R Examples of Using Some Prediction Tools
(Highlight: Random Forest)

January 9, 2009

Contents

1 Introduction 1
2 A Collection of Prediction Tools 2
3 Comparison 8
4 Conclusion 9

1 Introduction

We intend to advocate the prediction tool Random Forest, which is very powerful yet easy to use. To help understanding, we set it in a context of other tools and talk about them in the following sequence:

1. Tree: A building block.
2. Bagging: Improvement by tree ensembles.
3. Random Forest: Injecting more randomness into tree ensembles.

Demo Data For illustration, we use the Forensic Glass data in the MASS package. The goal is predict type (of glass fragments) with a set of predictors (of chemical properties). For illustration, we sample 10 data points as test sample and use the rest as training sample.

> library(MASS)
> data(fgl)
> str(fgl)
'data.frame': 214 obs. of 10 variables:
$ RI  : num 3.01 -0.39 -1.82 -0.34 -0.58 ...  
$ Na  : num 13.6 13.9 13.5 13.2 13.3 ...  
$ Mg  : num 4.49 3.6 3.55 3.62 3.61 3.6 3.61 3.58 3.6 ...  
$ Al  : num 1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...  
$ Si  : num 71.8 72.7 73.0 72.6 73.1 ...  
$ K   : num 0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...  
$ Ca  : num 8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...  
$ Ba  : num 0 0 0 0 0 0 0 0 0 0 ...  
$ Fe  : num 0 0 0 0 0 0.26 0 0 0 0.11 ...  
$ type: Factor w/ 6 levels "WinF","WinNF",...: 1 1 1 1 1 1 1 1 1 1 ...  

> set.seed(1)  
> s <- sample(dim(fgl)[1], 10)  
> test <- fgl[s, ]  
> train <- fgl[-s, ]  

2 A Collection of Prediction Tools

**Tree**  
A tree-based prediction method (e.g. CART) partitions the feature (variables) space into a set of rectangles, on which fixed constants (predictions) are assigned. We can use the `rpart` function in the `rpart` package, which implements CART.

> library(rpart)  
> p1 <- rpart(type ~ ., data = train)  
> plot(p1)  
> text(p1)
**Bagging**  Bagging (Bootstrap Aggregation) simply grows multiple trees, each tree growing on a different bootstrap sample. It then reports the majority vote or mean response (across all trees) as the prediction. We can use the `bagging` function in the `ipred` package.

```r
> library(ipred)
> p2 <- bagging(type ~ ., data = fgl, coob = T)
> p2
```

Bagging classification trees with 25 bootstrap replications

Call: `bagging.data.frame(formula = type ~ ., data = fgl, coob = T)`

Out-of-bag estimate of misclassification error: 0.2477

The `coob` option requests the out-of-bag estimate of the misclassification error.

**Random Forest**  Random Forest injects additional randomness into the bagging procedure on trees: each node is split using the best among a *subset* of predictors randomly chosen at that node, instead of the full set. It has the following merits:
• Superior performance.
• Robust against overfitting.
• Easy to use, little tuning.

Thus this is a highly recommended prediction tool. My own hypothesis for its performance is that the additional randomness greatly diversifies the trees, resulting in expanded search space and noise profile, the former reduces the bias and the latter reduces the tendency for overfitting by keeping a healthy signal-noise ratio.

```r
> library(randomForest)
> p3 <- randomForest(type ~ ., data = train, importance = T)
> p3

Call:
  randomForest(formula = type ~ ., data = train, importance = T)
Type of random forest: classification
  Number of trees: 500
No. of variables tried at each split: 3

OOB estimate of error rate: 19.12%
Confusion matrix:
    WinF WinNF Veh Con Tabl Head class.error
  WinF   61    5  1  0  0  0  0.08955224
  WinNF   8  57  2  2  2  1  0.20833333
   Veh    6    4  7  0  0  0  0.58823529
   Con    0    2  0 10  0  1  0.23076923
   Tabl   0    2  0  0  7  0  0.22222222
   Head   0    3  0  0  0 23  0.11538462

> plot(p3)
```
The **plot** method traces the error rates (out-of-bag, and by each response category) as the number of trees increases. The **importance** option in the **randomForest** function requests the assessment of predictor importances. There are two global measures: one is the mean decrease in accuracy over all classes, the other is the mean decrease in Gini index. Here is a plot of the two measures:

```
> par(mfrow = c(2, 1))
> barplot(p3$importance[, 7], main = "Importance (Dec.Accuracy)")
> barplot(p3$importance[, 8], main = "Importance (Gini Index")
```
Boosting  Boosting is a method for starting with a simple/weak classifier (e.g. a tree) and gradually improving it by refitting the data giving higher weight to misclassified samples. The prediction is voted by the resulting ensemble/committee of classifiers. In essence, boosting is a way of fitting an additive expansion in a set of elementary “basis” functions ([1]).

Unfortunately, currently no boosting package can deal directly with multinomial response (only continuous and binary). So we will use the Fisher’s Iris Data (keep only two species) for illustration.

```r
> data(iris)
> iris <- iris[iris$Species %in% c("versicolor", "virginica"), +  ]
> iris$Species <- factor(iris$Species)
> library(ada)
> p4 <- ada(Species ~ ., data = iris)
> p4

Call:
  ada(Species ~ ., data = iris)

Loss: exponential  Method: discrete  Iteration: 50
```
Final Confusion Matrix for Data:

<table>
<thead>
<tr>
<th></th>
<th>versicolor</th>
<th>virginica</th>
</tr>
</thead>
<tbody>
<tr>
<td>versicolor</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>virginica</td>
<td>2</td>
<td>48</td>
</tr>
</tbody>
</table>

Train Error: 0.05

Out-Of-Bag Error: 0.04 iteration= 14

Additional Estimates of number of iterations:

train.err1 train.kap1
5 5

> plot(p4)

Following is the variable importance plot:

> varplot(p4)
3 Comparison

Prediction on the Test Sample  The prediction accuracy on the test sample for Tree, Bagging, and Random Forest is:

```r
> data.frame(Truth = test$type, Tree = predict(p1, test, type = "class"),
+ Bagging = predict(p2, test), Forest = predict(p3, test))

<table>
<thead>
<tr>
<th>Truth</th>
<th>Tree</th>
<th>Bagging</th>
<th>Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>WinF</td>
<td>Veh</td>
<td>WinF</td>
</tr>
<tr>
<td>80</td>
<td>WinNF</td>
<td>WinNF</td>
<td>WinNF</td>
</tr>
<tr>
<td>122</td>
<td>WinNF</td>
<td>WinNF</td>
<td>WinNF</td>
</tr>
<tr>
<td>192</td>
<td>Head</td>
<td>Head</td>
<td>Head</td>
</tr>
<tr>
<td>43</td>
<td>WinF</td>
<td>WinF</td>
<td>WinF</td>
</tr>
<tr>
<td>188</td>
<td>Head</td>
<td>WinF</td>
<td>Head</td>
</tr>
<tr>
<td>197</td>
<td>Head</td>
<td>Head</td>
<td>Head</td>
</tr>
<tr>
<td>137</td>
<td>WinNF</td>
<td>WinNF</td>
<td>WinF</td>
</tr>
<tr>
<td>130</td>
<td>WinNF</td>
<td>Con</td>
<td>WinNF</td>
</tr>
<tr>
<td>13</td>
<td>WinF</td>
<td>WinNF</td>
<td>WinF</td>
</tr>
</tbody>
</table>
```

Incidentally, Bagging performs better than Forest on this test sample. Note Bagging is a special case of Forest. For a more rigorous check, we shall estimate the test error rate.
Error Rate Estimation  To compare the performances of different prediction tools, we can do a 10-fold cross validation to estimate the test error, using the errorest function in the ipred package. This function requires a predict function that specifies only two arguments (object and newdata) and returns a predicted class (or scalar). So we first need to write a wrapper function for predict.rpart:

```r
> mypredict.rpart <- function(object, newdata) {
+  predict(object, newdata = newdata, type = "class")
+ }
```

We can see a significant improvement by Random Forest in the following error rate comparison:

```r
> c(Tree = errorest(type ~ ., data = train, model = rpart, predict = mypredict.rpart)$error,
+    Bagging = errorest(type ~ ., data = train, model = bagging)$error,
+    Forest = errorest(type ~ ., data = train, model = randomForest)$error)
```

<table>
<thead>
<tr>
<th></th>
<th>Tree</th>
<th>Bagging</th>
<th>Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>0.3186275</td>
<td>0.2303922</td>
<td>0.1911765</td>
</tr>
</tbody>
</table>

Following is the error rate comparison for the Fisher's Iris data:

```r
> c(Tree = errorest(Species ~ ., data = iris, model = rpart, predict = mypredict.rpart)$error,
+    Bagging = errorest(Species ~ ., data = iris, model = bagging)$error,
+    Forest = errorest(Species ~ ., data = iris, model = randomForest)$error,
+    Boosting = errorest(Species ~ ., data = iris, model = ada)$error)
```

<table>
<thead>
<tr>
<th></th>
<th>Tree</th>
<th>Bagging</th>
<th>Forest</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>0.10</td>
<td>0.09</td>
<td>0.07</td>
<td>0.08</td>
</tr>
</tbody>
</table>

4 Conclusion

We introduced a set of prediction tools (Tree, Bagging, Forest, Boosting). Tree is a nonlinear method and serves as the building block for the other three tools. The drawback of Tree is that it is instable (sensitive to data noise) and has high variance in the prediction. Bagging reduces such variance by bootstrapping the samples. In contrast, Boosting can be think of as a bias reduction tool.

We recommend the Random Forest for routine prediction tasks.

References