Continuous soil attribute modeling and mapping: Cubist Models

Soil Security Laboratory

2018

1 Cubist Models

The Cubist model is currently a very popular model structure used within the DSM community. Its popularity is due to its ability to "mine" non-linear relationships in data, but does not have the issues of finite predictions that occur for other decision and regression tree models. In similar vain to regression trees however, Cubist models also are a data partitioning algorithm. The Cubist model is based on the M5 algorithm of Quinlan (1992), and is implemented in the `R` Cubist package.

The Cubist model first partitions the data into subsets within which their characteristics are similar with respect to the target variable and the covariates. A series of rules (a decision tree structure may also be defined if requested) defines the partitions, and these rules are arranged in a hierarchy. Each rule takes the form:

if [condition is true]

then [regress]

else [apply the next rule]

The condition may be a simple one based on one covariate or, more often, it comprises a number of covariates. If a condition results in being true then the next step is the prediction of the soil property of interest by ordinary least-squares regression from the covariates within that partition. If the condition is not true then the rule defines the next node in the tree, and the sequence of *if, then, else* is repeated. The result is that the regression equations, though general in form, are local to the partitions and their errors smaller than they would otherwise be. More details of the Cubist model can be found in the `Cubist` help files or Quinlan (1992).

Luckily, fitting a Cubist model in `R` is not too difficult — although it will be useful to spend some time playing around with many of the controllable parameters the function has. In the example below we can control the number of potenital rules that could potenitally partition the data (note this limits the number of possible rules, and does not necessarily mean that those number of rules will acutallu be realised i.e. the outcome is internally optimised). We can also limit the extrapolation of the model predictions, which is a useful model

constraint feature. These various control parameters plus others can be adjusted within the `cubistControl` parameter. The committees parameter is specified as an integer of how many committee models (e.g.. boosting iterations) are required. Here we just set it to 1, but naturally it is possible to perform some sort of sensitivity analysis when this committee model option is set to greater than 1. In terms of specifying the target variable and covariates, we do not define a formula as we did earlier fo the MLR model. Rather we specify the columns explicitly — those that are the target variable (`x`), and those that are the covariates (`y`).

First get the data and perform the covariate data intersection

```
library(ithir)
library(raster)
library(rgdal)
library(sp)

# point data
data(HV_subsoilpH)

# Start afresh round pH data to 2 decimal places
HV_subsoilpH$pH60_100cm <- round(HV_subsoilpH$pH60_100cm, 2)

# remove already intersected data
HV_subsoilpH <- HV_subsoilpH[, 1:3]

# add an id column
HV_subsoilpH$id <- seq(1, nrow(HV_subsoilpH), by = 1)

# re-arrange order of columns
HV_subsoilpH <- HV_subsoilpH[, c(4, 1, 2, 3)]

# Change names of coordinate columns
names(HV_subsoilpH)[2:3] <- c("x", "y")


# grids (covariate raster)
data(hunterCovariates_sub)
```

Perform the covariate intersection.

```
coordinates(HV_subsoilpH) <- ~x + y

# extract
DSM_data <- extract(hunterCovariates_sub, HV_subsoilpH, sp = 1, method = "simple")
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
```

```
## 'data.frame': 506 obs. of  15 variables:
##  $ id                      : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ x                       : num  340386 340345 340559 340483 340734 ...
##  $ y                       : num  6368690 6368491 6369168 6368740 6368964 ...
```

```
##  $ pH60_100cm              : num  4.47 5.42 6.26 8.03 8.86 7.28 4.95 5.61 5.39 3.44 ...
##  $ Terrain_Ruggedness_Index: num  1.34 1.42 1.64 1.04 1.27 ...
##  $ AACN                    : num  1.619 0.281 2.301 1.74 3.114 ...
##  $ Landsat_Band1           : num  57 47 59 52 62 53 47 52 53 63 ...
##  $ Elevation               : num  103.1 103.7 99.9 101.9 99.8 ...
##  $ Hillshading             : num  1.849 1.428 0.934 1.517 1.652 ...
##  $ Light_insolation        : num  1689 1701 1722 1688 1735 ...
##  $ Mid_Slope_Positon       : num  0.876 0.914 0.844 0.848 0.833 ...
##  $ MRVBF                   : num  3.85 3.31 3.66 3.92 3.89 ...
##  $ NDVI                    : num  -0.143 -0.386 -0.197 -0.14 -0.15 ...
##  $ TWI                     : num  17.5 18.2 18.8 18 17.8 ...
##  $ Slope                   : num  1.79 1.42 1.01 1.49 1.83 ...
```

Often it is handy to check to see whether there are missing values both in the target variable and of the covariates. It is possible that a point location does not fit within the extent of the available covariates. In these cases the data should be excluded. A quick way to assess whether there are missing or NA values in the data is to use the `complete.cases` function.

```
which(!complete.cases(DSM_data))
```

```
## integer(0)
```

```
DSM_data <- DSM_data[complete.cases(DSM_data), ]
```

Now lets begin the Cubist model fitting

```
library(Cubist)
set.seed(875)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
mDat <- DSM_data[training, ]

# fit the model
hv.cub.Exp <- cubist(x = mDat[, c("AACN", "Landsat_Band1", "Elevation", "Hillshading",
    "Mid_Slope_Positon", "MRVBF", "NDVI", "TWI")], y = mDat$pH60_100cm,
    cubistControl(rules = 100,
    extrapolation = 15), committees = 1)
```

The output generated from fitting a Cubist model can be retrieved using the `summary` function. This provides information about the conditions for each rule, the regression models for each rule, and information about the diagnostics of the model fit, plus the frequency of which the covariates were used as conditions and/or within a model.

```
summary(hv.cub.Exp)
```

```
##
## Call:
## cubist.default(x = mDat[, c("AACN", "Landsat_Band1",
##  "NDVI", "TWI")], y = mDat$pH60_100cm, committees = 1, control
##  = cubistControl(rules = 100, extrapolation = 15))
##
##
```

```
## Cubist [Release 2.07 GPL Edition]  Wed Dec 20 11:21:24 2017
## ---------------------------------
##
##     Target attribute `outcome'
##
## Read 354 cases (9 attributes) from undefined.data
##
## Model:
##
##   Rule 1: [148 cases, mean 6.175, range 3.92 to 9.74, est err 0.798]
##
##     if
##  Elevation > 99.47475
##  MRVBF > 0.004846
##  NDVI <= -0.192982
##     then
##  outcome = 4.895 + 0.56 MRVBF + 0.0224 AACN + 0.093 Hillshading
##            + 0.92 Mid_Slope_Positon + 1.5 NDVI - 0.014 Landsat_Band1
##            + 0.024 TWI - 0.0025 Elevation
##
##   Rule 2: [148 cases, mean 6.500, range 3 to 9.41, est err 1.122]
##
##     if
##  Elevation > 99.47475
##  MRVBF > 0.004846
##  NDVI > -0.192982
##     then
##  outcome = 7.116 + 0.0398 AACN + 0.18 TWI - 0.074 Landsat_Band1
##            + 0.12 MRVBF + 1.6 NDVI + 0.025 Hillshading - 0.0028 Elevation
##            + 0.25 Mid_Slope_Positon
##
##   Rule 3: [37 cases, mean 7.042, range 5 to 9.25, est err 0.893]
##
##     if
##  Elevation <= 99.47475
##     then
##  outcome = 1.488 + 0.0582 Elevation + 0.0081 AACN + 0.07 MRVBF + 1.1 NDVI
##            + 0.023 TWI + 0.017 Hillshading - 0.009 Landsat_Band1
##            + 0.17 Mid_Slope_Positon
##
##   Rule 4: [21 cases, mean 7.878, range 6.26 to 9.45, est err 0.654]
##
##     if
##  MRVBF <= 0.004846
##     then
##  outcome = 8.122 + 422.73 MRVBF - 0.0118 AACN + 0.3 NDVI
##            - 0.0007 Elevation + 0.005 TWI + 0.004 Hillshading
##            - 0.002 Landsat_Band1
##
```

```
##
## Evaluation on training data (354 cases):
##
##     Average  |error|              1.128
##     Relative |error|               0.99
##     Correlation coefficient        0.35
##
##
##  Attribute usage:
##    Conds  Model
##
##     94%   100%    Elevation
##     90%   100%    MRVBF
##     84%   100%    NDVI
##           100%    AACN
##           100%    Landsat_Band1
##           100%    Hillshading
##           100%    TWI
##            94%    Mid_Slope_Positon
##
##
## Time: 0.0 secs
```

It appears the `hv.cub.Exp` model contains 4 rules in this case. The useful feature of the Cubist model is that it does not unnecessarily overfit and partition the data, although you do need to keep an eye on the number of observations within each ruleset. Lets see how well the model validates.

```
# Internal validation
Cubist.pred.C <- predict(hv.cub.Exp, newdata = DSM_data[training, ])
goof(observed = DSM_data$pH60_100cm[training], predicted = Cubist.pred.C)
```

```
##          R2 concordance      MSE     RMSE       bias
## 1 0.3227766   0.4884487 1.305012 1.142371 -0.1572845
```

```
# External validation
Cubist.pred.V <- predict(hv.cub.Exp, newdata = DSM_data[-training, ])
goof(observed = DSM_data$pH60_100cm[-training], predicted = Cubist.pred.V)
```

```
##          R2 concordance      MSE     RMSE       bias
## 1 0.2995264    0.460038 1.137435 1.066506 -0.1578419
```

The calibration model validates quite well, but its performance against the validation is comparable. From Figure 1 it appears a few observations are underpredicted (negative bias), otherwise the Cubist model performs reasonably well.

Creating the map resulting from the `hv.cub.Exp` model can be implemented as before using the raster `predict` function (Figure 2).

```
map.cubist.r1 <- predict(hunterCovariates_sub, hv.cub.Exp, "soilpH_60_100_cubist.tif",
    format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
```

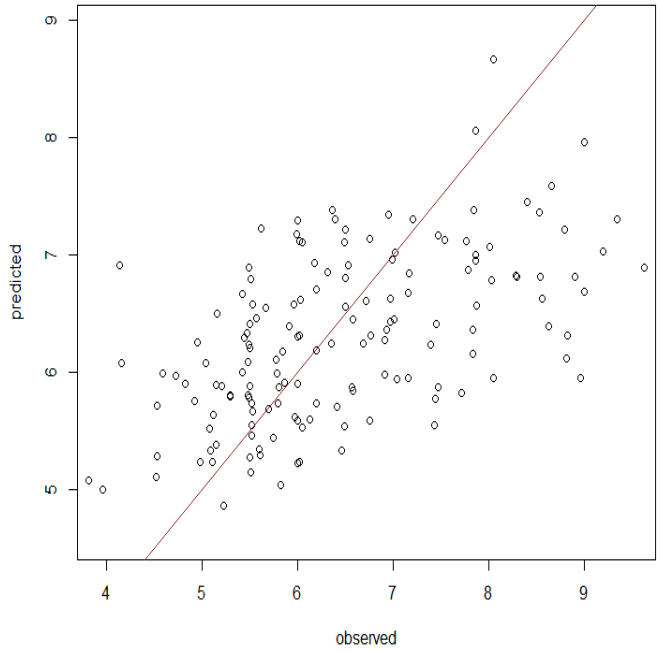

Figure 1: Cubist model xy-plot plot of predicted soil pH (60-100cm) (validation data set).

```
plot(map.cubist.r1, main = "Cubist model predicted Hunter Valley soil pH (60-100cm)")
```

References

Quinlan, J. R.
1992. *Proceedings of AI92, 5th Australian Conference on Artificial Intelligence, Singapore*, chapter Learning with continuous classes, Pp. 343–348. World Scientific.

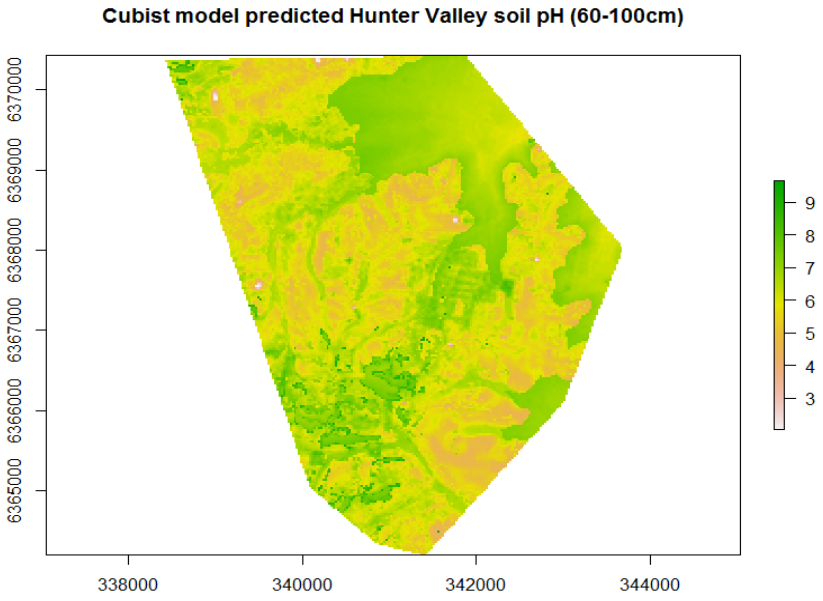

Figure 2: Cubist model predicted Hunter Valley soil pH (60-100cm.