Forest-type Regression with General Losses and Robust Forest

Alexander Hanbo Li [1] **Andrew Martin** [2]

Abstract

This paper introduces a new general framework for forest-type regression which allows the development of robust forest regressors by selecting from a large family of robust loss functions. In particular, when plugged in the squared error and quantile losses, it will recover the classical random forest (Breiman, 2001) and quantile random forest (Meinshausen, 2006). We then use robust loss functions to develop more robust forest-type regression algorithms. In the experiments, we show by simulation and real data that our robust forests are indeed much more insensitive to outliers, and choosing the right number of nearest neighbors can quickly improve the generalization performance of random forest.

1. Introduction

Since its development by Breiman (2001), random forest has proven to be both accurate and efficient for classification and regression problems. In regression setting, random forest will predict the conditional mean of a response variable by averaging predictions of a large number of regression trees. Later then, many other machine learning algorithms were developed upon random forest. Among them, robust versions of random forest have also been proposed using various methodologies. Besides the sampling idea (Breiman, 2001) which adds extra randomness, the other variations are mainly based on two ideas: (1) use more robust criterion to construct regression trees (Galimberti et al., 2007; Brence & Brown, 2006; Roy & Larocque, 2012); (2) choose more robust aggregation method (Meinshausen, 2006; Roy & Larocque, 2012; Tsymbal et al., 2006).

Meinshausen (2006) generalized random forest to predict quantiles by discovering that besides calculating the weighted mean of the observed response variables, one could also get information for the weighted distribution of observed response variables using the sets of local weights generated by random forest. This method is strongly connected to the adaptive nearest neighbors procedure (Lin & Jeon, 2006) which we will briefly review in section 1.2. Different from classical k-NN methods that rely on predefined distance metrics, the dissimilarities generated by random forest are data dependent and scale-invariant.

Another state-of-the-art algorithm AdaBoost (Freund & Schapire, 1995; Freund et al., 1996) has been generalized to be applicable to a large family of loss functions (Friedman, 2001; Mason et al., 1999; Li & Bradic, 2016). Recent development of more flexible boosting algorithms such as xgboost (Chen & Guestrin, 2016) have become the go-to forest estimators with tabular or matrix data. One way in which recent boosting algorithms have an advantage over the random forest is the ability to customize the loss function used to reduce the influence of outliers or optimize a metric more suited to the specific problem other than the mean squared error.

In this paper, we will propose a general framework for forest-type regression which can also be applied to a broad family of loss functions. It is claimed in (Meinshausen, 2006) that quantile random forest is another nonparametric approach which does not minimize an empirical loss. However, we will show in fact both random forest and quantile random forest estimators can be re-derived as regression methods using the squared error or quantile loss respectively in our framework. Inspired by the adaptive nearest neighbor viewpoint, we explore how random forest makes predictions using the local weights generated by ensemble of trees, and connect that with locally weighted regression (Fan & Gijbels, 1996; Tibshirani & Hastie, 1987; Staniswalis, 1989; Newey, 1994; Loader, 2006; Hastie & Loader, 1993). The intuition is that when predicting the target value (e.g. $\mathbb{E}[Y|X = x]$) at point x, the observations closer to x should receive larger weights. Different from predefining a kernel, random forest assigns the weights data dependently and adaptively. After we illustrate the relation between random forest and local regression, we will use random forest weights to design other regression algo-

[1]University of California at San Diego, San Diego, California, USA [2]Zillow, Seattle, Washington, USA. Correspondence to: Alexander Hanbo Li <alexanderhanboli@gmail.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

rithms. By plugging robust loss functions like Huber loss and Tukey's redescending loss, we get forest-type regression methods that are more robust to outliers. Finally, motivated from the truncated squared error loss example, we will show that decreasing the number of nearest neighbors in random forest will also immediately improve its generalization performance.

The layout of this paper is as follows. In Section 1.1 and 1.2 we review random forest and adaptive nearest neighbors. Section 2 introduces the general framework of forest-type regression. In Section 3 we plug in robust regression loss functions to get robust forest algorithms. In Section 4 we motivate from the truncated squared error loss and investigate the importance of choosing right number of nearest neighbors. Finally, we test our robust forests in Section 5 and show that they are always superior to the traditional formulation in the presence of outliers in both synthetic and real data set.

1.1. Random forest

Following the notation of Breiman (2001), let θ be the random parameter determining how a tree is grown, and data $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. For each tree $T(\theta)$, let L be the total number of leaves, and R_l denotes the rectangular subspace in $\mathcal{X}$ corresponding to the l-th leaf. Then for every $x \in \mathcal{X}$, there is exactly one leaf l such that $x \in R_l$. Denote this leaf by $l(x, \theta)$.

For each tree $T(\theta)$, the prediction of a new data point $X = x$ is the average of data values in leaf $l(x, \theta)$, that is, $\widehat{Y}(x, \theta) = \sum_{j=1}^{n} w(X_i, x, \theta) Y_i$, where

$$w(X_i, x, \theta) = \frac{\mathbb{1}_{\{X_i \in R_{l(x,\theta)}\}}}{\#\{j : X_j \in R_{l(x,\theta)}\}}. \tag{1}$$

Finally, the conditional mean $\mathbb{E}[Y|X = x]$ is approximated by the averaged prediction of m trees, $\widehat{Y}(x) = m^{-1} \sum_{t=1}^{m} \widehat{Y}(x, \theta_t)$. After rearranging the terms, we can write the prediction of random forest as

$$\widehat{Y}(x) = \sum_{i=1}^{n} w(X_i, x) Y_i, \tag{2}$$

where the averaged weight $w(X_i, x)$ is defined as

$$w(X_i, x) = \frac{1}{m} \sum_{t=1}^{m} w(X_i, x, \theta_t). \tag{3}$$

From equation (2), the prediction of the conditional expectation $\mathbb{E}[Y|X = x]$ is the weighted average of the response values of all observations. Furthermore, it is easy to show that $\sum_{i=1}^{n} w(X_i, x) = 1$.

1.2. Adaptive nearest neighbors

Lin and Jeon (2006) studies the connection between random forest and adaptive nearest neighbor. They introduced the so-called potential nearest neighbors (PNN): A sample point $\mathbf{x}_i$ is called a k-PNN to a target point $\mathbf{x}$ if there exists a monotone distance metric under which $\mathbf{x}_i$ is among the k closest to $\mathbf{x}$ among all the sample points.

Therefore, any k-NN method can be viewed as choosing k points from the k-PNNs according to some monotone metric. For example, under Euclidean metric, the classical k-NN algorithm sorts the observations by their Euclidean distances to the target point and outputs the k closest ones. This is equivalent to weighting the k-PNNs using inverse L_2 distance.

More interestingly, they prove that those observations with positive weights (3) all belong to the k-PNNs (Lin & Jeon, 2006). Therefore, random forests is another weighted k-PNN method, but it assigns weights to the observations different from any k-NN method under a pre-defined monotonic distance metric. In fact, the random forest weights are adaptive to the data if the splitting scheme is adaptive.

2. General framework for forest-type regression

In this section, we generalize the classical random forest to a general forest-type regression (FTR) framework which is applicable to a broad family of loss functions. In Section 2.1, we motivate the framework by connecting random forest predictor with locally weighted regression. Then in Section 2.2, we formally propose the new forest-type regression framework. In Section 2.3, we rediscover the quantile random forest estimator by plugging the quantile loss function into our framework.

2.1. Squared error and random forest

Classical random forest can be understood as an estimator of conditional mean $\mathbb{E}[Y|X]$. As shown in (2), the estimator $\widehat{Y}(x)$ is weighted average of all response Y_i's. This special form reminds us of the classical least squares regression, where the estimator is the sample mean. To be more precise, we rewrite (2) as

$$\sum_{i=1}^{n} w(X_i, x)(Y_i - \widehat{Y}(x)) = 0. \tag{4}$$

Equation (4) is the estimating equation (first order condition) of the locally weighted least squares regression (Ruppert & Wand, 1994):

$$\widehat{Y}(x) = \operatorname*{argmin}_{\lambda \in \mathbb{R}} \sum_{i=1}^{n} w(X_i, x)(Y_i - \lambda)^2 \tag{5}$$

In classical local regression, the weight $w(X_i, x)$ serves as a local metric between the target point x and observation X_i. Intuitively, observations closer to target x should be given more weights when predicting the response at x. One common choice of such local metric is kernel $K_h(X_i, x) = K((X_i - x)/h)$. For example, the tricube kernel $K(u) = (1 - |u|^3)^3 \mathbb{1}(|u| \leq 1)$ will ignore the impact of observations outside a window centered at x and increase the weight of an observation when it is getting closer to x. The form of kernel-type local regression is as follows:

$$\underset{\lambda \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} K_h(X_i - x)(Y_i - \lambda)^2,$$

The random forest weight $w(X_i, x)$ (3) defines a similar data dependent metric, which is constructed using the ensemble of regression trees. Using an adaptive splitting scheme, each tree chooses the most informative predictors from those at its disposal. The averaging process then assigns positive weights to these training responses, which are called voting points in (Lin & Jeon, 2006). Hence via the random forest voting mechanism, those observations close to the target point get assigned positive weights equivalent to a kernel functionality (Friedman et al., 2001).

2.2. Extension to general loss

Note that the formation (5) is just a special case when using squared error loss $\phi(a, b) = (a-b)^2$. In more general form, we have the following local regression problem:

$$\widehat{Y}(x) = \underset{s \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} w(X_i, x)\phi(s(X_i), Y_i) \tag{6}$$

where $w(X_i, x)$ is a local weight, $\mathcal{F}$ is a family of functions, and $\phi(\cdot)$ is a general loss. For example, when local weight is a kernel and $\mathcal{F}$ stands for polynomials of a certain degree, it reduces to local polynomial regression (Fan & Gijbels, 1996). Random forest falls into this framework with squared error loss, a family of constant functions and local weights (3) constructed from ensemble of trees.

Algorithm 1 Forest-type regression

Step 1: Calculate local weights $w(X_i, x)$ using ensemble or trees.

Step 2: Choose a loss $\phi(\cdot, \cdot)$ and a family $\mathcal{F}$ of function. Then do the locally weighted regression

$$\widehat{Y}(x) = \underset{s \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} w(X_i, x)\phi(Y_i, s(X_i)).$$

In Algorithm 1, we summarize the forest-type regression as a general two-step method. Note that here we only focus on local weights generated by random forest, which uses ensemble of trees to recursively partition the covariate space $\mathcal{X}$. However, there are many other data dependent dissimilarity measures that can potentially be used, such as k-NN, m_p-dissimilarity (Aryal et al., 2014), shared nearest neighbors (Jarvis & Patrick, 1973), information-based similarity (Lin et al., 1998), mass-based dissimilarity (Ting et al., 2016), etc. And there are many other domain specific dissimilarity measures. To avoid distraction, we will only use random forest weights throughout the rest of this paper.

2.3. Quantile loss and quantile random forest

Meinshausen (2006) proposed the quantile random forest which can extract the information of different quantiles rather than just predicting the average. It has been shown that quantile random forest is more robust than the classical random forest (Meinshausen, 2006; Roy & Larocque, 2012). In this section, we show quantile random forest estimator is also a special case of Algorithm 1. It is well known that the τ-th quantile of an (empirical) distribution is the constant that minimizes the (empirical) risk using τ-th quantile loss function $\rho_\tau(z) = z(\tau - \mathbb{1}_{\{z<0\}})$ (Koenker, 2005). Now let the loss function in Algorithm 1 be the quantile loss $\rho_\tau(\cdot)$, $\mathcal{F}$ be the family of constant functions, and $w(X_i, x)$ be random forest weights (3). Solving the optimization problem

$$\widehat{Y}_\tau(x) = \underset{\lambda \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} w(X_i, x)\rho_\tau(Y_i - \lambda),$$

we get the corresponding first order condition

$$\sum_{i=1}^{n} w(X_i, x)(\tau - \mathbb{1}\{Y_i - \widehat{Y}_\tau(x) < 0\}) = 0.$$

Recall that $\sum_{i=1}^{n} w(X_i, x) = 1$, hence, we have

$$\sum_{i=1}^{n} w(X_i, x)\, \mathbb{1}\{Y_i < \widehat{Y}_\tau(x)\} = \tau. \tag{7}$$

The estimator $\widehat{Y}_\tau(x)$ in (7) is exactly the same estimator proposed in (Meinshausen, 2006). In particular, when $\tau = 0.5$, the equation $\sum_{i=1}^{n} w(X_i, x)\, \mathbb{1}\{Y_i < \widehat{Y}_{0.5}(x)\} = 0.5$ will give us the median estimator $\widehat{Y}_{0.5}(x)$. Therefore, we have rediscovered quantile random forest from a totally different point of view as a local regression estimator with quantile loss function and random forest weights.

3. Robust forest

From the framework 1, quantile random forest is insensitive to outliers because of the more robust loss function. In this section, we test our framework on other robust losses and proposed fixed-point method to solve the estimating

equation. In Section 3.1 we choose the famous robust loss – (pseudo) Huber loss, and in Section 3.2, we further investigate a non-convex loss – Tukey's biweight.

3.1. Huber loss

The Huber loss (Huber et al., 1964)

$$H_\delta(y) = \begin{cases} \frac{1}{2}y^2 & \text{for } |y| \le \delta, \\ \delta(|y| - \frac{1}{2}\delta) & \text{elsewhere} \end{cases}$$

is a well-known loss function used in robust regression. The penalty acts like squared error loss when the error is within $[-\delta, \delta]$ but becomes linear outside this range. In this way, it will penalize the outliers more lightly but still preserves more efficiency than absolute deviation when data is concentrated in the center and has light tails (e.g. Normal). By plugging Huber loss into the FTR framework 1, we get a robust counterpart of random forest. The estimating equation is

$$\sum_{i=1}^{n} w_i(x)\,\mathrm{sign}(\widehat{Y}(x) - Y_i)\,\min(\widehat{Y}(x) - Y_i, \delta) = 0. \quad (8)$$

Direct optimization of (8) with local weights is hard, hence instead we will investigate the pseudo-Huber loss (see Figure 1),

$$L_\delta(y) = \delta^2\left(\sqrt{1 + \left(\frac{y}{\delta}\right)^2} - 1\right)$$

which is a smooth approximation of Huber loss (Charbonnier et al., 1997). The estimating equation

$$\sum_{i=1}^{n} w_i^{pH}(x)\left(\widehat{Y}_{pH}(x) - Y_i\right) = 0. \quad (9)$$

is very similar to that of square error loss if we define a new weight

$$w_i^{pH}(x) = \frac{w_i(x)}{\sqrt{1 + \left(\frac{\widehat{Y}_{pH}(x) - Y_i}{\delta}\right)^2}}. \quad (10)$$

Then the (pseudo) Huber estimator can be expressed as

$$\widehat{Y}_{pH}(x) = \frac{\sum_{i=1}^n w_i^{pH}(x) Y_i}{\sum_{i=1}^n w_i^{pH}(x)}. \quad (11)$$

Informally, the estimator (11) can be viewed as a weighted average of all the responses Y_i's. From (10), we know the new weight for pseudo-Huber loss has an extra scaling factor

$$\left(\sqrt{1 + (\delta^{-1}u)^2}\right)^{-1} \quad (12)$$

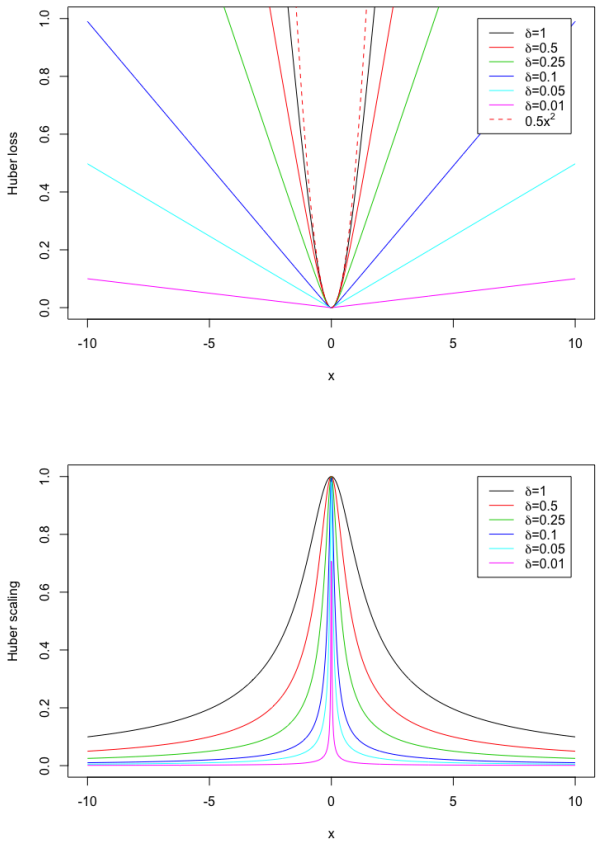

Figure 1. In the first row, we compare squared error loss $\frac{1}{2}x^2$ and pseudo-Huber loss with different δ. In the second row, we plot the scaling factor (12) of Huber loss. We observe that as δ decreases to zero, the Huber loss becomes more linear and flat, and the scaling factor shrinks more quickly as the input deviates from zero.

and hence will shrink more to zero whenever $\delta^{-1}|\widehat{Y}_{pH}(x) - Y_i|$ is large. The tuning parameter δ acts like a control of the level of robustness. A smaller δ will lead to more shrinkage on the weights of data that have responses far away from the estimator.

The estimating equation (9) can be solved by fix-point method which we propose in Algorithm 2. For notation simplicity, we will use $w_{i,j}$ to denote $w(X_i, x_j)$, where X_i is the i-th training point and x_j is the j-th testing point. The convergence to the unique solution (if exists) is guaranteed by Lemma 1.

Lemma 1. *Define*

$$K_\delta(y) = \frac{\sum_{i=1}^n \frac{w_i Y_i}{\sqrt{1+\left(\frac{y-Y_i}{\delta}\right)^2}}}{\sum_{i=1}^n \frac{w_i}{\sqrt{1+\left(\frac{y-Y_i}{\delta}\right)^2}}},$$

Algorithm 2 pseudo-Huber loss (δ)

Input: Test points $\{x_j\}_{j=1}^m$, initial guess $\{\widehat{Y}^{(0)}(x_j)\}$, local weights $w_{i,j}$, training responses $\{Y_i\}_{i=1}^n$, and error tolerance ϵ_0.
while $\epsilon > \epsilon_0$ **do**
(a) Update the weights

$$w_{i,j}^{(k)} = \frac{w_{i,j}}{\sqrt{1+\left(\frac{\widehat{Y}^{(k-1)}(x_j)-Y_i}{\delta}\right)^2}}$$

(b) Update the estimator

$$\widehat{Y}^{(k)}(x_j) = \frac{\sum_{i=1}^n w_{i,j}^{(k)} Y_i}{\sum_{i=1}^n w_{i,j}^{(k)}}$$

(c) Calculate error

$$\epsilon = \frac{1}{m}\sum_{j=1}^m \left(\widehat{Y}^k(x_j) - \widehat{Y}^{(k-1)}(x_j)\right)^2$$

(d) $k \leftarrow k+1$
end while
Output the pseudo-Huber estimator:

$$\widehat{Y}_{pH}(x_j) = \widehat{Y}^{(k)}(x_j)$$

where $\sum_{i=1}^n w_i = 1$. *Let* $K = \max_{i=1,\cdots,n}|Y_i|$. *Then Algorithm 2 can be written as* $\widehat{Y}^{(k)}(x) = K_\delta(\widehat{Y}^{(k-1)})$, *and converges exponentially to a unique solution as long as* $\delta > 2K$.

From Lemma 1, we know it is important to standardize the responses Y_i so that δ will be of the same scale for different problems. In practice, we observe that one will not need to choose δ that satisfies the worst-case condition $\delta > K$ in order for convergence, but making δ too small does lead to slow convergence rate. For assigning the initial guess $\widehat{Y}^{(0)}$, two simplest ways are to either take the random forest estimator we got or a constant vector equaling to the sample mean. Throughout the rest of this paper, we will choose the weights to be random forest weights (3).

3.2. Tukey's biweight

Non-convex function has played an important role in the context of robust regression (Huber, 2011; Hampel et al., 2011). Unlike convex losses, the penalization on the errors can be bounded and hence the contribution of outliers in the estimating equation will eventually vanish. Our forest regression framework 1 also incorporates the non-convex losses which will show through the Tukey's biweight function $T_\delta(\cdot)$ (Huber, 2011), which is an example

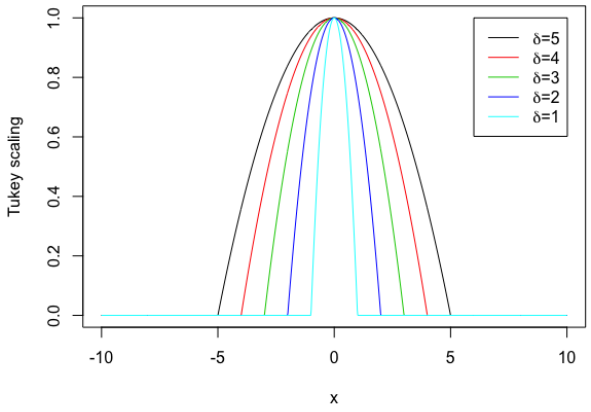

Figure 2. We plot the scaling factor (13) of Tukey's biweight. Compared to Huber scaling factor (see (12)), it has a hard threshold at δ.

of redescending loss whose derivative will vanish to zero as the input goes outside the interval $[-\delta, \delta]$. It is defined in the following way:

$$\frac{d}{dy}T_\delta(y) = \begin{cases} y\left(1-\frac{y^2}{\delta^2}\right)^2 & \text{for } |y| \le \delta, \\ 0 & \text{elsewhere.} \end{cases}$$

Similarly, by rearranging the estimating equation, we have

$$\widehat{Y}_{tukey}(x) = \frac{\sum_{i=1}^n w^{tukey}(X_i,x)Y_i}{\sum_{i=1}^n w^{tukey}(X_i,x)}$$

where

$$w^{tukey}(X_i,x) = w(X_i,x)\max\left\{1-\left(\frac{\widehat{Y}_{tukey}-Y_i}{\delta}\right)^2, 0\right\}$$

with an extra scaling factor (see Figure 2)

$$\max\left\{1-\left(\frac{u}{\delta}\right)^2, 0\right\}. \tag{13}$$

In another word, the final estimator actually only depends on data with responses inside $[-\delta, \delta]$, and the importance of any data (X_i, Y_i) will be shrinking to zero when $|\widehat{Y}_{tukey}(x) - Y_i|$ gets closer to the boundary value δ.

4. Truncated squared loss and nearest neighbors

In this section, we will further use the framework 1 to investigate truncated squared error loss, and use this example to motivate the relation between random forest generalization performance and the number of adaptive nearest neighbors.

4.1. Truncated squared error

For the truncated squared error loss

$$S_\delta(y) = \begin{cases} \frac{1}{2}y^2 & \text{for } |y| \le \delta, \\ \frac{1}{2}\delta^2 & \text{elsewhere} \end{cases}$$

the corresponding estimating equation is

$$\sum_{|\widehat{Y}_{trunc}(x) - Y_i| \le \delta} w(X_i, x)(\widehat{Y}_{trunc}(x) - Y_i) = 0.$$

If we define a new weight

$$w^{trunc}(X_i, x) = w(X_i, x)\, \mathbb{1}\{|\widehat{Y}_{trunc}(x) - Y_i| \le \delta\}, \quad (14)$$

then the estimator for truncated squared loss is

$$\widehat{Y}_{trunc}(x) = \frac{\sum_{i=1}^n w^{trunc}(X_i, x) Y_i}{\sum_{i=1}^n w^{trunc}(X_i, x)}. \quad (15)$$

The estimator (15) is like a trimmed version of the random forest estimator (2). We first sort $\{Y_i\}_{i=1}^n$ and trim off the responses where $|\widehat{Y}_{trunc}(x) - Y_i| > \delta$. Therefore, for any truncation level δ, the estimator $\widehat{Y}_{trunc}(x)$ only depends on data satisfying $|\widehat{Y}_{trunc}(x) - Y_i| \le \delta$ with the same local random forest weights (1).

4.2. Random Forest Nearest Neighbors

In classical random forest, all the data with positive weights (3) are included when calculating the final estimator $\widehat{Y}(x)$. However, from section 4.1, we know in order to achieve robustness, some of the data should be dropped out of consideration. For example, using the truncated squared error loss, we will only consider the data satisfying $|Y_i - \widehat{Y}_{trunc}(x)| \le \delta$. In classical random forest, the criterion of tree split is to reduce the mean squared error, then in most cases, data points inside one terminal node will tend to have more similar responses. So informally larger $|\widehat{Y}_{trim}(x) - Y_i|$ will indicate smaller local weight $w(X_i, x)$. Therefore, instead of solving for (15), we investigate a related estimator

$$\widehat{Y}_{wt}(x) = \frac{\sum_{w(X_i,x) \ge \epsilon} w(X_i, x) Y_i}{\sum_{w(X_i,x) \ge \epsilon} w(X_i, x)} \quad (16)$$

where $\epsilon > 0$ is a constant in $(0, 1)$. Recall that in (Lin & Jeon, 2006), they show all the observations with positive weights are considered voting points for random forest estimator. However, (16) implies that we should drop observations with weights smaller than a threshold in order for the robustness. More formally, let σ be a permutation such that $w(X_{\sigma(1)}, x) \ge \cdots \ge w(X_{\sigma(n_0)}, x) > 0$, then (2) is equivalent to

$$\widehat{Y}(x) = \sum_{i=1}^{n_0} w(X_{\sigma(i)}, x) Y_{\sigma(i)}.$$

Then we can define the k random forest nearest neighbors (k-RFNN) of x to be $\{X_{\sigma(1)}, \cdots, X_{\sigma(k)}\}$, $k \le n_0$, and get predictor

$$\widehat{Y}_k(x) = \sum_{i=1}^{k} \widetilde{w}(X_{\sigma(i)}, x) Y_{\sigma(i)}, \quad (17)$$

where $\widetilde{w}(X_{\sigma(i)}, x) = w(X_{\sigma(i)}, x) / \sum_{j=1}^k w(X_{\sigma(i)}, x)$. In the numerical experiments (Section 5.3), we will test the performance of the estimator (17) with different k, and show that by merely choosing the right number of nearest neighbors, one can largely improve the performance of classical random forest.

Shi and Horvath (2006) proposed a similar ensemble tree based nearest neighbor method. In their approach, if the observations X_i and X_j lie in the same leaf, then the similarity between them is increased by one. At the end, the similarities are normalized by dividing the total number of trees in the forest. Therefore, their weights (similarities) $w(X_i, x)$ will be $m^{-1} \sum_{t=1}^m \mathbb{1}_{\{X_i \in R_{l(x,\theta)}\}}$ contrast to (3). So different from their approach, for random forest, the similarity between X_i and X_j will be increased by $1/\#\{p : X_p \in R_{l(X_i,\theta)}\}$ if they both lie in the same leaf $l(X_i, \theta)$. This means the increment in the similarity also depends on the number of data points in the leaf.

5. Experiments

In this section, we plug in the quantile loss, Huber loss and Tukey's biweight loss into the general forest framework and compare these algorithms with random forest. Unless otherwise stated, for both Huber and Tukey forest, the error tolerance is set to be 10^{-6}, and every forest is an ensemble of 1000 trees with maximum terminal node size 10. The robust parameter δ are set to be 0.005 and 0.8 for Huber and Tukey forest, respectively.

5.1. One dimensional toy example

We generate 1000 training data points from a Uniform distribution on $[-5, 5]$ and another 1000 testing points from the same distribution. The true underlying model is $Y = X^2 + \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$. But on the training samples, we choose 20% of the data and add noise $2\mathcal{T}_2$ to the responses, where $\mathcal{T}_2$ follows t-distribution with degree of freedom 2.

In Figure 3, we plot the true squared curve and different forest predictions. It is clear that Huber and Tukey forest achieve competitive robustness as quantile random forest, and can almost recover the true underlying distribution, but random forest is largely impacted by the outliers. We also repeat the experiments for 20 times, and report the average mean squared error (MSE), mean absolute deviation (MAD) and median absolute percentage error (MAPE) in Table 1.

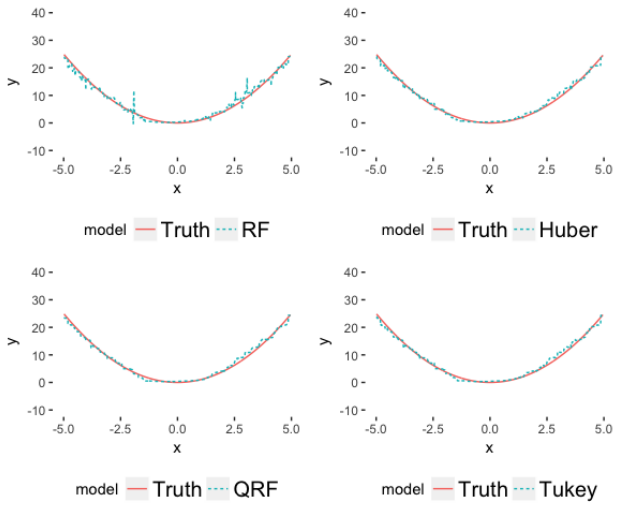

Figure 3. One dimensional comparison of random forest, quantile random forest, Huber forest and Tukey forest. All forests are ensemble of 500 regression trees and the maximum number of points in terminal nodes is 20.

Table 1. Comparison of random forest (RF), quantile random forest (QRF), Huber forest (Huber) and Tukey forest (Tukey) on one dimensional example.

MEASURE	RF	QRF	HUBER	TUKEY
MSE	2.56	1.88	1.85	**1.82**
MAD	1.20	1.07	**1.06**	1.07
MAPE	0.16	0.13	**0.12**	0.12

5.2. Multivariate example

We generate data from 10 dimensional Normal distribution, i.e. $X \sim \mathcal{N}_{10}(\vec{0}, \Sigma)$. Then we test out algorithms on following models.

(1) $Y = \sum_{i=1}^{10} X_i^2 + \epsilon$ and $\epsilon \sim \mathcal{N}(0,1)$, $\Sigma = \mathbb{I}$.
(2) $Y = \sum_{i=1}^{10} X_i^2 + \epsilon$ and $\epsilon \sim \mathcal{N}(0,1)$, $\Sigma = \text{Toeplitz}(\rho = 0.7)$.

Then for each model, we randomly choose η proportion of the training samples and add noise $15\mathcal{T}_2$ where $\mathcal{T}_2$ follows t-distribution with degree of freedom 2. The noise level $\eta \in \{0, 0.05, 0.1, 0.15, 0.2\}$. The results are summarized in Table 2 and 3. On the clean data, random forest still play the best, however, Huber forest's performance is also competitive and lose less efficiency than QRF and Tukey forest. On the noisy data, all three robust methods outperform random forest. Among them, Huber forest is most robust and stable.

Table 2. Comparison of the four methods in the setting (1). The average MSE is reported in first row, and average MAD in second row.

MSE	0%	5%	10%	15%	20%
RF	**8.19**	12.14	20.32	22.61	25.23
QRF	9.80	11.63	13.30	13.83	14.71
HUBER	9.02	**9.86**	**10.40**	**10.49**	**10.88**
TUKEY	10.56	12.41	18.16	12.34	16.62
MAD	0%	5%	10%	15%	20%
RF	**2.10**	2.49	2.73	2.89	3.02
QRF	2.23	2.37	2.66	2.75	2.84
HUBER	2.20	**2.28**	**2.36**	**2.38**	**2.43**
TUKEY	2.37	2.45	2.54	2.52	2.66

Table 3. Comparison of the four methods in the setting (2).

MSE	0%	5%	10%	15%	20%
RF	**9.21**	13.00	13.69	14.92	17.78
QRF	11.47	**12.07**	12.21	12.29	13.16
HUBER	11.19	12.08	**12.15**	**12.20**	**12.74**
TUKEY	12.84	13.09	13.31	14.52	14.60
MAD	0%	5%	10%	15%	20%
RF	**1.88**	2.19	2.74	2.80	2.83
QRF	2.06	**2.13**	2.28	2.32	2.41
HUBER	2.04	2.15	**2.17**	**2.17**	**2.22**
TUKEY	2.26	2.34	2.39	2.35	2.41

5.3. Nearest neighbors

In this section, we check how the number of adaptive nearest neighbors k in (17) will have impact on the performance of k-RFNN. We consider the same two models (1) and (2), and keep both training sample size and testing sample size to be 1000. The relations between MSE, MAD and the number of adaptive nearest neighbors are illustrated in Figure 4. Recall that k-RFNN with all 1000 neighbors is equivalent to random forest. From the figures, we clearly observe a kink at $k = 15$, which is much less than 1000.

5.4. Real data

We take two regression datasets from UCI machine learning repository (Lichman, 2013), and one real estate dataset from OpenIntro. For each dataset, we randomly choose $2/3$ observations for training and the rest for testing. MSE and MAD are reported by averaging over 20 trials. The results are presented in Table 4. To further test the robustness, we then repeat the experiment but add extra $\mathcal{T}_2$ noise to 20%

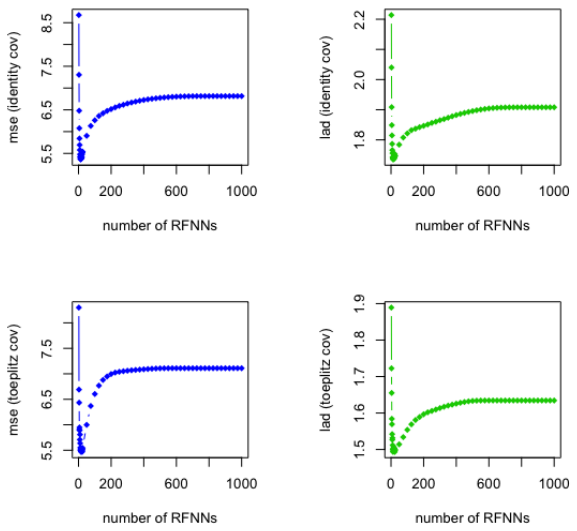

Figure 4. The performance of k-RFNN against the number of nearest neighbors.

of the standardized training data response variables everytime. The results are in Table 5. Robust forests outperform random forest in most of the cases except for Ames data sets, on which quantile random forest behaves poorly.

Table 4. Comparison of the four methods on two UCI repository datasets: (1) concrete compressive strength (CCS) (Yeh, 1998); (2) airfoil self-noise (Airfoil); and one OpenIntro dataset: Ames residential home sales (Ames).

MSE	RF	QRF	HUBER	TUKEY
CCS	37.22	34.79	**32.98**	34.42
AIRFOIL	18.22	**10.04**	14.28	16.55
AMES($\times 10^8$)	**4.51**	12.21	5.22	5.91
MAD	RF	QRF	HUBER	TUKEY
CCS	4.62	4.25	**4.17**	4.30
AIRFOIL	3.45	**2.30**	3.08	3.17
AMES($\times 10^4$)	1.34	2.44	**1.31**	1.36

6. Conclusion and discussion

The experimental results show that Huber forest, Tukey forest and quantile random forest are all much more robust than random forest in the presence of outliers. However, without outliers, Huber forest preserves more efficiency than the other two robust methods. We did not cross validate the parameter δ for different noise levels, so one would expect even better performance after carefully tuning the parameter.

Table 5. Test on real data sets with extra noise.

MSE	RF	QRF	HUBER	TUKEY
CCS	68.51	39.21	**39.05**	40.27
AIRFOIL	18.22	**10.04**	14.28	16.55
AMES($\times 10^8$)	5.77	18.20	**5.28**	5.39
MAD	RF	QRF	HUBER	TUKEY
CCS	5.46	**4.53**	4.57	4.80
AIRFOIL	3.45	**2.30**	3.08	3.17
AMES($\times 10^4$)	1.64	3.23	**1.47**	1.55

Besides random forest weights, other data dependent similarities could also be used in Algorithm 1. We could also design loss functions which optimizes a metric for specific problems. The fixed-point method could be replaced by other more efficient algorithms. The framework could be easily extended to classification problems. All these will be potential future work.

7. Appendix

7.1. Proof of Lemma 1

Proof. Because $\widehat{Y}^{(k)}(x) = K_\delta(\widehat{Y}^{(k-1)})$ which is a fixed-point method, we only need to show $\left|K_\delta^{'}(y)\right| < 1$ in order for the existence and uniqueness of the solution. Define the normalized weight

$$\widetilde{w}_i = \frac{w_i}{\sqrt{1+\left(\frac{y-Y_i}{\delta}\right)^2}} \Big/ \sum_{i=1}^{n} \frac{w_i}{\sqrt{1+\left(\frac{y-Y_i}{\delta}\right)^2}},$$

we have $\sum_{i=1}^n \widetilde{w}_i = 1$, and $\left|K_\delta^{'}(y)\right|$

$$\begin{aligned}
&\leq \left|\sum_{i=1}^{n} \widetilde{w}_i Y_i \left(\sum_{j=1}^{n} (\mathbb{I}(i=j) - \widetilde{w}_j) \frac{y-Y_j}{\delta^2+(y-Y_j)^2}\right)\right| \\
&\leq 2\sum_{i=1}^{n} \widetilde{w}_i |Y_i| \max_{i=1,\cdots,n} \left(\frac{|y-Y_i|}{\delta^2+(y-Y_i)^2}\right) \\
&= 2\sum_{i=1}^{n} \widetilde{w}_i |Y_i| \frac{1}{\min_{i=1,\cdots,n}\left(\frac{\delta^2}{|y-Y_i|}+|y-Y_i|\right)} \\
&\leq \max_{i=1,\cdots,n} |Y_i| \frac{1}{\delta}.
\end{aligned}$$

Therefore, $\left|K_\delta^{'}(y)\right| < \frac{1}{2}$ if $\delta > 2\max_{i=1,\cdots,n}|Y_i| = 2K$. □

Acknowledgements

We would like to thank Stan Humphrys and Zillow for supporting this research, as well as three anonymous referees for their insightful comments. Part of the implementation in this paper is based on Zillow code library.

References

Aryal, Sunil, Ting, Kai Ming, Haffari, Gholamreza, and Washio, Takashi. mp-dissimilarity: A data dependent dissimilarity measure. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pp. 707–712. IEEE, 2014.

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

Brence, MAJ John R and Brown, Donald E. Improving the robust random forest regression algorithm. *Systems and Information Engineering Technical Papers, Department of Systems and Information Engineering, University of Virginia*, 2006.

Charbonnier, Pierre, Blanc-Féraud, Laure, Aubert, Gilles, and Barlaud, Michel. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997.

Chen, Tianqi and Guestrin, Carlos. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016.

Fan, Jianqing and Gijbels, Irene. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, volume 66. CRC Press, 1996.

Freund, Yoav and Schapire, Robert E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pp. 23–37. Springer, 1995.

Freund, Yoav, Schapire, Robert E, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pp. 148–156, 1996.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Galimberti, Giuliano, Pillati, Marilena, and Soffritti, Gabriele. Robust regression trees based on m-estimators. *Statistica*, 67(2):173–190, 2007.

Hampel, Frank R, Ronchetti, Elvezio M, Rousseeuw, Peter J, and Stahel, Werner A. *Robust statistics: the approach based on influence functions*, volume 114. John Wiley & Sons, 2011.

Hastie, Trevor and Loader, Clive. Local regression: Automatic kernel carpentry. *Statistical Science*, pp. 120–129, 1993.

Huber, Peter J. *Robust statistics*. Springer, 2011.

Huber, Peter J et al. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101, 1964.

Jarvis, Raymond Austin and Patrick, Edward A. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on computers*, 100(11):1025–1034, 1973.

Koenker, Roger. *Quantile regression*. Number 38. Cambridge university press, 2005.

Li, Alexander Hanbo and Bradic, Jelena. Boosting in the presence of outliers: adaptive classification with nonconvex loss functions. *Journal of the American Statistical Association*, (just-accepted), 2016.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Lin, Dekang et al. An information-theoretic definition of similarity. In *ICML*, volume 98, pp. 296–304. Citeseer, 1998.

Lin, Yi and Jeon, Yongho. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

Loader, Clive. *Local regression and likelihood*. Springer Science & Business Media, 2006.

Mason, Llew, Baxter, Jonathan, Bartlett, Peter L, and Frean, Marcus R. Boosting algorithms as gradient descent. In *NIPS*, pp. 512–518, 1999.

Meinshausen, Nicolai. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999, 2006.

Newey, Whitney K. Kernel estimation of partial means and a general variance estimator. *Econometric Theory*, 10 (02):1–21, 1994.

Roy, Marie-Hélène and Larocque, Denis. Robustness of random forests for regression. *Journal of Nonparametric Statistics*, 24(4):993–1006, 2012.

Ruppert, David and Wand, Matthew P. Multivariate locally weighted least squares regression. *The annals of statistics*, pp. 1346–1370, 1994.

Shi, Tao and Horvath, Steve. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138, 2006.

Staniswalis, Joan G. The kernel estimate of a regression function in likelihood-based models. *Journal of the American Statistical Association*, 84(405):276–283, 1989.

Tibshirani, Robert and Hastie, Trevor. Local likelihood estimation. *Journal of the American Statistical Association*, 82(398):559–567, 1987.

Ting, Kai Ming, Zhu, Ye, Carman, Mark, Zhu, Yue, and Zhou, Zhi-Hua. Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1205–1214. ACM, 2016.

Tsymbal, Alexey, Pechenizkiy, Mykola, and Cunningham, Pádraig. Dynamic integration with random forests. In *European conference on machine learning*, pp. 801–808. Springer, 2006.

Yeh, I-C. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.