Rule induction: Ross Quinlan's ID3 algorithm

You are stranded on a desert island, with only a pile of records and a book of poetry, and so have no way to determine which of the many types of fruit available are safe to eat. They are of various colours and sizes, some have hairy skins and others are smooth, some have hard flesh and others are soft. After a great deal of stomach ache, you compile the table of data shown in table 2. This is pretty tedious to look up each time you feel hungry; is there some kind of pattern to it?

Table 2: Identifying what's good to eat

Conclusion	*Skin*	*Colour*	*Size*	*Flesh*
safe	hairy	brown	large	hard
safe	hairy	green	large	hard
dangerous	smooth	red	large	soft
safe	hairy	green	large	soft
safe	hairy	red	small	hard
safe	smooth	red	small	hard
safe	smooth	brown	small	hard
dangerous	hairy	green	small	soft
dangerous	smooth	green	small	hard
safe	hairy	red	large	hard
safe	smooth	brown	large	soft
dangerous	smooth	green	small	soft
safe	hairy	red	small	soft
dangerous	smooth	red	large	hard
safe	smooth	red	small	hard
dangerous	hairy	green	small	hard

This can be viewed as a set of examples, where each line states the values of certain attributes and a conclusion. On the other hand, it can be viewed as a set of rules: the first line is

```
if   skin = hairy
and  colour = brown
and  size = large
and  flesh = hard
then conclusion = safe
```

Such rules are not constrained to have a fixed number of preconditions, of course. Suppose that there was a

rule which had no precondition about the size of the fruit. Then it could be replaced by two rules (there are two possible values for `size`), that is, one in which `size = large` and one in which `size = small` were extra preconditions, or it could be replaced by a rule with the single equivalent extra precondition `size = Anything` or the precondition `(size = large or size = small)`.

It is possible to devise a *decision tree* to replace this set of rules, automatically. The decision tree is of the general form

```
if      attribute1 = value1 then <subtree 1>
else if attribute1 = value2 then <subtree 2>
else if ...
.........
else if attribute1 = valueN then <subtree N>
```

This corresponds directly to a set of rules, with as many rules as there are leaf nodes in the whole tree. Each rule is a tracing out of the path from the top of the tree to a leaf node. The process is not limited to preconditions of the form `This = That`, either, although for the purpose of explaining the idea only this kind of precondition will be considered.

The key question is which of the attributes is the most useful determiner of the conclusion of the rules. Whichever it is, that attribute ought to be the first or topmost one in the decision tree. Information theory provides one answer, as well as providing a meaning for the notion of `most useful determiner'. To formalise it, suppose that the conclusion $C$ can have any of the values $c_1 \cdots c_n$ (in the example above, $n = 2$ and the conclusions are `safe' or `dangerous'). Suppose that a certain attribute $A$ can take values $a_1 \cdots a_m$. The quantity $p(c_i|a_j)$ means `the probability that c_i, given a_j - that is, among the cases for which a_j is true, this is the probability that c_i is also true. For example, in figure 2:

$$p(\text{Conclusion=safe}|\text{Skin=hairy}) = 6/8 (= 3/4)$$

because there are eight lines with `Skin=hairy` and among those, six of them say `Conclusion=safe`.

Then the expression

$$-\log_2 p(c_i|a_j)$$

is deemed to be a measure of the `amount of information' that a_j has to offer about conclusion c_i. Information theory defines

$$\text{entropy} = -\sum_{i=1}^{n} p(c_i|a_j) \log_2 p(c_i|a_j)$$

to be the *entropy* of a_j with respect to the conclusion C. Entropy is a measure of `degree of doubt' - the higher it is, the more doubt there is about the possible conclusions.

This definition of entropy is not arbitrary. Suppose there is a set of possible events, but all that is known about them is their probabilities $p_1 \cdots p_k$. Any measure of the uncertainty about which event will occur should be

some function $H(p_1, \ldots, p_k)$ of the probabilities, and should satisfy the following very reasonable conditions:

1. it should be a continuous function of each argument
2. if the probabilities are all equal ($= 1/k$) then entropy should increase as k does. This is because, with equally likely events, when there are more events there is more uncertainty about the outcome.
3. If the `choice' between events is split into a set of successive `choices' then the entropy of the original set should be the same as the weighted sum of the entropies at each step of the successive `choices'. That is, the entropy should not depend on the `choice' method. For example, this means that if the events are concerned with throwing a die, and are specifically ``three or less", ``four or five" and ``six", then this could be cast as a two step process: ``is it three or less, or not?", then ``if high, is it six or not?". So, in mathematical terms, this independence means that

$$H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H(\frac{2}{3}, \frac{1}{3})$$

It can be shown (fairly easily) that the entropy function above is the only one which satisfies these conditions. If you are interested, see any good book on information theory, or consult Claude Shannon's seminal papers on the subject in the *Bell System Technical Journal* in 1948, or Shannon and Weaver's book *The Mathematical Theory of Communication*, which are very clearly written.

To return to the original problem now, the lower the entropy of a_j with respect to C is, the more information that a_j has to offer about C. Using this expression, the entropy of the attribute A with respect to C is

$$-\sum_{j=1}^{m} p(a_j) \sum_{i=1}^{n} p(c_i|a_j) \log_2 p(c_i|a_j)$$

and the attribute which has the lowest entropy is the most useful determiner.

Consider the attribute *Size* in the above example. From the table:

$$\begin{aligned} p(safe|large) &= 5/7 \\ p(dangerous|large) &= 2/7 \\ p(large) &= 7/16 \\ p(safe|small) &= 5/9 \\ p(dangerous|small) &= 4/9 \\ p(small) &= 9/16 \end{aligned}$$

Thus the entropy for *Size* as a determiner of *Conclusion* is, using logarithms to base 2:

$$7/16(5/7\log(5/7) + 2/7\log(2/7)) + 9/16(5/9\log(5/9) + 4/9\log(4/9)) = 0.9350955\ldots$$

However, since it is only necessary to compare entropies it makes no difference to use $log_e(\ldots)$ or $log_{10}(\ldots)$ rather than $log_2(\ldots)$. If you do all the calculations you will find that *Colour* has the smallest entropy. The next step is to partition the set of examples according to the possible values for the colour, and apply the same process to each subset. This eventually leads to the rules:

```
if   colour = brown
then conclusion = safe

if   colour = green
and  size = large
then conclusion = safe

if   colour = green
and  size = small
then conclusion = dangerous

if   colour = red
and  skin = hairy
then conclusion = safe

if   colour = red
and  skin = smooth
and  size = large
then conclusion = dangerous

if   colour = red
and  skin = smooth
and  size = small
then conclusion = safe
```

This set of six rules, each smaller than any of the originals, is an improvement on having those sixteen. They are all smaller because, as you might have guessed, the attribute *Flesh* is irrelevant. If you look at the table, you will find some examples of lines in it which differ only in the value of this attribute. From such an observation you can only conclude that the attribute is, under certain combinations of the other attributes, irrelevant; it should be a little surprising to you that it is always so.

To summarise the ID3 algoithm:

1. For each attribute, compute its entropy with respect to the conclusion
2. Select the attribute (say A) with lowest entropy.
3. Divide the data into separate sets so that within a set, A has a fixed value (eg Colour=green in one set, Colour=brown in another, etc).
4. Build a tree with branches:
   ```
   if A=a1 then ... (subtree1)
   if A=a2 then ... (subtree2)
   ...etc...
   ```
5. For each subtree, repeat this process from step 1.
6. At each iteration, one attribute gets removed from consideration. The process stops when there are no attributes left to consider, or when all the data being considered in a subtree have the same value for the conclusion (eg they all say Conclusion=safe).

The six generated rules can be even more succinctly put, however - only small green ones or large smooth red ones are dangerous. Of course, that word `only' conceals the fact that there are only two possible conclusions. Often such a further apparent compression of the rule set will not be easy to find, if it exists at all. One easy method is to try omitting each rule condition in turn, testing to see whether this results in any misclassification of data.

The rule induction algorithm was first used by Hunt in his CLS system in 1962! Then, with extensions for handling numeric data too, it was used by Ross Quinlan for his ID3 system (in 1979). Since then it has been

widely copied, and is the basis of a variety of commercial rule induction packages. ID3 and later packages did not apply the algorithm to the whole data set, since that might be wasteful; instead they would generate a set of rules from a small sampling of the data and then look for other data items which were misclassified by the current rules. A number of such anomalous data items would be added to the initial set and the rule induction algorithm re-run to generate a new set of rules. It has been reported that this iterative approach may not save much effort compared to running the algorithm on the whole set right at the start, unless the whole set is huge.

If you are considering using the algorithm then there are, as ever, points to bear in mind. First, if the table of data about edibility had included a record of which day of the week the sample had been eaten on, the algorithm would have blindly considered this as a factor. So, wholly spurious correlations are possible, since the algorithm takes no account of any meaning that the data it works on may have. Second, the algorithm considers just one attribute at a time. If two attributes turn out to have almost identical entropies it will still select the lower entropy, whereas it might have been better to consider them jointly (as though they made up a single attribute). Consider the following data:

Outcome	*X*	*Y*
yes	3	3
no	2	1
yes	4	4
no	2	4
no	1	3
yes	1	1
yes	2	2
no	2	3

The entropy-based ID3 algorithm is incapable of spotting the obvious summary, that:

```
if X = Y then outcome = yes else outcome = no
```

because it only ever considers one attribute at a time. Third, when inducing rules from large sets of examples in which there are a large number of possible outcomes (that is, n is large), then the algorithm can be very sensitive to apparently trivial changes in the set of examples. Quinlan's ID3 tried to cut down on effort by inducing a set of rules from a small subset of data, and then testing to see if those rules explained other data. Data not explained were then added to the chosen subset, and new rules induced. This process continued until all the data was accounted for. The letters ID stood for `iterative dichotomiser', a fancy name for this simple algorithm. Fourth, the algorithm cannot generate uncertain rules or handle uncertain data.

Although commercial rule induction systems have been used very successfully to condense sets of examples into rules, they must be used with care. You must be sure that the attributes you choose to include in describing the examples (the *Colour*, *Skin* and so on in the example above) must include the `right' ones. If the algorithm is given a complete set of examples then it can usually do a worthwhile job; if it is given only a subset then it is up to you to make sure that the subset is somehow representative of the whole. If a brown-skinned dangerous fruit were to be added to the data table above, the generated rules would change radically. For these reasons, rule induction systems are usually used to suggest correlations that can be confirmed by logical analysis afterward, rather than being trusted to get things right by themselves.

It is also important to remember that the generated rules are for solving the arbitrary classification task. Suppose you provided a set of attributes of motor cars, and used the algorithm to generate rules to classify an unspecified car. The rules might well have, as a first question, `How many wheels does it have?', then `how many cylinders?' and so on, ending up with, for instance, `does it have a statue of the Winged Victory on the bonnet? If yes, it is a Rolls Royce'. However, this last question is sufficient all by itself to determine whether a car is a Rolls Royce or not; if that is the only point at issue, all the earlier questions are unnecessary.

Peter Ross, peter@dcs.napier.ac.uk, x4437

This version: 2000-10-30