Noname manuscript No.
(will be inserted by the editor)

Optimal Predictive Clustering

Dimitris Bertsimas · Matthew Sobiesk · Yuchen Wang

Received: DD Month YEAR / Accepted: DD Month YEAR

Abstract Interpretability is an important part of building machine learning models. However, current interpretable machine learning methods can still improve in the areas of either out-of-sample performance or scalability. To address this, we combine clustering algorithms and linear regression to propose Optimal Predictive Clustering (OPC). By solving a mixed-integer optimization problem utilizing strong warm starts, we are able to find near-optimal clusters at high speeds while learning effective cluster-specific regression models. We also cluster points using both $\mathbf{X}$ and $\mathbf{y}$ values, which allows the clusters we find to be better suited for prediction tasks. We compare OPC, Lasso regression, Classification and Regression Trees (CART), Optimal Regression Trees (ORT, ORT-L), and XGBoost on 20 real-world datasets and show that OPC has a performance edge over all other interpretable methods, while also achieving better scaling than current state-of-the-art interpretable regression methods.

Keywords Clustering · Regression · Mixed-integer Optimization · Interpretability

Dimitris Bertsimas
Operations Research Center and Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA E-mail: dbertsim@mit.edu

Matthew Sobiesk
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, USA E-mail: msobiesk@mit.edu

Yuchen Wang
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, USA E-mail: yuchenw@mit.edu

1 Introduction

One of the current major goals in machine learning is to create and implement methods that achieve state-of-the-art performance, are scalable, and are interpretable. However, none of the currently widely used state-of-the-art regression methods achieve success in all three metrics at the same time.

Lasso regression (Tibshirani (1996)) is a widely used, fast, and fairly interpretable algorithm. One can solve for the model near-instantaneously, and by analyzing the magnitude and sign of the coefficients one can get an understanding of how a covariate contributed to the model's prediction. However, compared to methods such as XGBoost (Chen and Guestrin (2016)), it rarely achieves cutting edge performance.

CART (Breiman (1984)) is a decision tree algorithm that uses a greedy training process to learn splits in the tree in order to make predictions. Like Lasso regression it scales very well, and it is even more interpretable, as the path a data point takes down the tree gives a clear summary of exactly why a certain prediction was made. However, due to the greedy training algorithm, it also has a suboptimal performance compared to Optimal Regression Trees with Linear Predictions and XGBoost.

Optimal Regression Trees (ORT) and Optimal Regression Trees with Linear Predictions (ORT-L) (Bertsimas and Dunn (2019)) are both very interpretable decision tree methods like CART. The two methods differ in the type of prediction they output; ORT outputs point predictions, while ORT-L uses a Lasso regression model in each leaf to make predictions on data sorted to that leaf. While ORT trains at a reasonably fast speed, it is not as fast as Lasso regression or CART, and it also does not achieve the strong performance that ORT-L does. In contrast, ORT-L has cutting edge performance among interpretable regression methods, but it does not scale very well to larger datasets (on the scale of $p \geq 50$ and $n \geq 20000$).

Lastly, XGBoost is scalable and performs extremely well on a variety of datasets. However, it is not directly interpretable, which we define as a black box method. Table 1 summarizes our qualitative ranking of the five methods in the categories of performance, scalability, and interpretability.

	Lasso regression	**CART**	**ORT**	**ORT-L**	**XGBoost**
Performance	4	5	3	2	1
Scalability	1	1	4	5	3
Interpretability	3	1	1	3	5

Table 1 Comparison of major machine learning methods relative to each other across the metrics of performance (out-of-sample R^2), scalability and interpretability. 1 is the best, while 5 is the worst.

From Table 1, we observe all existing methods have weakness in at least one category. We therefore seek to design a method that has strong performance in all three categories at the same time. Optimal Predictive Clustering (OPC) is an algorithm that uses mixed integer optimization (MIO) to simultaneously cluster the data while learning cluster specific regression models. When clustering the data, the proposed method takes into account both $\mathbf{X}$ and $\mathbf{y}$ values, in order to create clusters better suited to prediction. This process results in Lasso regression models that are specialized for the data in the cluster they are applied to, which achieve stronger out-of-sample performance than a single model would. The resulting model is also interpretable, as we can create profiles for each cluster to understand why the models make the predictions they do. In addition, it is scalable, as the MIO can be solved quickly to near optimality using strong warm starts and early stopping. The warm starts considering both $\mathbf{X}$ and $\mathbf{y}$ values are generated by a advanced version of K-Means known as K-Means++ (Arthur and Vassilvitskii (2006)). In Sections 3, 4, and 5, we show the resulting method combines strong out-of-sample performance, scalability, and interpretability.

1.1 Literature

Combining clustering and regression algorithms is known as clusterwise linear regression. Originally proposed by Späth (1979, 1980), it is defined as finding a given number of clusters of observations such that the overall sum of squared errors within those clusters is minimized. Späth (1979) proposes a stepwise optimal solution which exchanges points in different clusters step by step to decrease the objective value. Späth (1986) and Meier (1987) extend this work to minimize the absolute deviations of the linear regression errors and provide further algorithmic improvements that speed up the learning process. However, these methods depend a lot on their initialization, which can result in the algorithm returning sub-optimal solutions.

Other researchers have tried a variety of algorithms and metaheuristics to solve clusterwise linear regression problems. These include Desarbo et al. (1989) using simulated annealing; Hansen and Caporossi (2005) using Variable Neighborhood Search; Bagirov et al. (2015) using a smoothed version of a nonlinear clusterwise linear regression model to iteratively find a solution; and DeSarbo and Cron (1988), Städler et al. (2010), Meynet and Maugis-Rabusseau (2012), and Devijver (2016) all applying mixture models trained using EM algorithms to learn the clusters and coefficients. However, these methods were either applied to small scale problems ($n \leq 500$) or involved a small number of clusters ($k \leq 3$), thus raising the question of how well they scale.

Lastly, Manwani and Sastry (2012) and Gitman et al. (2018) use an iterative algorithm that optimizes over both the sum of squared errors of the regression and the distance of $\mathbf{X}$ values from the centroids of the clusters they are assigned to. However, their clustering of the data does not include

y values, and they also apply their methods to mainly smaller scale problems ($n \leq 10000$).

We next describe global optimization solutions to the clusterwise linear regression problem. Lau et al. (1999) propose a nonlinear MIO formulation that maximizes a log-likelihood function objective and solve the problem without a guarantee of an optimal solution. Carbonneau et al. (2010) propose a mixed logical-quadratic programming formulation that provides a feasible method for solving the clusterwise regression problem to optimality. In later work they improve the solution by using branch and bound methods, column generation, and some other heuristics (Carbonneau et al. (2012), Carbonneau et al. (2014)). Bertsimas and Shioda (2007) apply linear mixed integer optimization to both regression and classification problems. Park et al. (2017) and Angün and Altınoy (2019) both extend this formulation to the case when there are multiple potential outputs. Park et al. (2017) use a mixed integer quadratic program combined with a column generation heuristic to find a solution, while Angün and Altınoy (2019) use a two step optimization framework to first find the number of clusters to use, and then find the solution of their novel MIO. Overall, none of these global optimization papers consider both the closeness between **X** and the cluster centroids and the prediction error of **y** in their objectives, which can result in space to improve out-of-sample performance. It is also unclear how well the proposed formulations and algorithms scale.

In summary, results from the literature indicate previous methods lack strong out-of-sample performance or scalability because they do not approach the globally optimal solution, do not consider both **X** and **y** values, or do not have strong warm starts. To solve this problem, the proposed method OPC takes into consideration not only the distance between the data and their corresponding centroids, but also the prediction error, which significantly improves the out-of-sample performance of the method. OPC also utilizes a strong warm start based on the K-Means++ algorithm, which improves the scalability of the algorithm.

1.2 Contribution

In this paper, we combine ideas from clustering and prediction to propose a new method called Optimal Predictive Clustering. We model it directly as a MIO and illustrate how to solve it with appropriate warm starts. Our main contribution in this paper is a novel MIO formulation of the clusterwise linear regression problem that generalizes some earlier approaches, alongside a faster way to solve it. A key part of this formulation is clustering points based on both **X** and **y** values, so that the clusters are better suited for learning regression models. We further show that this method achieves strong performance in the following three categories:

1. **Performance**: We demonstrate that OPC increases out-of-sample R^2 by 0.018 (2.96%) on average compared to ORT-L on 20 datasets from UCI and LIACC machine learning Repository (Dheeru and Karra Taniskidou

(2017),Torgo (2019)). Furthermore, OPC increases out-of-sample R^2 by 0.133 (26.92% improvement) on average over Lasso regression on the same 20 datasets. The method's performance also further closes the gap between the state-of-the-art interpretable methods and black box algorithms like XGBoost.
2. **Scalability**: We show the average time to solve OPC is about 800 seconds on the same 20 datasets, which is $\sim$30$\times$ faster than ORT-L. Furthermore, the maximum time to solve large scale problems ($n \sim 70000$, $p \sim 30$) is under 2000 seconds.
3. **Interpretability**: We show how to use the centroids the proposed method finds to create profiles of points in a given cluster and understand why it makes the predictions it does. We further compare OPC with ORT-L in two real world datasets, showing there is no significant loss in interpretability when using the proposed method.

For scalability, OPC is faster than ORT and ORT-L but slower than other methods. The performance of OPC is stronger than the rest methods except XGBoost. Finally, OPC's interpretability is similar to that of Lasso regression and ORT-L. Table 2 summarizes our qualitative ranking of the six methods.

	Lasso regression	**CART**	**ORT**	**ORT-L**	**XGBoost**	**OPC**
Performance	4	5	4	3	1	2
Scalability	1	1	5	6	3	4
Interpretability	3	1	1	3	6	3

Table 2 Comparison of major machine learning methods and OPC relative to each other across the metrics of performance (out-of-sample R^2), scalability and interpretability. 1 is the best, and 6 is the worst.

1.3 Structure

The paper is structured as follows. In Section 2, we describe how to find an optimal clustering of points for prediction using MIO. Then we present an algorithm to solve the optimization problem using warm starts. In Section 3, we show experimentally on synthetic data that the algorithm recovers the truth. We also present computational results on 20 real-world datasets and compare it with other methods including Lasso regression, CART, Optimal Regression Trees (ORT, ORT-L), and XGBoost. In Section 4, we compare the training times of these methods to that of OPC on both synthetic data and real-world datasets. In Section 5, we discuss how to interpret the results of the proposed method. We then go through two real-world examples of this process, while comparing the proposed method's interpretability to that of ORT-L. We conclude in Section 6,

2 The approach

In this section, we first describe how we apply a MIO to find an optimal clustering of points for prediction. After that, we illustrate how to find a strong warm start to speed up the optimization process. Lastly, we describe how we tune the algorithm's hyper-parameters.

We are given data $(\mathbf{x}_i, y_i)$, $i \in [n]$, where $[n] = \{1, \ldots, n\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. We normalize the data by applying the transformation

$$\hat{x}_{ij} := \frac{x_{ij} - m_j}{\sigma_j}, \ i \in [n], \ j \in [p], \tag{1}$$

where m_j and σ_j are the mean and standard deviation of dimension j of the training samples.

If we want to use clusters for prediction, one direct way is

1. Use K-Means to find the centroids for each cluster.
2. In each cluster, run a linear (Lasso) regression based on the y value.
3. When there is a new data point, assign it to the cluster with the closest centroid, then use the linear function for that cluster to predict the y value.

However, this only leads to a locally-optimal solution and also does not consider the $\mathbf{y}$ values. In order to achieve a globally optimal solution for this prediction problem, the objective function of the optimization problem should include:

1. The prediction error.
2. The distance between data and corresponding centroids.

We use a hyper-parameter α to decide the trade-off between them in the objective function.

To define the clusters in the MIO, assume we want to partition n points into K clusters. For each observation i, we introduce indicator variables $z_{ik} = \mathbb{1}\{x_i \text{ is in cluster } k\}$ to track the points assigned to each cluster. We force each point to be assigned to exactly one cluster by adding the constraint:

$$\sum_{k \in [K]} z_{ik} = 1, \quad i \in [n]. \tag{2}$$

We also force each cluster to have at least $N_{\min}$ points:

$$\sum_{i \in [n]} z_{ik} \geq N_{\min}, \quad k \in [K]. \tag{3}$$

Define the centroid of cluster k as

$$\mathbf{h_k} = (h_{k,1}, \ldots, h_{k,p})', \quad k \in [K]. \tag{4}$$

We add auxiliary variables $\mathbf{l_i} = (l_{i,1}, \ldots, l_{i,p})'$ to represent the centroid of the cluster point i belongs to. We apply constraints

$$h_{k,j} - l_{i,j} \geq -(1 - z_{ik})M_1,\ i \in [n],\ j \in [p],\ k \in [K], \tag{5}$$
$$h_{k,j} - l_{i,j} \leq (1 - z_{ik})M_1,\ i \in [n],\ j \in [p],\ k \in [K], \tag{6}$$

to enforce $h_{k,j} = l_{i,j} \iff z_{i,k} = 1,\ \forall i, j, k$ where M_1 is a sufficiently large number.

In each cluster, we want to use regression to make a prediction. We introduce variables $\mathbf{c_k} = (c_{k,1}, \ldots, c_{k,p})', d_k$ as the coefficients and intercept for the regression model in cluster k. We add auxiliary variables g_i to represent the predicted value for point i. We apply constraints

$$g_i - \sum_{j \in [p]} x_{i,j} c_{k,j} - d_k \geq -(1 - z_{ik})M_2, \quad i \in [n],\ k \in [K], \tag{7}$$
$$g_i - \sum_{j \in [p]} x_{i,j} c_{k,j} - d_k \leq (1 - z_{ik})M_2, \quad i \in [n],\ k \in [K], \tag{8}$$

enforcing $g_i = \sum_{j \in [p]} x_{i,j} c_{k,j} + d_k \iff z_{i,k} = 1,\ \forall i, j, k$ where M_2 is a sufficiently large number. We further apply L2 regularization to the coefficients $c_{k,j}$ weighted by hyper-parameter λ.

Putting all of this together gives the following MIO formulation for finding optimal clusters for prediction, which we call the OPC model:

$$\min \quad \sum_{i \in [n]} \left[\sum_{j \in [p]} \alpha (x_{i,j} - l_{i,j})^2 + (1 - \alpha)(y_i - g_i)^2 \right] + \lambda \sum_{k \in [K]} \sum_{j \in [p]} c_{k,j}^2 \tag{9}$$

$$\text{s.t.} \quad \sum_{k \in [K]} z_{ik} = 1, \quad i \in [n],$$
$$\sum_{i \in [n]} z_{ik} \geq N_{\min}, \quad k \in [K],$$
$$h_{k,j} - l_{i,j} \geq -(1 - z_{ik})M_1, \quad i \in [n],\ j \in [p],\ k \in [K],$$
$$h_{k,j} - l_{i,j} \leq (1 - z_{ik})M_1, \quad i \in [n],\ j \in [p],\ k \in [K],$$
$$g_i - \sum_{j \in [p]} x_{i,j} c_{k,j} - d_k \geq -(1 - z_{ik})M_2, \quad i \in [n],\ k \in [K],$$
$$g_i - \sum_{j \in [p]} x_{i,j} c_{k,j} - d_k \leq (1 - z_{ik})M_2, \quad i \in [n],\ k \in [K],$$
$$z_{ik} \in \{0, 1\}, \quad i \in [n], k \in [K].$$

Solving this MIO to optimality can be time consuming. We propose the following algorithm and use the solution as a warm start. For each number of clusters k and tuning parameter α we want to tune over, we cluster $[\alpha \mathbf{X}, (1-\alpha)\mathbf{y}]$ into k clusters using K-Means++ and train Lasso regression models within each one. We then compare the performance of each of these models on the validation set, and then find the k^* and α^* parameters where the model performs the best on the validation set. We lastly use these k^* and α^* values

to calculate warm starts for the cluster assignments, cluster centroids, and regression coefficients for the MIO.

Algorithm 1 Find warm starts for OPC

Select potential numbers of clusters $K \in \{K_1, K_2, K_3, \ldots, K_i\}$ and potential trade-off parameters $\alpha \in \{\alpha_1, \ldots, \alpha_j\}$ to tune over.
for $K = K_1, \ldots, K_i$ **do**
  for $\alpha = \alpha_1, \ldots, \alpha_j$ **do**
    Multiply the original $\mathbf{X}$ by α and add $(1-\alpha)\mathbf{y}$ as a new column.
    Run the K-Means++ algorithm to find K clusters for $[\alpha\mathbf{X},(1-\alpha)\mathbf{y}]$
    Get the centroids of each cluster, and then remove the $\mathbf{y}$ column
    Reweight the values by $\frac{1}{\alpha}$
    Re-assign points to the new cluster centroids
    Run a Lasso regression in each cluster
  end for
end for
Identify the solution (K^*, α^*) with largest R^2 on the validation set.
Multiply the original $\mathbf{X}$ by α^* and add $(1-\alpha^*)\mathbf{y}$ as a new column.
Run the K-Means++ algorithm to find K^* clusters for $[\alpha^*\mathbf{X},(1-\alpha^*)\mathbf{y}]$
Get the centroids of each cluster, and then remove the $\mathbf{y}$ column
Reweight the values by $\frac{1}{\alpha^*}$ to get cluster centroids $\mathbf{h}_j^*$ for each cluster j
Re-assign points to the new cluster centroids and get cluster assignments $\mathbf{z}^*$
Run a Lasso regression in each cluster j and get coefficients $\mathbf{c}_j^*, d_j^*$
Return $K^*, \alpha^*, \mathbf{z}^*, \mathbf{h}_j^*, \mathbf{c}_j^*, d_j^*$ $(j \in [K^*])$

3 Performance

In this section, we report the performance of OPC on a variety of synthetic and real-world datasets. In 3.1, we discuss data preprocessing methods and the hyper-parameters we validate for OPC as well as the other ML algorithms we use as benchmarks. In 3.2, we present OPC's performance on synthetic data and illustrate that it recovers the true underlying model from the data. In 3.3, we present OPC's performance on real-world datasets and then compare it to the performances of Lasso regression, CART, ORT, ORT-L and XGBoost methods. We also show the benefit of using both $\mathbf{X}$ and $\mathbf{y}$ values in the objective function by comparing OPC with methods that do not use $\mathbf{y}$ values.

3.1 Data Preprocessing and Model Validation

For a given dataset, we partition it into three parts: the training (60%), validation (20%), and testing sets (20%). The data are all standardized based on the mean and standard deviation of the combined training and validation sets.

For each of the models, we tune parameters by choosing those that lead to the best performance of the model on the validation set. For Lasso regression,

we validate regularization parameter λ using values between 0.0001 and 0.1. For CART, we validate depths from 1 to 10. For ORT and ORT-L, we tune both the depth of the tree from 1 to 10 and the complexity parameters between 0.001 and 0.1 based on the description from Interpretable AI (2019). For OPC, we use α values between 0.03 and 1.0, and K values between $\frac{n}{200}$ and $\frac{n}{1000}$ (for smaller datasets with $n \leq 1500$ we validate K= [1,2,3,4], for $1500 < n \leq 3000$ we validate K = [3,4,5,6,7]). We fix N_{min} to be 10 and λ to be 0.01. We set $M_1 = M_2 = 1000$. In simulation experiments, we fix K to be the true number of clusters. For XGBoost, we tune both the depth of the tree from 1 to 10 and the learning parameters between 0.1 and 0.5.

All problems are solved in parallel using the MIT Engaging Cluster (Intel Xeon 2.1 GHz) with 1 CPU core and 32GB of memory, and using the Python and Julia programming languages. Lasso regression and CART models are trained using the sklearn package in Python (Pedregosa et al. (2011)). ORT/ORT-L models are trained using Interpretable AI's Julia Interface (Interpretable AI (2019)), which applies the training algorithms found in Chapters 10 and 11 of Bertsimas and Dunn (2019). XGBoost models are trained using the XGBoost package in Python (Chen and Guestrin (2016)).

In order to avoid outcomes being influenced by one particular split of the data into training, validation, and test sets, we conduct the experiments five times with different splits each time. The final out-of-sample R^2 reported is the average of the five runs.

3.2 Performance on synthetic datasets

In this section, we investigate experimentally whether the algorithm converges to the ground truth we generate using synthetic datasets.

To create the datasets, we generate several different cluster centroids, generate points nearby the centroids using random perturbations, and then use cluster-specific random linear regression weights to define y values for each point. We use the following values of (K, n, p) to define the ground truths:

- Number of clusters K: 5, 10, 20
- Size of the data n: 1000, 10000, 100000
- Number of variables p: 10, 25, 40

Given this construction, the proposed method should be able to perfectly learn the true model, as we generate the data to match what the proposed method learns. The performance of the proposed method is shown in Table 3, where the Index columns are simply a numerical index of the datasets, columns K, p, and n are defined as above, and the Test R^2 column contain the test set R^2 of the proposed method.

Based on the results in Table 3, we observe the test R^2 is 1, which illustrates the proposed method perfectly learns the ground truth of the synthetic data. Warm starts play a key role here because they help our algorithm find the ground truth within the solving time. Empirically, when we give the optimization solver more than 300 seconds to solve the problem, the final solution

Index	K	p	n	Test R^2	Index	K	p	n	Test R^2
1	5	10	1000	1	15	20	25	10000	1
2	10	10	1000	1	16	5	40	10000	1
3	20	10	1000	1	17	10	40	10000	1
4	5	25	1000	1	18	20	40	10000	1
5	10	25	1000	1	19	5	10	100000	1
6	20	25	1000	1	20	10	10	100000	1
7	5	40	1000	1	21	20	10	100000	1
8	10	40	1000	1	22	5	25	100000	1
9	20	40	1000	1	23	10	25	100000	1
10	5	10	10000	1	24	20	25	100000	1
11	10	10	10000	1	25	5	40	100000	1
12	20	10	10000	1	26	10	40	100000	1
13	5	25	10000	1	27	20	40	100000	1
14	10	25	10000	1					

Table 3 Average out-of-sample R^2 of OPC on the synthetic data.

does not change. Thus, the algorithm finds the true optimum in that time span, and uses the rest of the time to guarantee the optimality. OPC is therefore able to recover the underlying structure of the data when such a structure exists, aided by using high quality warm starts.

We then experimented with adding noise to the $\mathbf{X}$ values of the data, to get better intuition of how model performance would change when applied to real world data. This noise was drawn from a uniform random variable Uniform$(-\epsilon, \epsilon)$, where $\epsilon = 0, 0.1, 0.2, \ldots, 1.9, 2$. This resulted in the following performance, aggregated by number of variables in the data, in Figure 1. As one can see, the method is generally less affected by increasing noise in the higher dimensional datasets. This makes sense, because in higher dimensional models the noise we add to the data gets added together in the linear model. Because the noise is Uniform$(-\epsilon, \epsilon)$, adding it together causes the summed noise to tend towards zero, lessening its effect.

Next, we analyze the performance of the model aggregated by number of data points, as seen in Figure 2. Here we see a much more striking decrease in performance as the amount of noise increases in all cases, although in general larger amounts of data still help the model make better inferences.

Overall, we observe in both graphs that for larger n and p values, the model is more robust to noise.

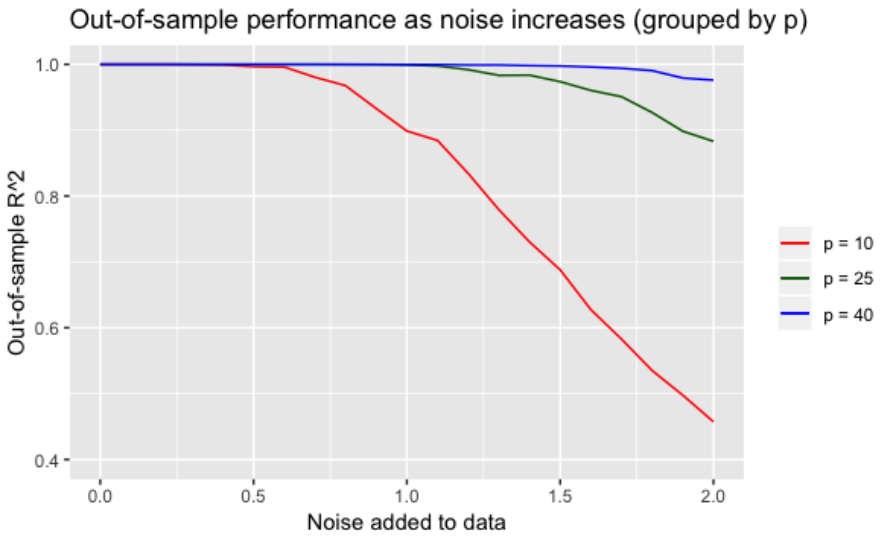

Fig. 1 Out-of-sample performance of the model with increasing levels of noise stratified by number of variables in the simulated data.

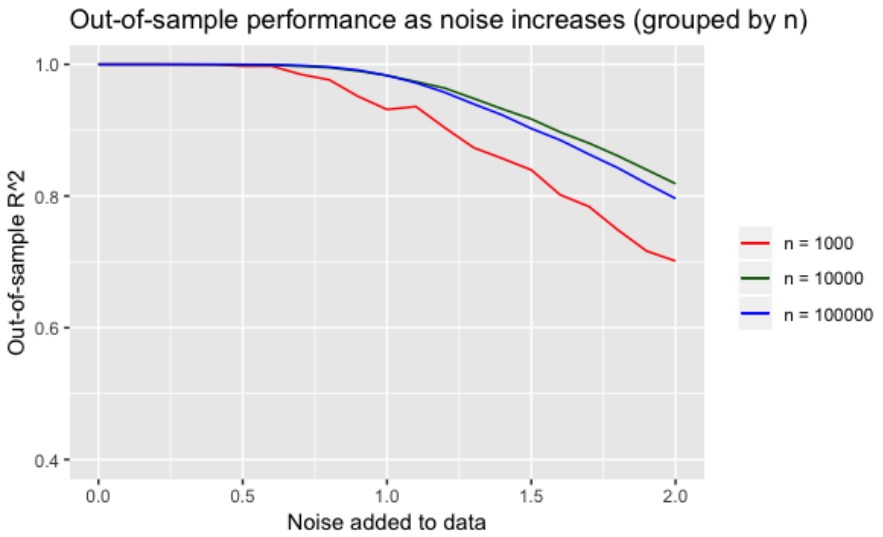

Fig. 2 Out-of-sample performance of the model with increasing levels of noise stratified by number of points in the simulated data.

Dataset	n	p	CART	Lasso regression	ORT	ORT-L	OPC	XGBoost
housing	505	13	0.732	0.751	0.77	0.805	**0.863**	0.863
geographic-origin	1059	68	0.467	0.657	0.422	0.482	**0.658**	0.627
wine-quality-red	1598	11	0.282	0.377	0.32	0.386	**0.394**	0.389
vote-for-clinton	2703	9	0.302	0.309	0.302	**0.398**	0.342	0.389
abalone	4176	7	0.456	0.525	0.461	0.538	**0.551**	0.528
wine-quality-white	4897	11	0.257	0.283	0.268	0.346	0.339	**0.422**
PTM	5874	16	0.166	0.09	0.143	0.123	0.215	**0.264**
PTT	5874	16	0.14	0.092	0.175	0.16	0.23	**0.27**
ailerons	7153	40	0.761	0.824	0.752	0.828	**0.839**	0.821
cpu-act	8191	21	0.967	0.722	0.964	**0.984**	0.936	0.98
cpu-small	8191	12	0.959	0.712	0.963	0.973	0.96	**0.973**
kin8nm	8191	8	0.434	0.421	0.518	**0.738**	0.655	0.671
elevators	8751	18	0.698	0.828	0.672	0.828	**0.864**	0.833
pole	15000	26	0.968	0.474	0.96	0.982	0.911	**0.984**
elevatorlarge	16559	17	0.715	0.816	0.704	0.824	**0.859**	0.836
energy	19735	29	0.227	0.161	0.209	0.293	0.351	**0.449**
californiahousing	20460	8	0.699	0.629	0.737	0.787	0.713	**0.801**
censusdomain	22784	16	0.398	0.265	0.363	0.417	0.439	**0.579**
CASP	45730	9	0.432	0.283	0.442	0.451	0.475	**0.583**
online_video	68784	25	0.956	0.651	0.958	0.845	0.947	**0.988**
AVERAGE			0.551	0.494	0.555	0.609	0.627	0.662

Table 4 Average out-of-sample R^2 of all methods on real-world datasets.

3.3 Performance on real-world datasets

In this section, we report the out-of-sample R^2 of OPC and other machine learning methods on 20 real-world datasets obtained from the UCI and LIACC Machine Learning Repositories. These results can be found in Table 4. The abbreviations PTM and PTT stand for parkinsons-telemonitoring-motor and parkinsons-telemonitoring-total, two UCI datasets.

In the left column the names of all datasets are listed from smallest n to largest n. The next two columns to the right show the number of data pointa n and the number of variables p in each dataset. The remaining columns contain the out-of-sample R^2 of all methods.

Based on these results, OPC has the best average out-of-sample performance of all interpretable regression methods. Compared with Lasso regression, OPC increases out-of-sample R^2 by 0.133 (26.92% improvement) on average in these datasets. Compared with the current cutting edge interpretable regression method ORT-L, OPC outperforms ORT-L in 13 out of 20 datasets with an average improvement of 0.018 (2.96% improvement). Compared with XGBoost, OPC outperforms XGBoost in 7 out of 20 datasets. XGBoost's average improvement over OPC is 0.035. Even though it still underperforms

XGBoost, OPC closes the gap between state of the art interpretable methods and black box algorithms.

In order to understand the benefit of considering $\mathbf{y}$ values when building the clusters, we compare the performance of OPC with other clusterwise regression methods in Table 5. A basic algorithm of building a clusterwise regression model is first running the K-Means++ algorithm to cluster all points. After that, we build a Lasso regression model in each cluster. The result of this basic algorithm is shown in the column named Basic. In Angün and Altınoy (2019), the authors propose a new MIO formulation for multiple response clustering that expanded upon previous formulations. We implement their algorithm with warm starts, and the result is shown in the column named Nested regression. The time constraints for solving MIO and the possible number of clusters are the same as for OPC. The regularization parameter is chosen to be either 0.01 or 0.1. We observe that the Angün and Altınoy (2019) result is similar to the basic method. Alongside this, OPC outperforms the basic method significantly in 10 out of 20 datasets and performs similarly to the basic method in the others. The average improvement is 0.011 (1.79% improvement). Overall, therefore, taking $\mathbf{y}$ values into account in the clustering is a useful technique to improve the performance of the method.

4 Scalability

In order to better understand the scalability of the proposed method, we also measure how long the training and validation process takes for each dataset. The training times for OPC on the synthetic data are in Table 6. Like Table 3, the Index columns are simply a numerical index of the datasets, and columns K, p, and n are defined as the number of clusters, the number of variables and the size of the data. The Time column contains the time it takes to build the OPC model. For the majority of the datasets, the training process takes less than ten minutes, and for all datasets it finishes within 3 hours.

Meanwhile, the training times of all methods for the real-world datasets are in Table 7. For these datasets, OPC always finishes running in an hour, finishes faster than ORT on average, and often finishes running an order of magnitude faster than ORT-L. Thus, the method achieves both strong out-of-sample performance and faster training times than comparable methods. It is still, however, slower than XGBoost (which finishes running in seconds), as well as CART and Lasso regression (which finish running almost instantaneously). Overall, though, the runtime is acceptable for most applications, and can be further improved by parallelizing the process.

Dataset	Basic	Nested regression	OPC
housing	0.846	0.846	**0.863**
geographic-origin	0.634	0.623	**0.658**
wine-quality-red	0.375	0.380	**0.394**
vote-for-clinton	**0.343**	0.335	0.342
abalone	0.55	0.545	**0.551**
wine-quality-white	**0.343**	**0.343**	0.339
PTM	**0.216**	**0.216**	0.215
PTT	0.21	0.21	**0.23**
ailerons	**0.841**	**0.841**	0.839
cpu-act	0.929	0.929	**0.936**
cpu-small	0.959	0.959	**0.960**
kin8nm	0.629	0.629	**0.655**
elevators	0.862	0.862	**0.864**
pole	0.876	0.876	**0.911**
elevatorlarge	0.857	0.857	**0.859**
energy	0.331	0.331	**0.351**
californiahousing	0.711	0.711	**0.713**
censusdomain	**0.44**	**0.44**	0.439
CASP	0.446	0.446	**0.475**
online_video	0.924	0.924	**0.947**
AVERAGE	0.616	0.615	0.627

Table 5 Average out-of-sample R^2 of K-Means++ with regression, Nested regression and OPC on real-world datasets. Bolded values are the highest values in a given row.

5 Interpretability

5.1 How to interpret the OPC model

Interpreting the results of OPC is a two-step process. First, we can analyze the clusters to create a profile of which points are sorted to which cluster. After that, we can study the regression coefficients to understand the particular model being used within a cluster.

To perform the first step, we can better understand why a point is sorted to a given cluster by comparing the centroids of the clusters. By looking at the differences in feature values for each centroid, we can find the key features that cause a point to be sorted to one cluster over another. Given that empirically we have found that the model can learn the variations in the data effectively using a small number of clusters, this process is easily scalable.

Index	K	p	n	Time	Index	K	p	n	Time
1	5	10	1000	305.2	15	20	25	10000	823.1
2	10	10	1000	316.8	16	5	40	10000	478.0
3	20	10	1000	322.0	17	10	40	10000	692.4
4	5	25	1000	312.6	18	20	40	10000	1095.8
5	10	25	1000	325.2	19	5	10	100000	770.6
6	20	25	1000	353.1	20	10	10	100000	1452.8
7	5	40	1000	318.0	21	20	10	100000	2510.9
8	10	40	1000	340.1	22	5	25	100000	1516.5
9	20	40	1000	380.4	23	10	25	100000	2756.2
10	5	10	10000	348.0	24	20	25	100000	5525.3
11	10	10	10000	404.1	25	5	40	100000	2137.2
12	20	10	10000	518.9	26	10	40	100000	4253.0
13	5	25	10000	411.9	27	20	40	100000	8699.6
14	10	25	10000	546.4					

Table 6 Training times of the OPC on synthetic data.

Dataset	n	p	CART	Lasso regression	ORT	ORT_L	OPC	XGBoost
housing	505	13	0	0	13.5	1902	357.1	0.1
geographic-origin	1059	68	0.2	0	78.8	104134.3	339	0.9
wine-quality-red	1598	11	0	0	31.8	2697.9	305.3	0.3
vote-for-clinton	2703	9	0.1	0	93.5	6698.6	339.4	0.5
abalone	4176	7	0.1	0	128.5	2734.6	351.9	0.5
wine-quality-white	4897	11	0.1	0	113.3	6720.6	477.9	0.8
PTM	5874	16	0.3	0.1	431.5	8432.6	676.1	1.6
PTT	5874	16	0.3	0.1	352.7	10815.6	665.3	1.6
ailerons	7153	40	0.2	0.5	765.6	7300.3	702.2	2.0
cpu-act	8191	21	0.4	0	771.2	25221.8	661.4	2.0
cpu-small	8191	12	0.2	0	536.7	14242.2	689.2	1.4
kin8nm	8191	8	0.2	0	786.8	25759.9	720.9	1.5
elevators	8751	18	0.2	0.3	470.5	1399.8	873.8	1.5
pole	15000	26	0.3	0.2	1762.8	38081.1	1009.1	2.9
elevatorlarge	16559	17	0.3	0.6	1506.2	5084.9	907.2	2.2
energy	19735	29	1.4	0.4	2003.2	87423.9	1219.5	5.1
californiahousing	20460	8	0.5	0.1	2209.6	28424.2	1075.3	2.4
censusdomain	22784	16	1.1	0.2	1616.9	49912.7	1317.6	4.9
CASP	45730	9	1.7	0.6	5781.4	75781.3	1589	6.6
online_video	68784	25	1.4	1.8	17933	10100.9	1995.1	10.2
AVERAGE			0.4	0.3	1869.4	25643.5	813.6	2.4

Table 7 Training times of the all methods on real-world datasets.

Variable	Meaning
CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
N0X	Nitric oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Boston employment centres
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	pupil-teacher ratio by town
B	$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	Percentage of lower status of the population

Table 8 The meaning of each variable.

Then, once a point is assigned to a given cluster, we can investigate the coefficients of that cluster's regression model. By analyzing their magnitude and sign, we're able to understand how different features contribute to the final prediction.

Thus, obtaining a prediction for a new data point is mainly deciding which cluster profile it is most similar to and then using a simple and easily understandable cluster-specific model to make a prediction. One can clearly see the process is interpretable. To illustrate this, in the following subsections, we will walk through two examples of this process to show empirically how interpretable the method is. We will also compare OPC with ORT-L trees trained on the same datasets, to show how the proposed method does not lose a significant amount of interpretability compared to decision tree models.

5.2 Housing Dataset Results

In this section, we illustrate how to interpret the OPC model on a real-world example about predicting housing values from Harrison Jr and Rubinfeld (1978). We also compare the interpretability of OPC with ORT-L. There are 505 samples and 13 variables in this dataset. The description of each variable is shown in Table 8. The target is to predict the housing value in suburbs of Boston.

For the hyper-parameters of OPC we use α between 0.03 and 1.0 and K between 3 and 7. We fix N_{min} to be 10 and λ to be 0.01. After we run the OPC algorithm, it returns 6 clusters and the final out-of-sample R^2 is 0.921.

	Centroid 1	Centroid 2	Centroid 3	Centroid 4	Centroid 5	Centroid 6
CRIM	19.09	0.96	7.18	0.20	0.05	0.89
ZN	0.00	0.34	0.00	6.25	60.29	20.06
INDUS	18.49	15.86	18.10	7.30	3.40	6.00
CHAS	0.00	0.12	0.09	0.05	0.00	0.18
N0X	0.68	0.62	0.67	0.48	0.42	0.51
RM	5.79	5.89	6.22	6.19	6.66	7.31
AGE	94.17	92.10	86.53	55.29	28.27	63.27
DIS	1.79	2.56	2.35	4.61	7.25	3.60
RAD	23.20	4.46	24.00	4.62	3.84	6.16
TAX	667.80	371.92	666.00	292.67	312.64	294.47
PTRATIO	20.20	18.44	20.20	18.42	16.99	16.38
B	212.65	353.71	358.77	390.38	390.15	388.38
LSTAT	23.86	16.97	15.29	10.15	5.89	5.30
Housing value	12.10	17.98	19.51	22.54	29.33	36.34
Number of data in this cluster	45	74	60	125	58	42

Table 9 Information about the centroid of each cluster and the number of data points within it.

The information about each cluster OPC finds is shown in Table 9. Each row represents the centroid of a given cluster and the last row shows the number of data points in the cluster.

In this model, we observe that all data points are nearly equally separated between clusters. There are also significant differences between the average housing values of each cluster. Looking at the centroids closely, we observe that each cluster can be interpreted as representing a specific type of house. In Table 10, we describe the characteristics of each centroid. When a new house comes, the model first indicates which type of house this new house is most similar to based on the distance to each centroid. After it is assigned to a specific cluster, we use linear regression to predict the exact housing value of this new house. The coefficients of linear regression in each cluster are shown in Table 11. In different clusters, variables have different levels of importance. For Cluster 1, CRIM, N0X and AGE are the most important variables in the Lasso regression, as they have the largest magnitude. For Cluster 6, however, the most important coefficients are INDUS, AGE, DIS, TAX, PTRATIO and LSTAT.

As a comparison, we also train an ORT-L model on this data. We tune both the depth of the tree from 1 to 10 and the complexity parameter to be between 0.001 and 0.1. The out-of-sample R^2 of this ORT is 0.884. The exact tree is shown in Figure 3 and the coefficients of the linear regression model of each leaf are shown in Table 12. Similarly to OPC, ORT-L also finds several clusters (leaves). In each leaf, the prediction is made by a linear regression. However, while the ORT-L's splits are easy to follow, the profiles they create at each leaf node are not always as detailed as the ones created by OPC, as if a variable does not appear in a split, it is basically ignored by the model.

	Characteristics of the house
Centroid 1	Extremely high crime rate, old house, close to employment centers, high property-tax rate, larger population of people classified as lower status
Centroid 2	Old house, less rooms in the house
Centroid 3	High crime rate, high property-tax rate
Centroid 4	Normal house
Centroid 5	Very large house, far from employment centres, better education environment, less lower status population, new house
Centroid 6	Large house, More rooms in the house, better education environment, much less lower status population, low property-tax rate

Table 10 Characteristics of the centroids of each cluster.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
CRIM	-0.081	0.000	-0.363	0	0	0.000
ZN	0	0	0	0	0.053	-0.023
INDUS	0	0.066	0	-0.144	0	0.459
CHAS	0	0.691	3.453	0	3.107	0
N0X	-38.996	-12.999	-30.585	-3.823	0	0
RM	-0.530	4.241	-1.855	5.964	11.397	9.409
AGE	0.109	0	0	-0.045	-0.006	-0.116
DIS	0	-0.217	-4.769	-0.954	-0.520	-1.561
RAD	0	0	0	0.010	0	0
TAX	0	-0.003	0	-0.011	-0.003	-0.020
PTRATIO	0	-0.871	0	-0.332	0.083	-1.120
B	-0.002	0.011	0	0	0	0
LSTAT	-0.253	-0.228	-1.303	0	-0.354	-0.696

Table 11 Coefficients of the linear regression in each cluster.

The data are also spread around much more unevenly in the tree, with some leaves having a single digit number of points while others have many more. Overall, though, observing a point's distance from different centroids is not significantly different from following the path a point takes down the tree, and there is thus not a significant loss of interpretability.

5.3 Wine Quality Dataset Results

In this section, we illustrate how to interpret OPC on another real-world example about wine quality from Cortez et al. (2009), and again compare the

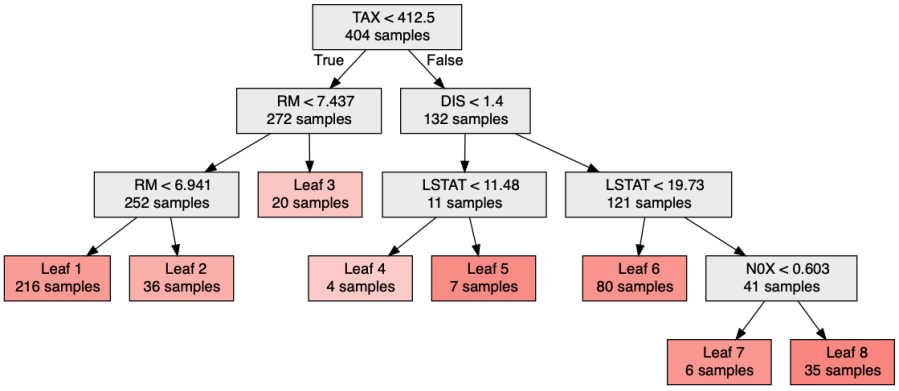

Fig. 3 ORT-L for predicting house values.

	Leaf 1	Leaf 2	Leaf 3	Leaf 4	Leaf 5	Leaf 6	Leaf 7	Leaf 8
CRIM	0	0	0	0	0	-0.050	0	-0.029
ZN	0.009	0.004	0.007	0	0	0	0	0
INDUS	-0.089	0	0	0	0	0	0	0
CHAS	0.953	-1.680	0	0	0	0	0	0
N0X	-7.244	0	0	0	0	0	0	0
RM	4.086	0	0	0	0	-0.499	0	0
AGE	-0.037	0	0	0	0	-0.001	0	0
DIS	-0.815	0	0	0	0	0	0	0
RAD	0.021	0	0	0	0	0.033	0	0
TAX	-0.008	0	0	0	0	0	0	-0.006
PTRATIO	-0.564	0	-0.731	0	0	0	0	0
B	0.008	0	0	0	0	0.009	0	0
LSTAT	-0.187	0	0	0	0	-0.501	0	0

Table 12 Coefficients of the linear regression in each leaf.

interpretability of OPC with ORT-L. There are 1598 samples and 11 variables in this dataset. The target is to predict the quality of red wine.

We tune both number of clusters K from 3 to 7 and α between 0.03 and 1.0. We fix N_{min} to be 10 and λ to be 0.01. After we run the OPC algorithm, it returns 4 clusters and the out-of-sample R^2 of OPC is 0.404. The information about each cluster OPC finds is shown in Table 13. Each row represents the centroid for a given cluster and the last row shows the number of data points in each cluster.

In this model, we observe all data points are once again nearly equally spread among each cluster. There is also a significant difference between the average wine quality values of each cluster. Looking at the centroids closely,

	Centroid 1	Centroid 2	Centroid 3	Centroid 4	Centroid 5
Fixed acidity	8.11	7.96	8.39	8.88	8.84
Volatile acidity	0.58	0.73	0.50	0.41	0.40
Citric acid	0.24	0.17	0.28	0.37	0.41
Residual sugar	2.57	2.56	2.46	2.74	2.76
Chlorides	0.09	0.10	0.09	0.08	0.07
Free sulfur dioxide	17.07	12.23	15.32	14.00	14.00
Total sulfur dioxide	55.96	33.51	40.35	34.18	30.86
Density	1.00	1.00	1.00	1.00	1.00
PH	3.31	3.38	3.31	3.29	3.23
Sulphates	0.62	0.60	0.68	0.74	0.78
Alcohol	9.94	10.20	10.63	11.45	11.95
Quality	5.25	5.34	5.68	6.11	6.55
Number of data in this cluster	371	330	236	205	136

Table 13 Information about the centroids of each cluster and the number of data points within it.

	Characteristics of the red wine
Centroid 1	High free sulfur dioxide, high total sulfur dioxide, high sulphates, low alcohol
Centroid 2	High volatile acidity, Low fixed acidity, high sulphates
Centroid 3	Normal Wine
Centroid 4	High fixed acidity, low volatile acidity
Centroid 5	High fixed acidity, low volatile acidity, low pH high citric acid, low chlorides,high sulphates, high alcohol

Table 14 Characteristics of the centroids of each cluster.

we observe that each cluster represents a specific type of red wine. In Table 14, we describe the characteristics of each centroid. When we want to make a prediction about a new wine, the model will first indicate which type of wine this new wine is most similar to based on the distance to each centroid. After it is assigned to a specific cluster, we use linear regression to predict the exact quality of this new wine. The coefficients of the cluster-specific linear regression models are shown in Table 15. We can see in different clusters, different variables are important in the Lasso regression. For example, alcohol and total sulfur dioxide are the most important variables in Cluster 1. However, fixed acidity, chlorides, and PH are the most important variables in Cluster 5.

As a comparison, we also train an ORT-L model. We tune both the depth of the tree from 1 to 10 and the complexity parameter to be between 0.001 and 0.1. The out-of-sample R^2 of this ORT-L is 0.376. The tree is shown in

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Fixed acidity	0	0.117	0	0.065	-0.033
Volatile acidity	-0.226	-1.404	-0.725	-0.181	-0.679
Citric acid	-0.167	0	-0.500	0	0.375
Residual sugar	0.023	0	0.200	0.029	0.035
Chlorides	0	0	-0.655	-1.638	-3.603
Free sulfur dioxide	0	0.014	0.004	0	0
Total sulfur dioxide	-0.003	0	0	-0.004	-0.002
Density	8.633	-56.112	0	-107.908	25.898
PH	0	-0.312	-0.156	-0.052	0.364
Sulphates	0	1.872	0.576	1.728	0.096
Alcohol	0.284	0.184	0.154	0.192	0.008

Table 15 Coefficients of the regression in each cluster.

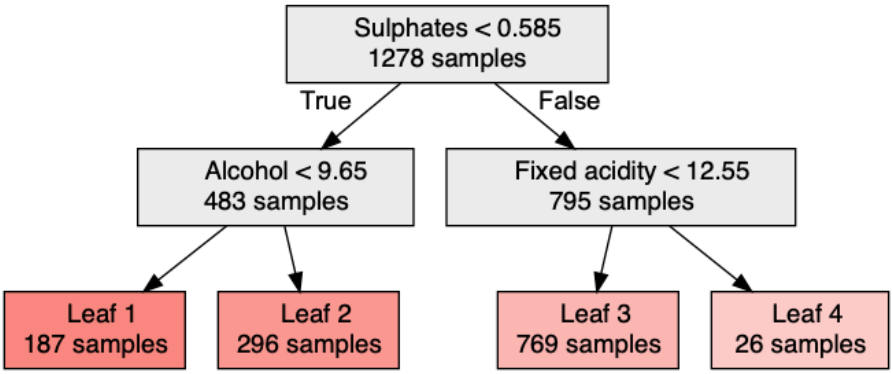

Fig. 4 The Optimal Tree for predicting wine quality.

Figure 4 and the coefficients of the linear regression model of each leaf are shown in Table 16.

Like in the last example, the tree's splits are easy to follow, but the resulting profiles are not the most informative compared to the OPC centroid profiles. The data are also spread more unevenly in the tree as in the last example, with the number of points sorted to a particular leaf ranging from 26 to 769. Overall, though, this process again shows that classifying points based on their distance from different centroids is still not significantly less interpretable than following the path a point takes down the tree.

These two real-world examples therefore illustrate how to interpret OPC models. In summary, OPC first finds which cluster the new data point belongs to and then uses linear regression to make a prediction. Compared to Optimal Regression Trees, OPC does not have the apparent geometric structures of

	Leaf 1	Leaf 2	Leaf 3	Leaf 4
Fixed acidity	0	0	0	0
Volatile acidity	0	-1.179	-0.939	0
Citric acid	-0.1443	0	-0.246	0
Residual sugar	0	-0.013	0.0213	0
Chlorides	0	0	-0.709	0
Free sulfur dioxide	0	0.009	0.002	0
Total sulfur dioxide	0	0	-0.006	0
Density	0	0	-0.435	0
PH	0	-0.652	-0.651	0
Sulphates	0	0	0.203	0
Alcohol	0	0.200	0.351	-0.123

Table 16 Coefficients of the linear regression in each leaf.

splits to differentiate the clusters. However, given the profiles one can create using the cluster centroids, OPC provides faster training times and better performance without losing too much interpretability.

6 Conclusions

In this paper, we introduced a methodology known as optimal predictive clustering that generalizes MIO-based approaches to the problem of clusterwise regression. This method clusters data using both $\mathbf{X}$ and $\mathbf{y}$ values, while also learning cluster specific models to make predictions. In terms of performance, scalability and interpretability, we illustrate OPC is competitive with previous methods, as shown in Table 2.

Experiments with synthetic data show strong evidence that OPC can recover the underlying structure of the data when such a structure exists. For real-world datasets, we show that the proposed algorithm achieves cutting edge performance at a much faster speed than the current state of the art of interpretable machine learning methods. OPC increases out-of-sample R^2 by 0.018 (2.96%) on average compared to ORT-L. Furthermore, OPC increases out-of-sample R^2 by 0.133 (26.92% improvement) on average over Lasso regression. Through a general description of the interpretation process and two examples, we also show that the proposed method maintains interpretability while achieving these results.

OPC therefore brings us closer to a current significant objective in machine learning – to create and implement models with state-of-the-art performance, scalability, and interpretability.

References

Angün, E. and Altınoy, A. (2019). A new mixed-integer linear programming formulation for multiple responses regression clustering, *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1634–1639.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding, *Technical report*, Stanford.

Bagirov, A. M., Ugon, J. and Mirzayeva, H. G. (2015). An algorithm for clusterwise linear regression based on smoothing techniques, *Optimization Letters* **9**(2): 375–390.
URL: *https://doi.org/10.1007/s11590-014-0749-3*

Bertsimas, D. and Dunn, J. (2019). *Machine Learning under a Modern Optimization Lens*, Dynamic Ideas LLC.

Bertsimas, D. and Shioda, R. (2007). Classification and regression via integer optimization, *Operations Research* **55**: 252–271.

Breiman, L. (1984). *Classification and Regression Trees*, CRC Press.

Carbonneau, R. A., Caporossi, G. and Hansen, P. (2010). Globally optimal clusterwise regression by mixed logical-quadratic programming, *European Journal of Operational Research* **212**: 213–222.

Carbonneau, R. A., Caporossi, G. and Hansen, P. (2014). Globally optimal clusterwise regression by column generation enhanced with heuristics, sequencing and ending subset optimization, *Journal of Classification* **31**(2): 219–241.
URL: *https://doi.org/10.1007/s00357-014-9155-x*

Carbonneau, R., Caporossi, G. and Hansen, P. (2012). Extensions to the repetitive branch and bound algorithm for globally optimal clusterwise regression, *Computers & Operations Research* **39**: 2748–.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system, *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, pp. 785–794.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties, *Decision Support Systems* **47**(4): 547–553.

Desarbo, W., Oliver, R. and Rangaswamy, A. (1989). A simulated annealing methodology for clusterwise linear regression, *Psychometrika* **54**: 707–736.

DeSarbo, W. S. and Cron, W. L. (1988). A maximum likelihood methodology for clusterwise linear regression, *Journal of Classification* **5**(2): 249–282.

Devijver, E. (2016). Model-based regression clustering for high-dimensional data: application to functional data, *Advances in Data Analysis and Classification* **11**: 243–279.

Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.
URL: *http://archive.ics.uci.edu/ml*

Gitman, I., Chen, J., Lei, E. and Dubrawski, A. (2018). Novel prediction techniques based on clusterwise linear regression, *ArXiv* **abs/1804.10742**.

Hansen, P. and Caporossi, G. (2005). Variable neighborhood search for least squares clusterwise regression.

Harrison Jr, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air, *Journal of environmental economics and management* **5**(1): 81–102.

Interpretable AI, L. (2019). Interpretable AI documentation.
URL: *https://www.interpretable.ai*

Lau, K.-N., Leung, P. L. and kit Tse, K. (1999). A mathematical programming approach to clusterwise regression model and its extensions, *European Journal of Operational Research* **116**: 640–652.

Manwani, N. and Sastry, P. S. (2012). K-Plane regression, *CoRR* **abs/1211.1513**.
URL: *http://arxiv.org/abs/1211.1513*

Meier, J. (1987). A fast algorithm for clusterwise linear absolute deviations regression, *Operations-Research-Spektrum* **9**(3): 187–189.
URL: *https://doi.org/10.1007/BF01721102*

Meynet, C. and Maugis-Rabusseau, C. (2012). A sparse variable selection procedure in model-based clustering.

Park, Y. W., Jiang, Y., Klabjan, D. and Williams, L. (2017). Algorithms for generalized clusterwise linear regression, *INFORMS Journal on Computing* **29**: 301–317.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Späth, H. (1979). Algorithm 39 clusterwise linear regression, *Computing* **22**(4): 367–373.
URL: *https://doi.org/10.1007/BF02265317*

Späth, H. (1980). Cluster analysis algorithms for data reduction and classification of objects.

Späth, H. (1986). Clusterwise linear least absolute deviations regression, *Computing* **37**(4): 371–377.
URL: *https://doi.org/10.1007/BF02251095*

Städler, N., Bühlmann, P. and van de Geer, S. A. (2010). L1-penalization for mixture regression models, *TEST* **19**: 209–256.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society: Series B (Methodological)* **58**(1): 267–288.

Torgo, L. (2019). LIACC regression data sets.
URL: *https://www.dcc.fc.up.pt/ ltorgo/Regression/DataSets.html*