Massachusetts Institute of Technology

Operations Research Center

15.083 Integer Programming

Instructor: Prof. Alex Jacquillat

Optimal Regression Trees with Monotonic Predictions

Angela Lin aglin@mit.edu

Cambridge, MA

Monday 21st August, 2023

1 Motivation

Fairness in Machine Learning (ML) models is critical for building safe and responsible AI systems. ML is increasingly being used in a wide range of decision-making scenarios that have serious implications for individuals and society, including criminal judgment, financial lending, hiring, admissions, health care, and social services. Therefore, we must consider the practical implications of deploying such models in real-world situations. Building fair models involves ensuring that the final model provides equitable, unbiased predictions. Further, **interpretability** in ML models, i.e. the ease with which humans can understand why and how predictions are made, is another desirable trait that if achieved can bridge the gap of trust between machines and humans. In a step towards addressing these goals, we aim to enforce **jointly monotonic predictions with respect to a group of predictive features**.

Features	Applicant A	Applicant B
GPA	4.5	3.5
Recommendation Letter	5	3
Research	2	0
Chance of Admit	0.5	0.9

Table 1: Example of unfair/uninterpretable prediction

To provide a concrete example: in the table above, a machine learning model suggests that Applicant B should be admitted over Applicant A even though Applicant A outperforms Applicant B in **all three** metrics. This prediction seems potentially unfair and hard to explain to a human. We seek to eliminate these kinds of predictions; our proposed model would necessarily make a higher prediction for Applicant A in this scenario.

Finally, a ML model that predicts monotonically can potentially make better, more logical predictions if monotonicity is a natural or **underlying property** in the data. For instance, when predicting demand, it is natural to assume that when price decreases, demand increases.

2 Methods

Let $f(x)$ be the prediction made by a machine learning model for a vector x with components x_i, $i \in [p]$. Suppose the modeler has chosen (using domain knowledge) a subset $\alpha \in [p]$ of the predictive features that it is desirable to enforce monotonic predictions with respect to. This group of features can also be seen as the "dominant" predictive features, as we require that the model make jointly monotonic predictions with respect to this group of features regardless of the values of the other features.

Our goal is to enforce that if x' **jointly** dominates x in **all features** in α (i.e. $x'_i \geq x_i \quad \forall i \in \alpha$), then $f(x') \geq f(x)$ no matter the values of the rest of the features in $[p] \setminus \alpha$. Formally, the constraints we want to impose on our model:

$$f(x) \leq f(x') \quad \forall x_\alpha \leq x'_\alpha \in \mathcal{X} \tag{1}$$

where $x_\alpha \leq x'_\alpha$ denotes $x_i \leq x'_i \quad \forall i \in \alpha$, and $\mathcal{X}$ is the universe of possible unseen points. $\mathcal{X}$ is a bounded set.

It is not obvious how one would embed jointly monotonic predictions with respect to a group of features into a ML model trained by a greedy algorithm (such as CART). However, a ML model that is trained using a global optimization formulation admits a natural way to enforce these requirements (as constraints in the optimization formulation). Therefore, we explore how to embed (1) into Optimal Regression Trees (ORT) from Bertsimas and Dunn [1], which are trained using a mixed integer linear program (MIP). The variables in the optimization problem are:

- f^i = the prediction for training point $i \in [n]$
- β_t = the prediction assigned to leaf node $t \in L$
- z_{it} = indicator for whether training point $i \in [n]$ is assigned to leaf node $t \in L$
- d_t = indicator for whether there's a split on branch node $t \in B$

- a_{jt} = indicator for whether feature $j \in [p]$ is used for the split at branch node $t \in B$
- b_t = the split value for the split at branch node $t \in B$
- l_t = indicator for whether leaf $t \in L$ has any points assigned to it
- L_i = the prediction error for point $i \in [n]$
- C = the complexity of the tree

The user-defined parameters are:

- N_{min} = the minimum number of points assigned to each leaf
- λ = the complexity regularization parameter
- max depth = the maximum depth of the tree

(1) is a set of infinitely many constraints, and if we had assignment variables z for each $x \in \mathcal{X}$, this would be an infinite number of variables as well. So we cannot directly add all constraints from (1) into the ORT MIP formulation. We will walk through our process for dealing with this.

As a first step, we should expect our model to at least make monotonic predictions on the training set, i.e. given a set of n training observations, we should expect

$$f(x^j) \leq f(x^i) \quad \forall x^j_\alpha \leq x^i_\alpha, (i,j) \in [n] \times [n]$$

We embed these constraints as (2b). Constraints (2c)-(2n) are exactly the formulation for ORT with parallel splits and constant predictions from Bertsimas and Dunn [1] (see Appendix for more detailed explanation of formulation).

$$\min \sum_{i=1}^{n} L_i + \lambda C \tag{2a}$$

$$\text{s.t. } f^j \leq f^i, \ \forall x^j_\alpha \leq x^i_\alpha, \ (i,j) \in [n] \times [n] \tag{2b}$$

$$L_i \geq |f^i - y^i|, \ \forall i \in [n] \tag{2c}$$

$$f^i - \beta_t \geq -M(1 - z_{it}), \ \forall i \in [n], t \in L \tag{2d}$$

$$f^i - \beta_t \leq M(1 - z_{it}), \ \forall i \in [n], t \in L \tag{2e}$$

$$C = \sum_{t \in B} d_t \tag{2f}$$

$$a_m^T x^i \geq b_m - M(1 - z_{it}), \ \forall i \in [n], t \in L, m \in R(t) \tag{2g}$$

$$a_m^T x^i + \epsilon \leq b_m + M(1 - z_{it}), \ \forall i \in [n], t \in L, m \in L(t) \tag{2h}$$

$$\sum_{t \in L} z_{it} = 1, \forall i \in [n] \tag{2i}$$

$$z_{it} \leq l_t \forall i \in [n], t \in L \tag{2j}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in L \tag{2k}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in B \tag{2l}$$

$$d_t \leq d_{p(t)}, \ \forall t \in B \setminus \{1\} \tag{2m}$$

$$z_{it}, l_t, a_{jt}, d_t \in \{0, 1\} \tag{2n}$$

L is the set of all leaf nodes (indexed by t); B is the set of all branch nodes (indexed by t); $R(t)$ is the set of right ancestors of leaf node t; and $L(t)$ is the set of left ancestors of leaf node t.

Further, we would like our model to make monotonic predictions on unseen data as well. If we have access to a test data set, we can re-train our model to ensure it makes monotonic predictions on the test data as well as the training data (taking the admissions data example, once we get our new batch of applications for the year, if we want to make sure our model makes monotonic predictions on this new batch of data, we can re-train our model as described below).

Given the test set of observations $\{x^1, ..., x^{n'}\}$, we ensure monotonic predictions on these data points, but still only optimize for predictive accuracy on the training set (we do not use the labels from the test set in the training of the model). We add variables z_{it} for each test point $i \in [n']$, and add constraints (3b), (3d), (3e), (3g), (3h), (3i) for the test points. Notice we do not use the labels $y^i, i \in [n']$ from the test set, and the objective of the model remains unchanged.

$$\min \sum_{i=1}^{n} L_i + \lambda C \tag{3a}$$

$$\text{s.t. } f^j \leq f^i, \ \forall x^j_\alpha \leq x^i_\alpha, \ (i,j) \in [n] \times [n] \cup [n'] \times [n'] \tag{3b}$$

$$L_i \geq |f^i - y^i|, \ \forall i \in [n] \tag{3c}$$

$$f^i - \beta_t \geq -M(1 - z_{it}), \ \forall i \in [n] \cup [n'], t \in L \tag{3d}$$

$$f^i - \beta_t \leq M(1 - z_{it}), \ \forall i \in [n] \cup [n'], t \in L \tag{3e}$$

$$C = \sum_{t \in B} d_t \tag{3f}$$

$$a_m^T x^i \geq b_m - M(1 - z_{it}), \ \forall i \in [n] \cup [n'], t \in L, m \in R(t) \tag{3g}$$

$$a_m^T x^i + \epsilon \leq b_m + M(1 - z_{it}), \ \forall i \in [n] \cup [n'], t \in L, m \in L(t) \tag{3h}$$

$$\sum_{t \in L} z_{it} = 1, \forall i \in [n] \cup [n'] \tag{3i}$$

$$z_{it} \leq l_t \forall i \in [n], t \in L \tag{3j}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in L \tag{3k}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in B \tag{3l}$$

$$d_t \leq d_{p(t)}, \ \forall t \in B \setminus \{1\} \tag{3m}$$

$$z_{it}, l_t, a_{jt}, d_t \in \{0, 1\} \tag{3n}$$

We hope that if the training and test set points offer "sufficient coverage"" of $\mathcal{X}$, then the ORT model from (3) (or even (2)) would make mostly monotonic predictions over $\mathcal{X}$.

However, we would like to check whether the ORT is guaranteed to make monotonic predictions over $\mathcal{X}$. To do this, we solve

$$\max_{x, x' \in X \quad s.t. x_\alpha \leq x'_\alpha} f(x) - f(x')$$

where $f(x)$ is the prediction made by the trained ORT for vector x. If the optimal objective value of this optimization problem is ≤ 0, then we know $f(x) \leq f(x') \quad \forall x_\alpha \leq x'_\alpha \in \mathcal{X}$, i.e. we have a certificate/guarantee that the ORT makes monotonic predictions (with respect to α) over the entire set X. Otherwise, if the optimal objective value is > 0, then we have found an $x, x' \in X$ that the ORT predicts non-monotonically, i.e. a constraint in (1) that is violated. We formulate a mixed integer linear program to solve this optimization problem:

Inputs: a decision tree (defined by a, b, β)
Decision variables: $x, x', z_t^{(x)}, z_t^{(x')}$ for $t \in L$ ($z_t^{(x)}$ indicates whether point x is assigned to leaf t)

$$\max_{x,x',z} \sum_{t\in L} \beta_t z_t^{(x)} - \sum_{t\in L} \beta_t z_t^{(x')} \tag{4a}$$
$$\text{s.t. } x_i \leq x'_i \quad \forall i \in \alpha \tag{4b}$$
$$a_m^T x \geq b_m - M(1 - z_t^{(x)}) \quad \forall t \in L, m \in R(t) \tag{4c}$$
$$a_m^T x + \epsilon \leq b_m + M(1 - z_t^{(x)}) \quad \forall t \in L, m \in L(t) \tag{4d}$$
$$a_m^T x' \geq b_m - M(1 - z_t^{(x')}) \quad \forall t \in L, m \in R(t) \tag{4e}$$
$$a_m^T x' + \epsilon \leq b_m + M(1 - z_t^{(x')}) \quad \forall t \in L, m \in L(t) \tag{4f}$$
$$\sum_{t\in L} z_t^{(x)} = 1 \tag{4g}$$
$$\sum_{t\in L} z_t^{(x')} = 1 \tag{4h}$$
$$x, x' \in X \tag{4i}$$
$$z_t^{(x)} \in \{0,1\} \quad \forall t \in L \tag{4j}$$
$$z_t^{(x')} \in \{0,1\} \quad \forall t \in L \tag{4k}$$
$$\tag{4l}$$

Given a decision tree, defined by a, b and β, this MIP finds two points $x_\alpha \leq x'_\alpha \in \mathcal{X}$ that maximize $f(x) - f(x')$. The objective is equal to $f(x) - f(x')$, as $f(x) = \sum_{t\in L} \beta_t z_t^{(x)}$ since $z_t^{(x)}$ is 1 if x is assigned to leaf t and 0 otherwise, and β_t is the prediction for leaf t. Constraints (4c) and (4d) ensure that $z_t^{(x)}$ accurately reflects whether point x belongs to leaf t of the tree, (4e) and (4f) play the same role for point x'.

If the optimal objective of (4) is > 0, we have found a violated constraint that corresponds to a pair of non-monotonically predicted points, and we would like to add this constraint to the ORT formualation. To do this, we must also add variables $z_t^{(x)}$ and $z_t^{(x')}$ to the master problem and ensure they take the correct values. The full algorithm is the following:

1. Let the Master Problem be formulation (2)
2. Solve the Master Problem and obtain an ORT defined by a, b, β
3. Solve sub-problem (4) with inputs a, b, β to get x, x'
   If the optimal objective of (4) is > 0, add new variables $z_t^{(x)}, z_t^{(x')}$ along with associated constraints (2d)-(2i) and monotonic prediction constraint (2b) to the Master Problem and return to step 2.
   If the optimal objective of (4) is ≤ 0, stop the algorithm. The current ORT is guaranteed to make monotonic predictions over $\mathcal{X}$.

By going through this iterative algorithm, we end up with an ORT that is guaranteed to make monotonic predictions over $\mathcal{X}$. Obtaining this certificate that the model will make monotonic predictions on any unseen points is is the ultimate goal, especially when we don't have full knowledge of the test set to be able to use it to train our model. For instance in a setting where we receive test points in an online fashion and must make predictions for these points immediately, it is imperative to know ahead of time that the model we're using is guaranteed to make monotonic predictions on these new points, as it is not reasonable to re-train the model each time we receive a new data point before making a prediction for that data point.

3 Data

3.1 Coffee Data Set

This data set is from an external retail collaborator of my advisor, Prof. Georgia Perakis. Each row corresponds to a coffee product in a particular week. The features are the price, the cumulative week, the week of the year, and the promotion percent (percent discounted). The target to be predicted is the total volume sold (which we assume to be the true demand). We subset the data to include only 1 brand and 1 product of coffee, which gives us 216 total observations. We choose our set of features to impose monotonic predictions with respect to as $\alpha = \{$- price, percent discounted$\}$ because we expect the underlying truth to be that the demand for a product goes up as both the price decreases and discount percent increases simultaneously.

3.2 Student Performance Data Set

This data set is the Student Performance Data Set from the UCI Machine Learning Repository. Each row corresponds to a secondary school student. The features are the student's internet access status, parents' educational status, student's higher education goals, number of previous class failures, amount of time spent studying, and number of absences. The target to be predicted is the student's final grade. This data set has 395 total observations. We choose our set of features to impose monotonic predictions with respect to as $\alpha = \{$study time, - number of absences$\}$ because it seems unfair and/or uninterpretable for a model to decide that a student who studies more and attends class more often should get a lower grade than a student who studies less and attends class less.

3.3 Graduate Admissions Data Set

This data set is the Graduate Admissions Data Set from Kaggle. Each row corresponds to a graduate school applicant. The features are undergraduate GPA, letter of recommendation score, statement of purpose score, research score, university rating, TOEFL score, and GRE score. The target to be predicted is the student's chances of being admitted to graduate school. We choose our set of features to impose monotonic predictions with respect to as α = $\{$letter of recommendation score, undergraduate GPA, research score, university rating$\}$ because it seems unfair and/or uninterpretable for a student with a higher score in all of these categories to be predicted a lower chance of admission than a student with a lower score in all of these categories.

4 Computational Results

We tested our methods by implementing the ORT MIP formulation from Bertsimas and Dunn [1] and adding monotonic prediction constraints for the training and test data. We also coded up the sub-problem for finding a non-monotonic prediction, as well as the full algorithm that generates and adds cuts and variables to the master problem until certified monotonicity on $\mathcal{X}$ is attained.

Upon running the code on our data sets, we saw that after adding the cuts from the training and test data, the sub-problem gave an optimal objective value of 0, meaning the trained ORT had already achieved certified monotonicity on $\mathcal{X}$, i.e. the training and test set cuts were sufficient to give a model that guaranteed monotonic predictions over $\mathcal{X}$. So then we decided to test what percentage of the training set cuts were necessary to get monotonic predictions on the test set and also what percentage of the training set cuts were necessary to get a "certified monotonic" ORT (we call an ORT that has a sub-problem optimal objective ≤ 0 "certified monotonic").

For each data set, we solved (2) with only a percentage of (randomly sampled) training set cuts added and then calculated how many non-monotonic predictions were made by the resulting ORT. Then we initiated the full algorithm using the resulting ORT, and generated and added cuts (and variables) from the sub-problem until a certified monotonic ORT was obtained. We repeated this 10 times for each percentage of training cuts tested and averaged the results. For each data set, features were normalized to take values between 0 and 1, and $\mathcal{X} = \{x | -5 \leq x_i \leq 5 \quad \forall i \in [p]\}$ was used. For all experiments, max depth = 3, $N_{min} = 5$, $\lambda = 0$ for the ORT.

4.1 Coffee Data set

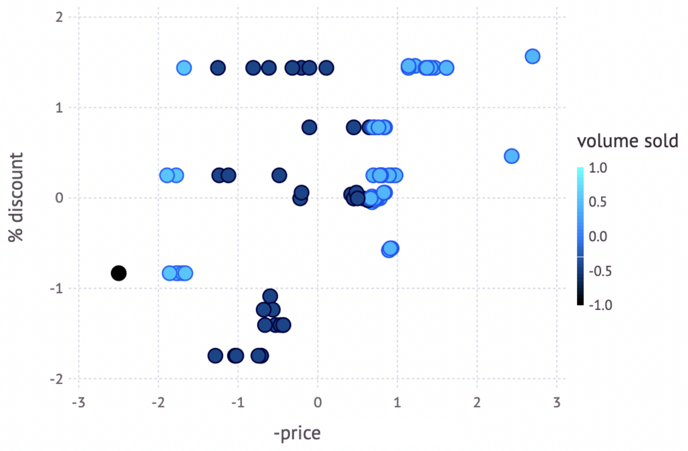
The training consisted of 130 samples and the test set 86 samples. The training set had 6083 pairs of points with monotonic relationship in $\alpha = \{$-price, percent discounted$\}$, which gave us accordingly many training cuts to add to the master problem.

% training cuts added	# non-monotonic predictions on test set	# master iterations*	time*
0 %	411	19	13 min
30 %	107	7	4 min
60 %	22	3	2 min
70 %	0	0	35 sec
100 %	0	0	21 sec

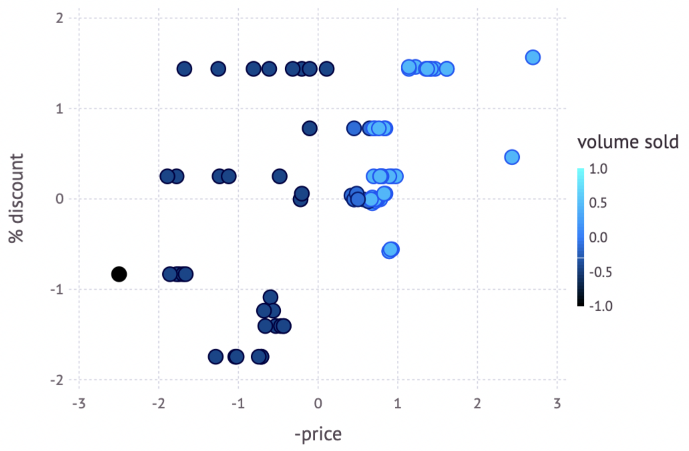
*until certified monotonicity obtained

Without adding any training cuts, there were many non-monotonic predictions made on the test set, and it took 19 iterations of generating a cut, adding the new cut and new variables, and re-solving the master problem to obtain a certified monotonic ORT. However, after adding at least 70% of the training cuts, not only did the resulting ORT make monotonic predictions on the test set, it was also a certified monotonic ORT. Each iteration of the master problem took about a minute to solve, but the solve time decreased as more cuts were added. The sub-problem solved in under a second.

Below are plots of the predictions (values represented by the shade of blue) against the α features on the axes. Indeed, the predictions were not monotonic on the left plot (without adding monotonic prediction constraints), but they did become jointly monotonic with respect to -price and discount on the right plot (after adding 70% of training cuts).



(a) test set predictions w/o monotonic constraints



(b) test set predictions with monotonic constraints

	w/o monotonic constraints	w/ monotonic constraints	percentage change
R^2	-0.087	0.008	+90.7%
MSE	1.007	0.904	-10.2%

Table 2: Out of Sample Metrics on Coffee Data Set

The R^2 improved significantly after adding the monotonic constraints, likely due to the fact that the predictive features were not good, so enforcing monotonic predictions helped the model because demand is usually monotonic with respect to price and discount percent. Injecting domain knowledge about the underlying truth into the model was able to improve prediction.

4.2 Student Performance Data Set

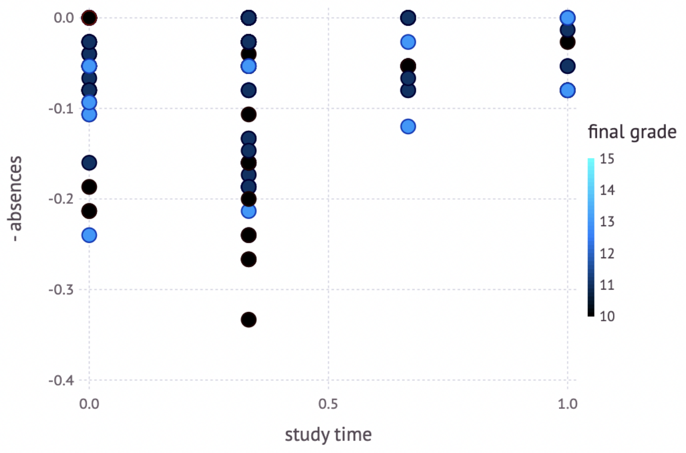
The training set consisted of 277 samples and the test set 118 samples. The training set had 30691 pairs of points with monotonic relationship in $\alpha = \{\text{study time, - number of absences}\}$, which gave us accordingly many training cuts to add to the master problem.

% training cuts added	# non-monotonic predictions on test set	# master iterations*	time*
0 %	1595	35	66 min
20 %	1132	22	39 min
40 %	220	5	8 min
50 %	0	1	2 min
55 %	0	0	49 sec
100 %	0	0	32 sec

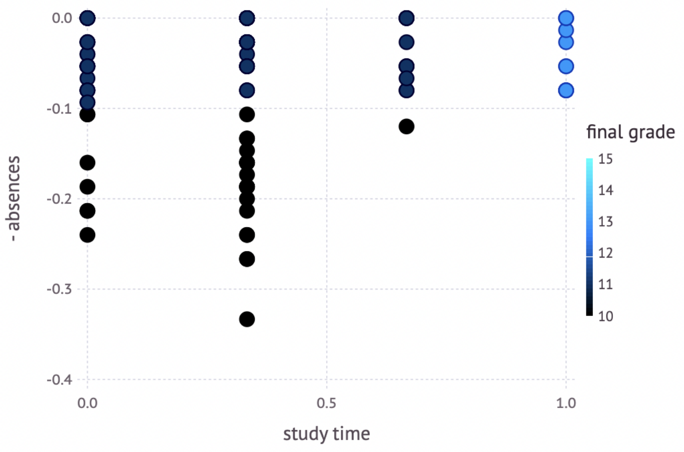
*until certified monotonicity obtained

Without adding any training cuts, there were many non-monotonic predictions made on the test set, and it took 35 iterations of the master problem to obtain a certified monotonic ORT. However, after adding 50% of the training cuts, all predictions on the test set were monotonic, and it only took 1 additional generated cut from the sub-problem to obtain a certified monotonic ORT. After adding at least 55 % of the training cuts, we got certified monotonic predictions over $\mathcal{X}$. Each iteration of the master problem took almost 2 minutes to solve on this data set, but the solve time decreased as more cuts were added. The sub-problem solved in under a second.

Below are plots of the predictions (values represented by the shade of blue) against the α features on the axes. Indeed, the predictions were not monotonic on the left plot (without adding monotonic prediction constraints), but became jointly monotonic with respect to study time and -number of absences on the right plot (after adding 55% of training cuts).



(a) test set predictions w/o monotonic constraints



(b) test set predictions with monotonic constraints

	w/o monotonic constraints	w/ monotonic constraints	percentage change
R^2	0.214	0.023	-89.4%
MSE	16.007	21.904	$+36.8\%$

Table 3: Out of Sample Metrics on Student Performance Data Set

The R^2 suffered quite a bit after adding the monotonic prediction constraints. For this example, incorporating this type of fairness into the predictions came at a price of lowered predictive performance.

4.3 Graduate Admissions Data Set

The training set consisted of 280 samples and the test set 120 samples. The training set had 27558 pairs of points with monotonic relationship in $\alpha = \{$letter of recommendation score, undergraduate GPA, research score, university rating$\}$.

After training an ORT without any monotonic prediction constraints, we saw that all predictions made on the test data were monotonic, and that the ORT was certified monotonic over $\mathcal{X}$. Our methods were not needed to get monotonic predictions on this data set; the predictions made by a vanilla ORT were naturally monotonic with respect to α. The ORT achieved an out of sample R^2 of 0.604 and took about 6 minutes to solve.

5 Limitations and Future Work

The optimal regression tree MIP is not very scalable, so we were only able to test our methods on small data sets (< 10 features and a couple hundred data points) and were limited to training small trees (depth ≤ 3). Since each master problem iteration took about a few minutes to solve, the full algorithm took very long if it was started without training any cuts, since it solved many iterations of the master problem before reaching a certified monotonic ORT. An alternative (non-iterative) approach for obtaining a certified monotonic ORT would be to take the dual of the LP-relaxation of sub-problem (4) and explicitly add constraints forcing the dual objective value to be ≤ 0 into the master problem. It is possible to directly embed the LP-dual of the sub-problem into the master problem because both are minimization problems. Requiring that $obj^*_{LPdual} \leq 0$ would guarantee that sub-problem (4)'s optimal objective, $obj^*_{MIP} \leq 0$, as $obj^*_{MIP} \leq obj^*_{LP} \leq obj^*_{LPdual}$. However, this dual approach may be too constraining if the LP-dual gives a loose upper bound on the MIP sub-problem objective value. The experiments we did in this project demonstrated the training cuts were sufficient for obtaining a certified monotonic ORT, so for these data sets, this approach was not needed.

Since we found experimentally that the training set cuts were sufficient for obtaining a certified monotonic ORT over $\mathcal{X}$, a next step would be to further explore this result from a theoretical perspective. A future goal is to characterize properties of a set of training cuts or training points that guarantee the solution will be a certified monotonic ORT over $\mathcal{X}$.

6 Conclusions

In this project, we worked towards building fair and interpretable machine learning models. As a step towards this goal, we embedded monotonic predictions within optimal regression trees, a machine learning model trained using a mixed integer program. We also formulated another mixed integer program that can be used to certify that a given decision tree always makes monotonic predictions, or to generate a new constraint and new variables if needed. We saw that adding monotonic prediction constraints on the training set was sufficient to guarantee monotonic predictions over a bounded set of possible unseen points. One of our experiments (student performance data set) showed that enforcing monotonic predictions led to a more fair and interpretable model but came at the price of worse predictive power. Another experiment (coffee data set) showed that adding monotonic constraints improved the predictive power of the model by providing additional information about the underlying truth. Finally, we note that monotonic prediction constraints can be incorporated into any machine learning model that is trained by solving a single global optimization formulation.

References

[1] D. Bertsimas and J. Dunn. *Machine Learning Under a Modern Optimization Lens*. Dynamic Ideas LLC, 2019.

A Appendix: Optimal Regression Trees Formulation

Given a set of n training observations $\{(x^1, y^1), ..., (x^n, y^n)\}$ where $x^i \in \mathbb{R}^p, y^i \in \mathbb{R}, \forall i \in [n]$, the following is the formulation for optimal regression trees with parallel splits and constant predictions from Bertsimas and Dunn [1]. L is the set of all leaf nodes (indexed by t); B is the set of all branch nodes (indexed by t); $R(t)$ is the set of right ancestors of leaf node t; and $L(t)$ is the set of left ancestors of leaf node t.
The variables in the optimization problem are:

- f^i = the prediction for training point $i \in [n]$
- β_t = the prediction assigned to leaf node $t \in L$
- z_{it} = indicator for whether training point $i \in [n]$ is assigned to leaf node $t \in L$
- d_t = indicator for whether there's a split on branch node $t \in B$
- a_{jt} = indicator for whether feature $j \in [p]$ is used for the split at branch node $t \in B$
- b_t = the split value for the split at branch node $t \in B$
- l_t = indicator for whether leaf $t \in L$ has any points assigned to it
- L_i = the prediction error for point $i \in [n]$
- C = the complexity of the tree

The user-defined parameters are:

- N_{min} = the minimum number of points assigned to each leaf
- λ = the complexity regularization parameter
- max depth = the maximum depth of the tree

$$\min \sum_{i=1}^{n} L_i + \lambda C \tag{5a}$$

$$\text{s.t. } L_i \geq |f^i - y^i|, \ \forall i \in [n] \tag{5b}$$

$$f^i - \beta_t \geq -M(1 - z_{it}), \ \forall i \in [n], t \in L \tag{5c}$$

$$f^i - \beta_t \leq M(1 - z_{it}), \ \forall i \in [n], t \in L \tag{5d}$$

$$C = \sum_{t \in B} d_t \tag{5e}$$

$$a_m^T x^i \geq b_m - M(1 - z_{it}), \ \forall i \in [n], t \in L, m \in R(t) \tag{5f}$$

$$a_m^T x^i + \epsilon \leq b_m + M(1 - z_{it}), \ \forall i \in [n], t \in L, m \in L(t) \tag{5g}$$

$$\sum_{t \in L} z_{it} = 1, \forall i \in [n] \tag{5h}$$

$$z_{it} \leq l_t \forall i \in [n], t \in L \tag{5i}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in L \tag{5j}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in B \tag{5k}$$

$$d_t \leq d_{p(t)}, \ \forall t \in B \setminus \{1\} \tag{5l}$$

$$z_{it}, l_t, a_{jt}, d_t \in \{0, 1\} \tag{5m}$$

- Constraint (b) encodes the loss (prediction error) for training point i
- Constraints (c) and (d) ensure that the prediction for each training point is equal to the prediction of the leaf it is assigned to
- Constraint (e) encodes the complexity of the tree by setting it equal to number of splits in the tree
- Constraints (f) and (g) ensure that the assignment of training points to leaves is consistent with the feature values of the training points and the split values in the tree
- Constraint (h) ensures each training point is assigned to exactly one leaf
- Constraint (i) ensures $l_t = 1$ if there are any points assigned to leaf t
- Constraint (j) ensures each leaf with points assigned to it has at least N_{min} points
- Constraint (k) ensures each potential split uses 1 variable if the split exists and 0 variables if the split does not exist
- Constraint (l) ensures that we can only make a split at a node if the parent node has a split