Quinlan's Cubist and "Divide and Conquer" Strategies

Stephen Vardeman

Analytics Iowa LLC

ISU Statistics and IMSE

Quinlan, cubist and C5.0

There is a line of algorithms associated with Ross Quinlan (including "Cubist" for SEL prediction and "C5.0" for classification). His company web site is `https://www.rulequest.com/index.html`. His algorithms are very complicated, and complete descriptions do not seem to be publicly available. (Though there are open source versions of some of his algorithms, much of his work seems to be proprietary and commercial versions of his software are no doubt more reliable than the open source versions.) Textbook descriptions of his methods are generally vague. Probably the best ones I know of are in the KJ book.

Basic notions

The basic notion of Cubist seems to be partition an input space, $\Re^p$, into rectangles and fit a (different) linear predictor for y in each rectangle. Consider the rectangle

$$R = \{\mathbf{x} |\ a_1 < x_1 < b_1, a_2 < x_2 < b_2, \ldots, a_p < x_p < b_p\}$$

where a_j and b_j can be finite or infinite. Where at least one of a_j or b_j is finite, a split has been made on coordinate j. Jargon typically used in describing these methods is that if one lists only rectangles where one or both of the a_j or b_j are finite, one has specified a "rule."

There are many implementation choices (the consequences of which are not transparent) that (much as with MARS) amount to a kind of "special sauce" owned by Quinlan and/or others who have followed him. Vague expositions of Cubist leave most users to treat SEL prediction based on it as a mysterious (albeit often effective) "black box."

Some things that can be said

Here are a few observations based on available information on Cubist for SEL prediction.

1. Trees of **regressions** (*not* trees constant inside each final rectangle) are at the heart of the methodology, both generating rectangles and making predictions. The "error" used to guide node splitting seems to be

$$\overline{\text{err}} \equiv \sum_l \left(\frac{N_l}{N}\right) \sqrt{MSE_l}$$

   where l indexes rectangles, N_l is the number of cases with $\mathbf{x}_i \in R_l$ and MSE_l is presumably from an OLS fit (of some linear model) in R_l.

2. Exactly what inputs x_j are used in each rectangle and how they are chosen is not clear. `Output` for an `R` implementation of Cubist lists different sets for the various rectangles.

Some more things

3. Exactly how to go from tree-building to the final set of rules/rectangles is not clear. Software seems to not allow control of this. Perhaps there is some kind of combining of final rectangles from a tree.
4. Some sort of "smoothing" is involved. This seems to be some averaging of regressions for bigger (containing) rectangles "up the tree branch" from a final rectangle. What this should mean is not absolutely clear if all one has is a set of "rules," particularly if there are cuts less extreme than a final pair defining R_l that have been eliminated from description of R_l. For example, "$3 < x_1 < 5$" is the same as "$3 < x_1 < 10$ *and* $3 < x_1 < 5$". Further, the form of weights used in the averaging seems completely ad hoc.

Cubist options

There are two serious modifications of the basic "tree of regressions" notion included in the R implementation of Cubist.

1. One may employ "**committees**." This seems to be **boosting** or something like it applied using the basic algorithm to create the corrections to successive versions of an approximate $\mathrm{E}[y|\mathbf{x}]$.
2. One may employ "**instances**." This seems to be (optionally) applied after the boosting. It is shrinkage of $\hat{y}$s in light of y_is for k-nearest neighbors, using weights depending upon the distances from $\mathbf{x}$ to the neighbors. For k cases closest to $\mathbf{x}$, say cases $i_1, i_2, \ldots, i_k$ and corresponding weights $w_1, w_2, \ldots, w_k$ (summing to 1?) the prediction used for input $\mathbf{x}$ seems from KJ to be

$$\hat{y}(\mathbf{x}) + \sum w_l \left(y_{i_l} - \hat{y}_{i_l}\right)$$

Divide and conquer

The "tree of regressions" basis of Cubist is motivated by the reality that in big problems, simple predictor forms can make sense only "locally." A sensible strategy in the face of that reality is to attempt to carve up an input space into pieces within each of which relatively simple prediction methods might be used. Cubist is built on using tress (based on regressions) to do the carving and linear predictors in the pieces. One might term it one of many possible "divide and conquer" prediction strategies.