Cubist Models For Regression

Max Kuhn (max.kuhn@pfizer.com)
Steve Weston,
Chris Keefer

April 22, 2011

1 Introduction

Cubist is an R port of the Cubist GPL C code released by RuleQuest at

http://rulequest.com/cubist-info.html

See the last section of this document for information on the porting. The other parts describes the functionality of the R package.

2 Model Trees

Cubist is a rule–based model that is an extension of Quinlan's M5 model tree. A tree is grown where the terminal leaves contain linear regression models. These models are based on the predictors used in previous splits. Also, there are intermediate linear models at each step of the tree. A prediction is made using the linear regression model at the terminal node of the tree, but is "smoothed" by taking into account the prediction from the linear model in the previous node of the tree (which also occurs recursively up the tree). The tree is reduced to a set of rules, which initially are paths from the top of the tree to the bottom. Rules are eliminated via pruning and/or combined for simplification.

This is explained better in Quinlan (1992). Wang and Witten (1997) attempted to recreate this model using a "rational reconstruction" of Quinlan (1992) that is the basis for the `M5P` model in Weka (and the R package RWeka).

An example of a model tree can be illustrated using the Boston Housing data in the mlbench package.

```
> library(Cubist)
> library(mlbench)
> data(BostonHousing)
> set.seed(1)
> inTrain <- sample(1:nrow(BostonHousing), floor(.8*nrow(BostonHousing)))
> trainingPredictors <- BostonHousing[ inTrain, -14]
> testPredictors     <- BostonHousing[-inTrain, -14]
> trainingOutcome <- BostonHousing$medv[ inTrain]
> testOutcome     <- BostonHousing$medv[-inTrain]
> modelTree <- cubist(x = trainingPredictors, y = trainingOutcome)
> modelTree

Call:
cubist.default(x = trainingPredictors, y = trainingOutcome)

Number of samples: 404
Number of predictors: 13

Number of committees: 1
Number of rules: 4


> summary(modelTree)

Call:
cubist.default(x = trainingPredictors, y = trainingOutcome)


Cubist [Release 2.07 GPL Edition]  Fri Apr 22 22:27:36 2011
-------------------------------

    Target attribute `outcome'

Read 404 cases (14 attributes) from undefined.data

Model:

  Rule 1: [88 cases, mean 13.81, range 5 to 27.5, est err 2.10]

    if
        nox > 0.668
    then
        outcome = 2.07 + 3.14 dis - 0.35 lstat + 18.8 nox + 0.007 b
                  - 0.12 ptratio - 0.008 age - 0.02 crim

  Rule 2: [153 cases, mean 19.54, range 8.1 to 31, est err 2.16]

    if
        nox <= 0.668
        lstat > 9.59
    then
```

```
        outcome = 34.81 - 1 dis - 0.72 ptratio - 0.056 age - 0.19 lstat + 1.5 rm
                  - 0.11 indus + 0.004 b

  Rule 3: [39 cases, mean 24.10, range 11.9 to 50, est err 2.73]

    if
        rm <= 6.23
        lstat <= 9.59
    then
        outcome = 11.89 + 3.69 crim - 1.25 lstat + 3.9 rm - 0.0045 tax
                  - 0.16 ptratio

  Rule 4: [128 cases, mean 31.31, range 16.5 to 50, est err 2.95]

    if
        rm > 6.23
        lstat <= 9.59
    then
        outcome = -1.13 + 1.6 crim - 0.93 lstat + 8.6 rm - 0.0141 tax
                  - 0.83 ptratio - 0.47 dis - 0.019 age - 1.1 nox


Evaluation on training data (404 cases):

    Average  |error|               2.27
    Relative |error|               0.34
    Correlation coefficient        0.94


        Attribute usage:
          Conds  Model

           78%   100%    lstat
           59%    53%    nox
           41%    78%    rm
                 100%    ptratio
                  90%    age
                  90%    dis
                  62%    crim
                  59%    b
                  41%    tax
                  38%    indus


Time: 0.0 secs
```

There is no formula method for `cubist`; the predictors are specified as matrix or data frame and the outcome is a numeric vector.

There is a predict method for the model:

```
> mtPred <- predict(modelTree, testPredictors)
> ## Test set RMSE
> sqrt(mean((mtPred - testOutcome)^2))

[1] 3.337924

> ## Test set R^2
> cor(mtPred, testOutcome)^2

[1] 0.8573504
```

3 Boosting

The Cubist model can also use a boosting–like scheme called *committees* where iterative model trees are created in sequence. The first tree follows the procedure described in the last section. Subsequent trees using a weighting scheme similar to gradient boosting where case weights are applied based on the errors from previous model trees. Unlike traditional boosting, stage weights for each committee are not used to average the predictions from each model tree; the final prediction is a simple average of the predictions from each model tree.

The `committee` option can be used to control number of model trees:

```
> set.seed(1)
> committeeModel <- cubist(x = trainingPredictors, y = trainingOutcome,
+                          committees = 5)
> summary(committeeModel)

Call:
cubist.default(x = trainingPredictors, y = trainingOutcome, committees = 5)


Cubist [Release 2.07 GPL Edition]  Fri Apr 22 22:27:36 2011
-------------------------------

    Target attribute `outcome'

Read 404 cases (14 attributes) from undefined.data

Model 1:

  Rule 1/1: [88 cases, mean 13.81, range 5 to 27.5, est err 2.10]

    if
        nox > 0.668
    then
        outcome = 2.07 + 3.14 dis - 0.35 lstat + 18.8 nox + 0.007 b
```

```
                - 0.12 ptratio - 0.008 age - 0.02 crim

  Rule 1/2: [153 cases, mean 19.54, range 8.1 to 31, est err 2.16]

    if
        nox <= 0.668
        lstat > 9.59
    then
        outcome = 34.81 - 1 dis - 0.72 ptratio - 0.056 age - 0.19 lstat + 1.5 rm
                  - 0.11 indus + 0.004 b

  Rule 1/3: [39 cases, mean 24.10, range 11.9 to 50, est err 2.73]

    if
        rm <= 6.23
        lstat <= 9.59
    then
        outcome = 11.89 + 3.69 crim - 1.25 lstat + 3.9 rm - 0.0045 tax
                  - 0.16 ptratio

  Rule 1/4: [128 cases, mean 31.31, range 16.5 to 50, est err 2.95]

    if
        rm > 6.23
        lstat <= 9.59
    then
        outcome = -1.13 + 1.6 crim - 0.93 lstat + 8.6 rm - 0.0141 tax
                  - 0.83 ptratio - 0.47 dis - 0.019 age - 1.1 nox

Model 2:

  Rule 2/1: [71 cases, mean 13.41, range 5 to 27.5, est err 2.62]

    if
        crim > 5.69175
        dis > 1.4254
    then
        outcome = 43.01 + 2.57 dis - 0.47 lstat - 0.7 ptratio - 2 rm

  Rule 2/2: [84 cases, mean 18.75, range 8.1 to 27.5, est err 2.25]

    if
        crim <= 5.69175
        nox > 0.532
        dis > 1.4254
        tax > 222
        ptratio > 17
    then
        outcome = 44.08 + 1.19 crim - 0.43 lstat - 1.05 ptratio - 0.011 age

  Rule 2/3: [15 cases, mean 23.43, range 5 to 50, est err 5.62]
```

```
  if
      dis <= 1.4254
      ptratio > 17
  then
      outcome = 174.86 - 100.95 dis - 1.07 lstat - 0.09 ptratio

Rule 2/4: [77 cases, mean 23.90, range 11.8 to 50, est err 2.35]

  if
      ptratio <= 17
      lstat > 5.12
  then
      outcome = -2.7 + 8.2 rm - 0.0228 tax - 1.68 dis - 0.064 age - 0.1 lstat
                - 0.24 ptratio - 3.7 nox + 0.009 zn - 0.02 crim - 0.02 indus
                + 0.001 b

Rule 2/5: [128 cases, mean 25.56, range 14.4 to 50, est err 3.12]

  if
      crim <= 5.69175
      nox <= 0.532
      ptratio > 17
  then
      outcome = -14.6 + 2.4 crim + 7 rm - 0.075 age + 0.23 lstat - 0.42 dis
                - 0.17 ptratio

Rule 2/6: [16 cases, mean 27.91, range 15.7 to 39.8, est err 5.25]

  if
      tax <= 222
      lstat > 5.12
  then
      outcome = 274.62 - 12.31 ptratio - 0.212 age - 0.03 lstat

Rule 2/7: [18 cases, mean 30.49, range 22.5 to 50, est err 3.76]

  if
      rm <= 6.861
      lstat <= 5.12
  then
      outcome = -58.67 + 10.99 crim + 13.3 rm - 0.08 dis - 0.02 lstat
                - 0.05 ptratio

Rule 2/8: [19 cases, mean 41.54, range 31.2 to 50, est err 3.64]

  if
      rm > 6.861
      age <= 71
      lstat <= 5.12
  then
```

```
        outcome = -57.01 + 14.2 rm - 0.23 dis - 0.05 lstat - 0.13 ptratio
                  - 2.2 nox + 0.006 zn - 0.01 crim

  Rule 2/9: [14 cases, mean 43.48, range 22.8 to 50, est err 5.55]

    if
        age > 71
        lstat <= 5.12
    then
        outcome = -24.48 + 1.99 crim + 0.467 age + 3.5 rm

Model 3:

  Rule 3/1: [88 cases, mean 13.81, range 5 to 27.5, est err 2.29]

    if
        nox > 0.668
    then
        outcome = -12.77 + 5.44 dis + 22.7 nox - 0.18 lstat + 0.013 b
                  - 0.07 crim

  Rule 3/2: [10 cases, mean 17.64, range 11.7 to 27.5, est err 11.51]

    if
        nox <= 0.668
        b <= 179.36
    then
        outcome = -2.19 + 0.149 b + 0.76 lstat

  Rule 3/3: [156 cases, mean 19.68, range 8.1 to 33.8, est err 2.23]

    if
        nox <= 0.668
        lstat > 9.53
    then
        outcome = 28.86 - 1.08 dis - 0.27 lstat - 0.067 age + 2.6 rm
                  - 0.62 ptratio

  Rule 3/4: [164 cases, mean 29.68, range 11.9 to 50, est err 3.44]

    if
        lstat <= 9.53
    then
        outcome = 6.37 + 4.09 crim - 0.75 lstat + 7.6 rm - 0.0303 tax
                  - 0.78 ptratio - 0.14 dis - 2.2 nox + 0.001 b

Model 4:

  Rule 4/1: [335 cases, mean 19.44, range 5 to 50, est err 2.60]

    if
```

```
        rm <= 7.079
        lstat > 5.12
    then
        outcome = 50.33 - 0.0168 tax - 0.39 lstat + 0.32 rad - 12.7 nox
                  - 0.65 ptratio - 0.59 dis - 0.11 crim

  Rule 4/2: [18 cases, mean 30.49, range 22.5 to 50, est err 4.90]

    if
        rm <= 6.861
        lstat <= 5.12
    then
        outcome = 20.98 + 8.17 crim - 0.54 lstat + 0.23 rad + 1.3 rm

  Rule 4/3: [5 cases, mean 31.10, range 13.8 to 50, est err 24.23]

    if
        dis <= 1.2852
        lstat > 5.12
    then
        outcome = 36.25

  Rule 4/4: [35 cases, mean 36.15, range 22.5 to 50, est err 3.57]

    if
        age <= 71
        lstat <= 5.12
    then
        outcome = -67.2 + 15.9 rm - 1.04 rad - 0.005 b - 0.05 lstat - 0.05 dis

  Rule 4/5: [43 cases, mean 39.37, range 15 to 50, est err 6.47]

    if
        rm > 7.079
    then
        outcome = -132.71 + 0.323 b + 9.1 rm - 0.48 rad - 1.4 ptratio
                  - 0.0015 tax - 0.03 lstat

  Rule 4/6: [14 cases, mean 43.48, range 22.8 to 50, est err 5.19]

    if
        age > 71
        lstat <= 5.12
    then
        outcome = -34.38 + 0.6 age - 0.75 lstat + 6.1 rm - 0.047 b + 0.16 rad

Model 5:

  Rule 5/1: [86 cases, mean 13.32, range 5 to 23.2, est err 2.71]

    if
```

```
      nox > 0.659
      lstat > 9.53
  then
      outcome = -28.42 + 7.76 dis + 35.5 nox + 0.017 b - 0.12 lstat
                - 0.18 ptratio + 0.4 rm - 0.01 age - 0.03 crim + 0.02 rad
                - 0.0006 tax

Rule 5/2: [154 cases, mean 19.76, range 8.1 to 33.8, est err 2.34]

  if
      nox <= 0.659
      lstat > 9.53
  then
      outcome = 32.87 - 1.36 dis - 0.1 age + 2.5 rm - 0.68 ptratio - 0.1 lstat
                - 2.8 nox - 0.03 crim + 0.002 b + 0.01 rad

Rule 5/3: [138 cases, mean 28.31, range 16.5 to 50, est err 2.52]

  if
      dis > 2.6403
      lstat <= 9.53
  then
      outcome = -34.37 + 11.1 rm + 0.81 crim - 0.2 lstat - 0.0064 tax
                - 0.03 age - 0.3 ptratio - 0.17 dis + 0.03 rad - 2.2 nox
                + 0.002 b

Rule 5/4: [26 cases, mean 36.97, range 11.9 to 50, est err 11.81]

  if
      dis <= 2.6403
      lstat <= 9.53
  then
      outcome = -1.18 + 2.48 crim + 3.6 rm

Rule 5/5: [21 cases, mean 41.33, range 21.9 to 50, est err 10.93]

  if
      nox > 0.573
      lstat <= 9.53
  then
      outcome = 61.04 + 4.95 crim - 51.6 nox - 0.0226 tax - 0.48 lstat
                + 4.3 rm - 0.64 ptratio


Evaluation on training data (404 cases):

    Average  |error|               1.94
    Relative |error|               0.29
    Correlation coefficient        0.96
```

```
Attribute usage:
  Conds  Model

   75%    98%    lstat
   45%    61%    nox
   28%    70%    rm
   16%    86%    dis
   14%    90%    ptratio
   13%    76%    crim
    5%    48%    tax
    4%    58%    age
          49%    b
          38%    rad
          11%    indus
           4%    zn


Time: 0.1 secs
```

For this model:

```
> cmPred <- predict(committeeModel, testPredictors)
> ## RMSE
> sqrt(mean((cmPred - testOutcome)^2))

[1] 2.863727

> ## R^2
> cor(cmPred, testOutcome)^2

[1] 0.8967124
```

4 Instance–Based Corrections

Another innovation in Cubist is the use of a nearest–neighbor to adjust the predictions from the rule–based model. First, a model tree (with or without committees) is created. Once a sample is predicted by this model, Cubist can find it's nearest neighbors and determine the average of these training set points. See Quinlan (1993a) for the details of the adjustment.

The development of rules and committees is independent of the choice of using instances. The original C code allowed the program to choose whether to use instances, not use them or let the program decide. Our approach is to build a model with the `cubist` function that is ignorant to the decision about instances. When samples are predicted, the argument `neighbors` can be used to adjust the rule–based model predictions (or not).

We can add instances to the previously fit committee model:

```
> instancePred <- predict(committeeModel, testPredictors, neighbors = 5)
> ## RMSE
> sqrt(mean((instancePred - testOutcome)^2))

[1] 2.685944

> ## R^2
> cor(instancePred, testOutcome)^2

[1] 0.9125262
```

Note that the previous models used the implicit default of `neighbors = 0` for their predictions.

5 Exporting the Model

As previously mentioned, this code is a port of the command–line C code. To run the C code, the training set data must be converted to a specific file format as detailed on the RuleQuest website. Two files are created. The `file.data` file is a header–less, comma delimited version of the data (the `file` part is a name given by the user). The `file.names` file provides infomraiton about the columns (eg. levels for categorical data and so on). After running the C program, another text file called `file.models`, which contains the information needed for prediction.

Once a model has been built with the R cubist package, the `exportCubistFiles` can be used to create the `.data`, `.names` and `.model` files so that the same model can be run at the command–line.

6 Current Limitations

There are a few features in the C code that are not yet operational in the R package:

- variable usage/importance haven't been ported into R objects
- only continuous and categorical predictors can be used (the C allows for other data types)
- there is an option to let the C code decide on using instances or not. The choice is more explicit in this package
- non–standard name are not currently checked
- the C code supports binning of predictors

Many of these features will be implemented in the future.

7 About the Cubist C Code and Our Approach

This section may be interesting or important to those of you who care about the implementation (if you exist at all).

The cubist sources are written to take specific data files from the file system, pull them into memory, run the computations, then write the results to a text file that is also saved to the file system. The code makes use of a lot of global variables (especially for the data). The code has been around for a while and, after reading it, one can tell that the author put in a lot of time to catch many special cases. At Pfizer, we have pushed millions of samples through the non–GPL code without any substantive errors.

So the approach here is to pass in the training data as strings that mimic the formats that one would use with the command line version and get back the textual representation that would be saved to the `.model` file also as a string. The prediction function would then pass the model text string (and the data text string if instances are used) to the C code for prediction.

We did this for a few reasons. First, this approach would require us to re–write `main()` and touch as little of the original code as possible (otherwise we would have to write a parser for the data and try to get it into the global variable structure with complete fidelity). Second, most modeling functions implicitly assume that the data matrix is all numeric, thus factors are converted to dummy variables etc. Cubist doesn't want categorical data split into dummy variables based on how it does splits. Thus, we would have to pass in the numeric and categorical predictors separately unless we want to get really fancy.

8 Session Information

- R version 2.11.1 (2010-05-31), `x86_64-apple-darwin9.8.0`
- Locale: `en_US/en_US/en_US/C/en_US/en_US`
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, stats4, utils
- Other packages: caret 4.85, cluster 1.12.3, coin 1.0-17, colorspace 1.0-1, Cubist 0.02, lattice 0.18-8, MASS 7.3-6, mlbench 2.1-0, modeltools 0.2-17, mvtnorm 0.9-95, party 0.9-99991, plyr 1.2.1, reshape 0.8.3, reshape2 1.1, RWeka 0.4-4, sandwich 2.2-6, strucchange 1.4-2, survival 2.35-8, vcd 1.2-9, zoo 1.6-4
- Loaded via a namespace (and not attached): rJava 0.8-7, RWekajars 3.7.2-1, stringr 0.4, tools 2.11.1

9 References

Quinlan. Learning with continuous classes. Proceedings of the 5th Australian Joint Conference On Artificial Intelligence (1992) pp. 343-348

Quinlan. Combining instance-based and model-based learning. Proceedings of the Tenth International Conference on Machine Learning (1993a) pp. 236-243

Quinlan. *C4.5: Programs For Machine Learning* (1993b) Morgan Kaufmann Publishers Inc. San Francisco, CA

Wang and Witten. Inducing model trees for continuous classes. Proceedings of the Ninth European Conference on Machine Learning (1997) pp. 128-137

http://rulequest.com/cubist-info.html