# Classification Using C5.0

## UseR! 2013

Max Kuhn

Pfizer Global R&D
Groton, CT
max.kuhn@pfizer.com

## Historical Notes

In the late 1970's Ross Quinlan was developing tree–based models such as ID3.

In the 1980's these methods evolved into a classification tree model called C4.5 (Quinlan, 1993)

Although Quinlan published very little on this model after his book, he was continually evolving the classification tree and rule–based models into the latest incarnation called C5.0

C5.0 was proprietary and commercially available until 2011 when a GPL version was released.

Kuhn and Johnson (2013) have a more complete description of C5.0 (and another unpublished model called Cubist).

## C4.5 And CART

*Some* of the differences between CART (Breiman *et al*, 1984) and C4.5 are:

- A different impurity measure is used (entropy).
- Tree pruning is done using *pessimistic pruning*.
- Splits on categorical predictors are handled very differently.
- Trees can be converted to rules.
- Missing values are handled using sending fractional samples into subsequent nodes.

(Another model, J48, is available in Weka and is very similar to C4.5. However, it does not combine categorical predictors, which leads to very different (= poor) trees and rules.)

## Rule–Based Models

`if-then` statements generated by a tree define a unique route to one terminal node for any sample.

A *rule* is a set of `if-then` conditions that have been collapsed into independent conditions.

For the example:

```
if X1 >= 1.7 and X2 >= 202.1 then Class = 1
if X1 >= 1.7 and X2 <  202.1 then Class = 1
if X1 <  1.7 then Class = 2
```

Rules can be simplified or pruned in a way that samples are covered by multiple rules, eg.

C4.5 (and C5.0) have options to convert the tree to rules.

## C5.0 vs C4.5

There are a number of subtle differences in the pruning process and a few other sub–routines.

C5.0 trees (and rulesets) are usually smaller than their C4.5 counterparts.

Rulesets have different class assignment algorithms.

The main differences between the two algorithms are: boosting, winnowing and asymmetric costs for specific errors.

## Boosting

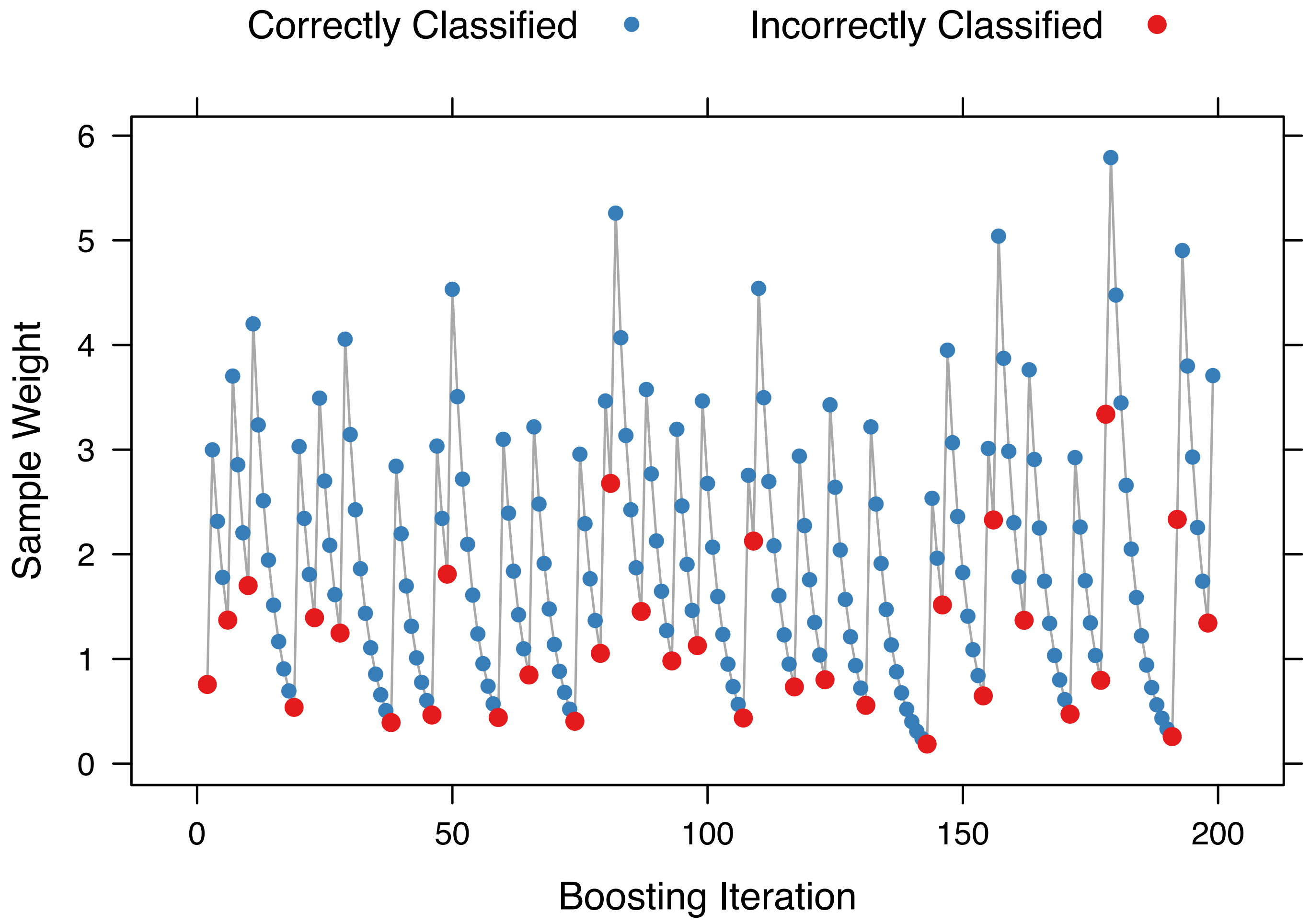
While the stochastic gradient boosting machines diverged from the original adaboost algorithm, C5.0 does something similar to adaboost.

After the first tree is created, weights are determined and subsequent iterations create weighted trees or rulesets.

Subsequent trees (or rulesets) are constrained to be about the same size as the initial model.

The final prediction is a simple averages of class probabilities generated from each tree or ruleset (i.e. no stage weights).

# C5.0 Weighting Scheme



## Example Data

We will use the HPC job scheduling data as an example. These data consist of information about compute jobs sent to a queuing system.

Given details about the job, predict how long the job will take (i.e. very fast, fast, moderate, long).

```
> str(schedulingData)

data.frame: 4331 obs. of  8 variables:
 $ Protocol   : Factor w/ 14 levels "A","C","D","E",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Compounds  : num  997 97 101 93 100 100 105 98 101 95 ...
 $ InputFields: num  137 103 75 76 82 82 88 95 91 92 ...
 $ Iterations : num  20 20 10 20 20 20 20 20 20 20 ...
 $ NumPending : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Hour       : num  14 13.8 13.8 10.1 10.4 ...
 $ Day        : Factor w/ 7 levels "Mon","Tue","Wed",..: 2 2 4 5 5 3 5 5 5 3 ...
 $ Class      : Factor w/ 4 levels "VF","F","M","L": 2 1 1 1 1 1 1 1 1 1 ...

> table(schedulingData$Class)

  VF    F    M    L
2211 1347  514  259
```

## Example Data

```
> library(AppliedPredictiveModeling)
> data(schedulingData)
> library(caret)
> set.seed(733)
> inTrain <- createDataPartition(schedulingData$Class, p = .75, list = FALSE)
> training <- schedulingData[ inTrain,]
> testing  <- schedulingData[-inTrain,]
```

## Syntax for a Single Tree

```
> library(C50)
> oneTree <- C5.0(Class ~ ., data = training)
> ## also C5.0(x, y) interface is available
> ## Factor predictors are not converted to dummy vars
> oneTree

Call:
C5.0.formula(formula = Class ~ ., data = training)


Classification Tree
Number of samples: 3251
Number of predictors: 7

Tree size: 199

Non-standard options: attempt to group attributes

> oneTreePred  <- predict(oneTree, testing)
> oneTreeProbs <- predict(oneTree, testing, type ="prob")
> postResample(oneTreePred, testing$Class)

 Accuracy     Kappa
0.8166667 0.7025836
```

## Single Tree

```
> summary(oneTree)

Call:
C5.0.formula(formula = Class ~ ., data = training)


C5.0 [Release 2.07 GPL Edition]   Wed Jul 10 21:28:01 2013
-------------------------------

Class specified by attribute outcome

Read 3251 cases (8 attributes) from undefined.data

Decision tree:

Protocol in {H,M,O}:
:...Iterations > 150: L (61)
:   Iterations <= 150:
:   :...Compounds <= 211:
:       :...Iterations > 50:

<snip>
```

## Single Ruleset

```
> rules <- C5.0(Class ~ ., data = training, rules = TRUE)
> postResample(predict(rules, testing), testing$Class)

 Accuracy     Kappa
0.8101852 0.6939689

> summary(rules)

<snip>

Rule 1: (400/2, lift 1.9)
Protocol in {C, D, I, K}
Compounds <= 640
Iterations <= 30
->  class VF  [0.993]

Rule 2: (213/2, lift 1.9)
Protocol in {A, D, E, G, J, N}
Compounds <= 640
InputFields <= 39
Iterations <= 30
NumPending <= 129
->  class VF  [0.986]

Rule 3: (118/1, lift 1.9)

<snip>
```

## Boosted Tree

```
> ## Rules can also be boosted
> bstTree <- C5.0(Class ~ ., data = training, trials = 10)
> bstTree

Call:
C5.0.formula(formula = Class ~ ., data = training, trials = 10)

Classification Tree
Number of samples: 3251
Number of predictors: 7

Number of boosting iterations: 10
Average tree size: 180.3

Non-standard options: attempt to group attributes

> bstTreePred <- predict(bstTree, testing)
> postResample(bstTreePred, testing$Class)

 Accuracy     Kappa
0.8342593 0.7322586
```

## Winnowing

Winnowing is a feature selection step conducted before modeling.

The data set is randomly split in half and an initial model is fit.

Each predictor is removed in turn and the effect on model performance is determined (using the other half of the random split).

Predictors are flagged if their removal does not increase the error rate.

The final model is fit to all of the training set samples using only the unflagged predictors.

This procedure can be used via

```
> mod <- C5.0(Class ~ ., data = training, control = C5.0Control(winnow = TRUE))
```

## Some Other Features

- The confidence factor for pruning can be changed (`C5.0Control(CF = .25)`)
- The minimum number of cases in a terminal node can also be adjusted (`C5.0Control(minCases = 2)`)
- "Fuzzy thresholding" can also be specified (`C5.0Control(fuzzyThreshold = FALSE)`)
- An optional global pruning algorithm can be turned on (`C5.0Control(noGlobalPruning = FALSE)`)
- By default, C5.0 will cancel boosting if it appears to be very ineffective. You can turn this off via (`C5.0Control(earlyStopping = FALSE)`)
- Asymmetric costs can be assigned to specific types of errors (`C5.0(costs = matrix())`). See Kuhn and Johnson (2013) for an example.

## Using train

The caret function train has bindings to C5.0 that will tune over the type of model, winnowing and boosting:

```
> tuned <- train(training[, 1:7], training$Class,
+                method = "C5.0", tuneLength = 11,
+                trControl = trainControl(method = "repeatedcv",
+                                         repeats = 5),
+                ## Other options can be passed thru too, e.g.
+                ## control = C5.0Control(earlyStopping = FALSE) etc.
+                metric = "Kappa")
```

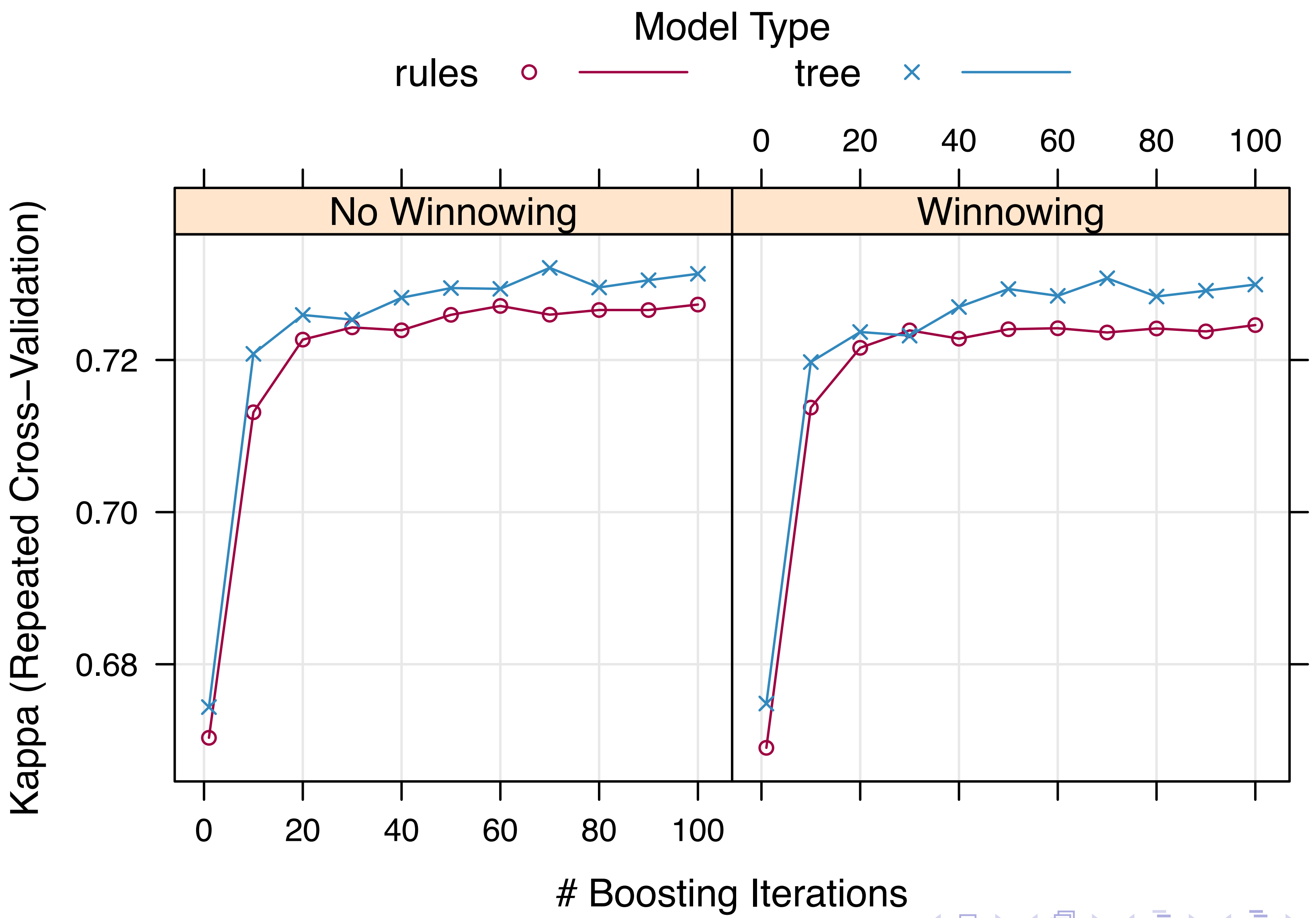
The optimal parameters used trees, no winnowing and 70 iterations of boosting.

```
> postResample(predict(tuned, testing), testing$Class)

 Accuracy     Kappa
0.8435185 0.7476887
```

## Resampling Profile

```
plot(tuned, metric = "Kappa")
```



## Acknowledgments

Thanks to:

- **Ross Quinlan**
- Steve Weston
- Nathan Coulter
- R Core and R-Forge Administrator

## References

Breiman L, Friedman J, Olshen R, Stone C (1984). *Classification and Regression Trees*. Chapman and Hall, New York.

Kuhn M, Johnson K (2013). *Applied Predictive Modeling*. Springer

Quinlan R (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

## Versions

- R version 3.0.0 (2013-04-03), `x86_64-apple-darwin10.8.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, splines, stats, tools, utils
- Other packages: AppliedPredictiveModeling 1.1-001, C50 0.1.0-15, caret 5.17-07, class 7.3-7, cluster 1.14.4, codetools 0.2-8, CORElearn 0.9.41, digest 0.6.3, doMC 1.3.0, e1071 1.6-1, foreach 1.4.1, Hmisc 3.10-1, iterators 1.0.6, lattice 0.20-15, MASS 7.3-26, mlbench 2.1-1, plyr 1.8, pROC 1.5.4, reshape2 1.2.2, rpart 4.1-1, survival 2.37-4, weaver 1.26.0
- Loaded via a namespace (and not attached): grid 3.0.0, stringr 0.6.2