_j, \underline{b}_j\} - 1\}$, in which case the box cannot lie below v and we split P into two children based on the disjunction that either $a_j \leq v$ (the box straddles v) or $a_j \geq v + 1$ (the box is above v). The third case occurs when $v \in \{\max\{\overline{a}_j, \underline{b}_j\}, \ldots, \overline{b}_j - 1\}$, in which case we split P into two children based on the disjunction that either $b_j \leq v$ (the box is does not extend above v) or $b_j \geq v + 1$ (the box extends above v). If no v falling under one of these three cases exists for any dimension j, then the subproblem represents a single possible box, that is, $\underline{a} = \overline{a}$ and $\underline{b} = \overline{b}$. Such a subproblem is a terminal node of the branch-and-bound tree, and in this case we simply compute the RMA objective value for $a = \underline{a} = \overline{a}$ and $b = \underline{b} = \overline{b}$ as the subproblem bound.

When more than one possible variable-cutpoint pair (j, v) exists, as is typically the case, our algorithm must select one. We use two related procedures for branching selection: strong branching and cutpoint caching. In strong branching, we simply experiment with all applicable variable-cutpoint pairs (j, v), and select one that the maximizes the minimum bound of the resulting two or three children. This is a standard technique in branch-and-bound algorithms, and involves evaluating the bounds of all the potential children of the current search node. To make this process as efficient as possible, we have developed specialized data structures for manipulating equivalence classes, and we analyze the branching possibilities in a particular order. In cutpoint caching, some subproblems use strong branching, while others select from a list of cutpoints that were chosen by strong branching for previously processed search nodes. The details of these procedures will be covered in a forthcoming companion publication.

4. Full Algorithm and Implementation

The pseudocode in Algorithm 2 describes our full REPR column generation procedure for solving (4), using the RMA preprocessing and branch-and-bound methods described above to solve the pricing problem. Several points bear mentioning: first, the nonnegative scalar parameter θ allows us to incorporate a tolerance into the column generation stopping criterion, so that we terminate when all reduced costs exceed $-\theta$ instead of when all reduced costs are nonnegative. This kind of tolerance is customary in column generation methods. The tolerance δ, on the other hand, controls the space of columns searched over. Furthermore, our implementation of the RMA branch-and-bound algorithm can identify any desired number $t \geq 1$ of the best possible RMA solutions, as opposed to just one value of k attaining the maximum in (11). This t is also a parameter to our procedure, so at each iteration of Algorithm 2 we may adjoin up to t new rules to K'. Adding multiple columns per iteration is a common technique in column generation methods. Finally, the algorithm has a parameter S specifying a limit on the number of column generation iterations, meaning that at the output model will contain at most St rules.

We implemented the algorithm in C++, using the GuRoBi

Algorithm 2 REPR: Rule-enhanced penalized regression

Input: data $X \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^m$, penalty parameters $C, E \geq 0$, column generation tolerance $\theta \geq 0$, integer $t \geq 1$, aggregation tolerance $\delta \geq 0$, iteration limit S
Output: $\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^n, K' \subset \mathcal{K}_\delta(X), \gamma \in \mathbb{R}^{|K'|}$
REPR
$K' \leftarrow \emptyset$
for $s = 1, \ldots, S$ **do**
  Solve the restricted master problem to obtain optimal primal variables $(\beta_0, \beta^+, \beta^-, \gamma^+, \gamma^-)$ and dual variables (ν, μ)
  Use the RMA branch-and-bound algorithm, with preprocessing as in Algorithm 1, to identify a t-best solution $k_1, \ldots, k_t$ to

$$\max_{k \in \mathcal{K}_\delta(X)} \left| \sum_{i=1}^{m} r_k(x_i)(\nu_i - \mu_i) \right|, \qquad (11)$$

  with $k_1, \ldots, k_t$ having respective objective values $z_1 \geq z_2 \geq \cdots \geq z_t$
  if $z_1 \leq E + \theta$ **break**
  for each $l \in \{1, \ldots, t\}$ **with** $z_l > E + \theta$ **do**
    $K' \leftarrow K' \cup \{k_l\}$
  end for
end for
return $(\beta_0, \beta := \beta^+ - \beta^-, K', \gamma := \gamma^+ - \gamma^-)$

commercial optimizer (Gurobi Optimization, 2016) to solve the restricted master problems. We implemented the RMA algorithm using using the PEBBL C++ class library (Eckstein et al., 2015), an open-source C++ framework for parallel branch and bound. PEBBL employs MPI-based parallelism (Gropp et al., 1994). Since solving the RMA pricing problem is by far the most time-consuming part of Algorithm 2, we used true parallel computing only in that portion of the algorithm. The remainder of the algorithm, including solving the restricted master problems, was executed in serial and redundantly on all processors.

5. Preliminary Testing of REPR

For preliminary testing of REPR, we selected 8 datasets from the UCI repository (Lichman, 2013), choosing small datasets with continuous response variables. The first four columns of Table 1 summarize the number of observations m, the number of attributes n, and the maximum number of distinguishable box-based rules $|K_0(X)|$ for these data sets.

In our initial testing, we focused on the $p = 2$ case in which fitting errors are penalized quadratically, and set $t = 1$, that is, we added one model rule per REPR iteration. We set the iteration limit S to 100 and effectively set the termination tolerance θ so that REPR terminated when

$$z_1 \leq \max\left\{0, E \cdot \left(|\mathbb{E}[y]| - 0.1\sigma[y]\right)\right\} + 0.001,$$

where $\mathbb{E}[y]$ denotes the sample mean of the response variable and $\sigma[y]$ its sample standard deviation. We found this rule of thumb to work well in pracice, but it likely merits further study. We also chose $C = 1$ and $E = 1$. We used $\delta = 0$ for SERVO, YACHT, and MPG, and $\delta = 0.005$ for the remaining datasets.

Dataset	m	n	$\lvert K_0(X)\rvert$	REPR Time	RMA Nodes
SERVO	167	10	9.8e05	0:00:11	3.6e2
CONCRETE	103	9	2.7e29	0:29:51	3.0e5
MACHINE	209	6	2.5e15	0:03:07	5.6e4
YACHT	308	6	2.6e10	0:00:36	5.3e2
MPG	392	7	3.3e19	13:09:58	2.4e6
COOL	768	8	1.1e10	0:04:48	3.7e3
HEAT	768	8	1.1e10	0:03:42	3.9e3
AIRFOIL	1503	5	1.0e11	2:13:04	5.1e3

Table 1. Summary of experimental datasets. The last two columns respectively show REPR's average run time for an 80% sample of the dataset (on a 16-core workstation, in *hh*:*mm*:*ss* format) and the average resulting number of RMA search nodes per RMA invocation.

With the fixed parameters given above, we tested REPR and some competing regression procedures on ten different randomly-chosen partitions of each dataset; each partition consists of 80% training data and 20% testing data. The competing procedures are RuleFit, random forests, LASSO, and classical linear regression. The penalty parameter in LASSO is the same as the value of C chosen for REPR. To implement RuleFit and random forests, we used their publicly available R packages. Table 2 shows the averages of the resulting mean square errors and Table 3 shows their standard deviations. REPR has the smallest average MSE for 5 of the 8 datasets and has the second smallest average MSE on the remaining 3 datasets, coming very close to random forests on MPG. For the standard deviation of the MSE, which we take as a general measure of prediction stability, REPR has the lowest values for 6 of the 8 datasets. The box plots in Figures 3 and 4 visualize these results in more detail for HEAT and MACHINE, respectively. Figure 5 displays the average MSEs in a bar-chart format, with the MSE of REPR normalized to 1.

Figures 6-9 give more detailed information for specific datasets. Figure 6 and 7 respectively show how REPR's prediction MSEs for HEAT and CONCRETE evolve with each iteration, with each data point averaged over the 10 different REPR runs; the horizontal lines indicate the average MSE level for the competing procedures. MSE generally declines as REPR adds rules, although some diminish-

Method **Dataset:**	SERVO	CONCRETE	MACHINE	YACHT	MPG	COOL	HEAT	AIRFOIL
REPR	**0.08228**	**0.00447**	**0.37128**	0.00630	0.01380	0.00218	**0.00220**	**0.00030**
RuleFit	0.10053	0.00736	0.97250	**0.00571**	0.01430	**0.00094**	0.01063	0.00032
Random forests	0.24305	0.01348	0.55456	0.13343	**0.01348**	0.00302	0.00544	0.00083
Lasso	0.67362	0.00573	0.48776	0.74694	0.01981	0.01757	0.01694	0.00150
Linear regression	0.67723	0.00573	0.48776	0.74455	0.02073	0.01708	0.01581	0.00149

Table 2. Average MSE over the experimental datasets. The smallest value in each column is bolded.

Method **Dataset:**	SERVO	CONCRETE	MACHINE	YACHT	MPG	COOL	HEAT	AIRFOIL
REPR	**0.03196**	0.00220	**0.14625**	**0.00186**	**0.00213**	0.00091	**0.00043**	**0.00006**
RuleFit	0.05173	**0.00188**	0.65957	0.00287	0.00430	**0.00007**	0.01453	0.00011
Random forests	0.07554	0.00521	0.31717	0.04922	0.00403	0.00044	0.00055	**0.00006**
Lasso	0.15302	0.00255	0.24303	0.10380	0.00300	0.00270	0.00145	0.00010
Linear regression	0.15814	0.00255	0.24303	0.09806	0.00251	0.00251	0.00192	0.00010

Table 3. Standard deviation of MSE over the experimental datasets. The smallest value in each column is bolded.

ing returns are evident for CONCRETE. Interestingly, neither of these figures shows appreciable evidence of overfitting by REPR, even when large numbers of rules are incorporated into the model. Figures 8 and 9 display testing-set predictions for specific (arbitrarily chosen) partitions of the MACHINE and CONCRETE datasets, respectively, with the observations sorted by response value. REPR seems to outperform the other methods in predicting extreme response values, although it is somewhat worse than the other methods at predicting non-extreme values for MACHINE.

The last two columns of Table 1 show, for a 16-core Xeon E5-2660 workstation, REPR's average total run time per data partition and the average number of search node per invocation of RMA. The longer runs could likely be accelerated by the application of more parallel processors.

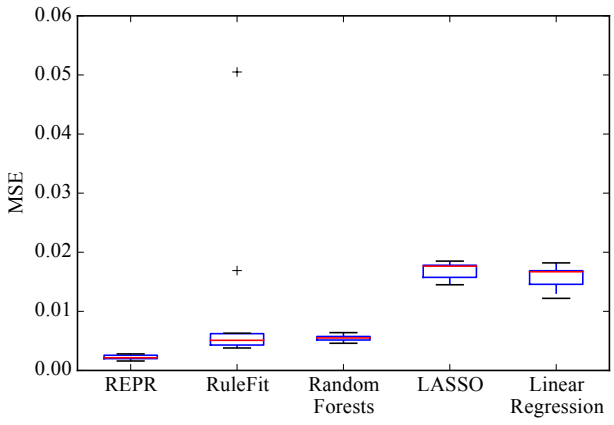

Figure 3. MSE box plots of for the HEAT data set.

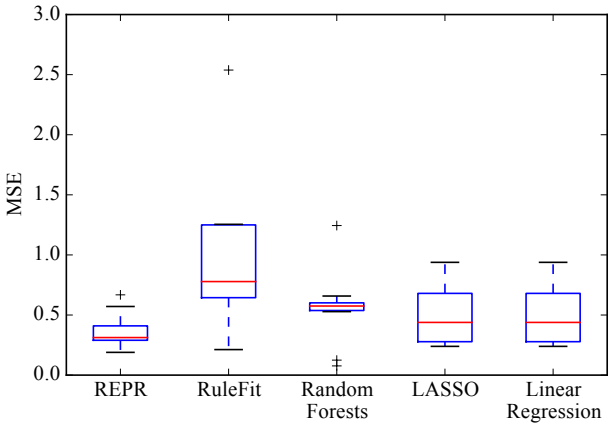

Figure 4. MSE box plots of for the MACHINE data set.

6. Conclusions and Future Research

The results presented here suggest that REPR has significant potential as a regression tool, at least for small datasets. Clearly, it should be tested on more datasets and larger datasets.

Here, we have tested REPR using fixed values of most of its parameters, and we expect we should be able to improve its performance by using intelligent heuristics or cross-validation procedures to select key parameters such as C and E. Improved preprocessing may also prove helpful: judicious normalization of the input data (X, y) should assist in finding good parameter choices, and we are also working on more sophisticated discretization technique for preprocessing the RMA solver input, as well as branch selection heuristics that are more efficient for large ℓ_j.

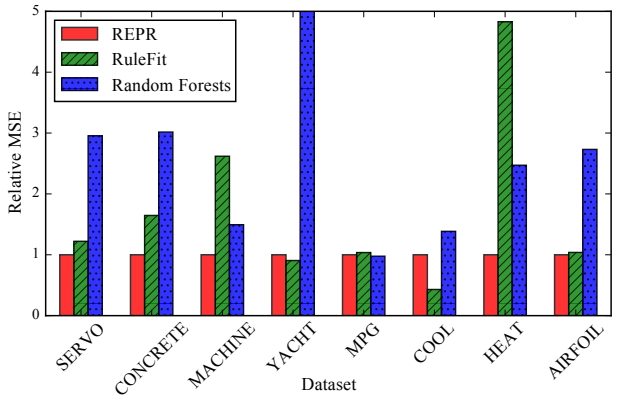

Figure 5. Comparison the average MSEs, with the MSE of REPR normalized to 1. The random forest value for YACHT is truncated.

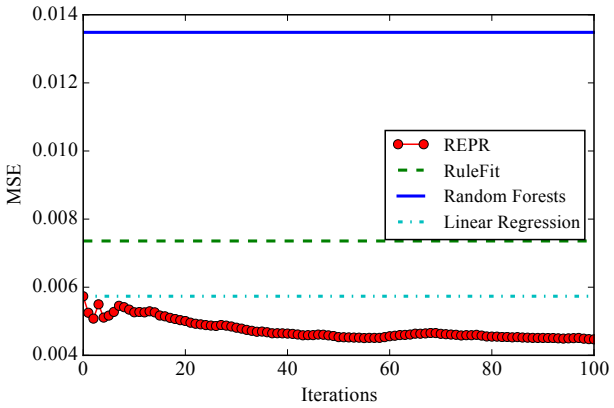

Figure 7. MSE as a function of iterations for the CONCRETE dataset.

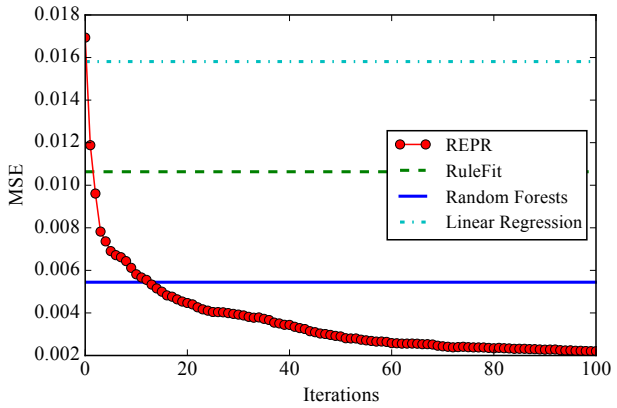

Figure 6. MSE as a function of iterations for the HEAT dataset.

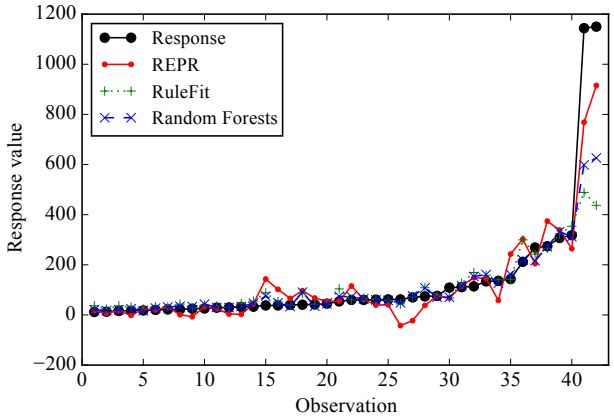

Figure 8. Sorted predictions for the MACHINE data set.

It would be interesting to see how well REPR performs if the pricing problems are solved less exactly. For example, one could use various techniques for truncating the branch-and-bound search, such as setting a limit on the number of subproblems explored or loosening the conditions for pruning unpromising subtrees. Or one could use, perhaps selectively, some entirely heuristic procedure to identify rules to add to the restricted master problem.

For problems with large numbers of observations m, it is conceivable that solving the restricted master problems could become a serial bottleneck in our current implementation strategy. If this phenomenon is observed in practice, it could be worth investigating parallel solution strategies for the restricted master.

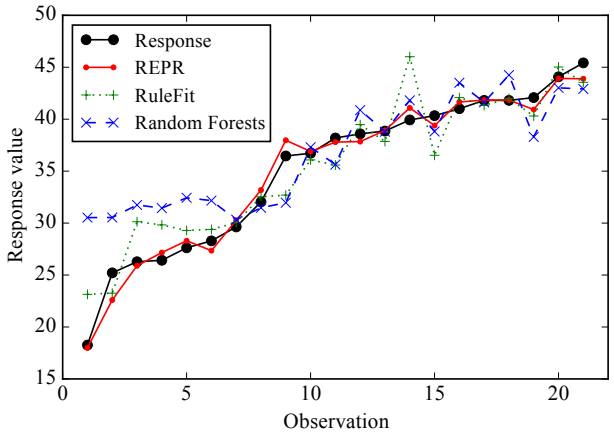

Figure 9. Sorted predictions for the CONCRETE data set.

References

Aho, Timo, Ženko, Bernard, Džeroski, Sašo, and Elomaa, Tapio. Multi-target regression with rule ensembles. *J. Mach. Learn. Res.*, 13(Aug):2367–2407, 2012.

Breiman, Leo. Random forests. *Mach. Learn.*, 45:5–32, 2001.

Cohen, William W. and Singer, Yoram. A simple, fast, and effective rule learner. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence*, pp. 335–342, 1999.

Dembczyński, Krzysztof, Kotłowski, Wojciech, and Słowiński, Roman. Solving regression by learning an ensemble of decision rules. In *International Conference on Artificial Intelligence and Soft Computing, 2008*, volume 5097 of *Lecture Notes in Artificial Intelligence*, pp. 533–544. Springer-Verlag, 2008a.

Dembczyński, Krzysztof, Kotłowski, Wojciech, and Słowiński, Roman. Maximum likelihood rule ensembles. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 224–231, New York, NY, USA, 2008b. ACM.

Demiriz, Ayhan, Bennett, Kristin P., and Shawe-Taylor, John. Linear programming boosting via column generation. *Mach. Learn.*, 46(1-3):225–254, 2002.

Eckstein, Jonathan and Goldberg, Noam. An improved branch-and-bound method for maximum monomial agreement. *INFORMS J. Comput.*, 24(2):328–341, 2012.

Eckstein, Jonathan, Hart, William E., and Phillips, Cynthia A. PEBBL: an object-oriented framework for scalable parallel branch and bound. *Math. Program. Comput.*, 7(4):429–469, 2015.

Ford, Jr., Lester R. and Fulkerson, David R. A suggested computation for maximal multi-commodity network flows. *Manage. Sci.*, 5:97–101, 1958.

Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Ann. of Stat.*, pp. 1189–1232, 2001.

Friedman, Jerome H. and Popescu, Bogdan E. Predictive learning via rule ensembles. *Ann. Appl. Stat.*, 2(3):916–954, 2008.

Gilmore, Paul C. and Gomory, Ralph E. A linear programming approach to the cutting-stock problem. *Oper. Res.*, 9:849–859, 1961.

Griva, Igor, Nash, Stephen G., and Sofer, Ariela. *Linear and Nonlinear Optimization*. SIAM, second edition, 2009.

Gropp, William, Lusk, Ewing, and Skjellum, Anthony. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1994.

Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2016. URL http://www.gurobi.com/documentation/7.0/refman/index.html.

Lichman, Moshe. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Morrison, David R., Jacobson, Sheldon H., Sauppe, Jason J., and Sewell, Edward C. Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning. *Discrete Optim.*, 19(C):79–102, 2016.

Tibshirani, Robert. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58(1):267–288, 1996.

Weiss, Sholom M. and Indurkhya, Nitin. Optimized rule induction. *IEEE Expert*, 8(6):61–69, 1993.