How are Random Forests not sensitive to outliers?

I've read in a few sources, including this one, that Random Forests are not sensitive to outliers (in the way that Logistic Regression and other ML methods are, for example).

However, two pieces of intuition tell me otherwise:

1. Whenever a decision tree is constructed, all of the points must be classified. This means that even outliers will get classified, and hence will affect the decision trees where they were selected during boosting.
2. Bootstrapping is a part of how a RandomForest does sub-sampling. Bootstrapping is susceptible to outliers.

Is there any way to reconcile my intuition about its sensitivity to outliers, with sources that disagree?

random-forest bootstrap outliers cart

edited Dec 19 '15 at 16:58
m0nhawk
150 2 8

asked Dec 17 '15 at 6:23
Hunle
87 1 9

The answer, below, is very good. The intuitive answer is that a decision tree works on splits and splits aren't sensitive to outliers: a split only has to fall anywhere between two groups of points to split them. – Wayne Dec 20 '15 at 15:15

So I suppose if the `min_samples_leaf_node` is `1`, then it could be susceptible to outliers. – Hunle Dec 21 '15 at 0:13

yes min_samples and bootstrap sample can completely remove the influence of 1b outliers in RF regression – Soren Havelund Welling Dec 21 '15 at 10:44

Some statisticians obtain a tunnel vision on those inliers, that one can predict and understand. Cherish the outliers as 'known unknowns' and wonder if your business model is fragile towards them. Some outliers are fundamentally unpredictable, but their impact is very real ... a paraphrase of N. Taleb's, 'Black Swan' – Soren Havelund Welling Dec 21 '15 at 10:52

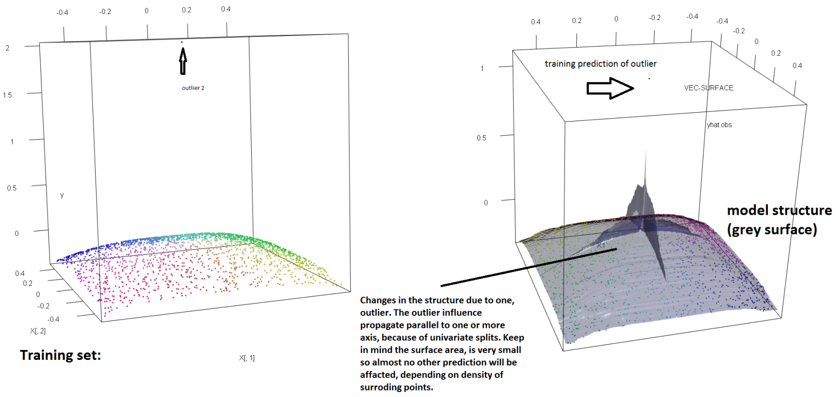
3 Answers

outlier 1a: This outlier has one or more extreme feature values and is placed distant to any other sample. The outlier will influence the initial splits of the trees as any other sample, so no strong influence. It will have low *proximity* to any other sample, and will only define the model structure in a remote part of feature space. During prediction most new samples are likely not to be similar to this outlier, and will rarely end up in the same terminal node. Moreover decision trees regards features as if they were ordinal(ranking). The value is either smaller/equal to or larger than break point, thus it does not matter if a feature value is an extreme outlier.

outlier 1b: For classification one single sample may be regarded as an outlier, when embedded in the middle of many sample of a different class. I described earlier how a default RF model will get influenced by this one sample of odd class, but only very close to the sample.

outlier 2: This outlier has an extreme target value perhaps many times higher than any other values, but the feature values are normal. A .631 fraction of the trees will have a terminal node with this sample. The model structure will get affected locally close to the outlier. Notice the model structure is affected mainly parallel to the feature axis, because nodes are split uni-variately.

I included a RF-regression simulation of outlier_2. 1999 points drawn from a smooth rounded structure $y = (x_1^4 + x_2^4)^{\frac{1}{2}}$ and one outlier with a much higher target value(y=2, x_1 =0,x_2 =0). The training set is shown to the left. The learned RF model-structure is shown the the right.



Training set:

model structure (grey surface)

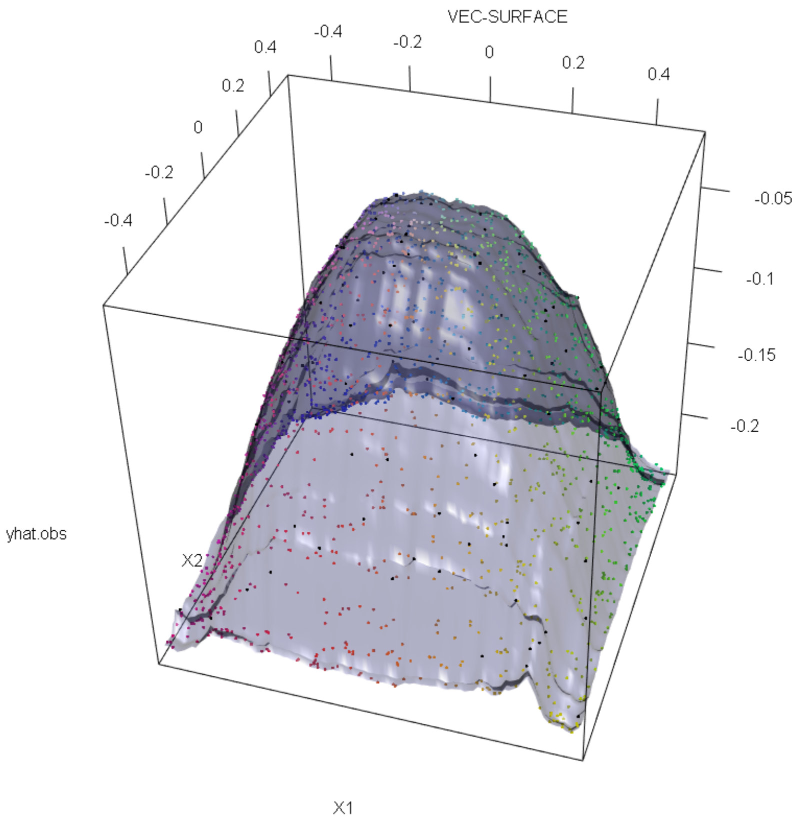
Changes in the structure due to one, outlier. The outlier influence propagate parallel to one or more axis, because of univariate splits. Keep in mind the surface area, is very small so almost no other prediction will be affected, depending on density of surroding points.

```
library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X = data.frame(replicate(2,runif(2000)-.5))
y = -sqrt((X[,1])^4+(X[,2])^4)^1
Col = fcol(X,1:2) #make colour pallete by x1 and x2
#insert outlier2 and colour it black
X[1,] = c(0,0);y[1]=2 ;Col[1] = "#000000FF" #black

#plot training set
plot3d(X[,1],X[,2],y,col=Col)

rf = randomForest(X,y)
vec.plot(rf,X,1:2,col=Col,grid.lines = 400)
```

EDIT: comment to user603

Yes for extreme outliers on target scale, one should consider to transform target scale before running RF. I added below a *robustModel()* function which tweaks randomForest. Another solutions would be to log transform before training.

```
.
##---code by user603
library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X<-data.frame(replicate(2,runif(2000)-.5))
y<--sqrt((X[,1])^4+(X[,2])^4)
Col<-fcol(X,1:2) #make colour pallete by x1 and x2

#insert outlier2 and colour it black
y2<-y;Col2<-Col
y2[1:100]<-rnorm(100,200,1);    #outliers
Col2[1:100]="#000000FF" #black
##---

#function to make models robust
robustModel = function(model,keep.outliers=TRUE) {
  f = function(X,y,lim=c(0.1,.9),keep.outliers="dummy",...) {
  limits = quantile(y,lim)
  if(keep.outliers) {#keep but reduce outliers
  y[limits[1]>y] = limits[1] #lower limit
  y[limits[2]<y] = limits[2] #upper limit
  } else {#completely remove outliers
    thrashThese = mapply("||",limits[1]>y,limits[2]>y)
    y = y[thrashThese]
    X = X[thrashThese,]
  }
  obj = model(x=X,y=y,...)
  class(obj) = c("robustMod",class(obj))
  return(obj)
  }
  formals(f)$keep.outliers = keep.outliers
  return(f)
}

robustRF = robustModel(randomForest) #make RF robust
rh = robustRF(X,y2,sampsize=250)     #train robustRF
vec.plot(rh,X,1:2,col=Col2)          #plot model surface
mean(abs(rh$predict[-c(1:100)]-y2[-c(1:100)]))
```



edited Dec 21 '15 at 15:58

answered Dec 19 '15 at 16:38
Soren Havelund Welling
2,871 5 17

You write "no other prediction will be affected". If you shift your single outlier to put `y[1]=200` you will see that it single handedly causes the prediction error *on the uncontaminated observations* to jump by a factor of 20! – user603 Dec 21 '15 at 14:08

@user603 True that, In such cases the target scale can be transformed monotonically before handed to RF. I added a 'robustModel: makes models robust' to my answer.....of course to predict such random target outlier(s) (type 2) remains impossible, but the remaining model structure do not have to suffer – Soren Havelund Welling Dec 21 '15 at 16:00

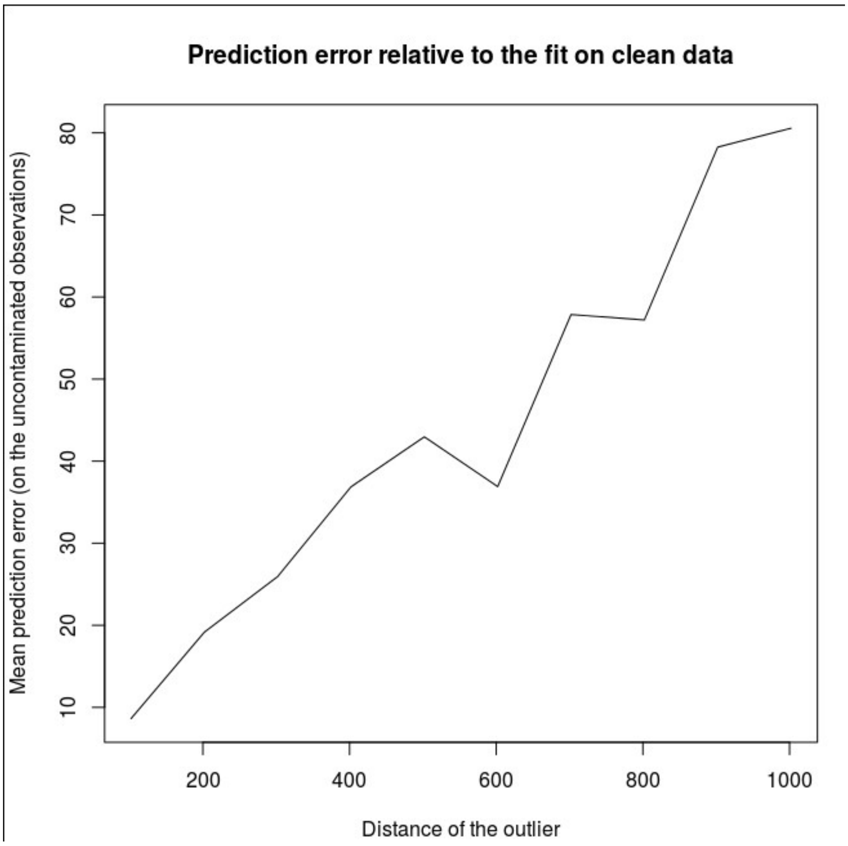
The Log transform is not, in general, a solution against outliers (it merely hides the problem). The robustification of RF you propose is essentially the approach advocated in Galimberti, G., Pillati, M., & Soffritti, G. (see my answer). The main difference is that your "robustModel" approach has a maximum breakdown point of 25% on the response space (it can withstand 25% or arbitrary 'y'-outliers) whereas theirs has a bdp of 50%. Note that neither approach is robust to outliers in the design space. – user603 Dec 21 '15 at 16:06

Your intuition is correct. This answer merely illustrates it on an example.

It is indeed a common misconception that CART/RF are somehow robust to outliers.

To illustrate the lack of robustness of RF to the presence of a single outliers, we can (lightly) modify the code used in Soren Havelund Welling's answer above to show that a **single** 'y'-outliers suffices to completely sway the fitted RF model. For example, if we compute the mean prediction error *of the uncontaminated observations* as a function of the distance between the outlier and the rest of the data, we can see (image below) that introducing *a single* outlier (by replacing one of the original observations by an arbitrary value on the 'y'-space) suffices to pull the predictions of the RF model arbitrarily far away from the values they would have had if computed on the original (uncontaminated) data:

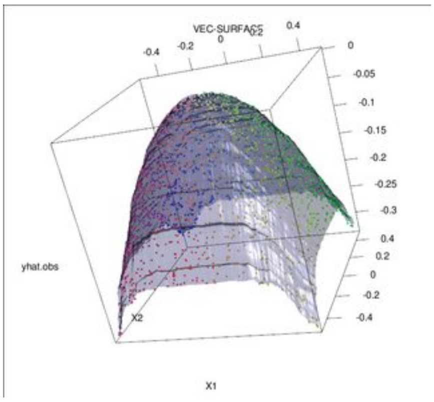
```
 library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X = data.frame(replicate(2,runif(2000)-.5))
y = -sqrt((X[,1])^4+(X[,2])^4)
X[1,]=c(0,0);
y2<-y
rg<-randomForest(X,y)   #RF model fitted without the outlier
outlier<-rel_prediction_error<-rep(NA,10)

for(i in 1:10){
    y2[1]=100*i+2
    rf=randomForest(X,y2)   #RF model fitted with the outlier
    rel_prediction_error[i]<-mean(abs(rf$predict[-1]-y2[-1]))/mean(abs(rg$predict[-1]-
y[-1]))
    outlier[i]<-y2[1]
}
plot(outlier,rel_prediction_error,type='l',ylab="Mean prediction error (on the
uncontaminated observations) \\\ relative to the fit on clean data",xlab="Distance of the
outlier")
```

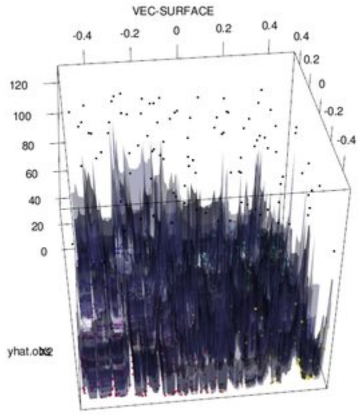


How far? In the example above, the single outlier has changed the fit so much that the mean prediction error (on the uncontaminated) observations is now *1-2 orders of magnitude* bigger than it would have been, had the model been fitted on the uncontaminated data.

So it is not true that a single outlier cannot affect the RF fit.

Furthermore, as I point out elsewhere, outliers are much harder to deal with when there are potentially *several* of them (though they don't need to be a large *proportion* of the data for their effects to show up). Of course, contaminated data can contain more than one outlier; to measure the impact of several outliers on the RF fit, compare the plot on the left obtained from the RF on the uncontaminated data to the plot on the right obtained by arbitrarily shifting 5% of the responses values (the code is below the answer).





Finally, in the regression context, it is important to point out that outliers can stand out from the bulk of the data in both the design and response space (1). In the specific context of RF, design outliers will affect the estimation of the hyper-parameters. However, this second effect is more manifest when the number of dimension is large.

What we observe here is a particular case of a more general result. The extreme sensitivity to outliers of multivariate data fitting methods based on convex loss functions has been rediscovered many times. See (2) for an illustration in the specific context of ML methods.

Edit.

Fortunately, while the base CART/RF algorithm is emphatically not robust to outliers, it is possible (and quiet easy) to modify the procedure to impart it robustness to "y"-outliers. I will now focus on regression RF's (since this is more specifically the object of the OP's question). More precisely, writing the splitting criterion for an arbitrary node t as:

$$s^* = \arg\max_s [p_L \mathrm{var}(t_L(s)) + p_R \mathrm{var}(t_R(s))]$$

where t_L and t_R are emerging child nodes dependent on the choice of s^* (t_L and t_R are implicit functions of s) and p_L denotes the fraction of data that falls to the left child node t_L and $p_R = 1 - p_L$ is the share of data in t_R. Then, one can impart "y"-space robustness to regression trees (and thus RF's) by replacing the variance functional used in the original definition by a robust alternative. This is in essence the approach used in (4) where the variance is replaced by a robust M-estimator of scale.

- (1) Unmasking Multivariate Outliers and Leverage Points. Peter J. Rousseeuw and Bert C. van Zomeren Journal of the American Statistical Association Vol. 85, No. 411 (Sep., 1990), pp. 633-639
- (2) Random classification noise defeats all convex potential boosters. Philip M. Long and Rocco A. Servedio (2008). http://dl.acm.org/citation.cfm?id=1390233
- (3) C. Becker and U. Gather (1999). The Masking Breakdown Point of Multivariate Outlier Identification Rules.
- (4) Galimberti, G., Pillati, M., & Soffritti, G. (2007). Robust regression trees based on M-estimators. Statistica, LXVII, 173–190.

```
library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X<-data.frame(replicate(2,runif(2000)-.5))
y<--sqrt((X[,1])^4+(X[,2])^4)
Col<-fcol(X,1:2) #make colour pallete by x1 and x2
#insert outlier2 and colour it black
y2<-y;Col2<-Col
y2[1:100]<-rnorm(100,200,1);    #outliers
Col2[1:100]="#000000FF" #black

#plot training set
plot3d(X[,1],X[,2],y,col=Col)
rf=randomForest(X,y)    #RF on clean data
rg=randomForest(X,y2)   #RF on contaminated data
vec.plot(rg,X,1:2,col=Col2,grid.lines=200)
mean(abs(rf$predict[-c(1:100)]-y[-c(1:100)]))
    mean(abs(rg$predict[-c(1:100)]-y2[-c(1:100)]))
```

edited Jan 2 at 1:58

answered Dec 20 '15 at 17:09

user603
14.1k 1 37 74

Thanks for your detailed answer. If there are several outliers in the same high dimensional space, it begs the question what is our criteria for calling an “outlier”? In that case, I wonder what hyper parameters may be set so that I can specify some kind of criteria for an outlier a priori? – Hunle Dec 21 '15 at 0:53

Right, so like I'm trying to say, I guess it's a matter of definition of "outlier". How do you build a definition of an "outlier" into the hyperparameters of an RF algorithm? – Hunle Dec 21 '15 at 1:19

1 I have added my earlier comments to my answer. I hope it now does a better job of answering your question! – user603 Dec 21 '15 at 1:46

1 Thanks. What are p and s in the formula? – Hunle Dec 21 '15 at 2:34

1 Why are combined outliers (1a+2) bad? In your example, the RF model fit the data structure perfectly, 99,99% OOB MSE. The model structure of the middle land between the two clusters is pretty rough, yes, and more a product of the model than of the data. But, no inference and/or predictions should be in this unknown area, so it does not matter. Absolute robustness toward outliers is inevitably to ignore rare but perhaps important possible events. Most ML algos would by default take a middle ground stance between robustness and 'flexibility' but can be tweaked to increase robustness. – Soren Havelund Welling Dec 21 '15 at 11:13

@SorenHavelundWelling: Thanks! fixed the example to show that the outliers can make the mean absolute error arbitrarily large. – user603 Dec 21 '15 at 16:15

When I first pondered this question, I assumed that the way an RF treats outliers would be the same regardless of classification or regression. I'm actually interested in how this would be different if, say, I were using RF for binary classification. – Hunle Dec 24 '15 at 19:42

If the response is binary, the only type of outliers that matter are those on the design space. Outliers on the design are not handled by the robustification strategy outlined in the edit of my answer and I do not know how to modify RF to handle them. – user603 Dec 26 '15 at 12:24

Is there any reason that in your code, @user603, that you eliminate the last value in the array in this line: `rel_prediction_error[i]<-mean(abs(rf$predict[-1]-y2[-1]))/mean(abs(rg$predict[-1]-y[-1]))` . Why do you use `[-1]` to eliminate the last value? – Hunle Jun 7 at 6:55

@Hunle: yes y[1] is the outlier. rel_prediction_error is the vector of prediction errors on the *non outlying observations*. This corresponds to the sentence "For example, if we compute the mean prediction error of the uncontaminated observations" in y text. – user603 Jun 7 at 7:15

This example seems a bit unfair, since the function you use is \-sqrt{x^4+x^4}\ and the outliers are positive. You say you're measuring the "distance" of the outler - distance from what? – Hunle Jun 8 at 7:35

@Hunle: a) why would this example be unfair? Could you elaborate? I do not understand the relevance of your argument ' the function...the outliers are positive'. b) distance to where the original, uncontaminated observation was. We measure the sensitivity of the RF to the presence of a single outlier in the data by showing that the *whole* fitted RF surface (at all values $\boldsymbol{x} \in \mathbb{R}^2$) can be affected at will by just moving one of the original data point ($\boldsymbol{x}_i$) to an arbitrary location in the design space ($\boldsymbol{x}'_i$). The distance is $||\boldsymbol{x}'_i - \boldsymbol{x}_i||$. – user603 Jun 8 at 8:01

It is not the Random Forest algorithm itself that is robust to outliers, but the base learner it is based on: the **decision tree**. Decision trees isolate atypical observations into small leaves (i.e., small subspaces of the original space). Furthermore, decision trees are **local** models. Unlike linear regression, where the same equation holds for the entire space, a very simple model is fitted locally to each subspace (i.e., to each leaf).

- In the case of regression, it is generally a very low-order regression model (usually only the average of the observations in the leaf).
- For classification, it is majority voting.

Therefore, for regression for instance, extreme values do not affect the entire model because they get averaged locally. So the fit to the other values is not affected.

Actually, this desirable property carries over to other tree-like structures, like dendograms. Hierarchical clustering, for instance, has long been used for data cleaning because it automatically isolates aberrant observations into small clusters. See for instance Loureiro et al. (2004). Outlier detection using clustering methods: a data cleaning application.

So, in a nutshell, RF inherits its insensitivity to outliers from **recursive portioning** and **local model fitting**.

Note that decision trees are low bias but high variance models: their structure are prone to changing upon a small modification of the training set (removal or addition of a few observations). But this should not be mistaken with sensitivity to outliers, this is a different matter.

edited Jan 1 at 19:33 answered Jan 1 at 19:27

Antoine
1,454 7 25

I actually considered using a clustering method, as you suggest, for the detection of outliers. But then, I'm unsure where to apply the clustering. Should it be applied to `labeled` or `unlabeled` data? And how would this clustering be achieved on heterogenous data that contains both categorical and numerical features? – Hunle Jan 6 at 6:33

Add Another Answer