

Random Forests

Developers: Leo Breiman (Berkeley)
and Adele Cutler (Utah State Univ.)

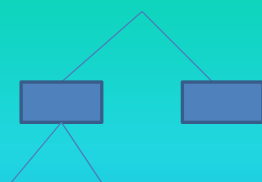


In Random Forests, many classification trees are grown (Random Forest).

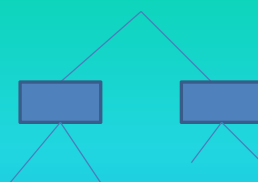
A new vector to be classified is classified using each of the tree in the Random Forest.

Each tree in the Random Forest classifies the new object, and "votes" for that class.

The classification having the most votes (over all the trees in the forest) is the RF classification.



New Obs
classified as 2



New Obs
classified as 1



New Obs
classified as 2

If 2 is the most common classification for the new observation, then RF classifies the new observation as 2.

Variable Importance (VI) Measure in Random Forests

- Class membership of each OOB (out of bag) sample is predicted using each tree in the forest, and # of correctly classified samples (CCO) is counted.
- Values of variable X_j are permuted in the OOB sample, and class membership of OOB samples are predicted again from the tree, and # of correctly classified samples after permutation (CCP) is counted.

$$VI = (CCO - CCP) / \#(\text{OOB samples})$$

We will use two packages – party and randomForest
for random forest method

Example 1: data file readingSkills of library party

The data frame readingSkills has 200 rows and 4 columns – data is simulated, has 4 variables:

"nativeSpeaker" "age" "shoeSize" "score"

Response variable = score, correlated with "nativeSpeaker" and "age", but not with shoeSize.

a) Use library randomForest for data of Example 1

```
treerf <- randomForest(score~., data = na.omit(readingSkills),  
importance=TRUE, proximity=TRUE)
```

```
importance(treerf) # print importance of variables
```

	%IncMSE	IncNodePurity
nativeSpeaker	52.42525	1957.751
age	31.21942	5775.828
shoeSize	23.18246	4526.170

Look at correlations among predictors:

```
pred0 <- cbind(readingSkills$nativeSpeaker,readingSkills$age,  
readingSkills$shoeSize)
```

```
pred00 <- na.omit(pred0)
```

```
cor(pred00)
```

	[,1]	[,2]	[,3]
[1,]	1.0000000	0.1355429	0.0958855
[2,]	0.1355429	1.0000000	0.8969335
[3,]	0.0958855	0.8969335	1.0000000

Age and shoeSize are highly correlated ($r = 0.87$) – this is the reason for shoeSize to turn out as more important than nativeSpeaker – see previous slide.

- Some of the recursive partitioning methods in R Are –
- CART in rpart (T. M. Therneau, B. Atkinson, and B. D. Ripley. rpart: Recursive partitioning. 2008. URL <http://CRAN.R-project.org/package=rpart>. R package version 3.1-41.)

Random Forests (L. Breiman. Random forests. Machine Learning, 45(1): 5–32, 2001.) available in R-package randomForests (A. Liaw and M. Wiener. randomForest: Breiman and Cutler’s Random Forests for Classification and Regression, 2008. URL <http://CRAN.R-project.org/package=randomForest>. R package version 4.5-28.)

- CART and random forests are commonly used methods in applied research.
- Several improvements have been suggested in statistical literature.
- The classical CART artificially favor splits in variables with large number of categories or continuous variables., and hence are BIASED.
- An unbiased tree algorithm is available in ctree function of party package, which also has cforest that yields unbiased random forests.

- Variable importance is computed by permuting each predictor (1 at a time) in OOB sample as discussed on slide 3. This is unconditional variable importance.

- The permutation importance can be misleading when predictors are correlated. Conditional importance measures eliminate this problem.

UNCONDITIONAL IMPORTANCE			CONDITIONAL IMPORTANCE		
Y	Xj	Z	Y	Xj	Z
Y1	permuted	Z1	Y1	permuted	Z1=a
.	.	.	.	mute	Z12=a
.	Z37=a
.	.	.	.	permuted	Z5=b
Yj	untouched	Zj	Yj	mute	Z14=b
.	.	.	.	permuted	Z140=b
.	.	.	.	mute	
.	.	.	.	permuted	
Yn	.	Zn	Yn	mute	

b) Use library party for data of Example 1

```
Tree.party <- cforest(score~., data = na.omit(readingSkills),
control=cforest_unbiased(mtry=2, ntree=50))
```

```
varimp(tree.party)
```

```
nativeSpeaker    age    shoeSize
12.18944    84.92342    14.71297
```

```
varimp(treepty, conditional = TRUE)
```

```
nativeSpeaker    age    shoeSize
11.1748705    50.8275283    0.5413227
```

shoeSize > nativeSpeaker

since shoeSize is correlated with Age.

Conditional importance takes care of the problem.

Example 2: data file cu.summary of library rpart

The data frame cu.summary has 17 rows and 5 columns – data is from April 1990 issue of Consumer Reports.

The columns are:

Price – quantitative – list price in US\$ of base model

Country – of origin (factor with levels

Brazil England France Germany Japan Japan/USA

Korea Mexico Sweden USA

Reliability – ordinal factor – Much worse < worse < average < better < Much better

Mileage – quantitative

Type - Compact Large Medium Small Sporty Van

Use the file R_code_random_forest.txt

a) Use library randomForest for data of Example 2

```
Tree2.rf <- randomForest(Mileage~Price + Country + Reliability +  
Type, data = na.omit(cu.summary), importance=TRUE,  
proximity=TRUE)
```

```
importance(tree2.rf) # importance measure in randomForest
```

	%IncMSE	IncNodePurity
Price	20.942227	403.3665
Country	2.529669	117.6796
Reliability	4.003398	97.8021
Type	16.703569	310.8141

STATS24x7.com© 2010 ADI-NV, INC

13

```
print(tree1)
```

Call:

```
randomForest(formula = Mileage ~ Price + Country  
+ Reliability + Type, data =  
na.omit(cu.summary), importance = TRUE,  
proximity = TRUE)
```

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 1

Mean of squared residuals: 9.9575

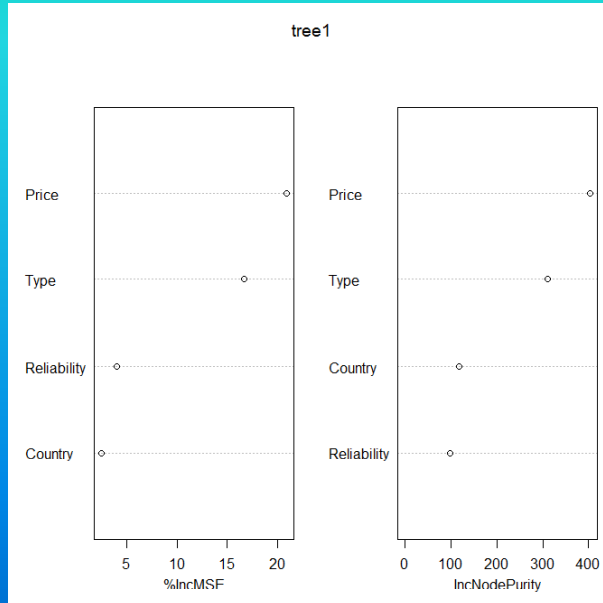
% Var explained: 56.12

STATS24x7.com© 2010 ADI-NV, INC

14

Plot the variable importance measure (in randomForest)

`varImpPlot(tree2.rf)`



Look at correlation matrix among predictors

	[,1]	[,2]	[,3]	[,4]
[1,]	1.00000000	0.006682906	-0.14709251	-0.20823617
[2,]	0.006682906	1.00000000	-0.67467732	-0.02974314
[3,]	-0.147092508	-0.674677317	1.00000000	-0.02352221
[4,]	-0.208236170	-0.029743141	-0.02352221	1.00000000

Predictors are not highly correlated.

b) Use library randomForest for data of Example 2

```
library(rpart)
```

```
tree2.party <- cforest(Mileage~Price + Country + Reliability + Type,  
data = na.omit(cu.summary), control = cforest_unbiased(mtry = 2,  
ntree = 50))
```

```
varimp(tree2.party)
```

```
# Price Country Reliability Type  
# 10.86457802 0.07326038 0.92528646 7.56058877
```

```
varimp(tree2.party, conditional = TRUE)
```

```
# Price Country Reliability Type  
# 10.4190442 0.1071173 0.8996090 5.4114401
```

Unconditional and conditional importance similar, since predictors are uncorrelated in Example 2.

Predict test data using randomForest

Use iris data.

```
iris <- read.csv("K:/TEACH/DataMining_Fall2009/Data/iris.csv",  
header=TRUE)  
attach(iris)
```

```
# split data into training and test, approximately 25% in test
```

```
set.seed(43)  
r <- nrow(iris)  
ind <- sample(r, round(.25*r), replace=TRUE)
```

```
iris.train <- iris[-ind,]  
iris.test <- iris[ind,]
```

```
iris.rf <- randomForest(Species ~ .,
data=iris.train)
iris.pred <- predict(iris.rf, iris.test)

table(observed = iris.test[,5], predicted =
iris.pred)
```

```
#           predicted
# observed  setosa versicolor virginica
# setosa    15      0      0
# versicolor 0     10      1
# virginica  0      0     12
```

```
> iris.rf
```

Call:

```
randomForest(formula = Species ~ ., data = iris.train)
```

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 5.98%

Confusion matrix:

```
      setosa versicolor virginica class.error
setosa   41      0      0 0.00000000
versicolor  0     35      3 0.07894737
virginica  0      4     34 0.10526316
```