Controlling for Monotonicity in Random Forest Regressors

Mark Seligman

Suiji

May 17, 2016
Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
- Background and motivation
- Arborist
- Training solution
- Simulated data
- Benchmark data
- Conclusions, future work.
Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
Binary decision trees, briefly

- Prediction method posing series of T/F questions about data set.
  - I.e., questions about observations of given predictors.
- Answer determines which (of two) questions to pose next: branch.
  - E.g., \( p \leq 1.0? \) : pose “left” question, else pose “right”.
  - Here, “pose” can be thought of as branching.
- Different data traverse different paths through the tree.
- Terminal (or “leaf”) node in path reports score for that path.
- Can build single tree and refine: “boosting”.
- Can build “forest” of (typically) 100 – 1000 trees.
  - Forest-wide score derived from aggregate of all trees’ scores.
Random Forests, in a nutshell

- Trademarked by Leo Breiman (dec.) and Adele Cutler.
- Predicts an outcome vector, either numerical or categorical.
  - Regression forests train numerical-valued trees.
  - Classification forests train category-valued trees.
- Trains on design matrix of observations: columns of predictors.
  - Columns individually either numerical or categorical ("factors").
- Trees trained on randomly-selected ("bagged") set of matrix rows.
- Predictors sampled randomly throughout training
  - Separately chosen for each node: \textit{mtry}.
- Validation on held-out subset: different for each tree.
- Independent prediction on separately-provided test sets.
- Focus here is on regression forests, numerical predictors.
Motivation

- Algorithm involves two levels of sampling.
  - Sampling original response to train individual tree.
  - Sampling predictors to split a node: $mtry$.
- Sampling helps reduce bias.
- But, also promotes variation among trees as estimators.
- Couple this with noise in the data itself:
  - Trees may predict values counter to knowledge or expectation.
- Is it possible to constrain training to expectation?
- Will show:
  - In the case of monotonicity: yes.
  - In fact, only a minor modification of existing algorithm.
Monotonicity

- Modeller may presume “hard” physical principle to hold.
  - E.g., signal intensity decreases with distance.
  - Predictive quality suspect if contradicted.
  - May wish to preclude from training.

- Might, alternatively, expect “soft” relationships to hold.
  - E.g., LGD increases with loan-to-value ratio.
  - Exceptions may or may not suggest poor predictive quality.
  - May wish to control in training.

- Monotonicity here either desirable or essential.
  - “Classical” RF appears not to offer constraining mechanism.
Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
Arborist

- General-purpose Random Forest (TM) implementation.
  - Regression, including quantiles.
  - Classification.
  - Numeric, factor predictors: no \textit{a priori} limit on cardinality.
- Tunable to commodity hardware: e.g., multicore, GPU.
- Language-agnostic Core, with separate language “front ends”.
  - \texttt{R} sets the standard.
  - \texttt{Python} under development.
  - Julia?
- Easily extended to support new features.
  - This talk emphasizes extensibility.
Arborist, cont.

- Currently in-memory only.
  - Out-of-core support planned.
- Scales well with sample count.
- Predictor scaling more nuanced:
  - Better scaling when predictor occupancy high:
  - i.e., $mtry$ as a fraction of predictor count.
  - See Wright and Ziegler [to appear].
- Number two position on airline-data benchmark.
Rborist package, version 0.1-1, now on CRAN

- Repairs errors in separate prediction.
- Reduces memory footprint.
- Improves core occupancy during parallel execution.
- New features and improvements, including:
  - Quantile training now default.
  - Case weighting, with auto mode for unbalanced data.
  - mtry semantics now default for low-predictor regimes.
  - preTrain option caches state for iterative workflows.
  - Supports forestFloor feature-contribution package.
  - Supported by Caret.
  - Monotonic regression, the subject of this talk.

- Infrastructural support for future releases: more later.
Training

- Helpful to think of individual tree nodes as sets of samples.
  - Begin by sampling original response vector, with or w/o replacement.
  - Root node "is" this sampled set of values.
  - Every node either splits into two successors or terminates.
  - Splitting effects a bipartition of a node’s sample set.
    Successor nodes are assigned complementary subsets.
    $n$ samples: $\mathcal{O}(2^n)$ potential assignments.
    How an assignment is selected will be shown.

- Training splits the root set in this way until exhausted.

- The leaves of a trained tree together partition its root set.
Prediction

- Regression defines a leaf’s score to be the mean of its sample values.
- Recall that prediction walks a data-dependent path down a tree.
  - The predicted value, then, is the score of the terminal leaf.
- A forest-wide prediction is the mean over all tree predictions.
- Quantile regression proceeds similarly.
  - Each leaf reports a collection of values, not just a mean.
  - The forest yields a collection of values, easily binned.
  - Quantiles can be inferred directly from the bins.
Regression: numeric splits

- Numeric splits involve a predictor and an order predicate on it.
  - E.g., predictor \( p \) with question “Is \( p < 3.1 \)?”
  - Branching direction at prediction determined by predicate’s value.
  - But how does training derive the predicate?

- Training evaluates trial predicates on the sample sets of a node.
- A trial optimal with respect to some metric is selected.
  - Over a random collection of predictors.
  - On all \textit{conforming} bipartitions of the node’s sample set.

- Numerical trial predicates, again, are order relationships.
  - Conforming bipartitions equivalent to cuts in ordered values.
  - For \( n \) samples, a predictor has at most \( n - 1 \) unique cuts.
  - For numeric regressors, then, splitting is \( O(n) \).
Node splitting involves both the sample set and its observations.
  ▶ *Observation* ordering dictates the biparition.
    I.e., which samples map L/R.
  ▶ *Sample* values yield information metric on the trial partition.
    Standard metric for numeric regressors is weighted sample variance.

Rejection scheme: do not accept trials violating the constraint.

Recall, subset assignments already in hand for each trial.

Will show:
  ▶ Straightforward to derive successor contributions from the assignments.
  ▶ In particular, constraint reuses information already at hand.
In symbols

Denote the original response vector as \( y = (y_0, y_1, y_2, \ldots, y_n)^T \).
Let \( T \) be a node with sample set \( S(T) \).
Denote the sample set cardinality by

\[
t \equiv |S(T)|.
\]

Sample set cardinality is conserved by (left, right) trial successors \( L, R \):

\[
l \equiv |S(L)|, \quad r \equiv |S(R)|; \quad l + r = t
\]

The impurity of \( T \) is a measure of variance (Ishwaran [2015]):

\[
\mathcal{I}(T) \equiv \frac{1}{t} \sum_{i \in S(T)} (y_i - \bar{y}_T)^2
\]
Optimality

Optimal trial pairs maximize weighted-variance impurity difference:

\[ I(T) - t^{-1} (lI(L) + rI(R)), \]

which can be shown to be equivalent to maximizing:

\[ l^{-1} \left( \sum_{i \in L} y_i \right)^2 + r^{-1} \left( \sum_{i \in R} y_i \right)^2 \]  \hspace{1cm} (1)

Recall that the respective left and right contributions are given by:

\[ \bar{y}_L = l^{-1} \sum_{i \in L} y_i \quad \text{and} \quad \bar{y}_R = r^{-1} \sum_{i \in R} y_i. \]  \hspace{1cm} (2)
Splitting, II

So monotonicity is enforced by maximizing (1), constraining for (2). Speed up by precomputing $\text{sum}_T$ and accumulating $\text{sum}_L$:

$$\text{sum}_T \equiv \sum_{i \in T} y_i, \quad \text{sum}_L \equiv \sum_{i \in L} y_i. \quad (3)$$

For example, (1) then simplifies to:

$$l^{-1}\text{sum}_L^2 + (t - l)^{-1} (\text{sum}_T - \text{sum}_L)^2$$

The nondecreasing constraint, for example, is:

$$\bar{y}_L \leq \bar{y}_R, \quad \text{or}$$

$$l^{-1}\text{sum}_L \leq (t - l)^{-1} (\text{sum}_T - \text{sum}_L) \quad (4)$$
Putting it all together

- A node is split by visiting a random subcollection of the predictors.
- For each selected predictor, the sample set is visited in *predictor* order.
- Numerical regressors: conforming bipartitions based on order.
  - Conforming left/right trials are characterized by the cuts.
- Optimal trials maximize (3), i.e., “standard” RF behavior.
- Constrained trials satisfy (4).
- Optimal trials not meeting the constraint are rejected.
Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
regMono option

- Vector of probabilities, with sign.
- Length == # predictors.
- Default is 0 for all predictors, if not passed.
- Values are probabilities, $p$, with sign indicating direction.
  - $p > 0$: nondecreasing.
  - $p == 0$: no constraint.
  - $p < 0$: nonincreasing.
  - $|p| \in [0, 1]$: reject with probability $p$.

- Will complain if:
  - Classification.
  - Vector length does not match predictor count.
  - Nonzero entries for factors.
  - $p \notin [-1, 1]$. 
Usage

```r
# Width == predictor count.
constraint <- rep(0.0, ncol(x))

# Predictor 5 rejects nondecreasing paths a.s.:
constraint[5] <- -1.0

# Predictor 2 rejects increasing paths with
# probability 0.7:
constraint[2] <- 0.7

rb <- Rborist(x, y, regMono = constraint)
```
Outline

1. Background and motivation
2. Arborist
3. RegMono
4. Simulated data
5. Boston housing data
6. Conclusions, future work
Strategy

Select several linear and quadratic transformations.
Choose several different dimensions, $n$.
Choose several magnitudes of noise jitter.
Apply 10,000 times per selected transformation:

- Sample points in $[0, 1]^n$ uniformly.
- Apply selected transformation.
- Jitter with Gaussian noise, selected variance.
- Train on resulting data set, with and w/o constraint.
  - Constraint parameter always 1.0, for simplicity.
- Separate prediction at 20 predetermined points.

Biases should be low in either case, unless constraints detrimental.
Variances are the quantities of interest.
Variance vs. distance from response center.
Transformation: $y = 0.1 + 2.5x_1 + 2.5x_2$
Bias vs. distance from response center.
Transformation: \( y = 0.1 + 2.5x_1 + 2.5x_2 \)
Variance vs. distance from response center.
Transformation: $y = 0.1 + 0.3x_1 + 0.8x_2 + 1.2x_3 + 1.6x_4 + 2.2x_5 + 2.7x_6 + 3.1x_7 + 3.5x_8 + 3.8x_9 + 4.2x_{10}$
Bias vs. distance from response center.
Transformation: \[ y = 0.1 + 0.3x_1 + 0.8x_2 + 1.2x_3 + 1.6x_4 + 2.2x_5 + 2.7x_6 + 3.1x_7 + 3.5x_8 + 3.8x_9 + 4.2x_{10} \]
Variance vs. distance from response center.
Transformation: $y = 0.1 + 2.5x_1^2 + 2.5x_2^2$
Variance vs. distance from response center.
Transformation: \( y = 0.1 - 2.5x_1 - 2.5x_2 \)

Unconstrained (blue) vs monotone (red) 100 repeats: sd = 1
Bias vs. distance from response center.
Transformation: $y = 0.1 - 2.5x_1 - 2.5x_2$
1 Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
Boston housing data

Well-known benchmark appearing in ML repositories: Lichman [2013]. Data gleaned from census reports from greater Boston area. 13 demographic and spatial covariates.
Median housing prices of 506 census tracts.
Originally studied by Harrison and Rubinfeld [1978].
## Covariate descriptions

<table>
<thead>
<tr>
<th>Covariate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crim</td>
<td>local per-capita crime rate</td>
</tr>
<tr>
<td>zn</td>
<td>proportion of residential land zoned over 25,000 sq.ft</td>
</tr>
<tr>
<td>indus</td>
<td>proportion of town area with non-retail business</td>
</tr>
<tr>
<td>chas</td>
<td>adjacency to Charles river: binary encoding</td>
</tr>
<tr>
<td>nox</td>
<td>concentration of NO in parts-per-10-million</td>
</tr>
<tr>
<td>rm</td>
<td>average number of rooms</td>
</tr>
<tr>
<td>age</td>
<td>proportion owner-occupied units built pre-1940</td>
</tr>
<tr>
<td>dis</td>
<td>weighted mean distances to nearby employment centers</td>
</tr>
<tr>
<td>rad</td>
<td>index of accessibility to radial highways</td>
</tr>
<tr>
<td>tax</td>
<td>full-value property-tax rate per $10,000</td>
</tr>
<tr>
<td>pratio</td>
<td>pupil-teacher ratio by town</td>
</tr>
<tr>
<td>black</td>
<td>weighting of local proportion of African-Americans</td>
</tr>
<tr>
<td>lstat</td>
<td>% population deemed “lower status”</td>
</tr>
</tbody>
</table>
## Correlation with price

<table>
<thead>
<tr>
<th>Covariate</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lstat</td>
<td>-0.7376627</td>
</tr>
<tr>
<td>ptratio</td>
<td>-0.5077867</td>
</tr>
<tr>
<td>indus</td>
<td>-0.4837252</td>
</tr>
<tr>
<td>tax</td>
<td>-0.4685359</td>
</tr>
<tr>
<td>nox</td>
<td>-0.4273208</td>
</tr>
<tr>
<td>crim</td>
<td>-0.3883046</td>
</tr>
<tr>
<td>rad</td>
<td>-0.3816262</td>
</tr>
<tr>
<td>age</td>
<td>-0.3769546</td>
</tr>
<tr>
<td>chas</td>
<td>0.1752602</td>
</tr>
<tr>
<td>dis</td>
<td>0.2499287</td>
</tr>
<tr>
<td>black</td>
<td>0.3334608</td>
</tr>
<tr>
<td>zn</td>
<td>0.3604453</td>
</tr>
<tr>
<td>rm</td>
<td>0.6953599</td>
</tr>
</tbody>
</table>
Constraints

Highest correlate: concentration of “low status” residents.
  ▶ Look for decreasing trend.
Room count also has high correlation.
  ▶ Look for increasing trend.
Crime, far from highest, might also contribute monotonically.
  ▶ Look for decreasing trend.
Hold-out experiments

- Repeat on all combinations of the three covariates:
- Apply 500 times, with and without constraint.
  - Train with random 20% of samples held out.
  - Predict held-out samples using trained forest.
- Compare MSE of predictions.
  - One-sided Wilcoxon paired test.
  - $H_0$ : unconstrained MSE $\geq$ constrained.
Boston: results

<table>
<thead>
<tr>
<th>Constraint</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>- crim</td>
<td>0.2392</td>
</tr>
<tr>
<td>+ rm</td>
<td>8.389e-14</td>
</tr>
<tr>
<td>- lstat</td>
<td>0.3015</td>
</tr>
<tr>
<td>- crim + rm</td>
<td>0.3517e-09</td>
</tr>
<tr>
<td>+ rm - lstat</td>
<td>1.489e-10</td>
</tr>
<tr>
<td>- crim - lstat</td>
<td>0.9554</td>
</tr>
<tr>
<td>- crim + rm - lstat</td>
<td>1.559e-10</td>
</tr>
</tbody>
</table>

- Only # rooms clearly benefits alone from monotone constraint.
  - Yet low-status covariate is more highly correlated.
  - Low status, in particular, may be effect rather than cause.

- Other two do benefit, but only in combination with # rooms.
Stochastic control

- Also examined effect of probability threshold.
- Same scheme as above.
- Following table considers room count alone:

<table>
<thead>
<tr>
<th>Rooms constraint: probability</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.1485</td>
</tr>
<tr>
<td>0.4</td>
<td>0.001191</td>
</tr>
<tr>
<td>0.6</td>
<td>4.743e-09</td>
</tr>
<tr>
<td>0.8</td>
<td>6.732e-11</td>
</tr>
<tr>
<td>1.0</td>
<td>8.389e-14</td>
</tr>
</tbody>
</table>
Outline

2 Background and motivation

3 Arborist

4 RegMono

5 Simulated data

6 Boston housing data

7 Conclusions, future work
Conclusions

- Monotonic constraints are straightforward to implement in RF.
- Can reduce variance when *a priori* evidence suggests their use.
- Can improve predictive quality in some cases.
- Correlation is not necessarily a guide to employing them.
Constraint-based regression

- Monotonic regression should extend to other tree-based approaches.
  - CART
  - PRIM
  - Gradient boosting.

- Note: **GBM** package offers a monotone option. (Scooped?)

- May extend to shape-based regression.
  - Rejection scheme, while simple, is a blunt instrument.
  - May benefit from utilizing *distribution* of sampled values.
  - Inference on convexity, for example.
  - Computationally more expensive, but software changes strictly local.
Rborist: nearer term

- Greedy restaging replaced by “patient” scheme.
  - Improves performance at medium/low predictor occupancy.
- Sparse data representations.
  - Infrastructural changes mostly complete.
- Specialized GPU version.
  - On-coprocessor restaging.
    - Experimental package completed.
    - High break-even point: > 50,000 rows.
    - Communication costs dominate.
  - On-coprocessor restaging + splitting.
    - Little/no communication cost.
    - Regression under development.
    - Factors a little tricky, especially at high cardinality.
    - Binary classification to follow, or accompany.
    - Full classification further out.
**Rborist, longer term**

- Cluster implementations.
  - Infrastructure in place to train tree blocks independently.
  - Head node / work node model.
  - MPI or Spark/Hadoop hooks still needed.

- Out-of-memory support.
  - General attack mapped out.
  - Combination of tiling and streaming approaches.

- Near-memory computing (much longer term).
  - Exciting features anticipated from 3D memory cubes.
  - In-memory transformations, such as sorting.
  - Could greatly diminish some key performance bottlenecks.
Acknowledgments

- Pr. Jean Opsomer, Colorado State University.
- Steve Miller and Ryan Ballantine.
- Szilard Pafka.
- Jeffry Howbert, formerly Zillow.
- Pr. Dr. Andreas Ziegler, Universität zu Lübeck.
- Christopher Brown.
- Grady Lemoine.

