2,871 5 17 review help

Obtaining knowledge from a random forest

Random forests are considered to be black boxes, but recently I was thinking what knowledge can be obtained from a random forest?

The most obvious thing is the importance of the variables, in the simplest variant it can be done just by calculating the number of occurrences of a variable.
The second thing I was thinking about are interactions. I think that if the number of trees is sufficiently large then the number of occurrences of pairs of variables can be tested (something like chi square independence). The third thing are nonlinearities of variables. My first idea was just to look at a chart of a variable Vs score, but I'm not sure yet whether it makes any sense.

These things are probably well studied (just an intuition). I would be grateful if anyone could point me how to examine those things properly.

Added 23.01.2012
Motivation

I want to use this knowledge to improve a logit model. I think (or at least I hope) that it is possible to find interactions and nonlinearities that were overlooked.

machine-learning | data-mining | interaction | random-forest | cart

edited Sep 19 '15 at 21:20

Antoine
1,454 7 25

asked Jan 16 '12 at 11:09

Tomek Tarczynski
1,062 4 15 24

Welcome to our site! – kjetil b halvorsen Sep 16 '15 at 15:03

a related thread on how variable importance measures are calculated for stochastic gradient tree boosting – Antoine Sep 19 '15 at 21:03

8 Answers

Random Forests are hardly a black box. They are based on decision trees, which are very easy to interpret:

```
#Setup a binary classification problem
require(randomForest)
data(iris)
set.seed(1)
dat <- iris
dat$Species <- factor(ifelse(dat$Species=='virginica','virginica','other'))
trainrows <- runif(nrow(dat)) > 0.3
train <- dat[trainrows,]
test <- dat[!trainrows,]

#Build a decision tree
require(rpart)
model.rpart <- rpart(Species~., train)
```

This results in a simple decision tree:

```
> model.rpart
n= 111

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 111 35 other (0.68468468 0.31531532)
  2) Petal.Length< 4.95 77  3 other (0.96103896 0.03896104) *
  3) Petal.Length>=4.95 34  2 virginica (0.05882353 0.94117647) *
```

If Petal.Length < 4.95, this tree classifies the observation as "other." If it's greater than 4.95, it classifies the observation as "virginica." A random forest is simple a collection of many such trees, where each one is trained on a random subset of the data. Each tree then "votes" on the final classification of each observation.

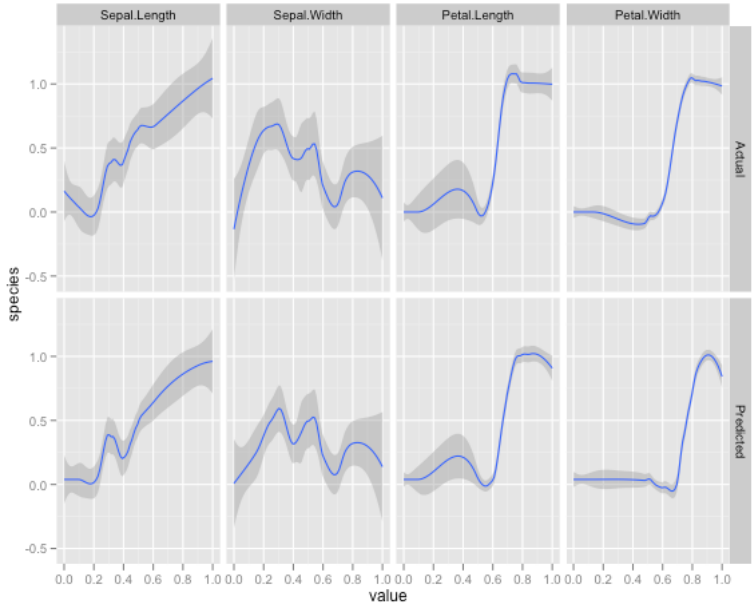
```
model.rf <- randomForest(Species~., train, ntree=25, proximity=TRUE, importance=TRUE,
nodesize=5)
> getTree(model.rf, k=1, labelVar=TRUE)
  left daughter right daughter    split var split point status prediction
1             2              3  Petal.Width        1.70      1       <NA>
2             4              5 Petal.Length        4.95      1       <NA>
3             6              7 Petal.Length        4.95      1       <NA>
4             0              0         <NA>        0.00     -1      other
5             0              0         <NA>        0.00     -1  virginica
6             0              0         <NA>        0.00     -1      other
7             0              0         <NA>        0.00     -1  virginica
```

You can even pull out individual trees from the rf, and look at their structure. The format is slightly different than for `rpart` models, but you could inspect each tree if you wanted and see how it's modeling the data.

Furthermore, no model is truly a black box, because you can examine predicted responses vs

actual responses for each variable in the dataset. This is a good idea regardless of what sort of model you are building:

```
library(ggplot2)
pSpecies <- predict(model.rf,test,'vote')[,2]
plotData <- lapply(names(test[,1:4]), function(x){
  out <- data.frame(
    var = x,
    type = c(rep('Actual',nrow(test)),rep('Predicted',nrow(test))),
    value = c(test[,x],test[,x]),
    species = c(as.numeric(test$Species)-1,pSpecies)
    )
  out$value <- out$value-min(out$value) #Normalize to [0,1]
  out$value <- out$value/max(out$value)
  out
})
plotData <- do.call(rbind,plotData)
qplot(value, species, data=plotData, facets = type ~ var, geom='smooth', span = 0.5)
```



I've normalized the variables (sepal and petal length and width) to a 0-1 range. The response is also 0-1, where 0 is other and 1 is virginica. As you can see the random forest is a good model, even on the test set.

Additionally, a random forest will compute various measure of variable importance, which can be very informative:

```
> importance(model.rf, type=1)
             MeanDecreaseAccuracy
Sepal.Length           0.28567162
Sepal.Width           -0.08584199
Petal.Length           0.64705819
Petal.Width            0.58176828
```

This table represents how much removing each variable reduces the accuracy of the model. Finally, there are many other plots you can make from a random forest model, to view what's going on in the black box:

```
plot(model.rf)
plot(margin(model.rf))
MDSplot(model.rf, iris$Species, k=5)
plot(outlier(model.rf), type="h", col=c("red", "green", "blue")[as.numeric(dat$Species)])
```

You can view the help files for each of these functions to get a better idea of what they display.

edited Jan 22 '12 at 5:52

answered Jan 21 '12 at 17:09

Zach
13.3k 7 63 118

5 Thanks for the answer, there is a lot of useful info, but it is not exactly what I was looking for. Maybe I need to clarify better the motivation that is behind this question. I want to use a random forest to improve a logit model, by improve I mean to add some interaction or to use a nonlinear transformation. – Tomek Tarczynski Jan 23 '12 at 10:47

@TomekTarczynski that's an interesting problem and similar to one I'm dealing with right now. I assume by "logit model" you mean logistic regression or something similar? I'm using lasso logistic regression (from the glmnet R package) to select predictors from a model with interactions between all pairs of variables. I haven't added in any nonlinear terms yet--but in principle that should be possible too. The only issue I guess is deciding what nonlinear terms to try (polynomial terms, exponential transforms, etc?). Also, I'm not picking up any higher-order interactions but that's easy too. – Anne Z. Jan 25 '12 at 13:23

2 @Tomek, what are you not getting from this answer? If you are using the randomForest package in R then the plots Zach describes should be very useful. Specifically, you could use varImpPlot for feature selection in your logit model and partialPlot to estimate the type of transformation to try on continuous predictors in the logit model. I would suggest that the latter plot be used to determine where nonlinear relationships between predictor and response exists and then allows you to make that transformation explicitly or to use a spline on that variable. –

B_Miner Jan 25 '12 at 14:14

2 @b_miner - just a guess, but it sounds like tomek is asking how to find non-linear interactions between variables because logistic regression already captures the linear relationships. – rm999 Jan 25 '12 at 15:04

@rm999 How do you define a non linear interaction in a logit model? Interaction terms created between transformed variables? – B_Miner Jan 25 '12 at 15:19

|

Some time ago I had to justify a RF model-fit to some chemists in my company. I spent quite time trying different visualization techniques. During the process, I accidentally also came up with some new techniques which I put into an R package (forestFloor) specifically for random forest visualizations.

The classical approach are partial dependence plots supported by: Rminer(data-based sensitivity analysis is reinvented partial dependence), or partialPlot in randomForest package. I find the partial dependence package iceBOX as an elegant way to discover interactions. Have not used edarf package, but seems to have some fine visualizations dedicated for RF. The ggRandomForest package also contain a large set of useful visualizations.

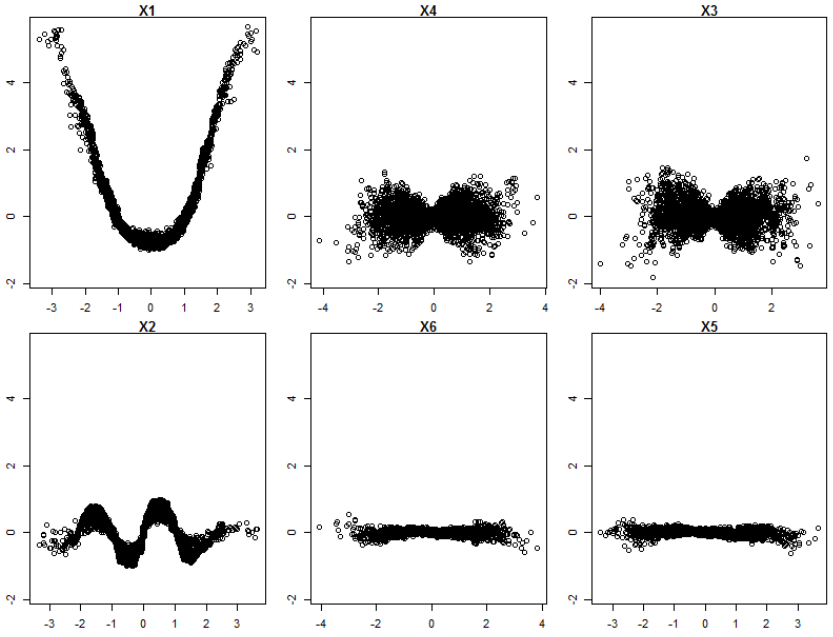
Currently forestFloor supports randomForest objects(support for other RF implementions is on its way). Also feature contributions can be computed for gradient boosted trees, as these trees after training are not much different from random forest trees. So forestFloor could support XGBoost in future. Partial dependence plots are completely model invariant.

All packages have in common to visualize the geometrical mapping structure of a model from feature space to target space. A sine curve y = sin(x) would be a mapping from x to y and can be plotted in 2D. To plot a RF mapping directly would often require too many dimensions. Instead the overall mapping structure can be projected, sliced or decomposed, such that the entire mapping structure is boiled down into a sequence of 2D marginal plots. If your RF model only has captured main effects and no interactions between variables, classic visualizations methods will do just fine. Then you can simplify your model structure like this $y = F(X) \approx f_1(x_1) + f_2(x_2) + \ldots + f_d(x_d)$ Then each partial function by each variable can be visualized just as the sine curve. If your RF model has captured sizable interactions, then it is more problematic. 3D slices of the structure can visualize interactions between two features and the output. The problem is to know which combination of features to visualize, (**iceBOX** does address this issue). Also it is not easy to tell if other latent interactions still are not accounted for.

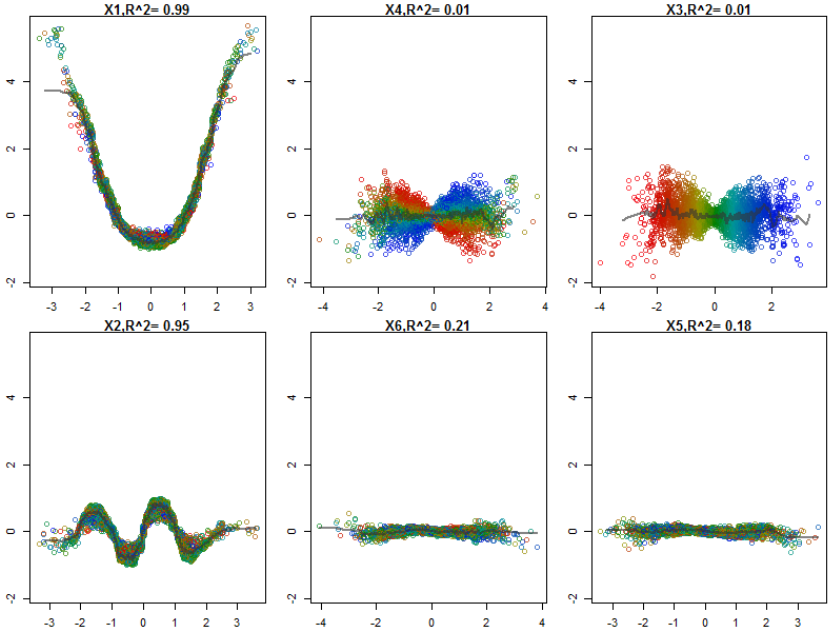
In this paper, I used an very early version of forestFloor to explain what actual biochemical relationship a very small RF model had captured. And in this paper we thoroughly describe visualizations of feature contributions, Forest Floor Visualizations of Random Forests.

I have pasted the simulated example from forestFloor package, where I show how to uncover a simulated hidden function $y = {x_1}^2 + sin(x_2\pi) + 2 * x_3 * x_4 +$ noise

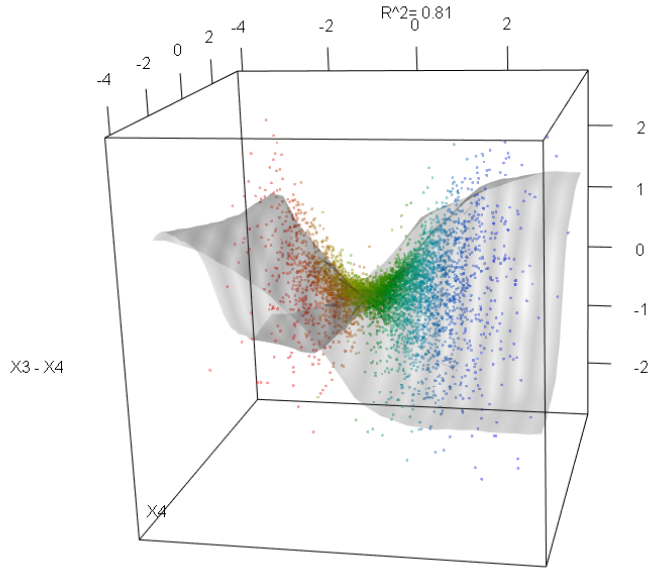
```
#1 - Regression example:
set.seed(1234)
library(forestFloor)
library(randomForest)

#simulate data y = x1^2+sin(x2*pi)+x3*x4 + noise
obs = 5000 #how many observations/samples
vars = 6   #how many variables/features
#create 6 normal distr. uncorr. variables
X = data.frame(replicate(vars,rnorm(obs)))
#create target by hidden function
Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 0.5 * rnorm(obs))

#grow a forest
rfo = randomForest(
  X, #features, data.frame or matrix. Recommended to name columns.
  Y, #targets, vector of integers or floats
  keep.inbag = TRUE,  # mandatory,
  importance = TRUE,  # recommended, else ordering by giniImpurity (unstable)
  sampsize = 1500 ,   # optional, reduce tree sizes to compute faster
  ntree = if(interactive()) 500 else 50 #speedup CRAN testing
)

#compute forestFloor object, often only 5-10% time of growing forest
ff = forestFloor(
  rf.fit = rfo,       # mandatory
  X = X,              # mandatory
  calc_np = FALSE,    # TRUE or FALSE both works, makes no difference
  binary_reg = FALSE  # takes no effect here when rfo$type="regression"
)


#plot partial functions of most important variables first
plot(ff,                        # forestFloor object
     plot_seq = 1:6,            # optional sequence of features to plot
     orderByImportance=TRUE     # if TRUE index sequence by importance, else by X column
)
```

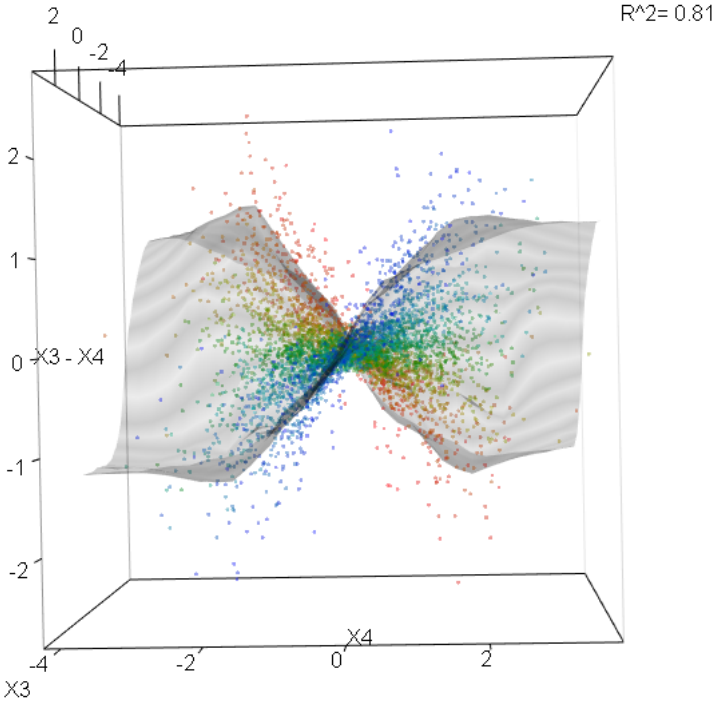



```
#Non interacting features are well displayed, whereas X3 and X4 are not
#by applying color gradient, interactions reveal themself
#also a k-nearest neighbor fit is applied to evaluate goodness-of-fit
Col=fcol(ff,3,orderByImportance=FALSE) #create color gradient see help(fcol)
plot(ff,col=Col,plot_GOF=TRUE)
```



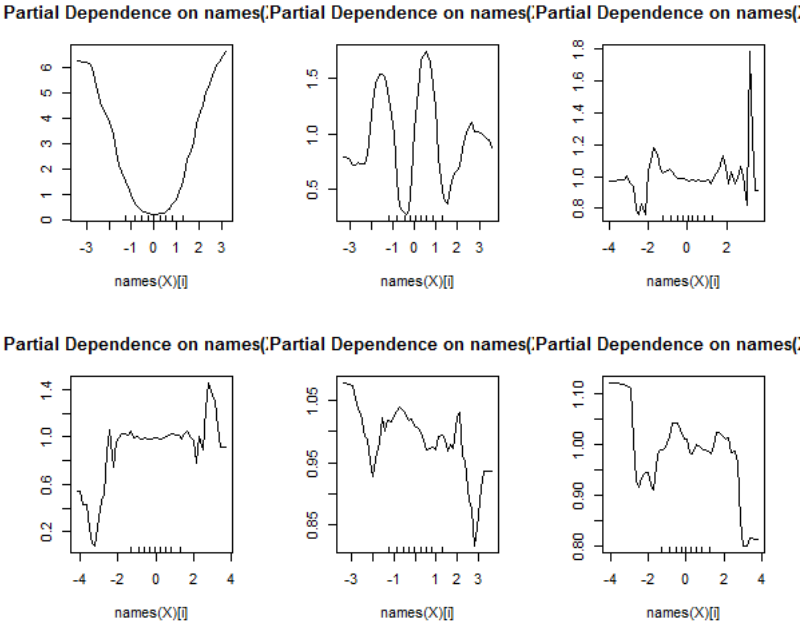

```
#feature contributions of X3 and X4 are well explained in the context of X3 and X4
# as GOF R^2>.8


show3d(ff,3:4,col=Col,plot_GOF=TRUE,orderByImportance=FALSE)
```





Lastly the code for partial dependence plots coded by A.Liaw described by J.Friedman. Which do fine for main effects.

```
par(mfrow=c(2,3))
for(i in 1:6) partialPlot(rfo,X,x.var=names(X)[i])
```



edited Jun 8 at 13:38

answered Sep 16 '15 at 23:36

Soren Havelund Welling
2,871 5 17

I postulated earlier that feature contributions also could be computed for boosted trees. I wrote a test algorithm and it is possible. Unfortunately, feature contributions for boosted trees, do not show the same beneficial properties as for bagged trees. Effects of one feature tend to scatter across all feature contributions, thus the visualization becomes quite confusing. – Soren Havelund Welling Apr 5 at 9:52

Oups! I found a bug in my test algorithm which made all the issues dissipate. forestFloor could be updated to work just fine for gradient boosted trees. – Soren Havelund Welling May 26 at 11:12

To supplement these fine responses, I would mention use of gradient boosted trees (e.g. the GBM Package in R). In R, I prefer this to random forests because missing values are allowed as compared to randomForest where imputation is required. Variable importance and partial plots are available (as in randomForest) to aid in feature selection and nonlinear transformation exploration in your logit model. Further, variable interaction is addressed with Friedman's H-statistic (`interact.gbm`) with reference given as `J.H. Friedman and B.E. Popescu (2005). "Predictive Learning via Rule Ensembles." Section 8.1` . A commercial version called TreeNet is available from Salford Systems and this video presentation speaks to their take on variable interaction estimation Video.

answered Jan 25 '12 at 14:24

B_Miner
873 3 40 73

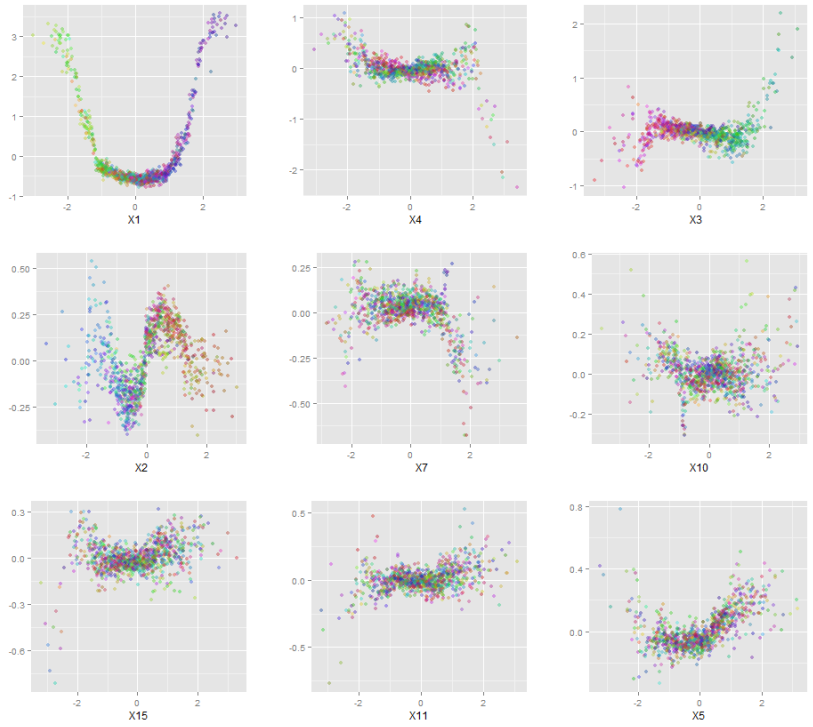
1 I agree, GBMs are a logical next step from random forests. – Zach Jan 25 '12 at 14:28

@B_miner: Great! I don't know how, but I have overlooked GBM. It seems that using GBM it is easy do detect interactions and nonlinearities. – Tomek Tarczynski Jan 26 '12 at 7:55

Late answer, but I came across a recent R package `forestFloor` (2015) that helps you doing this "unblackboxing" task in an automated fashion. It looks very promising!

```
library(forestFloor)
library(randomForest)
#simulate data
obs=1000
vars = 18
X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 1 * rnorm(obs))
#grow a forest, remeber to include inbag
rfo=randomForest(X,Y,keep.inbag = TRUE,sampsize=250,ntree=50)
#compute topology
ff = forestFloor(rfo,X)
#ggPlotForestFloor(ff,1:9)
plot(ff,1:9,col=fcol(ff))
```

Produces the following plots:



It also provides three-dimensional visualization if you are looking for interactions.

edited May 3 at 12:23
discipulus
112 6

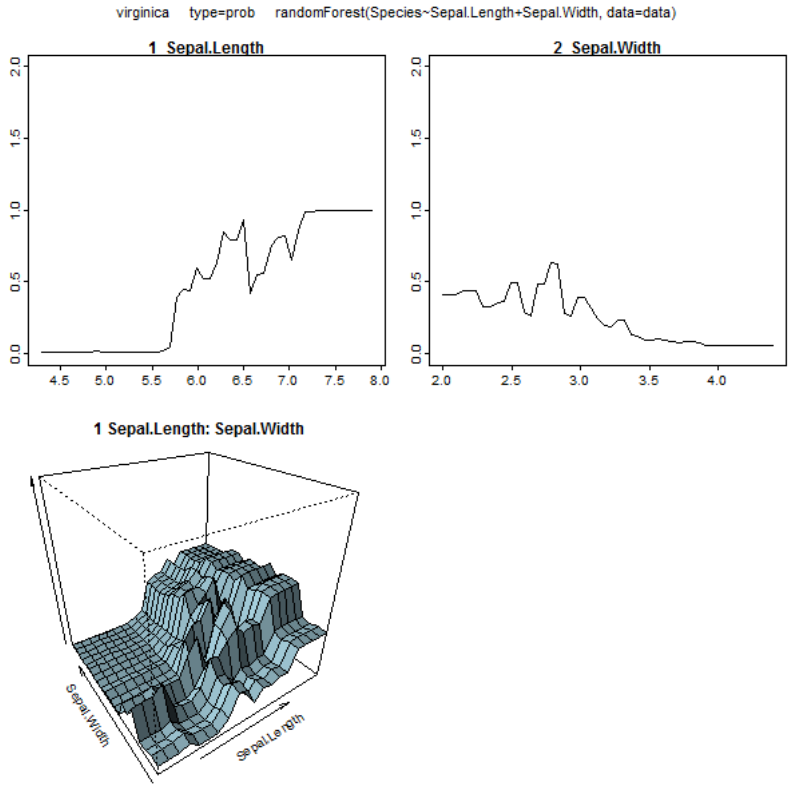
answered Oct 6 '15 at 14:22
RUser4512
2,839 6 28

3 I removed support for ggplot2, instead of last line try e.g.: **plot(ff,1:9,col=fcol(ff,1:4))** – Soren Havelund Welling Dec 22 '15 at 13:10

As mentioned by Zach, one way of understanding a model is to plot the response as the predictors vary. You can do this easily for "any" model with the plotmo package. For example

```
library(randomForest)
data <- iris
data$Species <- factor(ifelse(data$Species=='virginica','virginica','other'))
mod <- randomForest(Species~Sepal.Length+Sepal.Width, data=data)
library(plotmo)
plotmo(mod, type="prob")
```

which gives



This changes one variable while holding the others at their median values. For interaction plots, it changes two variables.

The example above uses only two variables; more complicated models can be visualized in a piecemeal fashion by looking at one or two variables at a time. Since the "other" variables are held at their median values, this shows only a slice of the data, but can still be useful. Some examples can be found in Chapter 10 of Plotting rpart trees with the rpart.plot package.

edited Jun 11 at 7:51

answered Jun 30 '15 at 12:38

Stephen Milborrow
141 1 4

I'm very interested in these type of questions myself. I do think there is a lot of information we can get out of a random forest.

About Interactions, it seems like Breiman and Cultier have already tried to look at it, especially for classification RFs.

To my knowledge, this has not been implemented in the randomForest R package. Maybe because it might not be as simple and because the meaning of "variable interactions" is very dependent of your problem.

About the nonlinearity, I'm not sure what you are looking for, regression forest are used for nonlinear multiple regression problems without any priors on what type of nonlinear function to use.

edited Jan 25 '12 at 14:04

answered Jan 25 '12 at 11:52

Rémy Nicolle
41 3

Using R you can produce a Dotchart of variable importance as measured by a Random Forest.

edited Mar 14 '12 at 21:32

Etiennebr
139 7

answered Jan 16 '12 at 11:21

George Dontas
3,669 1 22 38

Late answer related to my question here (*Can we make Random Forest 100% interpretable by fixing the seed?*):

> Let z_1 be the seed in the creation of boostrapped training set, and z_2 be the seed in the selection of feature's subset (for simplification, I only list 2 kinds of seeds here).
>
> 1. From z_1, m boostrapped training sets are created: $D_1(z_1)$, $D_2(z_1)$, $D_3(z_1)$, ..., $D_m(z_1)$.

2. From those traning sets, m corresponding decision trees are created, and tuned via cross-validation: $T_1(z_1, z_2)$, $T_2(z_1, z_2)$, $T_3(z_1, z_2)$,..., $T_m(z_1, z_2)$.
3. Let's denote predictions from the $j^{\text{th}}{}_{(j=1,2,...,m)}$ tree for an individual x_i (from training or testing set, whatever) as $\hat{f}^j(x_i)_{(i\leq n, j\leq m)}$. Hence the final predictions by the ensemble trees are:

$$\hat{F}(x_i) => \frac{1}{m}\sum_{j=1}^{m}\hat{f}^j(x_i)$$

4. Once the model is validated, and is **stable** (meaning $\hat{F}(x_i)$ doesn't depend strongly on the pair (z_1, z_2)). I start to create **every possible combinations of my features**, which give me a very big set (x_i').
5. Applying my forest on each x_i' gives me the corresponding predictions:

$$x_1' \rightarrow \hat{F}(x_1') \text{ - which is fixed} > \text{thanks to } (z_1, z_2)$$

$$x_2' \rightarrow \hat{F}(x_2') \text{ - } > \text{which is fixed thanks to } (z_1, z_2)$$

$$x_3' \rightarrow > \hat{F}(x_3') \text{ - which is fixed thanks to } (z_1, z_2)$$

$$x_4' > \rightarrow \hat{F}(x_4') \text{ - which is fixed thanks to } (z_1, > z_2)$$

$$....$$

6. The latter can be easily represented in form of **a single** (huge) **tree**. For example: x_1': (Age = 18, sex = M, ...), x_2' = (Age = 18, sex = F, ...), ... could be regrouped to create a leaf.

This works also for every ensemble methods based on aggregation of trees.

answered Apr 5 at 10:10

Matemattica
529 4 20

Add Another Answer