OPERATIONS RESEARCH
Vol. 55, No. 2, March–April 2007, pp. 252–271
ISSN 0030-364X | EISSN 1526-5463 | 07 | 5502 | 0252

DOI 10.1287/opre.1060.0360

Classification and Regression via Integer Optimization

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E53-363, Cambridge, Massachusetts 02139, dbertsim@mit.edu

Romy Shioda
Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, rshioda@uwaterloo.ca

Motivated by the significant advances in integer optimization in the past decade, we introduce mixed-integer optimization methods to the classical statistical problems of classification and regression and construct a software package called CRIO (classification and regression via integer optimization). CRIO separates data points into different polyhedral regions. In classification each region is assigned a class, while in regression each region has its own distinct regression coefficients. Computational experimentations with generated and real data sets show that CRIO is comparable to and often outperforms the current leading methods in classification and regression. We hope that these results illustrate the potential for significant impact of integer optimization methods on computational statistics and data mining.

Subject classifications: programming: integer, applications; statistics: nonparametric.
Area of review: Financial Services.
History: Received November 2002; revisions received April 2005, April 2006; accepted April 2006.

1. Introduction

In the last 20 years, the availability of massive amounts of data in electronic form and the spectacular advances in computational power have led to the development of the field of data mining (or knowledge discovery) that is at the center of modern scientific computation. Two central problems in this development (as well as in classical statistics) are *data classification* and *regression*. To the best of our knowledge, popular methods for these problems include:

(1) Decision trees for classification and regression (Breiman et al. 1984). Decision trees recursively split the data along a variable into hyper-rectangular regions. After this forward propagation is complete, backward propagation (or pruning) is performed to prevent over-fitting the model to the data set. Its main shortcoming is its fundamentally greedy approach. Classification and regression trees (CART) (Breiman et al. 1984) is the leading work for this model. Bennett and Blue (1984) find a globally optimal decision tree, but the structure of the tree needs to be prefixed. C5.0 (Quinlan 1993) is another popular method that is similar in spirit to CART. CRUISE (Kim and Loh 2000) is a more recent version of CART that allows for multivariate partitions (thus, heuristically partitioning the input regions into polyhedral regions) and minimizes bias in the variable selection step.

(2) Multivariate adaptive regression splines (MARS) (Friedman 1991) for regression. Like CART, MARS also partitions the data into regions but fits continuous splines or basis functions to each region, thus maintaining continuity of the predicted values among neighboring regions.

(3) Support vector machines (SVM) (Vapnik 1999, Rifkin 2002, Mangasarian 1965) for classification. In its simplest form, SVM separates points of different classes by a single hyperplane. More sophisticated SVM methods find a separating hyperplane in a higher dimensional space, leading to a nonlinear partitioning. In all cases, the optimal separating hyperplane is modeled as a convex quadratic programming problem.

Decision trees and MARS are heuristics in nature and are closer to the tradition of statistical inference methods. SVM belongs to the category of separating hyperplane methods, utilize formal continuous optimization techniques (quadratic optimization), and are closer to the tradition of machine learning and mathematical programming. It is fair to say that all these methods (to various degrees) are at the forefront of data mining and have had significant impact in practical applications. Commercial software is available for all these methods, facilitating their wide applicability.

While continuous optimization methods have been widely used in statistics and have had a significant impact in the last 30 years (a classical reference is Arthanari and Dodge 1993), integer optimization has had very limited impact in statistical computation. While the statistics community has long recognized that many statistical problems, including classification and regression, can be formulated as integer optimization problems (Arthanari and Dodge 1993 contains several integer optimization formulations), the belief was formed in the early 1970s that these methods are not tractable in practical computational settings. Due to the success of the above methods and the belief

of integer optimization's impracticality, the applicability of integer optimization methods to the problems of classification and regression has not been seriously investigated.

Our objective in this paper is to do exactly this—to develop a methodology for classification and regression that utilizes state-of-the-art integer optimization methods to exploit the discrete character of these problems. We were motivated by the significant advances in integer optimization in the past decade that make it possible to solve large-scale problems within practical times. We have created a software package based on integer optimization that we call *classification and regression via integer optimization* (CRIO), which we compare to the state-of-the-art methods outlined earlier. While CRIO's distinguishing feature is mixed-integer optimization, we have incorporated certain elements of earlier methods that have been successful in our attempt to make CRIO applicable in diverse real-world settings.

We view our contribution as twofold:

(1) To bring to the attention of the statistics and data mining community that integer optimization can have a significant impact on statistical computation, and thus motivate researchers to revisit old statistical problems using integer optimization.

(2) To show that CRIO is a promising method for classification and regression that matches and often outperforms other state-of-the-art methods outlined earlier.

The structure of this paper is as follows. In §2, we give an intuitive presentation of the geometric ideas of our approach for both classification and regression to facilitate understanding, and provide motivation to the globally optimal nature of CRIO. In §§3 and 4, we develop CRIO for classification and for regression, respectively. In §5, we compare CRIO with logistic regression, least-square regression, neural networks, decision trees, SVMs, and MARS on generate and real data sets. We present our conclusions in the final section.

2. An Intuitive Overview of CRIO

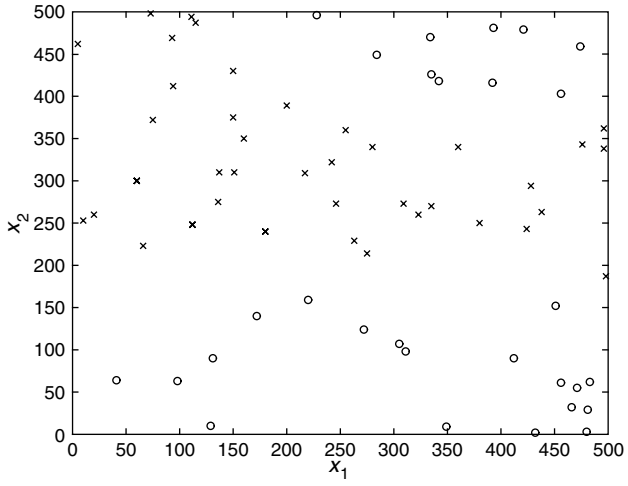
In this section, we present the geometric ideas of our approach intuitively (first with respect to classification and then with respect to regression) to facilitate understanding of the mathematical presentation in §§3 and 4.

2.1. The Geometry of the Classification Approach

Given n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$, and $i = 1, \ldots, n$, we want to partition $\mathbb{R}^d$ into a small number of regions that only contain points of the same class. Consider, for example, the points in Figure 1. Figure 2 illustrates the output of CRIO.

In the first step, we use a mixed-integer optimization model to assign Class 1 points into K groups[1] (K is a user-defined parameter), such that no Class 0 point belongs in the convex hull of a Class 1 group. We group Class 1 points instead of Class 0 points, without loss of generality,

Figure 1. A given set of training data. Class 0 points are represented with an "o" and Class 1 points are represented by an "x."



throughout the paper. Clearly, we want to keep K small (e.g., less than five) to avoid over-fitting. Given the presence of outliers, it might be infeasible to use K groups (see Figure 3). For this purpose, we further enhance the mixed-integer optimization model by enabling it to eliminate a pre-specified number of outliers. Having defined K groups at the end of this phase, we use quadratic optimization methods to represent groups by polyhedral regions—an approach inspired by SVMs. Finally, we eliminate redundant faces of these polyhedra by iteratively solving linear optimization problems. After the final sets of polyhedra are defined, we classify a new point $\mathbf{x}_0$ as Class 1, if it is contained in any of the K polyhedra, and as Class 0, otherwise. To reduce the dimension of the mixed-integer optimization model and thus reduce computation time, we preprocess the data so that points form small clusters using a clustering algorithm (see Figure 4).

Figure 2. The output of CRIO.

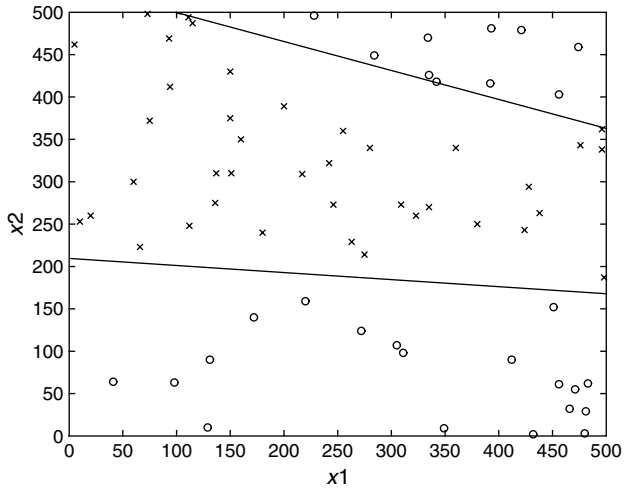


Figure 3. Illustration of outliers in the classification data. Points A and B are Class 0 outliers.

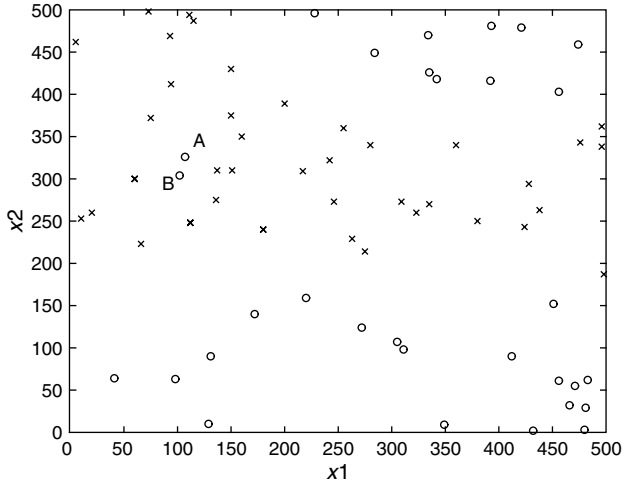
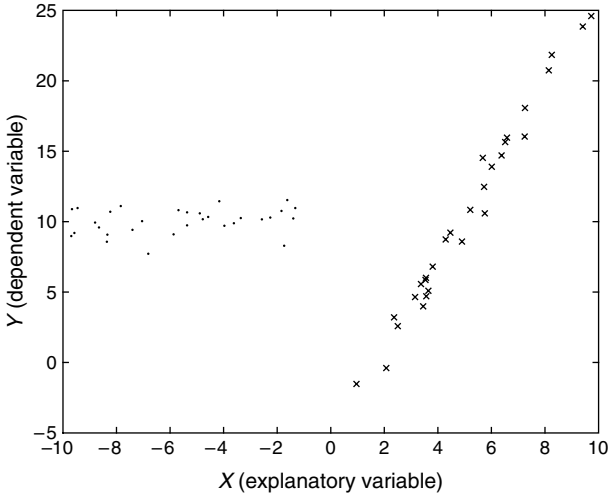


Figure 5. A given set of training data for regression with $d = 1$.



2.2. The Geometry of the Regression Approach

In a classical regression setting, we are given n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and $i = 1, \ldots, n$. We wish to find a linear relationship between $\mathbf{x}_i$ and y_i, i.e., $y_i \approx \boldsymbol{\beta}'\mathbf{x}_i$ for all i, where the coefficients $\boldsymbol{\beta} \in \mathbb{R}^d$ are found by minimizing $\sum_{i=1}^n (y_i - \boldsymbol{\beta}'\mathbf{x}_i)^2$ or $\sum_{i=1}^n |y_i - \boldsymbol{\beta}'\mathbf{x}_i|$. CRIO seeks to find K disjoint regions $P_k \subset \mathbb{R}^d$ and corresponding coefficients $\boldsymbol{\beta}_k \in \mathbb{R}^d$, $k = 1, \ldots, K$, such that if $\mathbf{x}_0 \in P_k$, our prediction for y_0 will be $\hat{y}_0 = \boldsymbol{\beta}_k'\mathbf{x}_0$. Given the points in Figure 5 where $\mathbf{x}_i \in \mathbb{R}$, Figure 6 illustrates the output of CRIO.

CRIO first solves a mixed-integer optimization model to assign the n points into K groups (where K is a user-defined parameter). In addition, the mixed-integer optimization is further enhanced to detect and eliminate outlier points in the data set (see Figure 7). In contrast, traditional regression models deal with outliers *after* the slopes have been determined, by examining which points contribute the most to the total prediction error (see Rousseeuw and Leroy 1987 and Ryan 1997). This procedure can often be deceiving because the model is heavily influenced by the outlier points. After the points are assigned to the K groups, we determine the coefficients $\boldsymbol{\beta}_k$ that best fit the data for group k, for $k = 1, \ldots, K$, and define polyhedra P_k to represent each group using linear optimization methods. After the coefficients and polyhedra are defined, we predict the y_0 value of a new point $\mathbf{x}_0$ as we outlined earlier. CRIO does not, in fact, create a partition of $\mathbb{R}^d$, so it is possible that a new point $\mathbf{x}_0$ might not belong to any of the P_ks.[2] In such a case, we assign it to the region P_r that contains the majority among its F (a user-defined number) nearest neighbors in the training set, and make the prediction $\hat{y}_0 = \boldsymbol{\beta}_r'\mathbf{x}_0$. Similarly to the classification model, we preprocess the data

Figure 4. Data points are grouped into small clusters in $\mathbf{x}$-space.

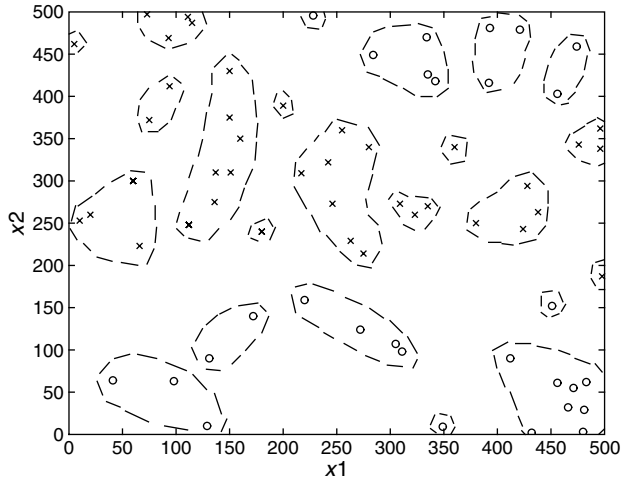


Figure 6. The output of CRIO, where the regression coefficient or slope is different for the two regions.

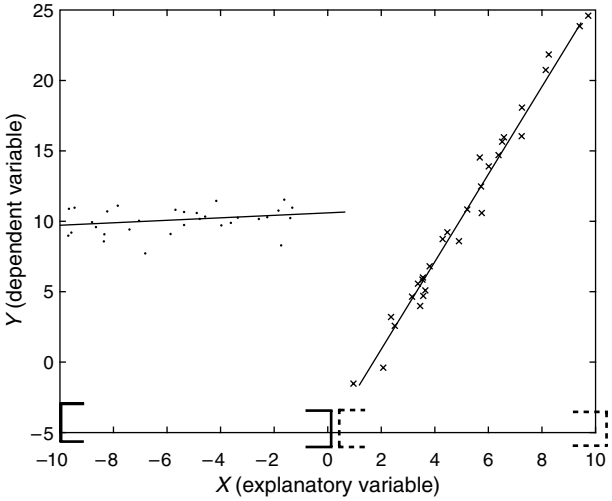
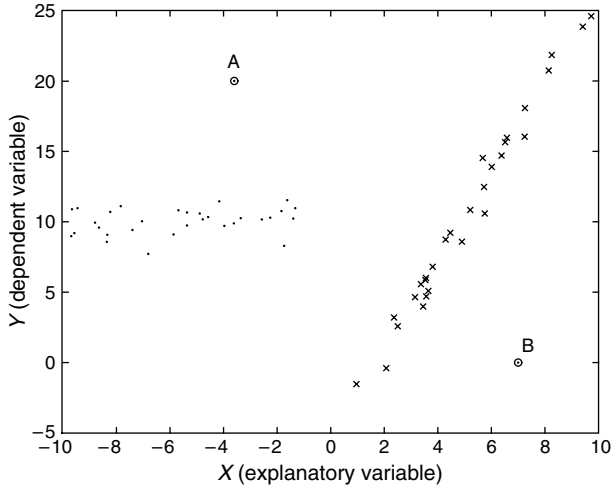


Figure 7. Illustration of outliers in regression data.



by clustering them into small clusters to reduce the dimension and thus the computation time of the mixed-integer optimization model (see Figure 8).

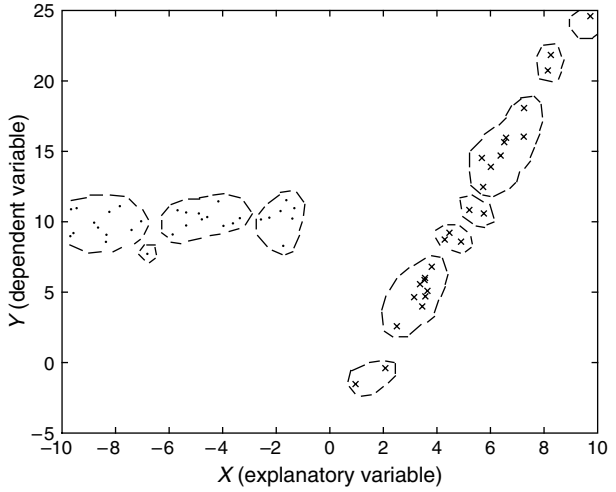
In summary, CRIO has a common approach to both problems of classification and regression: (a) preprocess data by assigning points to clusters to reduce the dimensionality of the mixed-integer optimization problems solved next; (b) solve a mixed integer-optimization problem that assigns clusters to groups and removes outliers. In the case of regression, the model also selects the regression coefficients for each group; and (c) solve continuous optimization problems (quadratic optimization problems for classification and linear optimization problems for regression) that assign groups to polyhedral regions.

2.3. Motivation for CRIO

The initial motivation for both the classification and regression component of CRIO was to develop a globally optimal version of CART (Breiman et al. 1984). We soon discovered that a method that partitions the input space with hyperplanes can be modeled far more tractably than a model using univariate division rules, which might be surprising, given that the former model might appear to be a generalization of the latter. Thus, our approach can be considered to be a *globally optimal classification and regression tree* that, unlike the original version of CART, partitions the input space with hyperplanes. Our method is also distinct from SVMs with linear kernels because we might use multiple hyperplanes to partition the input space, whereas SVMs use a single hyperplane.

We provide Figure 9 as a simple illustration of the behavior of classification trees, SVMs, and CRIO, on the famous checkerboard data set. A point shown as "x" is one class and "o" is another. Although the data set might appear to be ideal for univariate classification trees, we see that the method generates many more divisions than necessary. This is because the model made mistakes in the initial partitions, thus requiring several small subsequent partitions to correct the initial ones. We see similar behaviors with multivariate classification trees, where many more partitions are made than necessary to correct initial errors. These examples illustrate the significant advantage of a globally optimal solution. The classification component of CRIO makes very similar assumptions as multivariate classification trees; however, the global optimal solution of CRIO results in a more robust solution. The globally optimal nature of SVM's solutions, as well as its flexibility, also results in a good partitioning of this data set, although the model assumptions of CRIO are better suited for this particular example.

Figure 8. Data points clustered in $(\mathbf{x}, y)$-space.



3. CRIO for Classification

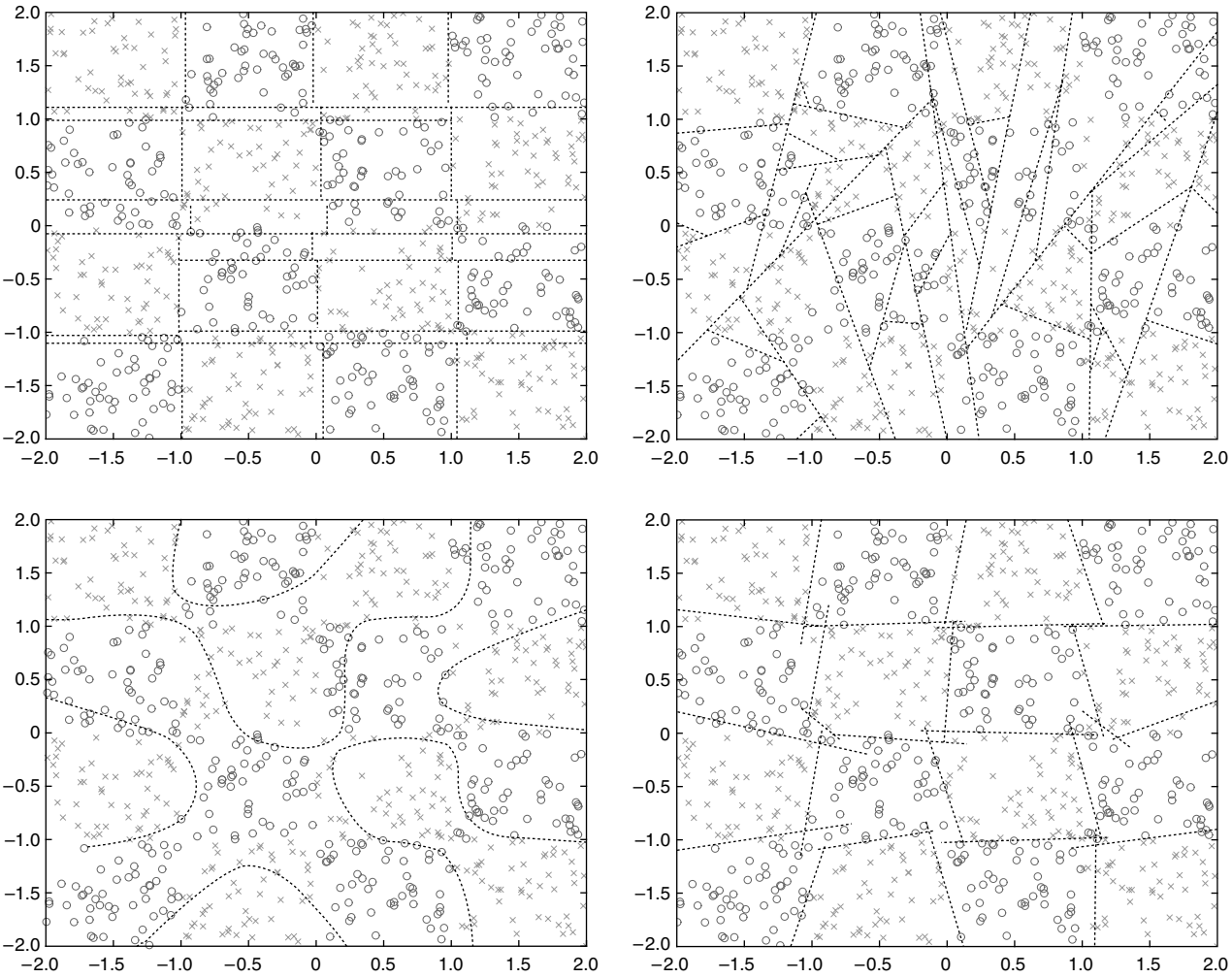
In this section, we present our methods for two-class classification problems. As outlined in §2.1, we first assign both Class 1 and Class 0 points into clusters (§3.2), then assign Class 1 clusters to groups via the use of a mixed-integer optimization model that also detects outliers (§3.3). Section 3.4 presents the methodology of assigning polyhedra to groups, and §3.5 summarizes the overall algorithm for binary classification. We start by presenting the basic mixed-integer optimization model in §3.1, which forms the basis of our final approach.

3.1. The Mixed-Integer Optimization Model

The training data consist of n observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. Let m_0 and m_1 be the number of Class 0 and Class 1 points, respectively. We denote Class 0 and Class 1 points by $\mathbf{x}_i^0$, $i = 1, \ldots, m_0$, and $\mathbf{x}_i^1$, $i = 1, \ldots, m_1$, respectively. Let $M_0 = \{1, \ldots, m_0\}$, $M_1 = \{1, \ldots, m_1\}$, and $\bar{K} = \{1, \ldots, K\}$.

We want to partition Class 1 points into K disjoint groups, so that no Class 0 point can be expressed as a convex combination of these points. Let G_k be the set indices

Figure 9. Input space partition of the checkerboard data set using univariate classification trees (top left), multivariate classification trees (top right), SVM with Gaussian kernels (bottom left), and CRIO (bottom right).



of Class 1 points that are in group k, where $\bigcup_{k\in\bar{K}} G_k = M_1$ and $G_k \cap G_{k'} = \varnothing$, $k, k' \in \bar{K}, k \neq k'$. Thus, we require that the following system is infeasible for all $i \in M_0$ and $k \in \bar{K}$:

$$\begin{aligned} &\sum_{j\in G_k} \lambda_j \mathbf{x}_j^1 = \mathbf{x}_i^0, \\ &\sum_{j\in G_k} \lambda_j = 1, \\ &\lambda_j \geqslant 0, \quad j \in G_k. \end{aligned} \tag{1}$$

From Farkas' lemma, system (1) is infeasible if and only if the following problem is feasible:

$$\begin{aligned} &\mathbf{p}'\mathbf{x}_i^0 + q < 0, \\ &\mathbf{p}'\mathbf{x}_j^1 + q \geqslant 0, \quad j \in G_k. \end{aligned} \tag{2}$$

We consider the optimization problem:

$$\begin{aligned} z_{k,i} = \text{maximize} \ & \epsilon \\ \text{subject to} \ & \mathbf{p}'\mathbf{x}_i^0 + q \leqslant -\epsilon, \\ & \mathbf{p}'\mathbf{x}_j^1 + q \geqslant 0, \quad j \in G_k, \\ & 0 \leqslant \epsilon \leqslant 1. \end{aligned} \tag{3}$$

If $z_{k,i} > 0$, system (2) is feasible and thus problem (1) is infeasible. If $z_{k,i} = 0$, system (2) is infeasible and thus problem (1) is feasible, i.e., $\mathbf{x}_i^0$ is in the convex hull of the points $\mathbf{x}_j^1$, $j \in G_k$. We add the constraint $\epsilon \leqslant 1$ to prevent unbounded solutions. Note that problem (3) seeks to find a hyperplane $\mathbf{p}'\mathbf{x} + q = 0$ that separates point $\mathbf{x}_i^0$ from all the Class 1 points in group k.

We want to expand problem (3) for all $k \in \bar{K}$ and $i \in M_0$. If we knew which group each Class 1 point belonged to, then we would consider

$$\begin{aligned} z = \text{maximize} \ & \delta \\ \text{subject to} \ & \mathbf{p}'_{k,i}\mathbf{x}_i^0 + q_{k,i} \leqslant -\delta, \quad i \in M_0;\ k \in \bar{K}, \\ & \mathbf{p}'_{k,i}\mathbf{x}_j^1 + q_{k,i} \geqslant 0, \quad i \in M_0;\ k \in \bar{K};\ j \in G_k, \\ & \delta \leqslant 1. \end{aligned} \tag{4}$$

To determine if we can assign Class 1 points into K groups such that $z > 0$, we define decision variables for

$k \in \bar{K}$ and $j \in M_1$:

$$a_{k,j} = \begin{cases} 1 & \text{if } \mathbf{x}_j^1 \text{ is assigned to group } k \text{ (i.e., } j \in G_k), \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

We include the constraints $\mathbf{p}'_{k,i}\mathbf{x}_j^1 + q_{k,i} \geqslant 0$ in problem (4) if and only if $a_{k,j} = 1$, i.e.,

$$\mathbf{p}'_{k,i}\mathbf{x}_j^1 + q_{k,i} \geqslant M(a_{k,j} - 1),$$

where M is a large positive constant. Note, however, that we can re-scale the variables $\mathbf{p}_{k,i}$ and $q_{k,i}$ by a positive number, and thus we can take $M = 1$, i.e.,

$$\mathbf{p}'_{k,i}\mathbf{x}_j^1 + q_{k,i} \geqslant a_{k,j} - 1.$$

Thus, we can check whether we can partition Class 1 points into K disjoint groups such that no Class 0 points are included in their convex hull, by solving the following mixed-integer optimization problem:

$$\begin{aligned} z^* = \text{maximize } & \delta \\ \text{subject to } & \mathbf{p}'_{k,i}\mathbf{x}_i^0 + q_{k,i} \leqslant -\delta, \quad i \in M_0;\ k \in \bar{K}, \\ & \mathbf{p}'_{k,i}\mathbf{x}_j^1 + q_{k,i} \geqslant a_{k,j} - 1, \\ & \qquad i \in M_0;\ k \in \bar{K};\ j \in M_1, \quad (6) \\ & \textstyle\sum_{k=1}^{K} a_{k,j} = 1, \quad j \in M_1, \\ & \delta \leqslant 1, \\ & a_{k,j} \in \{0, 1\}. \end{aligned}$$

If $z^* > 0$, the partition into K groups is feasible; while if $z^* = 0$, it is not, requiring us to increase the value of K.

3.2. The Clustering Algorithm

Problem (6) has $Km_0(d+1)+1$ continuous variables, Km_1 binary variables, and $Km_0 + Km_0m_1 + m_1$ rows. For large values of m_0 and m_1, problem (6) becomes expensive to solve. Alternatively, we can drastically decrease the dimension of problem (6) by solving a hyperplane for clusters of points at a time instead of point by point.

We develop a hierarchical clustering-based algorithm that preprocesses the data to create clusters of Class 0 and Class 1 points. Collections of Class 0 (Class 1) points are considered a *cluster* if there are no Class 1 (Class 0) points in their convex hull. If we preprocess the data to find K_0 Class 0 clusters and K_1 Class 1 clusters, we can modify problem (6) (see formulation (9) below) to have $KK_0(d+1)+1$ continuous variables, KK_1 binary variables, and $Km_0 + KK_0m_1 + K_1$ rows.

The clustering algorithm applies the hierarchical clustering methodology (see Johnson and Wichern 1998) where points or clusters with the shortest distances are merged into a cluster until the desired number of clusters is achieved. For our purposes, we need to check whether a merger of Class 0 (Class 1) clusters will not contain any Class 1 (Class 0) points in the resulting convex hull. We solve the following linear optimization problem to check whether Class 1 clusters r and s can be merged:

$$\begin{aligned} \delta^* = \text{maximize } & \delta \\ \text{subject to } & \mathbf{p}'_i\mathbf{x}_i^0 + q_i \leqslant -\delta, \quad i \in M_0, \quad (7) \\ & \mathbf{p}'_i\mathbf{x}_j^1 + q_i \geqslant \delta, \quad j \in C_r \cup C_s, \end{aligned}$$

where C_r and C_s are the set of indices of Class 1 points in clusters r and s, respectively.

If $\delta^* > 0$, then clusters r and s can merge; while if $\delta^* = 0$, they cannot because there is at least one Class 0 point in the convex hull of the combined cluster. The overall preprocessing algorithm that identifies clusters of Class 1 points is as follows:

1. **Initialize:** $K := m_1$, $k := 0$.
2. **while** $k < K$ **do**
3. Find the clusters with minimum pairwise distance—call these r and s.
4. Solve problem (7) on clusters r and s.
5. **if** $\delta^* = 0$ **then**
6. $k := k + 1$
7. **else**
8. Merge clusters r and s.
9. $K := K - 1$, $k := 0$.
10. **end if**
11. $k := k + 1$.
12. **end while**

In the start of the algorithm, each point is considered a cluster, thus $K = m_1$. On line 3, the minimum pairwise distances are calculated by comparing the statistical distances[3] between the centers of all the clusters. We define the center of a cluster as the arithmetic mean of all the points that belong to that cluster. In the merging step on line 4, these centers are updated. Finding clusters for Class 0 follows similarly.

After we have K_0 and K_1 clusters of Class 0 and Class 1 points, respectively, we run a modified version of problem (6) to assign the K_1 Class 1 clusters to K groups, where $K < K_1 \ll m_1$. Let $\bar{K}_0 = \{1, \ldots, K_0\}$ and $\bar{K}_1 = \{1, \ldots, K_1\}$. Let C_t^0, $t \in \bar{K}_0$, be the set of indices of Class 0 points in cluster t and C_r^1, $r \in \bar{K}_1$, be the set of indices of Class 1 points in Cluster r. We define the following binary variables for $r \in \bar{K}_1$ and $k \in \bar{K}$:

$$a_{k,r} = \begin{cases} 1 & \text{if cluster } r \text{ is assigned to group } k, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Analogously to problem (6), we formulate the following mixed-integer optimization problem for clusters:

$$\begin{aligned} & \text{maximize } \delta \\ & \text{subject to } \mathbf{p}'_{k,t}\mathbf{x}_i^0 + q_{k,t} \leqslant -\delta, \quad i \in C_t^0;\ t \in \bar{K}_0;\ k \in \bar{K}, \end{aligned}$$

$$\begin{aligned}
&\mathbf{p}'_{k,t}\mathbf{x}^1_j + q_{k,t} \geqslant a_{k,r} - 1,\\
&\qquad\qquad t \in \bar{K}_0;\ r \in \bar{K}_1;\ k \in \bar{K};\ j \in C^1_r,\\
&\sum_{k=1}^{K} a_{k,r} = 1, \quad r \in \bar{K}_1,\\
&a_{k,r} \in \{0,1\}.
\end{aligned} \tag{9}$$

If $a_{k,r} = 1$ in an optimal solution, then all Class 1 points in cluster r are assigned to group k, i.e., $G_k = \bigcup_{\{r \mid a_{k,r}=1\}} C^1_r$.

3.3. Elimination of Outliers

In the presence of outliers, it is possible that we may need a large number of groups—possibly leading to over-fitting. A point can be considered an outlier if it lies significantly far from any other point of its class (see Figure 3 for an illustration). In this section, we outline two methods that remove outliers: (a) based on the clustering algorithm of the previous section, and (b) via an extension of problem (9).

Outlier Removal via Clustering. The clustering method of §3.2 applied on Class 0 points would keep outlier points in its own cluster without ever merging them with any other Class 0 cluster. Thus, after K_0 clusters are found, we can check the cardinality of each of the clusters and eliminate those with very small cardinality—perhaps less than 1% of m_0. Such a procedure can similarly be done on Class 1 points.

Outlier Removal via Optimization. Figure 3 illustrates how outlier points can prevent CRIO from grouping Class 1 points with small K, i.e., problem (9) can return only a trivial solution where $\delta = 0$, $\mathbf{p}_{k,i} = \mathbf{0}$, $q_{k,i} = 0$, and the $a_{k,j}$s are assigned arbitrarily. We want to modify problem (9) to eliminate or ignore outlier points that prevent us from grouping Class 1 points into K groups.

One possible approach is to assign a binary decision variable to each point, so that it is removed if it is equal to one and not removed otherwise. Such a modification can be theoretically incorporated into formulation (9), but the large increase in binary variables can make the problem difficult to solve.

We propose a different approach that modifies problem (9) to always return a partition of Class 1 points so that the total margin of the violation is minimized. Problem (10) is such a model, where ϵ^0_i and ϵ^1_j are violation margins corresponding to Class 0 and Class 1 points, respectively. The model is as follows:

$$\begin{aligned}
\text{minimize}\quad & \sum_{i=1}^{m_0} \epsilon^0_i + \sum_{j=1}^{m_1} \epsilon^1_j\\
\text{subject to}\quad & \mathbf{p}'_{k,t}\mathbf{x}^0_i + q_{k,t} \leqslant -1 + \epsilon^0_i, \quad i \in C^0_t;\ t \in \bar{K}_0;\ k \in \bar{K},\\
& \mathbf{p}'_{k,t}\mathbf{x}^1_j + q_{k,t} \geqslant -M + (M+1)a_{k,r} - \epsilon^1_j,\\
& \qquad\qquad t \in \bar{K}_0;\ k \in \bar{K};\ r \in \bar{K}_1;\ j \in C^1_r,\\
& \sum_{k=1}^{K} a_{k,r} = 1, \quad r \in \bar{K}_1,\\
& a_{k,r} \in \{0,1\}, \quad \epsilon^0_i \geqslant 0, \quad \epsilon^1_j \geqslant 0,
\end{aligned} \tag{10}$$

where M is a large positive constant.

As in problem (9), the first constraint of problem (10) requires $\mathbf{p}'_{k,t}\mathbf{x}^0_i + q_{k,t}$ to be strictly negative for all Class 0 points. However, if a point $\mathbf{x}^0_i$ cannot satisfy the constraint, problem (10) allows the constraint to be violated, i.e., $\mathbf{p}'_{k,t}\mathbf{x}^0_i + q_{k,t}$ can be positive if $\epsilon^0_i > 1$. Similarly, problems (9) and (10) require $\mathbf{p}'_{k,t}\mathbf{x}^1_j + q_{k,t}$ to be nonnegative when $a_{k,r} = 1$ and arbitrary when $a_{k,r} = 0$. However, (10) allows $\mathbf{p}'_{k,t}\mathbf{x}^1_j + q_{k,t}$ to be negative even when $a_{k,r} = 1$ because the second constraint becomes $\mathbf{p}'_{k,t}\mathbf{x}^1_j + q_{k,t} \geqslant 1 - \epsilon^1_j$ when $a_{k,r} = 1$, and the left-hand side can take on negative values if $\epsilon^1_j > 1$. Thus, by allowing ϵ^0_i and ϵ^1_j to be greater than one when necessary, problem (10) will always return K Class 1 groups by ignoring those points that initially prevented the groupings. These points with $\epsilon^0_i > 1$ and $\epsilon^1_j > 1$ can be considered outliers and be eliminated.

3.4. Assigning Groups to Polyhedral Regions

The solution of problem (10) results in K disjoint groups of Class 1 points, such that no Class 0 point is in the convex hull of any of these groups. Our objective in this section is to represent each group k geometrically with a polyhedron P_k. An initially apparent choice for P_k is to use the solution of problem (10), i.e.,

$$P_k = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{p}'_{k,t}\mathbf{x} \geqslant -q_{k,t},\ k \in \bar{K};\ t \in \bar{K}_0\}.$$

Motivated by the success of SVMs (see Vapnik 1999), we present an approach of using hyperplanes that separate the points of each class such that the minimum Euclidean distance from any point to the hyperplane is maximized. This prevents over-fitting the model to the training data set because it leaves as much distance between the boundary and points of each class as possible.

Our goal is to find a hyperplane

$$\boldsymbol{\pi}'_{k,t}\mathbf{x} = \alpha_{k,t}$$

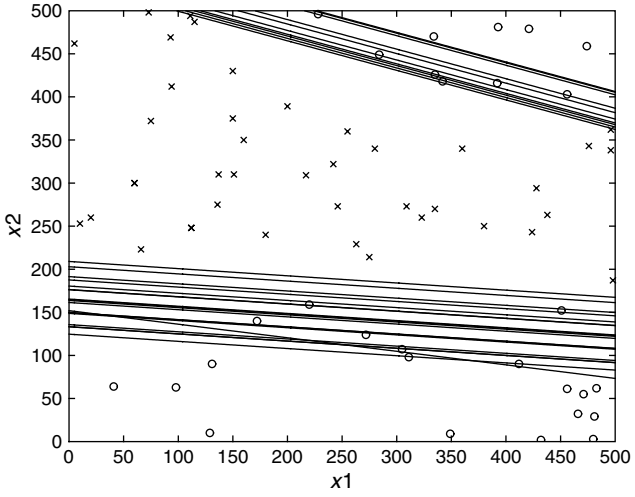
for every group k, $k \in \bar{K}$, of Class 1 points and for every cluster t, $t \in \bar{K}_0$, of Class 0 points such that all points in cluster t are separated from every point in group k so that the minimum distance between every point and the hyperplane is maximized. The distance, $d(\mathbf{x}, \boldsymbol{\pi}_{k,t}, \alpha_{k,t})$, between a point $\mathbf{x}$ and the hyperplane $\boldsymbol{\pi}'_{k,t}\mathbf{x} = \alpha_{k,t}$, is

$$d(\mathbf{x}, \boldsymbol{\pi}_{k,t}, \alpha_{k,t}) = \frac{|\gamma|}{\|\boldsymbol{\pi}_{k,t}\|}, \quad \text{where } \gamma = \boldsymbol{\pi}'_{k,t}\mathbf{x} - \alpha_{k,t}.$$

Thus, we can maximize $d(\mathbf{x}, \boldsymbol{\pi}_{k,t}, \alpha_{k,t})$ by fixing $|\gamma|$ and minimize $\|\boldsymbol{\pi}_{k,t}\|^2 = \boldsymbol{\pi}'_{k,t}\boldsymbol{\pi}_{k,t}$, thus solving the quadratic optimization problem:

$$\begin{aligned}
\text{minimize}\quad & \boldsymbol{\pi}'_{k,t}\boldsymbol{\pi}_{k,t}\\
\text{subject to}\quad & \boldsymbol{\pi}'_{k,t}\mathbf{x}^0_i \geqslant \alpha_{k,t} + 1, \quad i \in C^0_t,\\
& \boldsymbol{\pi}'_{k,t}\mathbf{x}^1_j \leqslant \alpha_{k,t} - 1, \quad j \in G_k.
\end{aligned} \tag{11}$$

Figure 10. Before eliminating redundant constraints.



We solve problem (11) for each $t \in \bar{K}_0$ and $k \in \bar{K}$, and find KK_0 hyperplanes. Thus, for each group k, the corresponding polyhedral region is

$$P_k = \{\mathbf{x} \in \mathbb{R}^d \mid \boldsymbol{\pi}'_{k,t}\mathbf{x} \leqslant \alpha_{k,t},\ t \in \bar{K}_0\}. \tag{12}$$

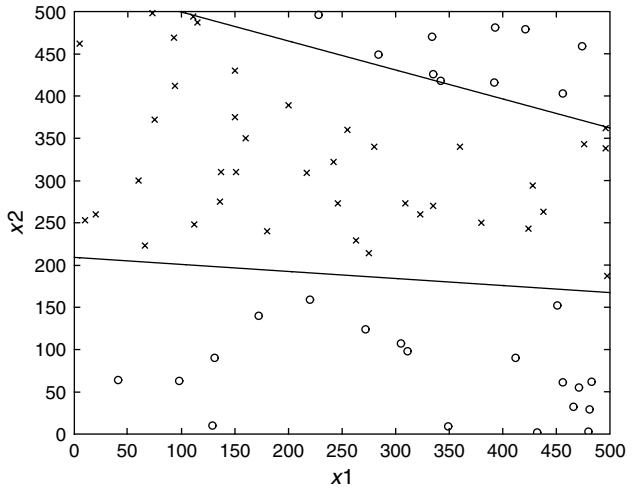
The last step in our process is the elimination of redundant hyperplanes in the representation of polyhedron P_k given in Equation (12). Figures 10 and 11 illustrate this procedure. We can check whether the constraint

$$\boldsymbol{\pi}'_{k,t_0}\mathbf{x} \leqslant \alpha_{k,t_0} \tag{13}$$

is redundant for the representation of P_k by solving the following linear optimization problem (note that the decision variables are $\mathbf{x}$):

$$\begin{aligned} w_{k,t_0} = \text{maximize}\ & \boldsymbol{\pi}'_{k,t_0}\mathbf{x} \\ \text{subject to}\ & \boldsymbol{\pi}'_{k,t}\mathbf{x} \leqslant \alpha_{k,t}, \quad t \in \bar{K}_0 \backslash \{t_0\}, \\ & \boldsymbol{\pi}'_{k,t_0}\mathbf{x} \leqslant \alpha_{k,t_0} + 1. \end{aligned} \tag{14}$$

Figure 11. After eliminating redundant constraints.



We have included only the last constraint to prevent problem (14) from becoming unbounded. If $w_{k,t_0} \leqslant \alpha_{k,t_0}$, then constraint (13) is implied by the other constraints defining P_k, and thus it is redundant. However, if $w_{k,t_0} > \alpha_{k,t_0}$, then constraint (13) is necessary for describing the polyhedron P_k. To summarize, the following algorithm eliminates all redundant constraints.

1. **for** k = 1 to K **do**
2. **for** t_0 = 1 to K_0 **do**
3. Solve problem (14).
4. **if** $w_{k,t_0} \leqslant \alpha_{k,t_0}$ **then**
5. Eliminate constraint $\boldsymbol{\pi}'_{k,t_0}\mathbf{x} \leqslant \alpha_{k,t_0}$.
6. **end if**
7. **end for**
8. **end for**

3.5. The Overall Algorithm for Classification

The overall algorithm for classification is as follows:

1. **Preprocessing**. Use the clustering algorithm outlined in §3.2 to find clusters. Eliminate clusters with cardinality less than 1% of m_0 (m_1) for Class 0 (Class 1) clusters.
2. **Assign clusters to groups**. Solve the mixed-integer optimization problem (10) to assign clusters of Class 1 points to groups, while eliminating potential outliers.
3. **Assign groups to polyhedral regions**. Solve the quadratic optimization problem (11) to find hyperplanes that define the polyhedron of each Class 1 group.
4. **Eliminate redundant constraints**. Remove redundant constraints from the polyhedra following the algorithm outlined at the end of §3.4.

After CRIO determines the nonredundant representations of the polyhedra, the model is used to predict the class of new data points. If the point lies in any of the K polyhedra, we label the point as a Class 1 point. If the point is not contained in any of the polyhedra, then we label the point as a Class 0 point.

4. CRIO for Regression

In this section, we present in detail our approach for regression. For presentation purposes, we start in §4.1 with an initial mixed-integer optimization model to assign points to groups, which, while not practical because of dimensionality problems, forms the basis of our approach. As outlined in §2.2, we first assign points to clusters (§4.2), then assign clusters to groups of points (§4.3), which we then represent by polyhedral regions P_k (§4.4). In §4.5, we propose a method of automatically finding nonlinear transformations of the explanatory variables to improve the predictive power of the method. Finally, we present the overall algorithm in §4.6.

4.1. Assigning Points to Groups: An Initial Model

The training data consist of n observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We let $N = \{1, \ldots, n\}$,

$\bar{K} = \{1, \ldots, K\}$, and M be a large positive constant. We define binary variables for $k \in \bar{K}$ and $i \in N$:

$$a_{k,i} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is assigned to group } k, \\ 0 & \text{otherwise.} \end{cases}$$

The mixed-integer optimization model is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \delta_i \\ \text{subject to} \quad & \delta_i \geqslant (y_i - \boldsymbol{\beta}_k' \mathbf{x}_i) - M(1 - a_{k,i}), \quad k \in \bar{K};\ i \in N, \\ & \delta_i \geqslant -(y_i - \boldsymbol{\beta}_k' \mathbf{x}_i) - M(1 - a_{k,i}), \quad k \in \bar{K};\ i \in N, \\ & \sum_{k=1}^{K} a_{k,i} = 1, \quad i \in N, \\ & a_{k,i} \in \{0, 1\}, \quad \delta_i \geqslant 0. \end{aligned} \tag{15}$$

From the first and second constraints, δ_i is the absolute error associated with point $\mathbf{x}_i$. If $a_{k,i} = 1$, then $\delta_i \geqslant (y_i - \boldsymbol{\beta}_k' \mathbf{x}_i)$, $\delta_i \geqslant -(y_i - \boldsymbol{\beta}_k' \mathbf{x}_i)$, and the minimization of δ_i sets δ_i equal to $|y_i - \boldsymbol{\beta}_k' \mathbf{x}_i|$. If $a_{k,i} = 0$, the right-hand side of the first two constraints becomes negative, making them irrelevant because δ_i is nonnegative. Finally, the third constraint limits the assignment of each point to just one group.

We have found that even for relatively small n ($n \approx 200$), problem (15) is difficult to solve in reasonable time. For this reason, we initially run a clustering algorithm, similar to that of §3.2, to cluster nearby $\mathbf{x}_i$ points together. After L such clusters are found, for $L \ll n$, we can solve a mixed-integer optimization model, similar to problem (15), but with significantly fewer binary decision variables.

4.2. The Clustering Algorithm

We apply a nearest-neighbor clustering algorithm in the combined $(\mathbf{x}, y)$ space, as opposed to just the $\mathbf{x}$ space, to find L clusters. Specifically, the clustering algorithm initially starts with n clusters, then continues to merge the clusters with points close to each other until we are left with L clusters. More formally, the clustering algorithm is as follows:

1. **Initialize:** $k = n$. $C_i = \{i\}$, $i = 1, \ldots, n$.
2. **while** $l < L$ **do**
3. Find the points $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$, $i < j$, with minimum pairwise statistical distance. Let $l(i)$ and $l(j)$ be the indices of the clusters that $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ currently belong to, respectively.
4. Add cluster $l(j)$'s points to cluster $l(i)$, i.e., $C_{l(i)} := C_{l(i)} \cup C_{l(j)}$.
5. Let the pairwise statistical distance between all the points in $C_{l(i)}$ be ∞.
6. $l = l - 1$.
7. **end while**

In the clustering algorithm for classification problems (§3.2), we merged clusters that had centers of close proximity. However, in the present clustering algorithm, we merge clusters that contain points of close proximity. Computational experimentations showed that this latter method of clustering suited the regression problem better.

4.3. Assigning Points to Groups: A Practical Approach

Although we can continue the clustering algorithm of the previous section until we find K clusters, define them as our final groups, and find the best $\boldsymbol{\beta}_k$ coefficient for each group by solving separate linear regression problems, such an approach does not combine points to minimize the total absolute error. For this reason, we use the clustering algorithm until we have L, $L > K$, clusters and then solve a mixed-integer optimization model that assigns the L clusters into K groups to minimize the total absolute error. Another key concern in regression models is the presence of outliers. The mixed-integer optimization model we present next is able to remove potential outliers by eliminating points in clusters that tend to weaken the fit of the predictor coefficients.

Let C_l, $l \in \bar{L} = \{1, \ldots, L\}$, be cluster l, and denote $l(i)$ as $\mathbf{x}_i$'s cluster. Similarly to problem (15), we define the following binary variables for $k \in \bar{K} \cup \{0\}$ and $l \in \bar{L}$:

$$a_{k,l} = \begin{cases} 1 & \text{if cluster } l \text{ is assigned to group } k, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

We define $k = 0$ as the outlier group, in the sense that points in cluster l with $a_{0,l} = 1$ will be eliminated. The following model assigns clusters to groups and allows the possibility of eliminating clusters of points as outliers:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \delta_i \\ \text{subject to} \quad & \delta_i \geqslant (y_i - \boldsymbol{\beta}_k' \mathbf{x}_i) - M(1 - a_{k,l(i)}), \quad k \in \bar{K};\ i \in N, \\ & \delta_i \geqslant -(y_i - \boldsymbol{\beta}_k' \mathbf{x}_i) - M(1 - a_{k,l(i)}), \\ & \qquad\qquad k \in \bar{K};\ i \in N, \\ & \sum_{k=0}^{K} a_{k,l} = 1, \quad l \in \bar{L}, \\ & \sum_{l=1}^{L} |C_l| a_{0,l} \leqslant \rho |N|, \\ & a_{k,l} \in \{0, 1\}, \quad \delta_i \geqslant 0, \end{aligned} \tag{17}$$

where M is a large positive constant, and ρ is the maximum fraction of points that can be eliminated as outliers.

From the first and second set of constraints, δ_i is the absolute error associated to point $\mathbf{x}_i$. If $a_{k,l(i)} = 1$, then $\delta_i \geqslant (y_i - \boldsymbol{\beta}_k' \mathbf{x}_i)$, $\delta_i \geqslant -(y_i - \boldsymbol{\beta}_k' \mathbf{x}_i)$, and the minimization of δ_i

sets it equal to $|y_i - \boldsymbol{\beta}_k' \mathbf{x}_i|$. If $a_{k,l(i)} = 0$, the first two constraints become irrelevant because δ_i is nonnegative. The third set of constraints limits the assignment of each cluster to just one group (including the outlier group). The last constraint limits the percentage of points eliminated to be less than or equal to a pre-specified number ρ. If $a_{k,l} = 1$, then all the points in cluster l are assigned to group k, i.e., $G_k = \bigcup_{\{l \mid a_{k,l}=1\}} C_l$.

Problem (17) has KL binary variables as opposed to Kn binary variables in problem (15). The number of clusters L controls the trade-off between the quality of the solution and the efficiency of the computation. As L increases, the quality of the solution increases, but the efficiency of the computation decreases. In §5.2, we discuss appropriate values for K, L, and ρ from our computational experimentation.

4.4. Assigning Groups to Polyhedral Regions

We identify K groups of points solving problem (17). In this section, we establish a geometric representation of group k by a polyhedron P_k.

It is possible for the convex hulls of the K groups to overlap, and thus we might not be able to define disjoint regions of P_k that contain all the points of group k. For this reason, our approach is based on separating pairs of groups with the objective of minimizing the sum of violations. We first outline how to separate group k and group r, where $k < r$. We consider the following two linear optimization problems:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in G_k} \epsilon_i + \sum_{l \in G_r} \epsilon_l \\
\text{subject to} \quad & \mathbf{p}_{k,r}' \mathbf{x}_i' - q_{k,r} \leqslant -1 + \epsilon_i, \quad i \in G_k, \\
& \mathbf{p}_{k,r}' \mathbf{x}_l - q_{k,r} \geqslant 1 - \epsilon_l, \quad l \in G_r, \\
& \mathbf{p}_{k,r}' \mathbf{e} \geqslant 1, \\
& \epsilon_i \geqslant 0, \quad \epsilon_l \geqslant 0;
\end{aligned} \tag{18}$$

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in G_k} \epsilon_i + \sum_{l \in G_r} \epsilon_l \\
\text{subject to} \quad & \mathbf{p}_{k,r}' \mathbf{x}_i - q_{k,r} \leqslant -1 + \epsilon_i, \quad i \in G_k, \\
& \mathbf{p}_{k,r}' \mathbf{x}_l - q_{k,r} \geqslant 1 - \epsilon_l, \quad l \in G_r, \\
& \mathbf{p}_{k,r}' \mathbf{e} \leqslant -1, \\
& \epsilon_i \geqslant 0, \quad \epsilon_l \geqslant 0,
\end{aligned} \tag{19}$$

where $\mathbf{e}$ is a vector of ones.

Both problems (18) and (19) find a hyperplane $\mathbf{p}_{k,r}' \mathbf{x} = q_{k,r}$ that softly separates points in group k from points in group r, i.e., points in either group can be on the wrong side of this hyperplane if necessary. The purpose of the third constraint is to prevent getting the trivial hyperplane $\mathbf{p}_{k,r} = \mathbf{0}$ and $q_{k,r} = 0$ for the optimal solution. Problem (18) sets the sum of the elements of $\mathbf{p}_{k,r}$ to be strictly positive, and problem (19) sets the sum of the elements of $\mathbf{p}_{k,r}$ to be strictly negative. Both problems need to be solved because we do not know a priori whether the sum of the elements of the optimal nontrivial $\mathbf{p}_{k,r}$ is positive or negative. The optimal solution of the problem that results in the least number of violated points is chosen as our hyperplane.

After we solve problems (18) and (19) for every pair of groups, we let

$$\begin{aligned}
P_k = \{\mathbf{x} \mid \mathbf{p}_{k,i}' \mathbf{x} \leqslant q_{k,i},\ i = 1, \ldots, k-1, \\
\mathbf{p}_{k,i}' \mathbf{x} \geqslant q_{k,i},\ i = k+1, \ldots, K\}.
\end{aligned} \tag{20}$$

After P_k is defined, we recompute $\boldsymbol{\beta}_k$ using all the points contained in P_k because it is possible that they are different from the original G_k that problem (17) found. We solve a linear optimization problem that minimizes the absolute deviation of all points in P_k to find the new $\boldsymbol{\beta}_k$.

4.5. Nonlinear Data Transformations

To improve the predictive power of CRIO, we augment the explanatory variables with nonlinear transformations. In particular, we consider the transformations x^2, $\log x$, and $1/x$ applied to the coordinates of the given points. We can augment each d-dimensional vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})'$ with $x_{i,j}^2$, $\log x_{i,j}$, $1/x_{i,j}$, $j = 1, \ldots, d$, and apply CRIO to the resulting $4d$-dimensional vectors, but the increased dimension slows down the computation time. For this reason, we use a simple heuristic method to choose which transformation of which variable to include in the data set.

For $j = 1, \ldots, d$, we run the one-dimensional regressions: (a) $(x_{i,j}, y_i)$, $i \in N$, with the sum of squared errors equal to $f_{j,1}$; (b) $(x_{i,j}^2, y_i)$, $i \in N$, with the sum of squared errors equal to $f_{j,2}$; (c) $(\log x_{i,j}, y_i)$, $i \in N$, with the sum of squared errors equal to $f_{j,3}$; and (d) $(1/x_{i,j}, y_i)$, $i \in N$, with the sum of squared errors equal to $f_{j,4}$. If $f_{j,2} < f_{j,1}$, we add $x_{i,j}^2$ and eliminate $x_{i,j}$. If $f_{j,3} < f_{j,1}$, we add $\log x_{i,j}$ and eliminate $x_{i,j}$. If $f_{j,4} < f_{j,1}$, we add $1/x_{i,j}$ and eliminate $x_{i,j}$. Otherwise, we do not add any nonlinear transformation to the data set.

4.6. The Overall Algorithm for Regression

The overall algorithm for regression is as follows:

1. **Nonlinear transformation**. Augment the original data set with nonlinear transformations using the method discussed in §4.5.
2. **Preprocessing**. Use the clustering algorithm to find $L \ll n$ clusters of the data points.
3. **Assign clusters to groups**. Solve problem (17) to determine which points belong to which group, while eliminating potential outliers.
4. **Assign groups to polyhedral regions**. Solve the linear optimization problems (18) and (19) for all pairs of groups, and define polyhedra as in Equation (20).
5. **Re-computation of βs**. Once the polyhedra P_k are identified, recompute $\boldsymbol{\beta}_k$ using only the points that belong in P_k.

Given a new point $\mathbf{x}_0$ (augmented by the same transformations as applied in the training set data), if $\mathbf{x}_0 \in P_k$, then we predict $\hat{y}_0 = \boldsymbol{\beta}_k' \mathbf{x}_0$. Otherwise, we assign $\mathbf{x}_0$ to the region P_r that contains the majority among its F nearest neighbors in the training set, and make the prediction $\hat{y}_0 = \boldsymbol{\beta}_r' \mathbf{x}_0$.

5. Computational Results

In this section, we report on the performance of CRIO on several generated and widely circulated real data sets. In §5.1, we present CRIO's performance on classification data sets and compare it to the performances of logistic regression, neural networks, classification trees, tree-boosting methods, and SVM. In §5.2, we present CRIO's performance on regression data sets and compare it to the performances of least-square regression, neural networks, tree-boosting methods, and MARS.

5.1. Classification Results

We tested the classification component of CRIO on two sets of generated data and five real data sets, and we compared its performance against logistic regression (via MATLAB's® "Logist" procedure), neural network classifier (via MATLAB's® neural network toolbox), SVM (via SVMfu[4]) (Rifkin 2002), and classification trees with univariate and multivariate partition rules (via CRUISE[5]—a classification tree software with both univariate and multivariate partitioning (Kim and Loh 2000, Loh and Shih 1997, Kim and Loh 2001)). We also tested two tree-boosting methods: the generalized boosted method (GBM) implemented in R (Ridgeway 2005), and TreeNet™ by Salford Systems, which is an implementation of multiple additive regression trees (Friedman 2002). Although both methods are based on Friedman (1999a, b), we present them both because they exhibited varying behaviors on different data sets.

Each data set was split into three parts: the training set, the validation set, and the testing set. The training set, comprising 50% of the data, was used to develop the model. The validation set, comprising 30% of the data, was used to select the best values of the model parameters. Finally, the remaining points were part of the testing set, which ultimately decided the prediction accuracy and generalizability of the model. This latter set was put aside until the very end, after the parameters of the model were finalized. The assignment to each of the three sets was done randomly, and this process was repeated 10 times for each data set.

In CRIO, we used the validation set to decide on the appropriate value for K and which class to assign to groups. In all cases, we used the the mixed-integer programming model (10), set the parameters M, K_0, and K_1 to 1,000, 10, and 10, respectively, and deleted clusters with cardinality less than 1% of the total number of points as outliers. In logistic regression, we used the validation set for deciding the cut-off level. In neural networks, we used the validation set to fine-tune several parameters, including the number of epochs, activation function, and number of nodes in the hidden layer. In classification trees, the validation set was used to set parameters for CRUISE, such as the variable selection method, split selection method, and the α value. For SVM, linear, polynomial (with degree 2 and 3), and Gaussian kernels were all tested, as well as different cost per unit violation of the margin. The kernel and parameters resulting in the best validation accuracy were chosen to classify the testing set. For GBM, we tested shrinkage values between 0.001 and 0.01, and 3,000 to 10,000 tree iterations, as suggested by Ridgeway (2005). We use the Bernoulli distribution for the loss criterion distribution. In almost all cases, shrinkage of 0.01 and 3,000 iterations gave the best cross-validation accuracy. Similarly, for TreeNet™, we tested shrinkage values between 0.001 to 0.01, 3,000 to 5,000 tree iterations, and the logistic loss criterion. In all cases, shrinkage of 0.01 and 5,000 iterations gave the best cross-validation performance.

We solved all optimization problems (mixed-integer, quadratic, and linear) using CPLEX 8.0[6] (ILOG 2001) running on a Linux desktop.

Classification Data. We generated two sets of data sets for experimentation. First, we randomly generated data from mixtures of Gaussian distributions. For each class, 0 and 1, we generated covariance matrices and the mean vectors for two Gaussian distributions, and we randomly generated data points from the distributions. Second, we generated a data set partitioned into polyhedral regions constructed by a decision tree with multivariate division rules. The tree was generated by randomly generating a hyperplane to partition the input space, then for each partition, a new subtree or leaf was generated recursively. The leaf nodes were uniformly assigned to Classes 0 or 1. For every generated data set, we mislabeled the class of 5% of the points, making them act as outliers. The Gaussian data sets were generated to favor SVMs with Gaussian kernels, and the polyhedral data sets were generated to favor CRIO and classification trees with multivariate splits.

In addition to the generated data sets, we tested our models on four real data sets found on the UCI data repository (http://www.ics.uci.edu/~mlearn/MLSummary.html). The "Cancer" data are from the Wisconsin Breast Cancer database, with 682 data points and nine explanatory variables. The "Liver" data are from BUPA Medical Research Ltd., with 345 data points and six explanatory variables. The "Diabetes" data are from the Pima Indian Diabetes database, with 768 data points and seven explanatory variables. The "Heart" data are from the SPECT heart database, where we combined the given training and testing set to get 349 data points and 44 explanatory variables. All data sets involve binary classification.

Results. For logistic regression, neural networks, univariate and multivariate classification trees, GBM, TreeNet™, SVM, and CRIO, Tables 1 and 2 summarize their classification accuracy on the Gaussian data set,

Table 1. Classification accuracy of logistic regression, neural networks, and univariate and multivariate classification trees on generated Gaussian data with n data points in $\mathbb{R}^d$.

Gaussian data	n	d	Logistic regression			Neural networks		
			Train	Validation	Test	Train	Validation	Test
G1	500	2	0.976 (0.010)	0.976 (0.012)	0.972 (0.017)	0.999 (0.003)	0.993 (0.004)	0.995 (0.007)
G2	500	10	0.606 (0.021)	0.595 (0.025)	0.592 (0.034)	1.000 (0.000)	0.995 (0.008)	0.996 (0.005)
G3	1,000	2	0.726 (0.017)	0.738 (0.022)	0.742 (0.028)	0.918 (0.011)	0.914 (0.016)	0.912 (0.018)
G4	1,000	10	0.608 (0.016)	0.591 (0.022)	0.595 (0.029)	1.000 (0.000)	0.996 (0.005)	0.992 (0.009)
G5	1,000	20	0.603 (0.012)	0.598 (0.018)	0.595 (0.027)	0.999 (0.001)	0.974 (0.010)	0.970 (0.009)
G6	2,000	2	0.716 (0.013)	0.729 (0.018)	0.721 (0.014)	0.928 (0.007)	0.929 (0.010)	0.934 (0.015)
G7	2,000	10	0.934 (0.154)	0.933 (0.148)	0.930 (0.154)	0.997 (0.007)	0.991 (0.005)	0.989 (0.009)
G8	2,000	20	0.949 (0.005)	0.943 (0.010)	0.938 (0.011)	0.992 (0.004)	0.955 (0.014)	0.949 (0.015)
G9	4,000	2	0.904 (0.005)	0.912 (0.008)	0.899 (0.008)	0.981 (0.002)	0.982 (0.003)	0.980 (0.005)
G10	4,000	10	0.502 (0.005)	0.499 (0.009)	0.499 (0.013)	1.000 (0.001)	0.999 (0.001)	0.998 (0.001)
G11	4,000	20	0.740 (0.257)	0.743 (0.253)	0.744 (0.251)	0.996 (0.003)	0.984 (0.003)	0.982 (0.006)
			Univariate classification trees			Multivariate classification trees		
G1	500	2	0.964 (0.018)	0.939 (0.025)	0.930 (0.054)	0.978 (0.007)	0.970 (0.020)	0.970 (0.012)
G2	500	10	0.988 (0.008)	0.967 (0.009)	0.974 (0.014)	0.997 (0.002)	0.997 (0.006)	0.998 (0.004)
G3	1,000	2	0.914 (0.011)	0.888 (0.018)	0.892 (0.022)	0.868 (0.017)	0.854 (0.021)	0.857 (0.022)
G4	1,000	10	0.964 (0.006)	0.947 (0.013)	0.947 (0.012)	0.998 (0.003)	0.989 (0.008)	0.989 (0.010)
G5	1,000	20	0.915 (0.027)	0.865 (0.028)	0.860 (0.033)	0.986 (0.009)	0.968 (0.009)	0.975 (0.014)
G6	2,000	2	0.934 (0.016)	0.907 (0.028)	0.920 (0.018)	0.931 (0.012)	0.931 (0.013)	0.935 (0.013)
G7	2,000	10	0.960 (0.009)	0.947 (0.011)	0.943 (0.010)	0.991 (0.004)	0.983 (0.006)	0.987 (0.006)
G8	2,000	20	0.899 (0.012)	0.854 (0.012)	0.848 (0.022)	0.955 (0.010)	0.942 (0.011)	0.945 (0.014)
G9	4,000	2	0.979 (0.003)	0.973 (0.009)	0.971 (0.005)	0.957 (0.008)	0.959 (0.012)	0.955 (0.010)
G10	4,000	10	0.985 (0.002)	0.976 (0.005)	0.974 (0.006)	0.997 (0.002)	0.996 (0.003)	0.994 (0.003)
G11	4,000	20	0.951 (0.009)	0.918 (0.009)	0.922 (0.012)	0.983 (0.003)	0.981 (0.005)	0.981 (0.005)

Note. Standard deviations are in parentheses.

Table 2. Classification accuracy of generalized boosted methods (GBM), TreeNet™, logistic regression, neural networks, univariate and multivariate classification trees, and SVM and CRIO on generated Gaussian data with n data points in $\mathbb{R}^d$.

Gaussian data	n	d	GBM			TreeNet™		
			Train	Validation	Test	Train	Validation	Test
G1	500	2	1.000 (0)	0.984 (0.01)	0.987 (0.011)	0.990 (0.005)	0.985 (0.005)	0.982 (0.020)
G2	500	10	1.000 (0)	0.986 (0.01)	0.991 (0.007)	0.984 (0.015)	0.977 (0.011)	0.974 (0.014)
G3	1,000	2	0.930 (0.007)	0.910 (0.018)	0.911 (0.018)	0.923 (0.017)	0.912 (0.015)	0.899 (0.027)
G4	1,000	10	1.000 (0)	0.990 (0.004)	0.984 (0.007)	0.994 (0.003)	0.984 (0.009)	0.977 (0.008)
G5	1,000	20	1.000 (0.001)	0.967 (0.009)	0.97 (0.012)	0.998 (0.002)	0.963 (0.010)	0.962 (0.015)
G6	2,000	2	0.879 (0.013)	0.865 (0.016)	0.868 (0.014)	0.929 (0.008)	0.919 (0.018)	0.924 (0.012)
G7	2,000	10	0.996 (0.002)	0.979 (0.004)	0.982 (0.004)	0.992 (0.002)	0.978 (0.004)	0.976 (0.007)
G8	2,000	20	0.974 (0.004)	0.932 (0.007)	0.934 (0.01)	0.978 (0.003)	0.948 (0.007)	0.944 (0.013)
G9	4,000	2	0.977 (0.002)	0.975 (0.004)	0.972 (0.004)	0.979 (0.003)	0.978 (0.003)	0.973 (0.005)
G10	4,000	10	1.000 (0)	0.997 (0.002)	0.996 (0.002)	0.994 (0.003)	0.992 (0.004)	0.989 (0.007)
G11	4,000	20	0.99 (0.001)	0.975 (0.004)	0.976 (0.005)	0.987 (0.002)	0.973 (0.003)	0.972 (0.006)
			SVM			CRIO		
G1	500	2	0.996 (0.004)	0.993 (0.003)	0.993 (0.008)	0.998 (0.003)	0.991 (0.008)	0.994 (0.005)
G2	500	10	0.998 (0.002)	0.999 (0.002)	0.998 (0.004)	1.000 (0)	1.000 (0)	1.000 (0)
G3	1,000	2	0.924 (0.010)	0.918 (0.012)	0.915 (0.018)	0.918 (0.011)	0.910 (0.014)	0.921 (0.019)
G4	1,000	10	0.998 (0.003)	0.996 (0.003)	0.991 (0.007)	1.000 (0.001)	0.997 (0.003)	0.997 (0.005)
G5	1,000	20	0.999 (0.001)	0.986 (0.006)	0.990 (0.007)	1.000 (0)	0.981 (0.006)	0.985 (0.011)
G6	2,000	2	0.955 (0.008)	0.953 (0.008)	0.952 (0.009)	0.946 (0.005)	0.948 (0.010)	0.952 (0.011)
G7	2,000	10	0.992 (0.002)	0.986 (0.003)	0.987 (0.005)	0.999 (0.002)	0.992 (0.003)	0.993 (0.004)
G8	2,000	20	0.995 (0.001)	0.975 (0.007)	0.971 (0.010)	1.000 (0)	0.962 (0.006)	0.964 (0.005)
G9	4,000	2	0.982 (0.002)	0.983 (0.004)	0.980 (0.004)	0.982 (0.002)	0.983 (0.004)	0.981 (0.004)
G10	4,000	10	0.999 (0.001)	0.999 (0.001)	0.972 (0.084)	1.000 (0)	1.000 (0)	1.000 (0)
G11	4,000	20	0.998 (0.001)	0.989 (0.003)	0.990 (0.002)	0.993 (0.003)	0.987 (0.002)	0.984 (0.003)

Note. Standard deviations are in parentheses.

Table 3. Average CPU time for logistic regression (LR), neural networks (NN), univariate classification trees (CTreeU), multivariate classification trees (CTreeM), generalized boosted methods (GBM), TreeNet™, SVM, and CRIO for the Gaussian data sets.

Data	LR	NN	CTreeU	CTreeM	GBM	TreeNet™	SVM	CRIO
G1	0.004	0.424	0.067	0.023	1.065	[6.117, 37.71]	0.014	0.048
G2	0.012	0.375	0.090	0.029	1.767	[10.37, 39.75]	0.010	0.085
G3	0.003	0.667	0.325	0.120	1.931	[8.040, 38.25]	0.060	0.443
G4	0.018	0.522	0.317	0.102	3.711	[11.58, 36.22]	0.047	0.620
G5	0.025	0.601	0.543	0.324	5.489	[18.57, 40.29]	0.075	1.223
G6	0.006	1.485	1.097	0.288	4.166	[12.04, 37.79]	0.492	1.977
G7	0.021	0.965	1.254	0.292	7.509	[22.79, 41.30]	0.111	3.477
G8	0.034	0.867	1.697	0.973	11.60	[34.81, 43.38]	3.898	11.77
G9	0.019	2.268	2.684	0.388	8.872	[19.37, 36.72]	0.736	1.442
G10	0.073	1.958	4.180	0.379	15.31	[35.90, 37.72]	0.139	11.52
G11	0.109	1.877	5.056	1.614	23.44	[71.08, 71.08]	41.13	36.04

Tables 4 and 5 summarize their accuracy on the polyhedral data set, and Table 7 summarizes their accuracy on the real data sets. The columns labeled "Train," "Validation," and "Test" illustrate the fraction of the training, validation, and testing set, respectively, that the models correctly classified. Each entry is the average of 10 independent runs. The numbers in the parentheses are the corresponding standard deviation.

Tables 3, 6, and 8 summarize the average computation time (in CPU seconds) of the algorithms on the respective data sets. We were not able to get the exact total running time from TreeNet™, but were able to calculate the lower and upperbound of the total CPU seconds.[7]

We compared the performance of the various classification algorithms in terms of its predictive ability and the stability of its solutions. We use the classification accuracy of the methods on the testing set as the measure of predictive accuracy, and we use the standard deviation as a measure of stability of the prediction over different partitions of the data set.

For the Gaussian data sets, CRIO, SVM, and neural networks did consistently well, and multivariate classification

Table 4. Classification accuracy of logistic regression, neural networks, and univariate and multivariate classification trees on generated polyhedral data with n data points in $\mathbb{R}^d$.

Polyhedral data	n	d	Logistic regression			Neural networks		
			Train	Validation	Test	Train	Validation	Test
P1	500	2	0.728 (0.020)	0.752 (0.033)	0.727 (0.036)	0.823 (0.111)	0.821 (0.083)	0.806 (0.110)
P2	500	10	0.571 (0.030)	0.597 (0.033)	0.563 (0.051)	0.538 (0.033)	0.575 (0.048)	0.547 (0.078)
P3	1,000	2	0.625 (0.067)	0.612 (0.058)	0.620 (0.071)	0.783 (0.180)	0.772 (0.170)	0.777 (0.170)
P4	1,000	10	0.903 (0.011)	0.897 (0.010)	0.892 (0.018)	0.759 (0.020)	0.765 (0.019)	0.751 (0.033)
P5	1,000	20	0.945 (0.015)	0.934 (0.013)	0.927 (0.021)	0.541 (0.049)	0.548 (0.034)	0.541 (0.048)
P6	2,000	2	0.867 (0.006)	0.865 (0.013)	0.852 (0.016)	0.945 (0.017)	0.943 (0.017)	0.938 (0.014)
P7	2,000	10	0.899 (0.009)	0.895 (0.009)	0.899 (0.013)	0.709 (0.158)	0.701 (0.155)	0.698 (0.155)
P8	2,000	20	0.880 (0.008)	0.875 (0.011)	0.880 (0.022)	0.538 (0.103)	0.549 (0.090)	0.544 (0.104)
P9	4,000	2	0.900 (0.010)	0.905 (0.009)	0.894 (0.014)	0.949 (0.033)	0.949 (0.036)	0.952 (0.034)
P10	4,000	10	0.813 (0.006)	0.809 (0.008)	0.805 (0.015)	0.703 (0.034)	0.692 (0.036)	0.690 (0.024)
P11	4,000	20	0.855 (0.007)	0.855 (0.007)	0.847 (0.013)	0.702 (0.044)	0.703 (0.047)	0.704 (0.052)
			Univariate classification trees			Multivariate classification trees		
P1	500	2	0.906 (0.025)	0.848 (0.055)	0.837 (0.046)	0.884 (0.025)	0.881 (0.026)	0.880 (0.025)
P2	500	10	0.592 (0.062)	0.582 (0.035)	0.581 (0.050)	0.566 (0.036)	0.581 (0.036)	0.579 (0.052)
P3	1,000	2	0.898 (0.034)	0.847 (0.040)	0.835 (0.047)	0.942 (0.026)	0.940 (0.039)	0.931 (0.023)
P4	1,000	10	0.876 (0.014)	0.842 (0.022)	0.828 (0.030)	0.905 (0.011)	0.886 (0.017)	0.892 (0.018)
P5	1,000	20	0.771 (0.026)	0.718 (0.028)	0.697 (0.038)	0.931 (0.008)	0.916 (0.017)	0.915 (0.018)
P6	2,000	2	0.975 (0.005)	0.965 (0.006)	0.963 (0.012)	0.949 (0.006)	0.943 (0.006)	0.944 (0.010)
P7	2,000	10	0.824 (0.030)	0.751 (0.029)	0.754 (0.020)	0.909 (0.012)	0.894 (0.011)	0.899 (0.011)
P8	2,000	20	0.758 (0.044)	0.675 (0.038)	0.693 (0.026)	0.911 (0.018)	0.886 (0.009)	0.890 (0.026)
P9	4,000	2	0.969 (0.004)	0.958 (0.005)	0.957 (0.006)	0.950 (0.008)	0.945 (0.011)	0.947 (0.009)
P10	4,000	10	0.826 (0.014)	0.779 (0.012)	0.776 (0.016)	0.884 (0.011)	0.849 (0.013)	0.848 (0.017)
P11	4,000	20	0.801 (0.013)	0.762 (0.011)	0.755 (0.018)	0.869 (0.015)	0.856 (0.007)	0.859 (0.017)

Note. Standard deviations are in parentheses.

Table 5. Classification accuracy of logistic regression, neural networks, univariate and multivariate classification trees, SVM, and CRIO on generated polyhedral data with n data points in $\mathbb{R}^d$.

Polyhedral data	n	d	GBM			TreeNet™		
			Train	Validation	Test	Train	Validation	Test
P1	500	2	0.825 (0.016)	0.778 (0.032)	0.755 (0.033)	0.950 (0.017)	0.932 (0.014)	0.919 (0.035)
P2	500	10	0.623 (0.064)	0.585 (0.041)	0.544 (0.053)	0.758 (0.061)	0.567 (0.033)	0.508 (0.051)
P3	1,000	2	0.666 (0.022)	0.621 (0.023)	0.630 (0.021)	0.963 (0.007)	0.938 (0.018)	0.921 (0.017)
P4	1,000	10	0.946 (0.008)	0.880 (0.011)	0.872 (0.022)	0.948 (0.012)	0.891 (0.011)	0.877 (0.023)
P5	1,000	20	0.972 (0.006)	0.887 (0.014)	0.881 (0.015)	0.972 (0.007)	0.855 (0.019)	0.859 (0.022)
P6	2,000	2	0.977 (0.003)	0.967 (0.005)	0.966 (0.007)	0.979 (0.004)	0.974 (0.006)	0.971 (0.006)
P7	2,000	10	0.930 (0.008)	0.881 (0.014)	0.880 (0.014)	0.937 (0.015)	0.866 (0.026)	0.867 (0.021)
P8	2,000	20	0.920 (0.009)	0.850 (0.016)	0.864 (0.014)	0.937 (0.011)	0.840 (0.015)	0.852 (0.021)
P9	4,000	2	0.975 (0.004)	0.965 (0.007)	0.964 (0.008)	0.979 (0.003)	0.972 (0.004)	0.970 (0.009)
P10	4,000	10	0.842 (0.004)	0.809 (0.011)	0.808 (0.011)	0.851 (0.006)	0.805 (0.009)	0.812 (0.008)
P11	4,000	20	0.876 (0.008)	0.843 (0.003)	0.838 (0.01)	0.870 (0.013)	0.818 (0.014)	0.806 (0.012)
			SVM			CRIO		
P1	500	2	0.965 (0.018)	0.956 (0.014)	0.953 (0.016)	0.962 (0.017)	0.948 (0.017)	0.953 (0.014)
P2	500	10	1.000 (0.000)	0.595 (0.040)	0.567 (0.057)	0.841 (0.113)	0.605 (0.017)	0.563 (0.023)
P3	1,000	2	0.980 (0.006)	0.970 (0.012)	0.963 (0.014)	0.987 (0.003)	0.984 (0.006)	0.981 (0.010)
P4	1,000	10	0.906 (0.008)	0.889 (0.0.010)	0.892 (0.017)	0.990 (0.009)	0.917 (0.009)	0.922 (0.018)
P5	1,000	20	0.940 (0.025)	0.927 (0.0.018)	0.930 (0.016)	0.963 (0.007)	0.936 (0.018)	0.938 (0.013)
P6	2,000	2	0.869 (0.007)	0.854 (0.016)	0.854 (0.014)	0.975 (0.007)	0.968 (0.011)	0.968 (0.009)
P7	2,000	10	0.894 (0.012)	0.883 (0.011)	0.887 (0.019)	0.945 (0.015)	0.909 (0.011)	0.910 (0.016)
P8	2,000	20	0.886 (0.019)	0.874 (0.019)	0.883 (0.018)	0.906 (0.015)	0.877 (0.013)	0.885 (0.018)
P9	4,000	2	0.905 (0.006)	0.905 (0.010)	0.901 (0.014)	0.983 (0.002)	0.985 (0.004)	0.984 (0.005)
P10	4,000	10	0.811 (0.008)	0.802 (0.011)	0.804 (0.008)	0.924 (0.007)	0.894 (0.008)	0.881 (0.012)
P11	4,000	20	0.791 (0.070)	0.789 (0.071)	0.787 (0.064)	0.927 (0.008)	0.885 (0.006)	0.880 (0.014)

Note. Standard deviations are in parentheses.

trees and tree-boosting methods did well in a majority of the cases. In terms of prediction accuracy, CRIO outperformed SVM in seven out of the 11 cases, outperformed neural networks in 10 out of the 11 cases, and outperformed multivariate classification trees and both tree-boosting methods in all 11 cases. The high prediction accuracies and low standard deviation of the results show that CRIO's results are consistently accurate and stable across the 10 partitions. It is interesting to note that CRIO has comparably low standard deviations as SVMs because the latter has several theoretical stability and generalizability results (Bousquet and Elisseeff 2002, Rifkin 2002). As expected, SVM used the Gaussian kernel for all cases. CRIO used $K = 2$ for data set G6 and G8, and $K = 1$ for the rest.

CRIO, SVM, and the multivariate classification tree performed consistently well in the polyhedral data set. CRIO outperformed SVM in nine out of the 11 cases and outperformed CRUISE in 10 out of the 11 cases. The multivariate classification tree does not do as well as expected, given that the data set was partitioned into polyhedral regions. It is clear in this case that CRIO's major advantage over classification trees is the *global optimization* of its polyhe-

Table 6. Average CPU time for logistic regression (LR), neural networks (NN), univariate classification trees (CTreeU), multivariate classification trees (CTreeM), generalized boosted methods (GBM), SVM, and CRIO for the polyhedral data sets.

Data	LR	NN	CTreeU	CTreeM	GBM	TreeNet™	SVM	CRIO
P1	0.003	0.537	0.077	0.035	0.508	[6.279, 37.52]	0.015	0.248
P2	0.001	0.333	0.201	0.175	0.232	[8.277, 38.81]	0.026	0.147
P3	0.001	0.679	0.265	0.074	0.570	[6.989, 34.08]	0.084	0.152
P4	0.009	0.395	0.312	0.125	2.449	[11.99, 41.16]	946.2	1.435
P5	0.017	0.376	0.581	0.298	5.362	[21.94, 46.96]	1,757	0.372
P6	0.006	1.374	0.528	0.160	4.276	[10.74, 38.59]	183.7	2.574
P7	0.015	0.486	1.052	0.500	6.990	[22.32, 45.43]	2,721	3.745
P8	0.030	0.595	1.554	1.370	10.14	[35.40, 50.37]	2,314	5.868
P9	0.018	2.561	0.524	0.496	8.092	[22.36, 45.53]	827.5	1.985
P10	0.032	0.738	2.538	1.729	11.12	[41.53, 53.74]	2,689	16.36
P11	0.066	0.911	3.655	3.069	20.09	[63.08, 68.94]	3,578	30.29

Table 7. Classification accuracy of logistic regression, neural networks, univariate and multivariate classification trees, SVM, and CRIO on data sets Cancer, Liver, Diabetes, and Heart, each with n data points in $\mathbb{R}^d$.

Real data			Logistic regression			Neural networks		
Data set	n	d	Train	Validation	Test	Train	Validation	Test
Cancer	683	9	0.970 (0.008)	0.976 (0.011)	0.964 (0.009)	0.975 (0.007)	0.970 (0.012)	0.974 (0.011)
Liver	345	6	0.702 (0.031)	0.702 (0.045)	0.676 (0.040)	0.699 (0.090)	0.689 (0.037)	0.639 (0.098)
Diabetes	768	8	0.774 (0.023)	0.770 (0.034)	0.775 (0.033)	0.704 (0.087)	0.686 (0.087)	0.699 (0.089)
Heart	349	44	0.283 (0.024)	0.252 (0.031)	0.276 (0.050)	0.713 (0.030)	0.743 (0.030)	0.727 (0.053)
			Univariate classification trees			Multivariate classification trees		
Cancer	683	9	0.957 (0.011)	0.941 (0.024)	0.942 (0.013)	0.969 (0.009)	0.960 (0.008)	0.962 (0.017)
Liver	345	6	0.653 (0.075)	0.584 (0.043)	0.616 (0.043)	0.697 (0.048)	0.658 (0.044)	0.644 (0.045)
Diabetes	768	8	0.779 (0.026)	0.728 (0.026)	0.736 (0.037)	0.781 (0.019)	0.760 (0.035)	0.766 (0.025)
Heart	349	44	0.828 (0.061)	0.769 (0.027)	0.783 (0.073)	0.874 (0.095)	0.764 (0.033)	0.754 (0.063)
			GBM			TreeNet™		
Cancer	683	9	0.978 (0.005)	0.964 (0.012)	0.968 (0.016)	0.977 (0.005)	0.972 (0.012)	0.968 (0.013)
Liver	345	6	0.844 (0.019)	0.724 (0.030)	0.720 (0.048)	0.818 (0.057)	0.723 (0.039)	0.710 (0.043)
Diabetes	768	8	0.811 (0.013)	0.752 (0.020)	0.773 (0.021)	0.818 (0.027)	0.760 (0.032)	0.758 (0.027)
Heart	349	44	0.969 (0.022)	0.849 (0.034)	0.840 (0.050)	0.919 (0.036)	0.853 (0.034)	0.844 (0.044)
			SVM			CRIO		
Cancer	683	9	0.970 (0.006)	0.969 (0.010)	0.968 (0.013)	0.977 (0.006)	0.973 (0.007)	0.977 (0.009)
Liver	345	6	0.788 (0.046)	0.704 (0.035)	0.697 (0.045)	0.783 (0.042)	0.741 (0.026)	0.717 (0.046)
Diabetes	768	8	0.789 (0.028)	0.749 (0.021)	0.764 (0.046)	0.805 (0.032)	0.777 (0.028)	0.777 (0.032)
Heart	349	44	1.000 (0.000)	0.867 (0.032)	0.848 (0.061)	0.998 (0.004)	0.866 (0.016)	0.851 (0.056)

Note. Standard deviations are in parentheses.

dral partition. Again, the low standard deviation of CRIO's prediction results implies the consistency of the prediction results. SVM used Gaussian kernels for P1, P2, and P3, and linear kernels for the rest. CRIO used $K = 2$ for P6 and $K = 1$ for the rest.

CRIO outperformed all methods, in terms of prediction accuracy, for all four of the real data sets. CRIO's results, again, correspond to low standard deviations. SVM used Gaussian kernels for all data sets, and CRIO used $K = 2$ for the "Diabetes" data set and $K = 1$ for all other data sets.

For cases where CRIO used $K = 1$, 80% to 99% of the total CPU time was spent on finding a polyhedron for each Class 1 group (§3.4), i.e., solving the convex quadratic programming problem (11) for each Class 1 group and Class 0 cluster. Note that for $K = 1$, the mixed-integer programming problem (10) is essentially a linear programming problem. Thus, the significant portion of the total computation time is spent on solving the sets of quadratic programming problems, not the mixed-integer programming problem. We used CPLEX 8.0 to solve each of the separating hyperplane problems (11), without taking advantage of the special structures of this quadratic programming problem, as done by all SVM algorithms, including SVMfu. We believe that using tailored quadratic solvers for the separating hyperplane problem would significantly speed up the computation time in these cases. In cases when $K = 2$, about 10% to 15% of the time was used to solve the quadratic program. The remainder of the time was used to solve the mixed-integer programming problem. Again, given the particular properties of the mixed-integer programming problem, a more tailored solution method might speed up the computation time.

Table 8. Average CPU time for logistic regression (LR), neural networks (NN), univariate classification trees (CTreeU), multivariate classification trees (CTreeM), generalized boosted methods (GBM), TreeNet™, SVM, and CRIO for data sets Cancer, Liver, Diabetes, and Heart.

Data	LR	NN	CTreeU	CTreeM	GBM	TreeNet™	SVM	CRIO
Cancer	0.007	0.357	0.092	0.066	0.546	[2.725, 32.39]	0.013	0.290
Liver	0.003	0.443	0.070	0.054	0.606	[3.618, 37.73]	0.031	0.065
Diabetes	0.004	0.387	0.216	0.185	0.608	[3.782, 35.72]	0.071	0.250
Heart	0.023	0.301	0.233	0.835	1.642	[5.071, 36.34]	0.021	0.330

5.2. Regression Results

We tested the regression component of CRIO on three real data sets found on the UCI data repository and Friedman's (1999b) generated data set. We compared the results to linear regression, neural networks with radial basis functions and generalized regression (via MATLAB's® neural networks toolbox), tree-boosting methods GBM and Treenet™ described in §5.1, and MARS (via R's "polymars" package).

Similar to the experiments on the classification data sets in §5.1, each of the regression data was split into three parts, with 50%, 30%, and 20% of the data used for training, validating, and testing, respectively. The assignment to each set was done randomly, and the process was repeated 10 times.

We used the validation set for CRIO to fine-tune the value of parameter K. In all cases, we solved the mixed-integer programming problem (17) and set the parameters M, L, and ρ to 10,000, 10, and 0.01, respectively. In neural networks, we use the validation set to select the appropriate model (radial basis function versus generalized regression), adjust the number of epochs, the number of layers, the spread constant, and the accuracy parameter. For GBM and TreeNet™, we used the validation set to select the best shrinkage value and maximum number of tree iterations. In MARS, we used the validation set to choose the appropriate maximum number of basis functions and generalized cross validation (gcv) value.

Regression Data. The "Boston" data, with 13 explanatory variables and 506 observations, are the Boston housing data set with the dependent variable being the median value of houses in the suburbs of Boston. The "Abalone" data, with seven explanatory variables and 4,177 observations, attempt to predict the age of an abalone given its physiological measurements. The original "Abalone" data set had an additional categorical input variable, but we omitted it for our experimentations because not all methods were capable of treating such attributes. The "Auto" data, with five explanatory variables and 392 observations, are the auto-mpg data set that determines the miles-per-gallon fuel consumption of an automobile, given the mechanical attributes of the car. The "Friedman" data set is a generated data set resulting from using a random function generating method proposed by Friedman (1999b). All nine of the generated data sets consist of 10 explanatory variables.

Results. Tables 9 and 10 illustrate the mean-absolute error and mean-squared error, respectively, of linear least squares regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO, averaged over the 10 random partitions on the Friedman generated data sets. Tables 12 and 13 illustrate the mean-absolute error and mean-squared error, respectively, of the regression methods, averaged over the 10 random partitions on the "Boston," "Abalone," and "Auto" data sets. The numbers in parentheses are the corresponding standard deviations. Tables 11 and 14 illustrate the average running time, in terms of CPU seconds, of the methods for the Friedman generated data sets and the real data sets, respectively. Again, for TreeNet™, the lower and upper bounds of the total CPU seconds are shown because it does not currently report the exact total running time.

As in the classification case, we measure the performance of the various regression methods by their predictive ability and stability of their solutions. We measure the prediction accuracy using both mean-absolute errors and mean-squared errors between the predicted versus the actual response variable in the testing set, given that the mean-absolute error is used as the goodness-of-fit criterion for CRIO and mean-squared error is used as the goodness-of-fit criterion for linear least squares and the neural network model. GBM, TreeNet™, and MARS can be adjusted to deal with either loss criterion.

CRIO used $K = 2$ for all data sets, neural networks always used radial basis functions as the preferred model with just one layer of nodes, GBM used shrinkage value of 0.01 and 3,000 tree iterations, and TreeNet™ used shrinkage of 0.01 and 4,000 tree iterations.

Table 14 shows that CRIO has relatively reasonable average running time as the other methods for the smaller data sets, but its run time explodes for the larger "Abalone" data set. Unlike the classification component of CRIO, the regression component experiences a dramatic increase in run time with larger data sets mainly due to the M parameter in models (15) and (17). Because a tight estimate of this "big-M" parameter cannot be determined a priori, the large value of M seriously hampers the efficiency of the integer programming solver.

Clearly, the performance of the regression component of CRIO is not as convincing as the classification component, and more testing and modifications are needed. Namely, we would need to make improvements to the model (e.g., enforce continuity in the boundaries and find stronger approximations of the parameter M) and run more computational experiments before making any conclusions.

5.3. Discussion on Shortcomings

The computational experiments illustrated some benefits and shortcomings of CRIO compared to other existing methods in data mining and machine learning. The main advantage, at least in classification, is its prediction accuracy and relative stability. However, there are many shortcomings that need to be addressed.

As mentioned in the previous section, the regression component of CRIO clearly needs further development in its mathematical model and computational experimentations. Its main weakness arises from the discontinuity of the regression line at the boundaries of the polyhedral regions. This is not the case of MARS, which maintains continuity of its regression function throughout the domain. Thus, CRIO's predictive performance of general continuous functions are significantly hampered by this shortcoming, as seen with the Friedman data set. Continuity can be imposed by modifications to the mixed-integer programming problem in the one-dimensional case, but the extension to higher

Table 9. Mean absolute error of models linear least square regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on Friedman generated data sets.

Friedman data	n	Linear least squares			Neural networks		
		Train	Validation	Test	Train	Validation	Test
F1	500	0.890 (0.040)	0.954 (0.045)	0.913 (0.066)	0.875 (0.045)	0.918 (0.054)	0.887 (0.061)
F2	500	0.939 (0.021)	0.963 (0.053)	1.034 (0.068)	0.923 (0.044)	0.942 (0.052)	1.011 (0.057)
F3	500	0.905 (0.025)	0.960 (0.056)	0.949 (0.076)	0.895 (0.032)	0.940 (0.046)	0.929 (0.086)
F4	1,000	0.944 (0.022)	0.941 (0.047)	0.966 (0.048)	0.938 (0.023)	0.931 (0.046)	0.957 (0.048)
F5	1,000	0.903 (0.017)	0.914 (0.024)	0.931 (0.038)	0.805 (0.020)	0.898 (0.020)	0.913 (0.034)
F6	1,000	0.931 (0.016)	0.939 (0.024)	0.940 (0.032)	0.936 (0.016)	0.936 (0.028)	0.935 (0.038)
F7	4,000	0.951 (0.012)	0.965 (0.020)	0.964 (0.028)	0.948 (0.012)	0.959 (0.020)	0.960 (0.028)
F8	4,000	0.944 (0.007)	0.964 (0.019)	0.958 (0.033)	0.940 (0.007)	0.959 (0.019)	0.953 (0.029)
F9	4,000	0.953 (0.012)	0.950 (0.016)	0.947 (0.027)	0.951 (0.012)	0.949 (0.017)	0.944 (0.027)
		GBM			TreeNet™		
F1	500	0.897 (0.044)	0.919 (0.055)	0.889 (0.061)	0.885 (0.047)	0.922 (0.054)	0.892 (0.059)
F2	500	0.924 (0.033)	0.942 (0.055)	1.017 (0.065)	0.905 (0.059)	0.942 (0.054)	1.013 (0.069)
F3	500	0.894 (0.044)	0.937 (0.05)	0.931 (0.081)	0.834 (0.099)	0.941 (0.049)	0.934 (0.085)
F4	1,000	0.938 (0.021)	0.931 (0.045)	0.960 (0.049)	0.904 (0.045)	0.934 (0.043)	0.966 (0.052)
F5	1,000	0.895 (0.021)	0.898 (0.022)	0.913 (0.034)	0.883 (0.030)	0.900 (0.021)	0.916 (0.036)
F6	1,000	0.927 (0.025)	0.937 (0.028)	0.940 (0.037)	0.922 (0.024)	0.936 (0.029)	0.938 (0.037)
F7	4,000	0.948 (0.013)	0.959 (0.021)	0.959 (0.028)	0.941 (0.016)	0.959 (0.021)	0.959 (0.029)
F8	4,000	0.941 (0.009)	0.959 (0.019)	0.953 (0.031)	0.936 (0.014)	0.958 (0.019)	0.953 (0.031)
F9	4,000	0.943 (0.014)	0.948 (0.017)	0.946 (0.027)	0.935 (0.014)	0.948 (0.017)	0.945 (0.028)
		MARS			CRIO		
F1	500	0.904 (0.041)	0.921 (0.055)	0.891 (0.065)	0.875 (0.046)	0.938 (0.086)	0.913 (0.070)
F2	500	0.949 (0.025)	0.942 (0.052)	1.011 (0.057)	0.915 (0.027)	0.968 (0.056)	1.032 (0.066)
F3	500	0.927 (0.0276)	0.944 (0.047)	0.934 (0.083)	0.888 (0.028)	0.970 (0.071)	0.961 (0.074)
F4	1,000	0.949 (0.019)	0.931 (0.047)	0.957 (0.048)	0.931 (0.019)	0.943 (0.042)	0.965 (0.050)
F5	1,000	0.903 (0.017)	0.900 (0.024)	0.915 (0.033)	0.887 (0.015)	0.915 (0.021)	0.926 (0.033)
F6	1,000	0.942 (0.018)	0.938 (0.027)	0.936 (0.038)	0.921 (0.018)	0.944 (0.030)	0.945 (0.033)
F7	4,000	0.949 (0.012)	0.958 (0.021)	0.958 (0.028)	0.947 (0.012)	0.963 (0.020)	0.962 (0.028)
F8	4,000	0.943 (0.007)	0.958 (0.019)	0.957 (0.030)	0.940 (0.007)	0.963 (0.019)	0.956 (0.031)
F9	4,000	0.954 (0.012)	0.948 (0.016)	0.943 (0.027)	0.951 (0.013)	0.949 (0.016)	0.947 (0.027)

Note. The corresponding standard deviations are in parentheses.

dimensions is not evident with the current model. Thus, as it currently stands, if the underlying function is continuous, perhaps a continuous model would be more appropriate than CRIO.

Another apparent challenge for CRIO is maintaining reasonable computation time. Compared to heuristic-based methods in data mining such as neural networks and classification trees, CRIO has a much longer running time for larger data sets due to its use of integer programming. However, CRIO did have faster running times compared to the tree-boosting methods in almost all cases, and CRIO was faster than SVM in certain problems. The implementation of CRIO can be improved to speed up its running time. As mentioned earlier, it can implement a tailored quadratic programming solver for solving the SVM subproblems as done in all implementations of SVMs. The integer programming models for both classification and regression can be tailored by "implicitly" branching on the integer variables (Bienstock 1996, Bertsimas and Shioda 2004). Such an implementation will also eliminate the need for the "big-M" constraints in problems (15) and (17) that can significantly hamper the computation time of integer programming problems. In addition, because a provably optimal solution is not critical in this context, we can prematurely terminate the branch-and-bound procedure, say at 5% to 10% relative optimality gap or by a time limit. However, even with all these improvements, CRIO is not going to beat methods such as classification trees with respect to time. Thus, this method would be appropriate only for those who value prediction accuracy over computation time, which might be the case in areas of medical and genetic research.

The third major shortcoming of CRIO is its lack of interpretability, such as an ANOVA interpretation. This is a weakness that is shared by neural networks and SVMs as well. Unfortunately, there is not much that can be done to improve this problem for CRIO. Thus, if decision rules or variable importance information are vital to the data mining application, tools such as classification trees and tree-boosting methods would be more suitable. However, SVM is a very popular classification technique in machine learning even with this shortcoming. Thus, CRIO may find a similar audience that might find its classification accuracy more valuable. Also, CRIO is able to handle categorical

Table 10. Mean squared error of models linear least square regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on Friedman generated data sets.

Friedman data	n	Linear least squares			Neural networks		
		Train	Validation	Test	Train	Validation	Test
F1	500	1.299 (0.135)	1.485 (0.159)	1.370 (0.227)	1.245 (0.151)	1.389 (0.174)	1.303 (0.193)
F2	500	1.353 (0.057)	1.428 (0.127)	1.619 (0.165)	1.300 (0.126)	1.355 (0.129)	1.544 (0.119)
F3	500	1.286 (0.087)	1.466 (0.154)	1.498 (0.237)	1.245 (0.095)	1.401 (0.130)	1.442 (0.250)
F4	1,000	1.446 (0.053)	1.426 (0.111)	1.515 (0.124)	1.426 (0.066)	1.403 (0.119)	1.489 (0.122)
F5	1,000	1.342 (0.056)	1.378 (0.073)	1.416 (0.095)	1.295 (0.069)	1.334 (0.069)	1.356 (0.082)
F6	1,000	1.389 (0.042)	1.421 (0.081)	1.423 (0.105)	1.387 (0.045)	1.401 (0.083)	1.396 (0.102)
F7	4,000	1.442 (0.035)	1.471 (0.067)	1.482 (0.086)	1.431 (0.036)	1.455 (0.068)	1.469 (0.088)
F8	4,000	1.427 (0.014)	1.498 (0.051)	1.482 (0.074)	1.412 (0.015)	1.485 (0.054)	1.469 (0.069)
F9	4,000	1.439 (0.037)	1.432 (0.047)	1.427 (0.072)	1.433 (0.037)	1.428 (0.047)	1.417 (0.073)
		GBM			TreeNet™		
F1	500	1.32 (0.158)	1.388 (0.177)	1.308 (0.198)	1.305 (0.193)	1.390 (0.174)	1.304 (0.195)
F2	500	1.367 (0.062)	1.358 (0.13)	1.551 (0.128)	1.279 (0.167)	1.359 (0.128)	1.577 (0.147)
F3	500	1.308 (0.082)	1.402 (0.128)	1.442 (0.248)	1.161 (0.204)	1.396 (0.126)	1.447 (0.248)
F4	1,000	1.449 (0.061)	1.403 (0.121)	1.492 (0.122)	1.395 (0.100)	1.408 (0.120)	1.500 (0.122)
F5	1,000	1.331 (0.054)	1.335 (0.07)	1.358 (0.084)	1.287 (0.085)	1.335 (0.068)	1.372 (0.086)
F6	1,000	1.398 (0.048)	1.402 (0.083)	1.401 (0.102)	1.397 (0.034)	1.406 (0.085)	1.405 (0.101)
F7	4,000	1.434 (0.035)	1.451 (0.07)	1.464 (0.09)	1.430 (0.043)	1.452 (0.070)	1.465 (0.091)
F8	4,000	1.421 (0.017)	1.483 (0.053)	1.467 (0.071)	1.414 (0.021)	1.483 (0.054)	1.468 (0.070)
F9	4,000	1.419 (0.034)	1.422 (0.047)	1.42 (0.074)	1.399 (0.031)	1.422 (0.048)	1.416 (0.073)
		MARS			CRIO		
F1	500	1.350 (0.142)	1.399 (0.180)	1.313 (0.205)	1.313 (0.153)	1.419 (0.232)	1.356 (0.234)
F2	500	1.383 (0.065)	1.358 (0.129)	1.545 (0.119)	1.347 (0.065)	1.445 (0.144)	1.617 (0.155)
F3	500	1.339 (0.079)	1.417 (0.135)	1.451 (0.245)	1.301 (0.091)	1.498 (0.190)	1.534 (0.232)
F4	1,000	1.467 (0.052)	1.403 (0.120)	1.489 (0.122)	1.451 (0.053)	1.444 (0.103)	1.512 (0.137)
F5	1,000	1.339 (0.053)	1.338 (0.074)	1.359 (0.080)	1.319 (0.051)	1.375 (0.064)	1.391 (0.085)
F6	1,000	1.411 (0.046)	1.406 (0.081)	1.401 (0.103)	1.400 (0.048)	1.439 (0.088)	1.438 (0.108)
F7	4,000	1.439 (0.036)	1.451 (0.069)	1.463 (0.090)	1.439 (0.035)	1.462 (0.068)	1.474 (0.089)
F8	4,000	1.422 (0.015)	1.482 (0.053)	1.466 (0.070)	1.419 (0.017)	1.493 (0.054)	1.473 (0.072)
F9	4,000	1.442 (0.037)	1.424 (0.046)	1.415 (0.073)	1.441 (0.039)	1.428 (0.047)	1.427 (0.071)

Note. The corresponding standard deviations are in parentheses.

variables unlike SVMs, which might be an additional benefit in certain applications.

6. Conclusions

CRIO represents a new approach to solving classification and regression problems. The key components of CRIO include (1) clustering methods to reduce dimensionality; (2) nonlinear transformations of the variables to improve predictive power; (3) mixed-integer optimization methods to simultaneously group points together and eliminate outlier data; and (4) continuous optimization methods (linear and quadratic optimization) to represent groups by polyhedral regions.

Table 11. Average CPU time of linear least square regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on the Friedman generated data sets.

Data	LLS	NN	GBM	TreeNet™	MARS	CRIO
F1	0	0.159	0.145	[3.427, 37.97]	0.476	6.389
F2	0	0.146	0.122	[4.604, 38.93]	0.421	6.310
F3	0	0.171	0.155	[3.483, 37.84]	0.54	6.620
F4	0	0.298	0.168	[4.792, 38.10]	1.109	44.36
F5	0.001	0.290	0.165	[5.412, 38.44]	1.25	52.30
F6	0	0.284	0.177	[5.341, 38.36]	1.45	59.35
F7	0.001	4.711	0.511	[16.25, 42.78]	3.525	3,307
F8	0.001	4.701	0.365	[16.35, 42.79]	3.778	3,029
F9	0.001	4.671	0.894	[15.37, 42.25]	1.529	3,181

While clustering, nonlinear transformations, and continuous optimization methods have been used for these problems before, we believe that the key contribution of CRIO is its use of mixed-integer optimization methods. In contrast to the heuristic character of decision tree methods, CRIO produces splits of the explanatory variable space into disjoint regions in a globally optimal manner.

While more testing is clearly needed, we believe that the preliminary evidence on generated and widely circulated real data sets is encouraging because CRIO has matched and often outperformed current popular methods on these data sets. More generally, we hope that these encouraging results will motivate the statistics, data mining, and machine learning community to re-examine integer optimization methods as a viable tool in statistical computing.

Table 12. Mean absolute error of models linear regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on Boston, Abalone, and Auto data sets.

Real data			Linear least squares			Neural networks		
Data set	n	d	Train	Validation	Test	Train	Validation	Test
Boston	506	13	3.278 (0.233)	3.424 (0.232)	3.489 (0.325)	2.023 (0.075)	2.808 (0.283)	2.884 (0.253)
Abalone	4,177	7	1.619 (0.027)	1.642 (0.039)	1.667 (0.046)	1.572 (0.025)	1.598 (0.052)	1.611 (0.060)
Auto	392	7	2.532 (0.079)	2.533 (0.153)	2.601 (0.224)	1.673 (0.048)	2.494 (0.174)	2.698 (0.408)
			GBM			TreeNet™		
Boston	506	13	1.905 (0.140)	2.593 (0.227)	2.642 (0.175)	2.090 (0.124)	2.728 (0.268)	2.839 (0.239)
Abalone	4,177	7	1.611 (0.028)	1.696 (0.044)	1.714 (0.036)	0.524 (0.015)	0.547 (0.048)	0.554 (0.057)
Auto	392	7	1.639 (0.093)	2.051 (0.192)	2.083 (0.138)	1.642 (0.083)	2.070 (0.174)	2.092 (0.156)
			MARS			CRIO		
Boston	506	13	2.179 (0.231)	2.693 (0.195)	2.706 (0.252)	2.120 (0.119)	2.606 (0.226)	2.625 (0.181)
Abalone	4,177	7	1.507 (0.021)	1.563 (0.032)	1.575 (0.039)	1.482 (0.027)	1.513 (0.040)	1.536 (0.037)
Auto	392	7	1.994 (0.139)	2.088 (0.150)	2.152 (0.158)	1.856 (0.138)	2.055 (0.231)	2.095 (0.094)

Note. The corresponding standard deviations are in parentheses.

Table 13. Mean squared error of models linear least square regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on Boston, Abalone, and Auto data sets.

Real data			Linear least squares			Neural networks		
Data set	n	d	Train	Validation	Test	Train	Validation	Test
Boston	506	13	23.374 (3.292)	27.778 (5.180)	26.032 (6.503)	7.812 (0.068)	16.534 (3.680)	16.442 (3.773)
Abalone	4,177	8	4.959 (0.189)	5.316 (0.345)	5.291 (0.463)	4.778 (0.102)	4.941 (0.293)	4.942 (0.449)
Auto	392	7	11.437 (0.609)	11.334 (1.387)	12.057 (1.920)	5.052 (0.065)	12.257 (1.971)	15.988 (5.872)
			GBM			TreeNet™		
Boston	506	13	7.281 (1.715)	15.496 (4.055)	17.80 (4.090)	7.794 (1.233)	15.784 (4.601)	17.772 (3.990)
Abalone	4,177	7	5.065 (0.160)	5.692 (0.277)	5.657 (0.435)	0.655 (0.089)	0.973 (0.148)	0.929 (0.181)
Auto	392	7	6.079 (0.878)	8.267 (1.925)	8.752 (0.935)	4.985 (0.564)	8.052 (1.762)	8.252 (1.375)
			MARS			CRIO		
Boston	506	13	8.486 (2.114)	15.572 (3.490)	16.756 (5.086)	10.685 (1.785)	13.875 (3.393)	14.226 (4.925)
Abalone	4,177	8	4.347 (0.156)	4.786 (0.253)	4.778 (1.342)	4.571 (0.145)	4.785 (0.267)	4.794 (0.381)
Auto	392	7	7.457 (0.975)	8.226 (1.139)	8.912 (1.427)	7.300 (1.408)	8.641 (1.463)	9.872 (1.930)

Note. The corresponding standard deviations are in parentheses.

Table 14. Average CPU time of linear least square regression (LLS), neural networks (NN), GBM, TreeNet™, MARS, and CRIO on the Boston, Abalone, and Auto data sets.

Data	LLS	NN	GBM	TreeNet™	MARS	CRIO
Boston	0.000	1.092	1.369	[8.813, 36.671]	0.358	0.537
Abalone	0.000	13.103	6.472	[26.67, 43.609]	7.661	148.36
Auto	0.000	0.824	0.347	[8.403, 39.679]	0.182	0.132

Endnotes

1. We use the word *group* to mean a collection of points and the word *region* to mean a polyhedron that contains a group.

2. Note that in classification, the regions P_k also do not form a partition of $\mathbb{R}^d$, but if a point $\mathbf{x}_0$ belongs to any of the regions P_k, we classify it as Class 1; and if it does not belong to any of the regions P_k, we classify it as Class 0.

3. Given a collection $\mathcal{F}$ of points, the *statistical* distance between points $\mathbf{x} \in \mathcal{F} \subseteq \mathbb{R}^d$ and $\mathbf{z} \in \mathcal{F} \subseteq \mathbb{R}^d$ is defined as

$$d_{\mathcal{F}}(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{j=1}^{d} \frac{(x_j - z_j)^2}{s_j^2}},$$

where s_j^2 is the sample variance of the jth coordinate of all points in $\mathcal{F}$. Statistical distance is a widely accepted metric in data mining to measure the proximity of points.

4. SVMfu is a freeware developed by Ryan Rifkin from MIT that implements SVMs with linear, polynomial, and Gaussian kernels. It can be downloaded from http://five-percent-nation.mit.edu/SvmFu/.

5. CRUISE is a freeware developed by Hyung Joong Kim and Wei-Yin Loh (http://www.stat.wisc.edu/∼loh/cruise.

html). We also tested CART from Salford Systems, but CRUISE was comparable or outperformed the former in every case. Thus, we present only the performance of CRUISE in this paper.

6. CPLEX 8.0 is a product of ILOG (http://www.ilog.com/products/cplex/).

7. TreeNet™ reports the mean CPU seconds for every 10 trees. Because only the first two decimal places are shown (often being 0.00), we could not compute the exact total CPU seconds. However, we can clearly find the lower and upper bound of the total running time from this information.

Acknowledgments

The authors thank the referees for their recommendations and guidance. The research of both authors was partially supported by the Singapore-MIT Alliance. In addition, R. Shioda was funded by the Discovery grant from NSERC.

References

Arthanari, T. S., Y. Dodge. 1993. *Mathematical Programming in Statistics*. Wiley Classics Library Edition, Wiley–Interscience, New York.

Bennett, K. P., J. A. Blue. 1984. Optimal decision trees. R.P.I. Math Report 214, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY.

Bertsimas, D., R. Shioda. 2004. An algorithm for cardinality constraint quadratic optimization. In review.

Bienstock, D. 1996. Computational study of families of mixed-integer quadratic programming problems. *Math. Programming* **74** 121–140.

Bousquet, O., A. Elisseeff. 2002. Stability and generalization. *J. Machine Learn. Res.* **2** 445–498.

Breiman, L., J. Friedman, R. Olshen, C. Stone. 1984. *Classification and Regression Trees*. Wadsworth International, Belmont, CA.

Friedman, J. 1991. Multivariate adaptive regression splines. *Ann. Statist.* **19**(1) 1–67.

Friedman, J. H. 1999a. Stochastic gradient boosting. http://www-stat.stanford.edu/~jhf/ftp/stobst.ps.

Friedman, J. H. 1999b. Greedy function approximation: A gradient boosting machine. http://www-stat.stanford.edu/~jhf/ftp/trebst.ps.

Friedman, J. H. 2002. Getting started with MART in R. http://www-stat.stanford.edu/~jhf/r-mart/tutorial/tutorial.pdf.

Hastie, T., R. Tibshirani, J. Friedman. 2001. *The Elements of Statistical Learning*. Springer-Verlag, New York.

ILOG CPLEX 7.1 Reference Guide. 2001. ILOG CPLEX Division, Incline Village, NV.

Johnson, R. A., D. W. Wichern. 1998. *Applied Multivariate Statistical Analysis*, 4th ed. Prentice-Hall, Englewood Cliffs, NJ.

Kim, H., W. Y. Loh. 2000. CRUISE User Manual. Technical Report 989, Department of Statistics, University of Wisconsin, Madison, WI.

Kim, H., W. Y. Loh. 2001. Classification tree with unbiased multiway splits. *J. Amer. Statist. Assoc.* **96** 598–604.

Loh, W. Y., Y. Shih. 1997. Split selection methods for classification trees. *Statistica Sinica* **7** 815–840.

Mangasarian, O. L. 1965. Linear and nonlinear separation of patterns by linear programming. *Oper. Res.* **13**(3) 444–452.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA.

Ridgeway, G. 2005. Generalized boosted models: A guide to the gbm package. http://i-pensieri.com/gregr/papers/gbm-vignette.pdf.

Rifkin, R. 2002. Everything old is new again: A fresh look at historical approaches in machine learning. Ph.D. thesis, Electrical Engineering and Comuter Science and Operations Research, Massachusetts Institute of Technology, Cambridge, MA.

Rousseeuw, P. J., A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. Wiley, New York.

Ryan, T. P. 1997. *Modern Regression Methods*. Wiley Series in Probability and Statistics, John Wiley & Sons, New York.

Vapnik, V. 1999. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.