Classification and Regression via Integer Optimization

Dimitris Bertsimas * Romy Shioda †

September, 2002

Abstract

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 (**C**lassification and **R**egression via **I**nteger **O**ptimization). CRIO separates data points in different polyhedral regions. In classification each region is assigned a class, while in regression each region has its own distinct regression coefficients. Computational experimentation with real data sets shows that CRIO is comparable to and often outperforms the current leading methods in classification and regression. Specifically, in five widely circulated real classification data sets, CRIO had up to 13% and 10% higher classification accuracy than CART and support vector machines, respectively, and in three widely circulated real regression data sets, CRIO had up to 22% and 10% lower mean absolute regression error than CART and MARS, respectively. We hope that these results illustrate the potential for significant impact of integer optimization methods on computational statistics and data mining.

1 Introduction

In the last twenty 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) which 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, the state of the art methods for these problems include:

*Boeing Professor of Operations Research, Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E53-363, Cambridge, MA 02139, dbertsim@mit.edu. The research of the author was partially supported by the Singapore-MIT alliance.

†Operations Research Center, Massachusetts Institute of Technology, Cambridge, Mass. 02139. romy@mit.edu. The research of the author was partially supported by the Singapore-MIT alliance.

1. Classification And Regression Trees (CART) [3] for both classification and regression. CART recursively splits the data along a variable into hyper-rectangular regions. After this forward propagation is complete, backward propagation (or pruning) is performed in order to prevent over-fitting the model to the data set. Its main shortcoming is its fundamentally greedy approach. [2] finds a globally optimal decision tree, but the structure of the tree needs to be prefixed. C5.0 [9] is another popular method which is similar in spirit to CART.

2. Multivariate Adaptive Regression Splines (MARS) [4] 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) [5], [8], [13], for classification. In its simplest form, SVM separates points of different classes by a single hyperplane or a simple nonlinear function. More sophisticated SVM methods map the given points via a nonlinear transformation to a higher dimensional space and use continuous optimization methods to separate the points of different classes.

CART, MARS and C5.0 belong to the category of decision trees, are heuristic in nature and are closer to the tradition of artificial intelligence. SVM belongs to the category of separating hyperplane methods, utilize formal continuous optimization techniques (linear and quadratic optimization) and are closer to the tradition of mathematical programming. It is fair to say that all of 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 of 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 thirty years (see for example [1]), 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 ([1] contains several integer optimization formulations), the belief was formed in the early 1970's 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. Specifically, we have compared CRIO with CART and SVM for classification problems in five widely circulated real data sets. In all five data sets we tested, CRIO outperformed CART, with prediction accuracy up to 13% higher in the testing set. CRIO outperformed SVM in two of these data sets, with prediction accuracy up to 10% higher, performed similarly to SVM in two other data sets, and was outperformed by SVM in one of the data sets. Most importantly, the impact of CRIO was greatest in the data sets that tend to be difficult to classify for the current methods For regression problems, CRIO outperformed both CART and MARS in all three real data sets we tested. CRIO had up to 22% and 10% lower mean absolute error compared to that of CART and MARS, respectively, and 39% and 29% lower mean squared error, respectively. Thus, CRIO was able to give good predictions for all the data sets in reasonable computational times.

The structure of the paper is as follows. In Section 2, we give an intuitive presentation of the geometric ideas of our approach for both classification and regression to facilitate understanding. In Sections 3 and 4, we develop CRIO for classification and for regression, respectively. In Section 5, we compare CRIO with CART, SVM, and MARS on 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) in order to facilitate understanding of the mathematical presentation in Sections 3 and 4.

2.1 The Geometry of the Classification Approach

Given n data points $(\boldsymbol{x}_i, y_i)$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$ and $i = 1, \dots, n$, 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.

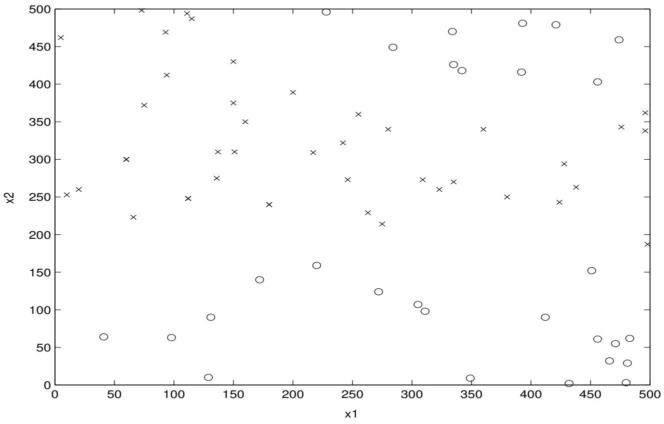

Figure 1: A given set of training data. Class 0 points are represented with an "o" and Class 1 points are represented by an "x".

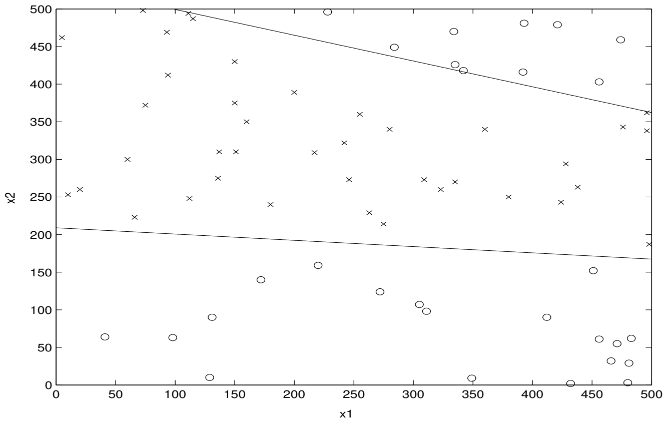

Figure 2: The output of CRIO.

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[2]. Clearly, we want to keep K small (typically 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 $\boldsymbol{x}_0$ as Class 1, if it is contained in any of the K polyhedra, and as Class 0, otherwise. In order to reduce the dimension of the mixed-integer optimization model and thus

[1] We use the word *group* to mean a collection of points, while we use the word *region* to mean a polyhedron that contains a group.

[2] We group Class 1 points instead of Class 0 points, without loss of generality, throughout the paper.

reduce computation time, we preprocess the data so that points form small clusters using a clustering algorithm (see Figure 4).

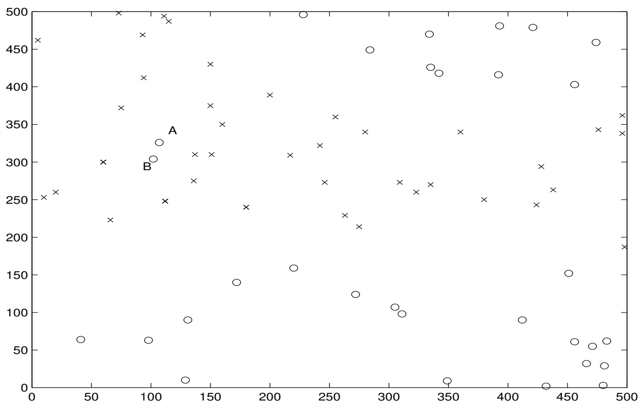

Figure 3: Illustration of outliers in the classification data. Points A and B are Class 0 outliers.

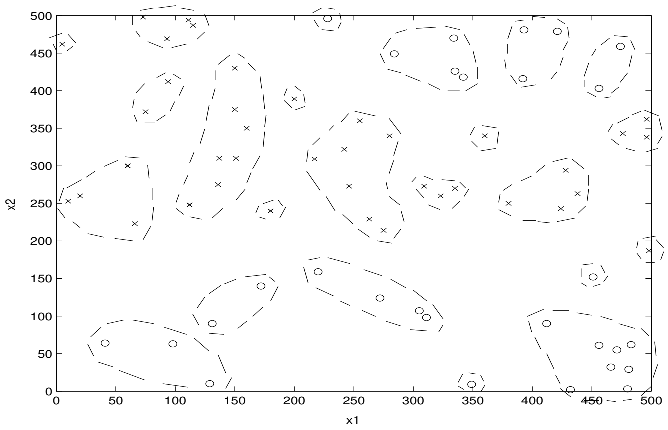

Figure 4: Data points are grouped into small clusters in $\boldsymbol{x}$-space.

2.2 The Geometry of the Regression Approach

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

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 [11], [12]). This procedure can often be deceiving since 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 $\boldsymbol{x}_0$ as we outlined earlier. CRIO does not in fact create

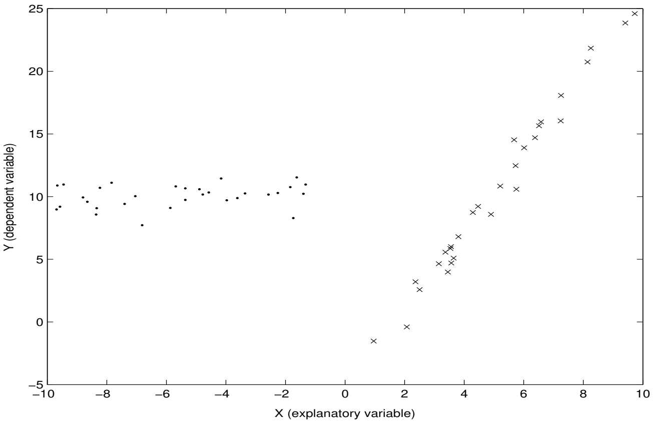

Figure 5: A given set of training data for regression with $d = 1$.

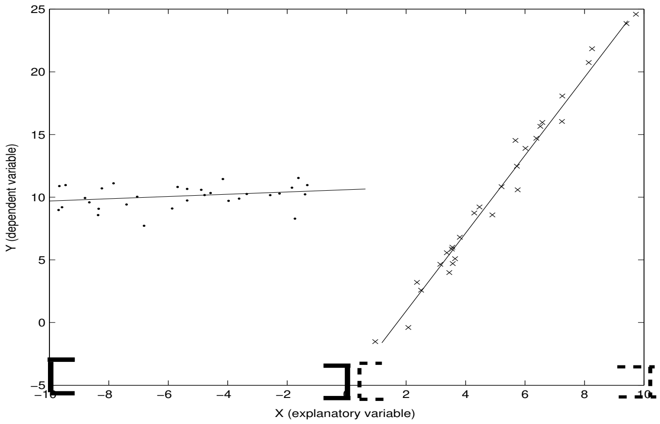

Figure 6: The output of CRIO, where the regression coefficient or slope is different for the two regions.

a partition of $\mathbb{R}^d$, so it is possible that a new point $\boldsymbol{x}_0$ may not belong to any of the P_k's [3]. 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 $\hat{y}_0 = \boldsymbol{\beta}_r' \boldsymbol{x}_0$. Similarly to the classification model, we preprocess the 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).

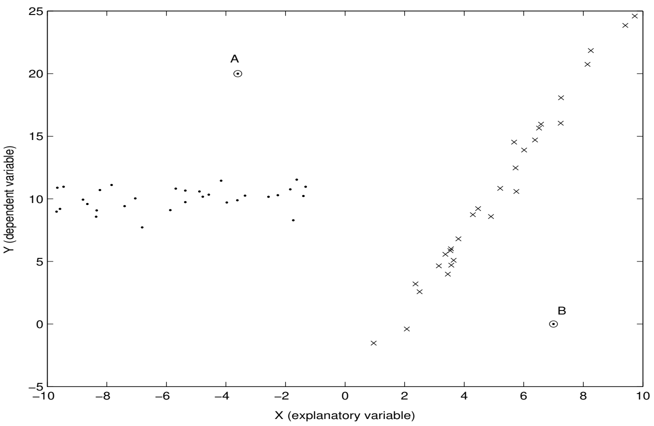

Figure 7: Illustration of outliers in regression data.

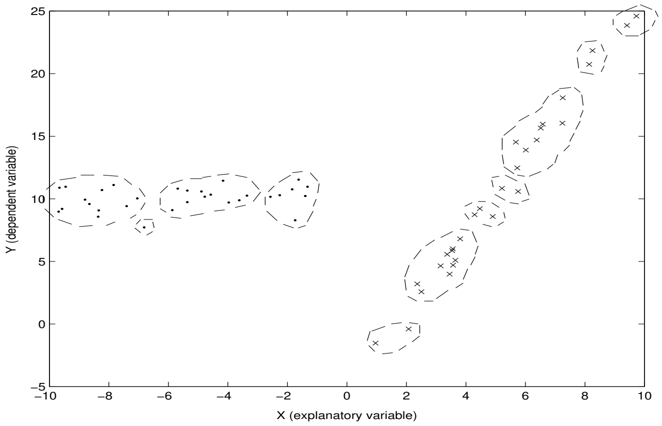

Figure 8: Data points clustered in $(\boldsymbol{x}, y)$-space.

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 op-

[3] Note that in classification, the regions P_k also do not form a partition of $\mathbb{R}^d$, but if a point $\boldsymbol{x}_0$ 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.

timization 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; (c) Solve continuous optimization problems (quadratic optimization problems for classification and linear optimization problems for regression) that assign groups to polyhedral regions.

3 CRIO for Classification

In this section, we present our methods first for binary and then multiple class classification problems. As outlined in Section 2.1, we first assign both Class 1 and Class 0 points into clusters (Section 3.2), then assign Class 1 clusters to groups via the use of a mixed-integer optimization model that also detects outliers (Section 3.3). Section 3.4 presents the methodology of assigning polyhedra to groups, and Section 3.5 summarizes the overall algorithm for binary classification. Section 3.6 extends these methods to multiple-class classification problems. We start by presenting the basic mixed-integer optimization model in Section 3.1, which forms the basis of our final approach.

3.1 The Mixed-Integer Optimization Model

The training data consists of n observations $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ 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 $\boldsymbol{x}_i^0$, $i = 1, \ldots, m_0$, and $\boldsymbol{x}_i^1$, $i = 1, \ldots, m_1$, respectively. Let $M_0 = \{1, \ldots, m_0\}$, $M_1 = \{1, \ldots, m_1\}$ and $\overline{K} = \{1, \ldots, K\}$.

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 of Class 1 points that are in Group k, where $\bigcup_{k \in \overline{K}} G_k = M_1$ and $\bigcap_{k \in \overline{K}} G_k = \emptyset$. Thus, we require that the following system is infeasible, for all $i \in M_0$ and $k \in \overline{K}$:

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

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

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

We consider the optimization problem:

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

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., $\boldsymbol{x}_i^0$ is in the convex hull of the points $\boldsymbol{x}_j^1$, $j \in G_k$. We add the constraint $\epsilon \le 1$ to prevent unbounded solutions. Note that Problem (3) seeks to find a hyperplane $\boldsymbol{p}'\boldsymbol{x} + q = 0$ that separates point $\boldsymbol{x}_i^0$ from all the Class 1 points in Group k.

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

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

In order to determine if we can assign Class 1 points into K groups such that $z > 0$, we define decision variables for $k \in \overline{K}$ and $j \in M_1$:

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

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

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

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

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

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:

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

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 [7]) 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:

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

where C_r and C_s are 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 can not since 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 δ* = 0 then
6:       k := k + 1
7:    else
8:       Merge Clusters r and s.
9:       K := K − 1, k := 0.
10:   k := k + 1.
```

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[4] 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 $\overline{K}_0 = \{1, \ldots, K_0\}$ and $\overline{K}_1 = \{1, \ldots, K_1\}$. Let C^0_t, $t \in \overline{K}_0$, be the set of indices of Class 0 points in Cluster t and C^1_r, $r \in \overline{K}_1$, be the set of indices of Class 1 points in Cluster r. We define the following binary variables for $r \in \overline{K}_1$ and $k \in \overline{K}$:

$$a_{k,r} = \begin{cases} 1, & \text{if Cluster } r \text{ is assigned to Group } k, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

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

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

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.

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

$$\begin{aligned}
\text{maximize} \quad & \delta & & \qquad (9)\\
\text{subject to} \quad & \boldsymbol{p}_{k,t}'\boldsymbol{x}_i^0 + q_{k,t} \le -\delta, & & i \in C_t^0; t \in \overline{K}_0; k \in \overline{K},\\
& \boldsymbol{p}_{k,t}'\boldsymbol{x}_j^1 + q_{k,t} \ge a_{k,r} - 1, & & t \in \overline{K}_0; r \in \overline{K}_1; k \in \overline{K}; j \in C_r^1,\\
& \sum_{k=1}^{K} a_{k,r} = 1, & & r \in \overline{K}_1,\\
& a_{k,r} \in \{0,1\}.
\end{aligned}$$

If $a_{k,r} = 1$ in an optimal solution, then all Class 1 points in Cluster r are assigned to Group k, i.e., $G_k = \bigcup_{\{r|a_{k,r}=1\}} C_r^1$.

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 Section 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 only return a trivial solution where $\delta = 0$, $\boldsymbol{p}_{k,i} = \boldsymbol{0}$, $q_{k,i} = 0$ and $a_{k,j}$'s 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 1, 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_i^0 and ϵ_j^1 are violation margins corresponding to Class 0 and Class 1 points, respectively. The model is as follows:

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{i=1}^{m_0} \epsilon_i^0 + \sum_{j=1}^{m_1} \epsilon_j^1 & \\
\text{subject to} & \boldsymbol{p}'_{k,t}\boldsymbol{x}_i^0 + q_{k,t} \le -1 + \epsilon_i^0, & i \in C_t^0; t \in \overline{K}_0; k \in \overline{K}, \\
& \boldsymbol{p}'_{k,t}\boldsymbol{x}_j^1 + q_{k,t} \ge -M + (M+1)a_{k,r} - \epsilon_j^1, & t \in \overline{K}_0; k \in \overline{K}; r \in \overline{K}_1; j \in C_r^1, \\
& \displaystyle\sum_{k=1}^{K} a_{k,r} = 1, & r \in \overline{K}_1, \\
& a_{k,r} \in \{0,1\}, \quad \epsilon_i^0 \ge 0, \quad \epsilon_j^1 \ge 0, &
\end{array}
\tag{10}
$$

where M is a large positive constant.

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

$$P_k = \{\boldsymbol{x} \in \mathbb{R}^d | \ \boldsymbol{p}'_{k,t}\boldsymbol{x} \ge -q_{k,t}, k \in \overline{K}; t \in \overline{K}_0\}.$$

Motivated by the success of SVMs (see [13]), 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, since it leaves as much distance between the boundary and points of each class as possible.

Our goal is to find a hyperplane

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

for every Group k, $k \in \overline{K}$, of Class 1 points and for every Cluster t, $t \in \overline{K}_0$, 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, $d(\boldsymbol{x}, \boldsymbol{\pi}_{k,t}, \alpha_{k,t})$, between a point $\boldsymbol{x}$ and the hyperplane $\boldsymbol{\pi}'_{k,t}\boldsymbol{x} = \alpha_{k,t}$ is

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

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

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

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

$$P_k = \{\boldsymbol{x} \in \mathbb{R}^d |\ \boldsymbol{\pi}'_{k,t}\boldsymbol{x} \leq \alpha_{k,t},\ t \in \overline{K}_0\}. \tag{12}$$

The last step in our process is the elimination of redundant hyperplanes in the representation of polyhedron P_k given in Eq. (12). Figures 9 and 10 illustrate this procedure. We can check whether the constraint

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

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

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

We have only included the last constraint to prevent Problem (14) from becoming unbounded. If $w_{k,t_0} \leq \alpha_{k,t_0}$, then the 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:

for $k = 1$ to K **do**
    for $t_0 = 1$ to K_0 **do**
        Solve Problem (14).
        if $w_{k,t_0} \leq \alpha_{k,t_0}$ **then**
            Eliminate Constraint $\boldsymbol{\pi}'_{k,t_0}\boldsymbol{x} \leq \alpha_{k,t_0}$.

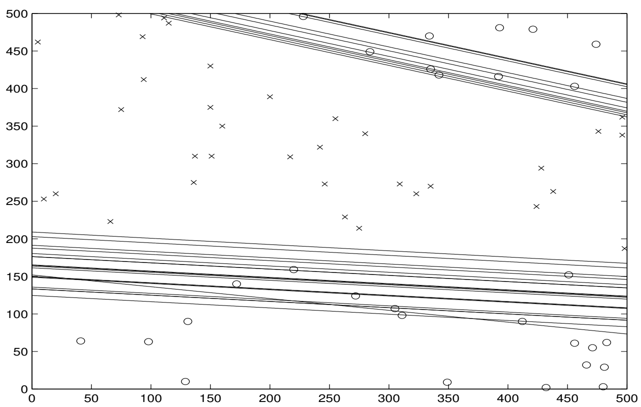

Figure 9: Before Eliminating Redundant Constraints.

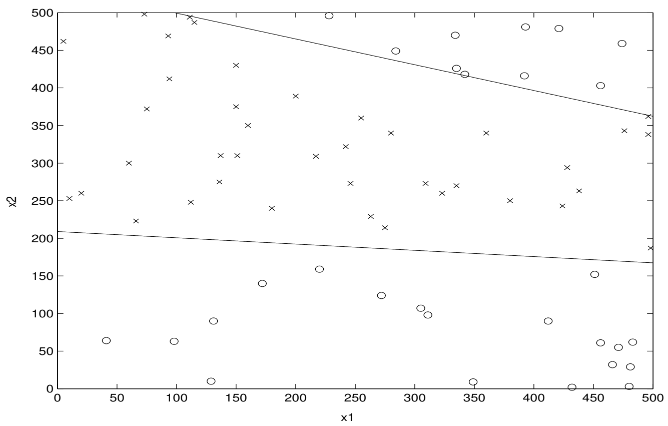

Figure 10: After Eliminating Redundant Constraints.

3.5 The Overall Algorithm for Classification

The overall algorithm for classification is as follows:

1. **Preprocessing**: Use the clustering algorithm outlined in Section 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 in the end of Section 3.4.

After CRIO determines the non-redundant 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.

3.6 Classification with Multiple Classes

We have only discussed binary classification problems so far, but the method can easily be extended to data sets with multiple classes by iteratively solving a binary classification problem. Suppose we have a data set with C classes, with N_i being the set of indices of class i points, $i = 1, 2, \dots, C$. The following algorithm classifies such a data set:

1: **for** $c = 1$ to C **do**

2: Let $M_1 = N_c$, and $M_0 = \bigcup_{i=c+1,\dots,C} N_i$.

3: Solve the binary classification problem using the method described in Section 3.5.

Thus, this algorithm finds polyhedra that separates Class 1 points from points of Class 2 through C, Class 2 points from points of Class 3 through C, etc. To classify a new point, we check whether it is contained in any Class 1 polyhedron, and if it does, we label the point Class 1; otherwise, we check whether it is contained in any Class 2 polyhedron. If it is not contained in any of the Class 1 through $C-1$ polyhedra, then we label it a Class C point. From computational experimentation, the method worked best when the classes are ordered in ascending cardinality, i.e., $|N_1| < |N_2| < \cdots < |N_C|$.

4 CRIO for Regression

In this section, we present in detail our approach for regression. For presentation purposes, we start in Section 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 Section 2.2, we first assign points to clusters (Section 4.2), then assign clusters to groups of points (Section 4.3), which we then represent by polyhedral regions P_k (Section 4.4). In Section 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 Section 4.6.

4.1 Assigning Points to Groups: An Initial Model

The training data consists of n observations $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We let $N = \{1, \ldots, n\}$, $\overline{K} = \{1, \ldots, K\}$ and M be a large positive constant. We define binary variables for $k \in \overline{K}$ and $i \in N$:

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

The mixed-integer optimization model is as follows:

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

From the first and second constraints, δ_i is the absolute error associated with point $\boldsymbol{x}_i$. If $a_{k,i} = 1$, then $\delta_i \geq (y_i - \boldsymbol{\beta}_k' \boldsymbol{x}_i)$, $\delta_i \geq -(y_i - \boldsymbol{\beta}_k' \boldsymbol{x}_i)$, and the minimization of δ_i sets δ_i equal to $|y_i - \boldsymbol{\beta}_k' \boldsymbol{x}_i|$. If $a_{k,i} = 0$, the right-hand-side of the first two constraints becomes negative, making them irrelevant since 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 Section 3.2, to cluster nearby $\boldsymbol{x}_i$ 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 $(\boldsymbol{x}, y)$ space, as opposed to just the $\boldsymbol{x}$ space, in order 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**

Find the points $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$, $i < j$, with minimum pairwise statistical distance. Let $l(i)$ and $l(j)$ be the indices of the clusters that $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$ currently belong to, respectively.

Add Cluster $l(j)$'s points to Cluster $l(i)$, i.e., $C_{l(i)} := C_{l(i)} \cup C_{l(j)}$.

Let the pairwise statistical distance between all the points in $C_{l(i)}$ be ∞.

$l = l - 1$.

In the clustering algorithm for classification problems (Section 3.2), we merged clusters who had centers of close proximity. However, in the present clustering algorithm, we merge clusters which contains 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 in order 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 in order 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, $l \in \overline{L} = \{1, \ldots, L\}$, be Cluster l, and denote $l(i)$ as $\boldsymbol{x}_i$'s cluster. Similarly to Problem (15), we define the following binary variables for $k \in \overline{K} \cup \{0\}$ and $l \in \overline{L}$:

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

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:

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{i=1}^{n} \delta_i & \\
\text{subject to} & \delta_i \geq (y_i - \boldsymbol{\beta}'_k \boldsymbol{x}_i) - M(1 - a_{k,l(i)}), & k \in \overline{K}; i \in N, \\
& \delta_i \geq -(y_i - \boldsymbol{\beta}'_k \boldsymbol{x}_i) - M(1 - a_{k,l(i)}), & k \in \overline{K}; i \in N, \\
& \displaystyle\sum_{k=0}^{K} a_{k,l} = 1, & l \in \overline{L}, \\
& \displaystyle\sum_{l=1}^{L} |C_l| a_{0,l} \leq \rho |N|, & \\
& a_{k,l} \in \{0,1\}, \quad \delta_i \geq 0, &
\end{array}
\tag{17}
$$

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 $\boldsymbol{x}_i$. If $a_{k,l(i)} = 1$, then $\delta_i \geq (y_i - \boldsymbol{\beta}'_k \boldsymbol{x}_i)$, $\delta_i \geq -(y_i - \boldsymbol{\beta}'_k \boldsymbol{x}_i)$, and the minimization of δ_i sets it equal to $|y_i - \boldsymbol{\beta}'_k \boldsymbol{x}_i|$. If $a_{k,l(i)} = 0$, the first two constraints become irrelevant, since 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., $G_k = \bigcup_{\{l | a_{k,l}=1\}} C_l$.

Problem (17) has KL binary variables as opposed to Kn binary variables in Problem (15). The number of clusters L controls the tradeoff between quality of the solution and efficiency of the computation. As L increases, the quality of the solution increases, but the efficiency of the computation decreases. In Section 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 may not be able to define disjoint regions of P_k that contains 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 Groups r, where $k < r$. We consider the following two linear optimization problems:

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in G_k} \epsilon_i + \sum_{l\in G_r} \epsilon_l \qquad (18) \\ \text{subject to} \quad & \boldsymbol{p}'_{k,r}\boldsymbol{x}'_i - q_{k,r} \le -1 + \epsilon_i, \quad i \in G_k, \\ & \boldsymbol{p}'_{k,r}\boldsymbol{x}_l - q_{k,r} \ge 1 - \epsilon_l, \quad l \in G_r, \\ & \boldsymbol{p}'_{k,r}\boldsymbol{e} \ge 1, \\ & \epsilon_i \ge 0, \quad \epsilon_l \ge 0, \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in G_k} \epsilon_i + \sum_{l\in G_r} \epsilon_l \qquad (19) \\ \text{subject to} \quad & \boldsymbol{p}'_{k,r}\boldsymbol{x}_i - q_{k,r} \le -1 + \epsilon_i, \quad i \in G_k, \\ & \boldsymbol{p}'_{k,r}\boldsymbol{x}_l - q_{k,r} \ge 1 - \epsilon_l, \quad l \in G_r, \\ & \boldsymbol{p}'_{k,r}\boldsymbol{e} \le -1, \\ & \epsilon_i \ge 0, \quad \epsilon_l \ge 0, \end{aligned}$$

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

Both Problems (18) and (19) find a hyperplane $\boldsymbol{p}'_{k,r}\boldsymbol{x} = q_{k,r}$ 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 $\boldsymbol{p}_{k,r} = \boldsymbol{0}$ and $q_{k,r} = 0$ for the optimal solution. Problem (18) sets the sum of the elements of $\boldsymbol{p}_{k,r}$ to be strictly positive and Problem (19) sets the sum of the elements of $\boldsymbol{p}_{k,r}$ to be strictly negative. Both of these problems need to be solved since we do not know *a priori* whether the sum of the elements of the optimal non-trivial $\boldsymbol{p}_{k,r}$ 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

$$P_k = \{\boldsymbol{x} |\ \boldsymbol{p}'_{k,i}\boldsymbol{x} \le q_{k,i}, i = 1, \ldots, k-1, \quad \boldsymbol{p}'_{k,i}\boldsymbol{x} \ge q_{k,i}, i = k+1, \ldots, K\}. \qquad (20)$$

After P_k is defined, we recompute $\boldsymbol{\beta}_k$ using all the points contained in P_k since 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

In order 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 $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d})'$ 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 sum of squared errors equal to $f_{j,1}$, (b) $(x_{i,j}^2, y_i)$, $i \in N$, with sum of squared errors equal to $f_{j,2}$, (c) $(\log x_{i,j}, y_i)$, $i \in N$, with sum of squared errors equal to $f_{j,3}$, (d) $(1/x_{i,j}, y_i)$, $i \in N$, with 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 Section 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 Eq. (20).

5. **Re-computation of $\boldsymbol{\beta}$s:** Once the polyhedra P_k are indentified, recompute $\boldsymbol{\beta}_k$ using only the points that belong in P_k.

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

5 Computational Results

In this section, we report on the performance of CRIO on several widely circulated real data sets. In Section 5.1, we present CRIO's performance on classification data sets and compared it to the performances of state of the art CART and SVM softwares. In Section 5.2, we present CRIO's performance on regression data sets and compared it to the performances of state of the art CART and MARS softwares.

5.1 Classification Results

We tested the classification component of CRIO on real data, and compared their prediction accuracy against the softwares CART[5] and SvmFu[6]. Each data set was split into three parts – the training set, the cross-validation set and the testing set. The training set, comprised of 50% of the data, was used to develop the model. The cross-validation set, composed of 30% of the data, was used to select the best values of K, K_0 and K_1. Finally, the remaining points are in the testing set, which ultimately decides the accuracy of the model. This set is put aside until the very end, after the parameters of the model are finalized. The assignment to each of the three sets was done randomly, and this process was repeated three times.

CART uses the cross-validation set for self-validating its initial decision tree built by the training set alone. The test set is classified using this final tree. For SvmFu, 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 cross-validation accuracy was chosen to classify the testing set.

We solved all optimization problems (mixed-integer, quadratic and linear) using CPLEX 7.1[7] [6] running in a Linux environment.

Classification Data

We tested our models on five real data sets found on the UCI data repository [8]. The "Cancer" data is from the Wisconsin Breast Cancer databases, with 682 data points and 9 explanatory variables. The "Liver" data is from BUPA Medical Research Ltd., with 345 data points and 6 explanatory variables. The "Diabetes" data is from the Pima Indian Diabetes Database, with 768 data points and 7 explanatory variables. The "Heart" data is from the SPECT heart database, where we combined the given training and testing set to get 349 data points and 44 explanatory variables. The "Iris" data determines the type of iris, given the plant's physical measurements, and has 150 data points, 4 explanatory variables and 3 classes. The "Cancer", "Liver", "Diabetes" and "Heart" involve binary classification, while the "Iris" data set involves classification with three classes.

---

[5] CART is a product of Salford Systems, http://www.salford-systems.com.

[6] 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/.

[7] CPLEX 7.1 is a product of ILOG http://www.ilog.com/products/cplex/

[8] http://www.ics.uci.edu/~mlearn/MLSummary.html

Results

Table 1 summarizes the classification accuracy of CART, SvmFu and CRIO for the five data sets. Each sub-panel in Table 1 corresponds to a data set. The columns labelled "Train", "Cross" and "Test" illustrate the percent of the training, cross-validation and testing set, respectively, that the models correctly classified. Each entry is the average of three independent runs.

In all data sets, CRIO was significantly more successful than CART, with prediction accuracy up to 13% higher in the testing set. CRIO outperformed SvmFu in the "Liver" and "Diabetes" data sets, with prediction accuracy up to 10% higher. In the "Cancer" and "Iris" data set CRIO and SvmFu had the same performance, and in the "Heart" data set SvmFu outperformed CRIO. Most importantly, the impact of CRIO was greatest in the data sets that tend to be difficult to classify for the current methods ("Liver" and "Diabetes"). CRIO runs in practical running times of under half a minute for all data sets on a personal workstation.

The parameter M of Problem (10) was set to 1000 for all of our experiments. Both the values of K_0 and K_1 ranged from 1 to 3 in "Cancer" and "Heart", 4 to 8 in "Iris" and 8 to 12 in "Liver" and "Diabetes". "Cancer","Heart" and "Iris" used $K = 1$ group for all three cases, "Liver" used up to $K = 2$ groups and "Diabetes" up to $K = 3$ groups. We feel that CRIO's ability, via mixed-integer optimization, to categorize the points in more than one region may explain the prediction accuracy of CRIO in the "Liver" and "Diabetes" data sets.

5.2 Regression Results

We tested the regression component of CRIO on three real data sets found on the UCI data repository and compared the results to commercial softwares of CART [3] and MARS [4][9]. Similar to the experiments on the classification data sets in Section 5.1, each of the regression data was split into three parts, with 50%, 30% and 20% of the data used for training, cross-validating and testing, respectively. The assignment to each set was done randomly, and the process was repeated three times.

We used the cross-validation set for CRIO to fine tune the values of parameters K, L and ρ of Problem (17). In CART, we use the cross-validation set in the pruning step to prevent over-fitting of the initial tree. In MARS, we use the cross-validation set to choose the appropriate maximum number of basis functions. Also, we adjusted the running time versus accuracy parameter in MARS to maximize accuracy over running time.

[9]MARS is a product of Salford Systems, http://www.salford-systems.com

Cancer $n = 682, d = 9$	**Train** (%)	**Cross** (%)	**Test** (%)
CART	95.37	93.83	95.41
SvmFu	97.07	96.41	98.79
CRIO	97.26	96.90	98.79
Liver $n = 345, d = 6$	**Train** (%)	**Cross** (%)	**Test** (%)
CART	76.93	68.97	63.33
SvmFu	80.43	70.23	64.76
CRIO	73.06	74.43	71.43
Diabetes $n = 768, d = 8$	**Train** (%)	**Cross** (%)	**Test** (%)
CART	83.20	71.27	69.05
SvmFu	80.99	71.74	70.56
CRIO	80.12	79.42	77.06
Heart $n = 349, d = 44$	**Train** (%)	**Cross** (%)	**Test** (%)
CART	83.37	76.93	74.18
SvmFu	100.00	88.46	86.38
CRIO	99.81	83.65	81.69
Iris $n = 150, d = 4$	**Train** (%)	**Cross** (%)	**Test** (%)
CART	97.78	96.30	91.11
SvmFu	98.67	98.52	96.67
CRIO	99.56	97.04	96.67

Table 1: Prediction accuracy rates of CART, SvmFu and CRIO.

Regression Data

The "Boston" data, with 13 explanatory variables and 506 observations, is the Boston housing data set with dependent variable being the median value of houses in the suburbs of Boston. The "Abalone" data, with 8 explanatory variables and 4177 observations, attempts to predict the age of an abalone given its physiological measurements. Finally, the "Auto" data, with 5 explanatory variables and 392 observations, is the auto-mpg data set which determines the miles-per-gallon fuel consumption of an automobile, given the mechanical attributes of the car.

Results

Table 2 illustrate the results of CART, MARS and CRIO for each data set. In addition, we ran MARS on the transformed data used in CRIO, creating MARS-transf. MAE is the mean absolute error and MSE is the mean squared error.

CRIO chose $K = 2$ for all data sets, $L = 3$ and 4 for "Boston" and "Auto" and $L = 10 - 14$ for "Abalone", and $\alpha = 1$ and 2% for all data sets. We commented earlier that it is possible for a new point not to be contained in any of the polyhedra P_k. We found this to be an extremely rare case for all the data sets, but when it did occur, we assigned the point to the region that contains the majority of $F = 11$ nearest neighbors. The nonlinear transformation step did improve the model, sometimes significantly, and log was the most commonly used transformation for each data set. The run times of CRIO for "Boston" and "Auto" is comparable to CART and MARS, taking couple CPU seconds on average. CRIO takes up to 13 minutes to solve for "Abalone", which is a significantly larger data set.

CRIO clearly outperformed both CART and MARS in the testing set accuracy. To our surprise, it had better MSE values although our model minimized the total absolute error, whereas CART's and MARS' goodness-of-fit criterion is proportional to the squared error. Compared to CART, CRIO had up to 21.7% lower mean absolute error and 38.6% lower mean squared error. Although MARS was more accurate than CART on average, CRIO still had up to 10.1% lower mean absolute error and 28.6% lower mean squared error.

We used MARS-transf to assess whether the reason of CRIO's superior performance was the nonlinear transformation of the variables or the mixed-integer optimization methodology. It is clear that MARS-transf improves upon MARS, but it is also clear that CRIO's performance is not only due to the nonlinear transformation. CRIO still has up to 8.1% lower mean absolute error and 3.6% lower mean squared error compared to MARS-transf.

Boston $n = 506, d = 13$	**Train** MAE	MSE	**Cross** MAE	MSE	**Test** MAE	MSE
CART	1.883	8.121	2.840	17.746	2.984	21.057
MARS	2.346	10.363	2.536	11.750	2.600	15.810
MARS-transf	2.283	9.907	2.463	11.391	2.543	14.543
CRIO	2.171	11.508	2.554	13.882	2.337	14.197
Abalone $n = 4177, d = 8$	**Train** MAE	MSE	**Cross** MAE	MSE	**Test** MAE	MSE
CART	1.387	4.676	1.577	5.655	1.628	5.978
MARS	1.492	4.263	1.563	6.518	1.552	6.127
MARS-transf	1.507	4.373	1.552	4.690	1.540	4.525
CRIO	1.506	4.594	1.515	4.880	1.469	4.376
Auto-mpg $n = 392, d = 5$	**Train** MAE	MSE	**Cross** MAE	MSE	**Test** MAE	MSE
CART	1.397	4.898	2.444	11.350	2.435	12.784
MARS	1.898	6.282	2.176	9.098	2.127	8.433
MARS-transf	2.084	7.386	1.884	7.029	1.922	8.101
CRIO	1.765	6.936	2.133	9.200	2.051	7.851

Table 2: Results of models CART, MARS, MARS-transf and CRIO on “Boston”, “Abalone” and “Auto” data sets.

6 Conclusions

CRIO represents a new approach to solving classifiction 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.

While clustering, nonlinear transformations, and continuous optimization methods have been used for these problems before, we feel that the key competitive advantage of CRIO is its use of mixed-integer optimization methods. In contrast to the heuristic character of CART and MARS, CRIO produces splits of the explanatory variable space into disjoint regions in a globally optimal manner. CRIO is also able to separate classification data into disjoint polyhedra, when SVM (even with the use of nonlinear kernels) is unable to do so.

While more testing is clearly needed, we feel that the preliminary evidence on widely circulated real data sets is encouraging, since CRIO has matched and often outperformed CART, MARS and SVM on these data sets. More generally, we hope that these encouraging results will motivate the statistics and data mining community to re-examine the long held belief on the ineffectiveness of integer optimization methods in statistical computing.

Acknowledgements

We would like to thank Professor Nitin Patel for his comments on an earlier draft of the paper.

References

[1] Arthanari, T.S., Dodge, Y. (1993). *Mathematical Programming in Statistics*, Wiley, Wiley Classics Library Edition, New York.

[2] Bennet, K. P., Blue, J. A. (1984). Optimal Decision Trees, *R.P.I. Math Report No.214*.

[3] Breiman, L., Friedman, J., Olshen, R., Stone, C. (1984).*Classification and Regression Trees*, Wadsworth International,CA.

[4] Friedman, J. (1991). "Multivariate Adaptive Regression Splines", *Annals of Statistics*, Vol. 19:1, 1-67.

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

[6] ILOG CPLEX 7.1 Reference Guide, (2001). ILOG CPLEX Division, Incline Village, NV.

[7] Johnson, R.A., Wichern, D.W. (1998) *Applied Multivariate Statistical Analysis*, 4th edt., Prentice Hall, NJ.

[8] Mangasarian, O.L. (1965). "Linear and Nonlinear Separation of Patterns by Linear Programming", *Operations Research*, Vol. 13:3, 444-452.

[9] Quinlan, R. (1993). *C4.5: Programns for Machine Learning*, Morgan Kaufman, San Mateo.

[10] Rice, J.A. (1995) *Mathematical Statistics and Data Analysis*, 2nd edt., Duxbury Press, CA.

[11] Rousseeuw, P.j., Leroy, A.M. (1987). *Robust regression and outlier detection*, Wiley, New York.

[12] Ryan, T.P. (1997) *Modern Regression Methods*,Wiley Series in Probabilty and Statistics, New York.

[13] Vapnik, V. (1999) *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.