

Generation of Boolean Classification Rules

Roger J. Marshall¹

¹ Department of Community Health, University of Auckland, New Zealand

Abstract. An algorithm to generate a class of Boolean classification rules is described. The algorithm is implemented in search partition analysis software (SPAN), a program designed to find an optimal binary data partition. Some comments on the relationship of the procedure with tree-based search procedures are discussed.

Keywords. Boolean classifiers, trees, decision rules

1 Introduction

Classification and regression trees are now well known statistical tools for data exploration and to develop classification rules. Other non-hierarchical search partitioning methods are less well known. One, search partition analysis (SPAN), is a procedure that I have proposed (Marshall, 1986,1995,1999). It has been applied primarily in medical problems. It is based on an algorithm to generate a binary partition from a class of Boolean classifiers in disjunctive normal form:

$$A = I_1 \cup I_2 \cup \dots \cup I_q \quad (1)$$

where I_i is the conjunction of p_i attributes from a pool of m potential predictive attributes. I will refer to q as the *order* of the expression (1).

For example, an optimal classification rule with $q = 3$ and $p_1 = p_2 = p_3 = 2$ for predicting death within 28 days for a person experiencing a stroke is (Broad et al, 2000)

$$A = (C \cap I) \cup (C \cap S) \cup (D \cap I) \quad (2)$$

where C , S , I and D are respectively the *attributes*: loss of consciousness, institutionalised pre-stroke, severe motor deficit and requires help dressing. This rule was found by generating the possible Boolean expressions of the form (1) that can be generated from a pool of, in this case, $m = 12$ of the best predictors of death.

Although the methodology has been published (Marshall, 1986,1995,1999), the algorithm to enumerate the class of possible expressions of the form (1) has not. It can be presented as a combinatoric problem: the enumeration of the ways of selecting q groups of objects, with replacement, from a pool of m objects such that there are p_i objects in group i . The pool of m objects is the set of attributes, say T_m , that are to be combined in the rule. In order that the expression does include redundant I_i 's, it is necessary to ensure that, for $p_j \leq p_i$, all the objects of I_j are not the same objects as those in I_i , for if they are $I_i \cup I_j = I_i$ and the order of the expression is no longer q .

2 The lock and key enumeration algorithm

Suppose the m objects of T_m are labelled $1, 2, \dots, m$. Let I_i denote the set of objects in group i . If, for the moment, the requirement that *all* objects in one group are not contained in another is ignored, that is, "embedding" $I_j \subseteq I_i$ is allowed, there are $\binom{m}{p_i}$ possibilities for each I_i . We identify each possibility

by a key K_i , where $K_i = 1, \dots, \binom{m}{p_i}$ and the objects of the q groups are determined by a set keys K_1, \dots, K_q . By generating a set of keys, the actual objects in each group can be obtained by "unlocking" the combination of each key to obtain I_i .

2.1 Unlocking a key

We discuss generating the keys in a moment. First, consider how to unlock a key K to obtain the combination $I = (i_1, \dots, i_p)$ of a group of p objects from m . There are $\binom{m}{p}$ combinations and suppose the keys correspond to an ordering in which i_p is incremented the fastest and i_1 the slowest. Thus $I = (1, 2, \dots, p)$ corresponds to $K = 1$, and $I = (m-p+1, m-p+2, \dots, m)$ corresponds to $K = \binom{m}{p}$ and $i_1 < i_2 < \dots < i_p$. Given i_1 there are $L_{i_1} = \binom{m-i_1}{p-1}$ possibilities for the remaining $p-1$ objects. Thus, if we define

$$S_i = \sum_{j=1}^i L_j$$

with $S_0 = 0$ and find an i such that $S_{i-1} < K \leq S_i$ then the first digit is $i_1 = i$. The remaining combination, i_2, \dots, i_p is now a permutation of the digits $i+1, \dots, m$ and $K - S_{i-1}$ is the key to unlock the remaining combination. Thus with K , m and p reset to $K - S_{i-1}$, $m - i$, and $p - 1$ respectively we proceed in the same way to find the second digit and so on.

2.2 Generating keys

To generate the key combinations K_1, \dots, K_q consider first the case when $p_1 = \dots = p_q = p$. There are $c = \binom{m}{p}$ possible values for each K_i and, to avoid embedding, these have to be distinct. Therefore the problem is that of enumerating the $\binom{c}{q}$ ways to select q objects from c , each enumeration satisfying $K_1 < K_2 < \dots < K_q$. When the p_i 's are not all equal, the keys do not necessarily have to satisfy the full inequality $K_1 < K_2 < \dots < K_q$ unless $p_{i-1} = p_i$ for some i and it is then necessary to ensure that $K_{i-1} < K_i$. Subject to these constraints the sequence of keys can be generated by the simulated nested loop device of Gentleman (1975). Except in the special case $p_1 = \dots = p_q = p$ the algorithm will generate some I_j that are embedded, and in this case it is necessary to check for embedding.

3 Size of the search

Let $N_{q,m}$ denote the number of expressions of the form of (1), that is, $N_{q,m}$ is the number of ways of selecting q groups of objects from a pool of m objects, with no embedding.

Consider first, the special case $p_1 = p_2 = \dots = p_q = p$. Then, there are $c = \binom{m}{p}$ possible groups and $N_{q,m} = \binom{c}{q}$ ways of picking q of them.

When the p_i 's are not all equal then there is no simple general formula for $N_{q,m}$. Consider first $q = 2$ and suppose $p_1 \geq p_2$. Let $c_i = \binom{m}{p_i}$ for $i = 1, 2$ and $c_{12} = \binom{p_1}{p_2}$. There are c_1 to select elements of I_1 and c_2 ways for I_2 . However the latter must exclude the c_{12} ways in which I_2 may be embedded in I_1 . Hence

$$N_{2,q} = c_1(c_2 - c_{12})/e!$$

where e is the number of times a p_i is repeated, that is, either 1 or 2. Note that $e = 2$ when $p_1 = p_2$ and the formula coincides with that given by $N_{q,m}$ above.

For the case $q = 3$ define, as above, $c_{jk} = \binom{p_j}{p_k}$ and, additionally, $c'_{jk} = \binom{m - p_k}{p_j - p_k}$, then it can be shown by extending the inclusion/exclusion argument above that

$$N_{q,3} = (c_1c_2c_3 - c_1c_3c_{12} - c_1c_2c_{13} - c_1c_2c_{23} + c_1c_{12}c_{13} + c_3c'_{13}c'_{23})/e!$$

where e is the number of times a p_i is repeated, either 1,2 or 3. Table 1 gives $N_{q,m}$ for some values of m and p_1, \dots, p_q .

I have been unable to derive a general formula for $N_{q,m}$ when $q \geq 4$ and when the p_i 's are unequal.

Table 1. $N_{q,m}$, the number of ways of selecting q groups of objects from m .

| p_1, \dots, p_q | 8 | 12 | 16 | 20 |
|-------------------|------|-------|--------|---------|
| 1 | 8 | 12 | 16 | 20 |
| 2 | 28 | 66 | 120 | 190 |
| 3 | 56 | 220 | 560 | 1140 |
| 1,1 | 28 | 66 | 120 | 190 |
| 2,1 | 168 | 660 | 1680 | 3420 |
| 3,1 | 280 | 1980 | 7280 | 19380 |
| 2,2 | 378 | 2145 | 7140 | 17955 |
| 3,2 | 1400 | 13860 | 65520 | 213180 |
| 3,3 | 1540 | 24090 | 156520 | 649230 |
| 1,1,1 | 56 | 220 | 560 | 1140 |
| 2,1,1 | 440 | 2970 | 10840 | 29070 |
| 2,2,1 | 1680 | 17820 | 87360 | 290700 |
| 2,2,2 | 3276 | 45760 | 280840 | 1125180 |

4 Practical aspects

In practice, to generate decision rules of the form (1), one would try different values of p_1, \dots, p_q . Specifically, combinations in which $q \leq Q$ and $p_i \leq P_i$. Obviously from Table 1, to make the search manageable P_1, \dots, P_Q need to be small, say 2, 2.

Also, one can consider interchanging \cup and \cap in (1), so admitting combinations in which the complement of the decision rule is in normal disjunctive form and written in terms of the complements of the elements of T_m . Once the lock and key combination is generated, two partitions of the data ensue.

4.1 Expanding a search recursively

A recursive procedure to allow more complex rules to be generated, but without doing an exhaustive search, was suggested in Marshall(1995). In this procedure, an initial search is done with fixed P_1, \dots, P_Q and the best rule determined. The rule is then used to create an attribute which is added to the attribute set for a subsequent search.

In terms of the analogy used in the lock and key algorithm, this procedure is tantamount to adding an extra object, the attribute generated on the first search, to the pool, T_m , of m objects and again generating q groups of objects. The number of ways of doing this, now $N_{q,m+1}$, will include the $N_{q,m}$ that are already generated plus new combinations that have the new object in at least one of the groups. Only the latter are of interest and there are $N_{q,m+1} - N_{q,m}$ of them. I have been unable to devise an algorithm which generates just these, however, it is easy and quick to generate the $N_{q,m+1}$ combinations and just immediately discard those which do not contain the new attribute.

4.2 Complexity penalising

In the recursive approach, the expression that are generated may become quite complex, in the sense that q and p_i may become large. Suppose a criterion, G , is used to choose the "best" classification rule. For instance reduction in mean square error. In practice G may reach a plateau as expression complexity increases. A mechanism for offsetting G against increasing complexity is desirable, to choose simple and effective decision rules.

There are different ways to assess the complexity of a Boolean expression. One is to choose the number of distinct elements that comprise it, for example, four in (2). Another, which takes account of the complexity of the complement of A in (1) is $c = q + q' - 1$ where q' is the order of \bar{A} when it is expressed in disjunctive normal form. For example, the complement of (2) is

$$\bar{A} = (\bar{C} \cap \bar{D}) \cup (\bar{C} \cap \bar{I}) \cup (\bar{S} \cap \bar{I})$$

so that the complexity is $c = 5$.

Since A and \bar{A} are two sides of the same coin, this definition of complexity seems most reasonable.

Complexity penalising can be achieved formally by using $G' = G - \beta c$ to select a decision rule, where β is a penalty parameter which can be arbitrarily imposed. In practice, however, the upper envelope of a plot of G versus c provides an effective means to select an optimal rule. Typically there are initial rapid gains in G with increasing c , but these fall away at some point that may provide an optimal rule.

5 Contents of the attribute set: regular rules

In Marshall(1986,1999) it was argued that, to generate rational classifiers, rules of the form of (1) should be in terms of attributes that are "positive" with respect to the outcome variable. In effect this requires that T_m consists of a pool of positive attributes. For example, to establish the decision rule (2), T_m was a set of 12 attributes which included the four adverse features U, C, S, I that are in rule (2) plus eight other adverse pre-stroke attributes: unable to dress, unable to walk, aphasia, incontinence and so on. I have termed this kind of rule *regular*.

Where it is anticipated that interactions in which an adverse factor may become beneficial in combination with other factor, it is possible to include *both* an attribute and its complement in T_m . For example, if in the stroke example, it was felt that *not* being institutionalised might also, in some circumstances, present a risk on stroke outcome, both I and \bar{I} could be included in T_m . The benefits of this, in the author's experience are dubious. Typically one ends up with a regular rule; the inclusion of complements only serve to make the search unnecessarily long.

6 Comparison to trees

One of the features of tree-based classification rules is that, typically, the rules that are generated are not regular. That is, when splits are made in terms of the presence or absence of an attribute, the routes to terminal nodes are generally combinations of some attributes that are present and some that are absent. Interpretation of the meaning of these combinations can often be difficult (Marshall, 2000).

However, in the process of amalgamating terminal nodes to form a decision rule, it is possible that the rule so formed, after Boolean simplification, *may* be regular. For example, Herman et al(1995) created a tree for classifying patients at high diabetes risk. There are four high risk terminal nodes which, when unioned, give the rule

$$(EO)(\bar{E}OS)(E\bar{O}H)(E\bar{O}\bar{H}G) \quad (3)$$

where O, E, H, S and G are respectively obesity, elderly (over 65), hypertension, sedentary and impaired glucose tolerance. Expression (3) reduces to simply

$$(EO)(OS)(EH)(EG)$$

which is regular. Also the reduced rule emphasises that, for example, being obese and sedentary is sufficient to satisfy (3), it is not necessary to be age under 65 (as the element $\bar{E}OS$ seems to imply). That is, some of the combinations are misleading and redundant.

On the other hand, because of the so-called replication problem (Pagallo and Haussler, 1990) it generally requires a complex tree to represent a relatively simple regular rule. For example, to represent (2) by a tree requires a tree with 11 terminal nodes.

These considerations, and others (Marshall, 2000) suggest that the rules derived by SPAN are often more interpretable, at least in the medical domain, than tree based rules. However, one could consider a hybrid of the two approaches by allowing splits of a tree based on combinations of the form (1). The SPAN software (mentioned below) allows this to be done. The benefits of this kind of approach, if any, over the direct non-hierarchical approach remain to be demonstrated.

7 Software

The algorithm describe is implemented in SPAN software, which can be obtained from URL <http://www.auckland.ac.nz/mch/span.htm>. SPAN is a Windows 95, 98 or NT program, which, as well as implementing the methods described above, also allows tree based analysis, with node splits that can be of the canonical form (1).

Generally, the time required to generate Boolean expressions is negligible compared to the time spent applying the expression to the data. SPAN has an built in algorithm to collapse data into grouped form which gives substantial gains in performance. For example, in a data set of 50,000 observations and with $m = 12$ and a binary outcome variable, the data collapse to only about 500 distinct data combinations, making the search very fast.

References

- Broad, J., Marshall, R.J., Bonita, R. (2000) The prediction of death soon after a stroke: A case study comparing four statistically derived prediction rules. *Journal of Clinical Epidemiology* (submitted).
- Gentleman, J.F. (1975) Algorithm AS 88. Generation of all nC_r combinations by simulating nested Fortran DO loops. *Applied Statistics*, **24**, 374-376.
- Herman, W.H., Engelgau, M.M., Smith, P.J., Aubert, R.E., Thompson, T.J. (1995) A new and simple questionnaire to identify people at increased risk for undiagnosed diabetes. *Diabetes Care*, **18**,382-387.
- Marshall, R.J. (1986) Partitioning methods for classification and decision making in medicine. *Statistics in Medicine*, **5**, 517- 526.
- Marshall, R.J. (1995) A program to implement a search method for identification of clinical subgroups. *Statistics in Medicine*, **14**, 2645-2660.
- Marshall, R.J. (1999) Classification to ordinal categories using a search partition methodology with an application in diabetes screening. *Statistics in Medicine*, **20**, 2723-2736.
- Marshall, R.J. (2000) The use of classification and regression trees in clinical epidemiology. *Journal of Clinical Epidemiology* (in press)
- Pagallo, G., Haussler, D. (1990) Boolean feature discovery and empirical learning. *Machine Learning*, **5**, 71-99.