# 模型预测的利器 — 随机森林

刘成昊

浙江大学计算机学院
twinsken@gmail.com

October 27, 2012

# Introduction

## Definition

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest)[Brieman 2001]

# Introduction

## Definition

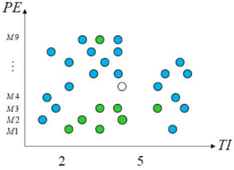
Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest)[Brieman 2001]

- the most successful general-purpose and good-performance algorithm in modern times
- are used not only for prediction ,but also to assess variable importance,outlier detection,clustering data etc.
- can handle "small n large p" -problem,high-order interactions,correlated predictor variable

# Decision Tree

binary-class with two predictor

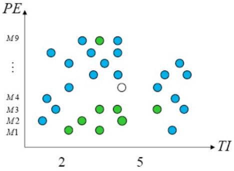
| TI | PE | Response |
|---|---|---|
| 1.0 | M2 | good |
| 2.0 | M1 | bad |
| ... | ... | ... |
| 4.5 | M5 | ? |



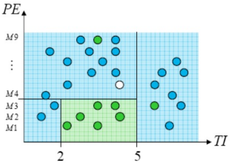
# Decision Tree

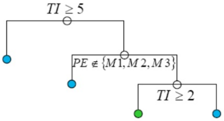
binary-class with two predictor

| TI | PE | Response |
|---|---|---|
| 1.0 | M2 | good |
| 2.0 | M1 | bad |
| ... | ... | ... |
| 4.5 | M5 | ? |




A simpler recursive partitioning tree



# Tree-Based Models

- Classification and regression trees (CART) can be generated through the rpart package.tend to select variables that have many possible splits or many missing values
information measure : information index ,Gini index

# Tree-Based Models

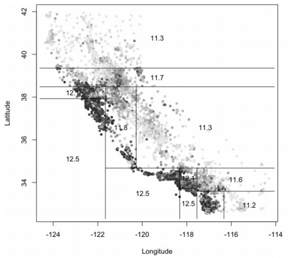
- Classification and regression trees (CART) can be generated through the rpart package.tend to select variables that have many possible splits or many missing values
  information measure : information index ,Gini index
- the ctree in the the party package. avoids the following variable selection bias of rpart,permutation tests to select variables instead of selecting the variable that maximizes an information measure

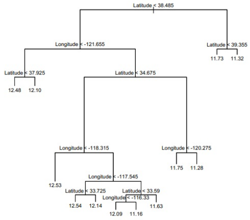
# Tree-Based Models

- Classification and regression trees (CART) can be generated through the rpart package.tend to select variables that have many possible splits or many missing values
  information measure : information index ,Gini index
- the ctree in the the party package. avoids the following variable selection bias of rpart,permutation tests to select variables instead of selecting the variable that maximizes an information measure
- other trees. such as oblique tree,rotation tree etc.

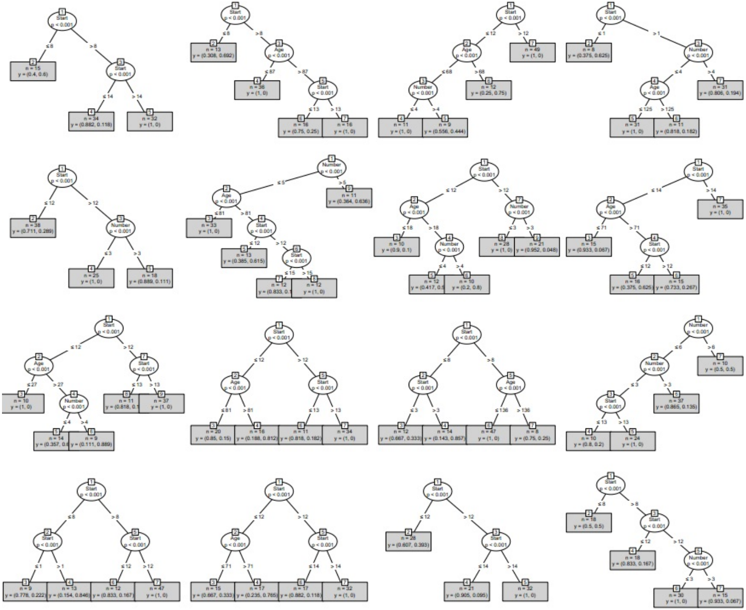
# A single tree can work?

- High variance.depend vary strongly on the particular learning sample used
- quite large and complex
- solution:pruning the tree with cross-validate



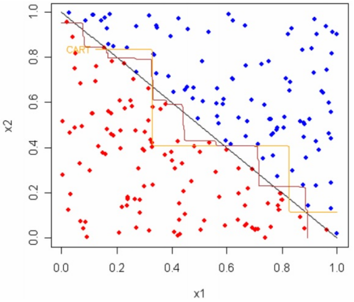


# what about a number of trees together?



# First idea — randomization with Bagging(Bagging Tree)

- create new training sets by random sampling with replacement
- reduce variance



# Second idea — randomization with predictor subsets(Random forest)

- Bagging use the same full set of predictors to determine split. RandomForest choose a random subset of predictors for each split.

# Second idea — randomization with predictor subsets(Random forest)

- Bagging use the same full set of predictors to determine split. RandomForest choose a random subset of predictors for each split.
- Bagging is a special case for randomforest when $m_{try} = K$ . $m_{try} = \sqrt{k}$ for classification and $m_{try} = \frac{k}{3}$ for regression . Empirically ,stronger than Bagging tree,especially K is small.

# Third idea — Extremely randomized tree

- higher randomization levels can improve the accuracy with respect to existing ensemble methods.
  each tree is built from the complete learning sample (no bootstrap copying).
  randomly selected at each interior node(cut-point is selected at random to define a split).

# Third idea — Extremely randomized tree

- higher randomization levels can improve the accuracy with respect to existing ensemble methods.
  each tree is built from the complete learning sample (no bootstrap copying).
  randomly selected at each interior node(cut-point is selected at random to define a split).
- $m_{try} = \sqrt{k}$ for classification and $m_{try} = k$ for regression
  $n_{min}$ the number of samples required for splitting a node.Larger $n_{min}$ lead to smaller trees,higher bias and smaller variance
  $M$ denote the number of trees.compromise between computational requirement and accuracy.

# Random forest in R

- randomForest (pkg:randomForest)
  based on CART trees,in favor of continuous variables and variable with many categories

# Random forest in R

- randomForest (pkg:randomForest)
  based on CART trees,in favor of continuous variables and variable with many categories
- cforest (pkg:party)
  unbiased conditional inference tree

# Random forest in R

- randomForest (pkg:randomForest)
  based on CART trees,in favor of continuous variables and variable with many categories
- cforest (pkg:party)
  unbiased conditional inference tree
- obliqueRF (pkg:obliqueRF)
  the optimal split is sought in the subspace spanned by those features

# cforest

a covariate selection scheme based on statistical theory.
selection by permutation-based significance tests.it can avoid bias

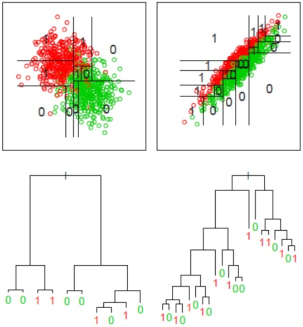
# cforest

a covariate selection scheme based on statistical theory.
selection by permutation-based significance tests.it can avoid bias

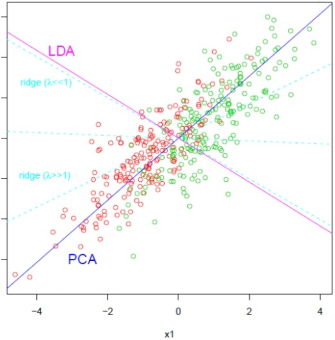
```
require(gtools)
dv <- c(1,3,4,5,5); covariate <- c(2,2,5,4,5)
# all possible permutations of dv, length(120):
perms <- permutations(5,5,dv,set=FALSE)
# now calculate correlations for all perms with covariate:
cors <- apply(perms, 1, function(perms_row) cor(perms_row,covariate))
cors <- cors[order(cors)]
# now p-value: compare cor(dv,covariate) with the
# sorted vector of all permutation correlations
length(cors[cors>=cor(dv,covariate)])/length(cors)
# result: [1] 0.1, i.e. a p-value of .1
# note that this is a one-sided test
```

# oblique Randomforest

- base learner: orthogonal split
- correlated feature values



# oblique Randomforest




**Recursive binary splits** :

$$f_m(\mathbf{x}) : \beta_m^T \mathbf{x} > c_m$$

with coefficients $\beta_m$ and threshold $c_m$

**Coefficients** $\beta_m$ : ridge regression

$$\beta_{ridge}(\lambda) \sim \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left( y_i - \sum_{j=1}^{2} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{P} |\beta_j|^2$$

$$\beta_{ridge}(\lambda') \sim \underset{||\beta=1||}{\operatorname{argmax}}\, corr^2(\beta X, Y) * \frac{var(\beta X)}{var(\beta X) + \lambda'}$$

# oblique Randomforest

oblique random forest with recursive linear model split:

- oRF outperforms RF on spectral data or nominal data especially on few samples,many irrelevant features and correlated predictors
- simpler feature importance/proximity measure

# Variable importance

- Gini importance: Gini gain produced by $X_j$ over all trees

# Variable importance

- Gini importance: Gini gain produced by $X_j$ over all trees
- Permutation importance: decrease in classification accuracy after permuting $X_j$ over all trees

# Variable importance

- Gini importance: Gini gain produced by $X_j$ over all trees
- Permutation importance: decrease in classification accuracy after permuting $X_j$ over all trees
- oRF importance:calculate ANOVA at every split

# Variable importance

- Gini importance: Gini gain produced by $X_j$ over all trees
- Permutation importance: decrease in classification accuracy after permuting $X_j$ over all trees
- oRF importance:calculate ANOVA at every split
- Conditional importance: Some Variable has no effect of its own,but correlated with a relevant predictor

# Variable importance code

```
# pkg :randomForest
# type = 1 Permutation importance,2 Gini importance
obj <- randomForest(...,importance=TURE)
importance(obj,type=1)
# pkg :party
# Permutation importance
obj <- cforest(...)
varimp(obj)
# oRF importance
obj <- obluqyeRF(...,bImportance=TRUE)
importance(obj)
# pkg :party
# Conditional importance
obj <- cforest(...)
varimp(obj,conditional = TRUE)
```

# Imbalance problem

- down-sampling the majority,over-sampling the minority

# Imbalance problem

- down-sampling the majority,over-sampling the minority
- cost-sensitive learning.place a heavier penalty on misclassifying the minority class.weights for finding splits (weighted loss function) and weights in the terminal node(weighted majority vote)

# Missing value problem

- pkg:randomForest need to impute the missing data
  - impute with median or maximum category
  - Impute missing values in predictor data using proximityrfImpute(x, y, iter=5, ntree=300, ...

# Missing value problem

- pkg:randomForest need to impute the missing data
  - impute with median or maximum category
  - Impute missing values in predictor data using proximityrfImpute(x, y, iter=5, ntree=300, ...
- pkg:party can handle missing value with surrogate split

# Missing value problem

- pkg:randomForest need to impute the missing data
  - impute with median or maximum category
  - Impute missing values in predictor data using proximityrfImpute(x, y, iter=5, ntree=300, ...
- pkg:party can handle missing value with surrogate split
- other impute methods
  - knnimpute,multiply imputationpkg:mice
  - pkg:missForest.don't need response variable

# missForest

1. The observed values of variable $\mathbf{X}_s$, denoted by $\mathbf{y}_{obs}^{(s)}$;
2. the missing values of variable $\mathbf{X}_s$, denoted by $\mathbf{y}_{mis}^{(s)}$;
3. the variables other than $\mathbf{X}_s$ with observations $\mathbf{i}_{obs}^{(s)} = \{1, \dots, n\} \setminus \mathbf{i}_{mis}^{(s)}$ denoted by $\mathbf{x}_{obs}^{(s)}$;
4. the variables other than $\mathbf{X}_s$ with observations $\mathbf{i}_{mis}^{(s)}$ denoted by $\mathbf{x}_{mis}^{(s)}$.

**Require:** $\mathbf{X}$ an $n \times p$ matrix, stopping criterion $\gamma$

Make initial guess for missing values;
$\mathbf{k} \leftarrow$ vector of sorted indices of columns in $\mathbf{X}$ w.r.t. increasing amount of missing values;
**while** not $\gamma$ **do**
    $\mathbf{X}_{old}^{imp} \leftarrow$ store previously imputed matrix;
    **for** $s$ in $\mathbf{k}$ **do**
        Fit a random forest: $\mathbf{y}_{obs}^{(s)} \sim \mathbf{x}_{obs}^{(s)}$;
        Predict $\mathbf{y}_{mis}^{(s)}$ using $\mathbf{x}_{mis}^{(s)}$;
        $\mathbf{X}_{new}^{imp} \leftarrow$ update imputed matrix, using predicted $\mathbf{y}_{mis}^{(s)}$;
    **end for**
    update $\gamma$.
**end while**
**return** the imputed matrix $\mathbf{X}^{imp}$

Thank you!