RFCDE: Random Forests for Conditional Density Estimation

Taylor Pospisil and Ann B. Lee

Department of Statistics & Data Science
Carnegie Mellon University
Pittsburgh, PA 15289, USA

Abstract

Random forests is a common non-parametric regression technique which performs well for mixed-type data and irrelevant covariates, while being robust to monotonic variable transformations. Existing random forest implementations target regression or classification. We introduce the `RFCDE` package for fitting random forest models optimized for nonparametric conditional density estimation, including joint densities for multiple responses. This enables analysis of conditional probability distributions which is useful for propagating uncertainty and of joint distributions that describe relationships between multiple responses and covariates. `RFCDE` is released under the MIT open-source license and can be accessed at `https://github.com/tpospisi/rfcde`. Both R and Python versions, which call a common C++ library, are available.

1 Introduction and Motivation

Conditional density estimation (CDE) is the estimation of the density $f(z \mid X = x)$ where we condition the response Z on observed covariates X. In a prediction context, CDE provides a more nuanced accounting of uncertainty than a point estimate or prediction interval, especially in the presence of heteroskedastic or multimodal responses. These conditional densities can be used to propagate uncertainty through further analyses or to minimize expected prediction loss for non-standard loss functions.

Our main contribution in `RFCDE` is two-fold: (1) we provide software for extending random forest estimation to conditional density estimation including joint densities for multivariate responses, and (2) we fit our trees based upon minimizing conditional density estimation loss. This overcomes the limitations of the usual regression loss functions due to heteroskedasticity and multimodality while still remaining computationally feasible.

We extend random forests Breiman 2001 to CDE, inheriting the benefits of random forests with respect to mixed-data types, irrelevant covariates, and data transformations. We take advantage of the fact that random forests can be viewed as a form of adaptive nearest-neighbor method with the aggregated tree structures determining a weighting scheme. This weighting scheme can then be used to estimate quantities other than the conditional mean; in this case the conditional density.

Other existing random forest implementations such as `quantileregressionForests` Meinshausen 2006 and `trtf` Hothorn and Zeileis 2017 can be used for CDE. In Section 3.3 we show that our method achieves lower CDE loss in competitive computational time.

For ease of use in the statistics and machine-learning communities, we provide packages in both R and Python. Both packages call a common C++ library which implements the core training functions for performance and avoidance of redundancies. The core library can easily be wrapped in other languages. Source code and a bug tracker can be found at `https::/github.com/tpospisi/rfcde`.

2 Overview

To construct our estimators we largely follow the usual random forest construction. At their simplest, random forests are ensembles of regression trees. Each tree is trained on a bootstrapped sample of the data. The training process involves recursively partitioning the covariate space through splitting rules taking the form of splitting into the sets $\{X_i \leq v\}$ and $\{X_i > v\}$ for a particular covariate X_i and split point v. Once a partition becomes small enough (controlled by the tuning parameter `nodesize`), it becomes a leaf node and is no longer partitioned.

For prediction we use the tree structure to calculate weights for the training data from which we perform a *weighted kernel density estimate* using "nearby" points. Borrowing the notation of Breiman 2001 and Meinshausen 2006, let θ_t denote the tree structure for a single tree. Let $R(x, \theta_t)$ denote the region of covariate space covered by the leaf node for input x. Then for a new observation x^* we use t-th tree to calculate weights for each training point x_i as

$$w_i(x^*, \theta_t) = \frac{\mathbb{1}\left[X_i \in R(x^*, \theta_t)\right]}{\sum_{i=1}^{n} \mathbb{1}\left[X_i \in R(x^*, \theta_t)\right]}.$$

We then aggregate over trees setting $w_i(x^*) = T^{-1} \sum_{t=1}^{T} w_i(x^*, \theta_t)$. The weights are finally used for the weighted kernel density estimate.

$$\widehat{f}(z \mid x^*) = \frac{1}{\sum_{i=1}^{n} w_i(x^*)} \sum_{i=1}^{n} w_i(x^*) K_h(Z_i - z) \tag{1}$$

where K_h is a kernel function integrating to one.

Our main departure from the standard random forest algorithm is choosing the splits of the partitioning. In regression contexts, the splitting variable and

split point are often chosen to minimize the mean-squared error loss. For CDE, we instead choose to minimize a *loss specific to CDE* [Izbicki and Lee, 2017]

$$L(f, \widehat{f}) = \int \int \left(f(z \mid x) - \widehat{f}(z \mid x) \right)^2 \mathrm{d}z \, \mathrm{d}P(x).$$

This loss is the L^2 error for density estimation weighted by the marginal density of the covariates. To conveniently estimate this loss we can expand the square and rewrite the loss as

$$L(f, \widehat{f}) = \mathbb{E}_X \left[\int \widehat{f}^2(z \mid X) \, \mathrm{d}z \right] - 2 \, \mathbb{E}_{X,Z} \left[\widehat{f}(Z \mid X) \right] + C_f \tag{2}$$

with C_f as a constant which doesn't depend on $\widehat{f}$. The first expectation is with respect to the marginal distribution of X and the second with respect to the joint distribution of X and Z. We estimate these expectations by their empirical expectation on observed data. While we use kernel density estimates for predictions on new observations, we do not use kernel density estimates when evaluating splits because of computationally expensive calculations in Equation 2 due to the dependence of $\widehat{f}$ on the $O(n^2)$ pairwise-distances between all training points.

For fast computations, we instead use *orthogonal series* to compute density estimates for splitting. Given an orthogonal basis $\{\Phi_j(z)\}$ such as a cosine basis or wavelet basis, we can express the density as $\widehat{f}(z \mid x) = \sum_j \widehat{\beta}_j \phi_j(z)$ where $\widehat{\beta}_j = \frac{1}{n} \sum_{i=1}^n \phi_j(z_i)$. This choice is motivated by a convenient formula for the CDE loss associated with an orthogonal series density estimate

$$\widehat{L}(f, \widehat{f}) - C_f = -\sum_j \widehat{\beta}_j^{\,2}.$$

The above expression only depends upon the quantities $\left\{ \widehat{\beta}_j \right\}$ that themselves depend only upon *linear* sums of $\phi_j(z_i)$. This makes it computationally efficient to evaluate the CDE loss for each split.

3 RFCDE

We provide `RFCDE` packages for R and Python. To avoid redundancy and achieve better performance, the main model fitting code is written as a common C++ library. We use `Rcpp` [Eddelbuettel and François, 2011] and `Cython` [Behnel et al., 2011] to write wrappers for R and Python. Vignettes are included on Github for each language.

3.1 Usage

The R and Python packages expose three main functions:

- `RFCDE(x_train, z_train, ...)` trains the random forest and returns a wrapped C++ object. There are several tuning parameters; for details see the documentation on Github.
- `predict(x_new, z_grid, bandwidth)` uses the fitted forest to estimate the weighted kernel density estimate for all points in `z_grid`.
- `weights(forest, x_new)` which uses the fitted forest to calculate weights for each new observation. This is used internally in `predict` but could be adapted for other purposes.

3.2 Univariate Experiment

To illustrate the difference that training the random forest to minimize a CDE loss can have on performance, we will compare against an existing random forest implementation (that minimizes MSE loss). We adapt the `quantregForest` Meinshausen, 2006 package in R to perform conditional density estimation according to Equation 1. This is equivalent to our method except that the splits for `quantregForest` minimize mean-squared error rather than CDE loss. This provides a useful comparison to demonstrate the advantages of specifically training random forests for the goal of conditional density estimation. We also compare against the `trtf` package Hothorn and Zeileis, 2017 which trains forests for CDE using flexible parametric families.

We generate data from the following model

$$X_{1:10}, Y_{1:10} \sim \text{Uniform}(0,1), \quad S \sim \text{Multinomial}(2, \frac{1}{2})$$
$$Z \mid X, Y, S \sim \begin{cases} \text{Normal}(\lfloor \sum_i X_i \rfloor, \sigma) & S = 1 \\ \text{Normal}(-\lfloor \sum_i X_i \rfloor, \sigma) & S = 2 \end{cases}$$

where the X_i are the relevant covariates with the Y_i serving as irrelevant covariates. S is an unobserved covariate which induces multimodality in the conditional densities.

Under the true model $\mathbb{E}[Z \mid X, Y] = 0$, so there is no partitioning scheme that can reduce the MSE loss (as the conditional mean is always zero). As such, the trained `quantregForest` trees behave similarly to nearest neighbors as splits are effectively chosen at random. We evaluate the two methods on two criterion: training time and CDE loss on a validation set. All models are tuned with the same forest parameters (`mtry` $= 4$, `ntrees` $= 1000$). `RFCDE` has `n_basis` $= 15$. `trtf` is fit using a Bernstein basis of order 5. `RFCDE` and `quantregForest` have bandwidth 0.2.

Table 1 summarizes the results for three simulations of size 1000, 10000, and 100000 training observations. We use 1000 observations for the test set and calculate the CDE loss. We see that `RFCDE` performs substantially better on CDE loss. We also note that `RFCDE` has competitive training time especially for larger data sets. `trtf` is only run for the smallest data set due to its execution time.

Table 1: Performance of `RFCDE` and competing methods; smaller CDE loss implies better estimates.

Method	N	CDE Loss (SE)	Train Time (seconds)	Predict Time (seconds)
`RFCDE`	1,000	**-0.171** (0.004)	2.31	1.29
`quantregForest`	1,000	-0.152 (0.003)	**1.17**	**0.61**
`trtf`	1,000	-0.109 (0.001)	1013.54	123.47
`RFCDE`	10,000	**-0.194** (0.003)	**42.99**	1.95
`quantregForest`	10,000	-0.159 (0.003)	48.60	**0.47**
`RFCDE`	100,000	**-0.227** (0.004)	**904.76**	8.05
`quantregForest`	100,000	-0.173 (0.003)	1289.93	**0.61**

3.3 Joint Experiment

Another feature of `RFCDE` is the ability to target joint conditional density loss. The splitting process extends straightforwardly to the multivariate case through the use of a tensor basis. The density estimation similarly extends through the use of multivariate kernel density estimation. This allows for conditional density estimation in multiple dimensions (although in practice only two or three dimensions are computationally practical due to the limitations of the tensor basis). To showcase our model, we draw 10000 samples from the following distribution:

$$X \sim \text{Uniform}(0,1) \quad Z_1 \sim \text{Uniform}(0,X) \quad Z_2 \sim \text{Uniform}(X,Z).$$

We fit our model with parameters `ntrees` $= 1000$, `mtry` $= 1$, `nodesize` $= 20$, and `n_basis` $= 15$. The bandwidth is selected adaptively with respect to x^* for each density.

We cannot straightforwardly adapt `quantregForest` or `trtf` software to joint densities so we will not compare against them. Instead we provide a qualitative assessment of performance in Figure 1. The joint densities, and in particular the dependency between Z_1 and Z_2, are captured well.

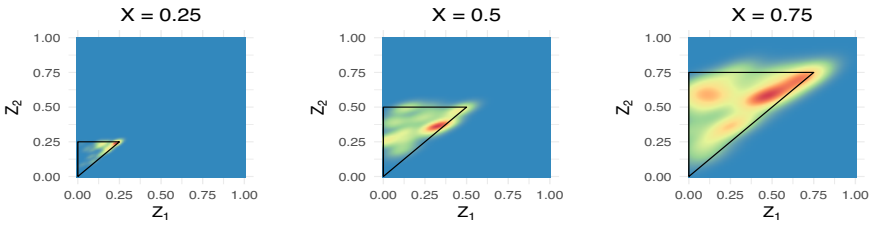

Figure 1: Joint conditional density estimates; the test values of X are listed above each joint density. We see that our method captures the joint dependence between the two responses as shown by the black lines which indicate the true support of the joint conditional density.

4 Conclusions

We provide `RFCDE`, a multi-language tool for fitting conditional density estimation (CDE) random forests for complex data and more than one response variable. `RFCDE` extends random forests by implementing a computationally efficient method for minimizing a CDE loss in each split. R and Python packages are available along with a common C++ library at `https://github.com/tpospisi/rfcde`.

Acknowledgements

We are grateful to Rafael Izbicki for helpful discussions and comments on the paper. This work was partially supported by NSF DMS-1520786.

References

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13 (2):31–39, 2011.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. doi: 10.18637/jss.v040.i08. URL `http://www.jstatsoft.org/v40/i08/`.

T. Hothorn and A. Zeileis. Transformation forests. *arXiv:1701.02110*, 2017.

R. Izbicki and A. B. Lee. Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronic Journal of Statistics*, 11(2):2800–2831, 2017.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.