A program for numerical classification

D. M. Boulton* and C. S. Wallace*

* *Department of Information Science, Monash University, Melbourne, Australia*

In a previous paper (Wallace & Boulton, 1968) the information measure was derived. It is designed to measure the objective goodness of a nonhierarchical taxanomic classification and can be used to choose the best of a number of different classifications of the one data set. The information measure can also form the basis of a classification algorithm which searches directly for that classification with the best information measure. In the present paper such a classification algorithm is described together with an ALGOL program called Snob incorporating this algorithm.

(Received November 1968)

Introduction

At present there are a number of different methods of numerical classification available to a user, but all will not necessarily produce the same solution to a given problem. It would be desirable to have an objective measure of classification goodness to enable the best of a number of different solutions to be chosen.

In Wallace and Boulton (1968), we presented a view of the aim of classification which allows a measure of goodness to be defined. The measure is called the information measure as it arises from information theory.

We derived the measure for non-hierarchical classifications with any mixture of continuous and unordered multistate attributes, both subject to missing values. The information measure can in principle be extended to other cases.

Rather than use the information measure simply to compare existing classifications, we have incorporated it in a program called Snob, which searches directly for the classification which minimises the information measure. Snob was written in ALGOL for the University of Sydney's KDF9 computer and is described here.

The information measure

Given a set of S things, usually considered as a sample from a large population, and for each a set of D attribute values, a classification (in the taxanomic sense) is a partition of the S things into T classes such that all things within a class can be treated as alike in certain discussions.

Our view of classification is that it is a method of modelling the population density distribution in D-dimensional measurement space (wherein each thing is represented by a point) by the union of T simple class distributions. Each class distribution applies over a region of measurement space and a thing belongs to the class in whose region of application it lies. The complete distribution model is thus composed of segments of each of the T simple class distribution models.

The probability of finding a thing with a given set of attribute values is estimated from the density of the composite model in the thing's vicinity. That is, if the thing lies in class t its probability is estimated from the density of the model for class t weighted by the relative abundance of class t. The class distribution parameters are estimated from the given population.

We now consider the length of a message which can convey the description of all S things as is contained in the $S \times D$ attribute values. The optimum method of encoding this message is to use a Shannon–Fano code (Oliver, 1952) where the length of message required is minus the logarithm of the probability of obtaining the given set of things. This probability is the product of the probabilities of obtaining each of the individual things if we assume the S things to be independent random selections from the population. If natural logarithms are used the message length is measured in ln2 bits (i.e. nits).

We have assumed that within each class different attribute values are uncorrelated. Hence, the probability of obtaining a particular thing, given that it belongs to class t, is simply the product of the marginal probabilities of each of its D attribute values within class t. If an attribute value is missing then its marginal probability is omitted from the product.

For multistate attribute d the class t marginal distribution is assumed to be multinomial with probabilities

$$p[m, d, t] \quad (m = 1, 2, \ldots, M[d]),$$

where $M[d]$ is the number of different states. Continuous attribute values are assumed to be accurate within a range $\epsilon[d]$ so there are only a finite number of different values possible in a finite range. The marginal distribution is assumed to be normal with a mean $\mu[d, t]$ and standard deviation $\sigma[d, t]$. The relative abundance of classes is assumed to be multinomial with probabilities

$$r[t] \quad (t = 1, 2, \ldots, T).$$

All these distribution parameters are estimated from the given population.

The length of the message describing the S things is minimised by using the maximum likelihood estimates for the values of the class distribution parameters. However, unless a receiver of the message has knowledge of these assumed values he will be unable to decode the message. Thus a further message, called the class description message, which specifies the assumed distribution parameter values, must accompany the descriptive message (Boulton and Wallace, 1969).

The information measure is equal to the length of the complete message: class description plus descriptive. We consider that the best classification of a given data set is that which results in the shortest message, that is minimises the information measure.

The information measure is discussed in more detail and the lengths of the two parts of the message are derived in Wallace and Boulton (1968).

The information measure is thus a function of the data $x[d, s]$, measurement accuracies $\epsilon[d]$, the number of classes T, the distribution parameters $r[t]$, $p[m, d, t]$ $\mu[d, t]$ and $\sigma[d, t]$, and the assignment of individual things to classes. Of these, the first two are fixed data and the remainder are at the disposal of the designer of the classification.

Minimisation methods

The program Snob is basically a collection of tactics for modifying a classification so as to decrease its information measure. The measure is a rather poorly behaved function of a large and variable number of variables. We may classify these as follows:

(*a*) The number of classes T.
(*b*) The presumed relative abundance $r[t]$ of each class.
(*c*) For each class:
  (i) for each multistate attribute the presumed probability of each state, $p[m, d, t]$;
  (ii) for each continuous attribute the presumed mean $\mu[d, t]$ and standard deviation $\sigma[d, t]$.
(*d*) For each thing the class to which it is presumed to belong, i.e. the class distribution chosen as the basis for encoding its attribute measurements.

No explicit method has been found for calculating the optima of these variables. However, optima of some variables can be found subject to the others being held constant.

Five different tactics are employed:

(i) Distribution Adjustment: with variables (*a*) and (*d*) held constant, the simultaneous optimisation of variables (*b*) and (*c*). Essentially, this optimisation amounts to a maximum likelihood estimate of variables (*b*) and (*c*). However, the likelihood functions of variables (*b*) and (*c*) are slightly modified by the inclusion of the message lengths needed to encode these variables.
(ii) Reclassifying: with variables (*a*), (*b*) and (*c*) held constant, simultaneous optimisation of variables (*d*).
(iii) Splitting: splitting of a single class into two and at the same time making an optimum choice of the variables (*b*) and (*c*) for the new classes.
(iv) Merging: merging two classes into one and at the same time making an optimum choice of variables (*b*) and (*c*) for the new class.
(v) Swapping: splitting a class into two and adding one of its parts to a second class. This will be explained in greater detail below.

The program Snob employs all five tactics to improve an initial classification and halts when no tactic yields an improvement. Unfortunately, this may occur before the true minimum of the information measure has been found. Further development of the program would require the development of other and more powerful tactics, but we feel it unlikely that any tactic will be found which will guarantee the true minimum.

Distribution adjustments

In this tactic, the number of classes and their membership are held constant. Thus the adjustment of the distribution parameters and class name labels for each class affects only the encoding of members of that class, and each class can be considered separately.

Within a class, the shortest encoding of its members' attribute value sets is obtained with that distribution function which fits the observed measurement points best in the maximum likelihood sense. The message length for the value sets is, in fact, minus the logarithm of the likelihood function. However, we show in Wallace and Boulton (1968) that including the description of the distribution parameters in the message length to be minimised leads to optimum parameter values which differ slightly from the normal maximum likelihood estimates. The length of the parameter descriptions play the role of minus the logarithm of an *a priori* probability.

The optimum estimates and corresponding message lengths are found to be as follows.

1. The optimum value for $r[t]$ is $n[t]/S$, where $n[t]$ is the number of things in class t. The resulting length of the class name label is

$$l[t] = \ln (S/n[t]) \tag{1}$$

The class name label is the message segment included in the description of each thing to indicate the class to which it belongs.

The total length of message for all classes due to the class name labels of the S things and the description of the manner of encoding class name labels is

$$\tfrac{1}{2}(T-1)(\ln (S/12) + 1) - \ln (T-1)! + \sum_{t=1}^{T} (n[t] + \tfrac{1}{2}) \ln (S/n[t]) \tag{2}$$

It is convenient to recast the above as the sum of two terms:

$$H(T) + \sum_{t=1}^{T} L[t] = \{-\ln (T-1)! - \tfrac{1}{2}(\ln (S/12) + 1)\} + \left\{ \sum_{t=1}^{T} (\tfrac{1}{2}(\ln (S/12) + 1) + (n[t] + \tfrac{1}{2}) \ln (S/n[t])) \right\}. \tag{3}$$

The second term contains contributions $L[t]$ which may be calculated for each class separately, and which will be called the class name term for that class. The first term, $H(T)$, which depends on the number of classes, will be called the class dictionary term.

2. For each unordered multistate attribute d, the probability of occurrence of state m of the attribute in class t is estimated by

$$(n[m, d, t] + 1)/(n[d, t] + M[d]) \quad (4)$$

where $n[m, d, t]$ is the number of things in class t having state m of attribute d, $n[d, t]$ is the number of things in class t having any known value of attribute d, and $M[d]$ is the number of states of attribute d.

The length of the label used in class t to indicate possession of state m of attribute d is therefore

$$c[m, d, t] = -\ln((n[m, d, t] + 1)/(n[d, t] + M[d])) \quad (5)$$

The estimate (4) is slightly biased to prevent divergence of (5) when $n[m, d, t] = 0$, and has the useful effect of allowing a thing to be assigned to a class without excessive cost even though it has a state m not possessed by any existing member of the class.

The total message length attributable to encoded values of multistate attribute d in class t, and to the description of the manner of encoding is given by

$$F[d, t] = \tfrac{1}{2}(M[d] - 1)\ln(n[d, t]/12 + 1) + \sum_{m=1}^{M[d]} (n[m, d, t] + \tfrac{1}{2})c[m, d, t] - \ln(M[d] - 1)! \quad (6)$$

3. For each continuous attribute d the distribution is assumed to be normal within the class t. Its mean is estimated by

$$\mu[d, t] = (\sum_{\text{in } t} x[d, s])/n[d, t] \quad (7)$$

(where $\sum_{\text{in } t}$ means summation over those things in class t having a known value of attribute d) and its standard deviation is estimated by

$$\sigma[d, t] = ((\sum_{\text{in } t} (x[d, s])^2) - n[d, t](\mu[d, t])^2)/(n[d, t] - 1) \quad (8)$$

The minus one in the denominator is not present in a normal maximum likelihood estimate, but arises from the need to quote the mean to an accuracy dependent on the standard deviation.

Because (8) fails when $n[d, t] = 1$, we use a modified estimate

$$\sigma[d, t] = ((\sigma[d, 0])^2 + (\epsilon[d])^2 + \sum_{\text{in } t} (x[d, s] - \mu[d, t])^2)/n[d, t], \quad (9)$$

where $\sigma[d, 0]$ is the standard deviation of the population as a whole.

The total message length attributable to the encoded values of continuous attribute d in class t and the description of the distribution parameters is

$$F[d, t] = \tfrac{1}{2}\ln(2(\sigma[d, 0])^4 n[d, t](n[d, t] - 1)/(9(\sigma[d, t]^4))) + n[d, t](\ln(\sigma[d, t]/k\epsilon[d]) + \tfrac{1}{2}) + \tfrac{1}{2}, \quad (10)$$

where $k = 1/\sqrt{2\pi}$.

This form assumes that the ranges of possible values within which the values of μ and σ must be located are $4\sigma[d, 0]$ and $\sigma[d, 0]$ respectively.

The program contains a routine '*typrob(t)*' which calculates the optimum parameters for a class t, and finds the total length of message contributed by the class. It requires as data:

(*a*) the fixed information:
S;
$M[d]$ for each discrete attribute;
$\epsilon[d]$, $\mu[d, 0]$ and $\sigma[d, 0]$ for each continuous attribute.
This data is set up at the beginning of the program, and is available to all routines.
(*b*) $n[t]$;
(*c*) for each state of each discrete attribute, $n[m, d, t]$;
(*d*) for each continuous attribute $\sum_{\text{in } t} x[d, s]$, $\sum_{\text{in } t} (x[d, s])^2$ and $n[d, t]$.

Typrob(t) calculates from this data the shortest possible total message length for the class, i.e.

$$F[t] = L[t] + \sum_{d=1}^{D} F[d, t]. \quad (11)$$

It also calculates and tabulates for later use:
(*a*) the class label length $l[t] = \ln(S/n[t])$;
(*b*) for each state of each discrete attribute, the state label length $c[m, d, t]$ given in (5);
(*c*) for each continuous attribute the mean $\mu[d, t]$ given in (7), the standard deviation $\sigma[d, t]$ given in (9), and a distribution normalising constant

$$g[d, t] = \ln(\sigma[d, t]/(k\epsilon[d])). \quad (12)$$

Where a class has no member having a known value for an attribute d, *typrob(t)* tabulates for $c[m, d, t]$ or μ, σ, $g[d, t]$ the appropriate values for the population as a whole.

The complete distribution adjustment process, which is performed by a procedure '*adjust*', consists of:
(*a*) scanning the classes to eliminate any which have lost all their members, at the same time reducing T, the number of classes, and renumbering the remaining classes so their numbers form a compact set from 1 to T;
(*b*) applying *typrob(t)* to all classes;
(*c*) calculating the total message length

$$F = H(T) + \sum_{t=1}^{T} F[t]. \quad (13)$$

Reclassifying

This tactic finds the optimum assignment of the S things to the T classes, the number, distribution parameters and class name labels of the classes being kept constant.

Reclassifying consists simply of finding, for each thing, the class distribution which allows the most economical description of its attributes. That is, the class is found which has the highest density in the neighbourhood of the thing in measurement space. A routine '*samprob*' calculates the message length $F[s, t]$ required to encode the attributes of s using the density distribution of class t. *Samprob* uses the values $c[m, d, t]$, $l[t]$ and μ, σ, $g[d, t]$ tabulated by *typrob(t)* to calculate

$$F[s, t] = l[t] + \sum_{d \text{ discrete}} c[x[d, s], d, t] + \sum_{d \text{ continuous}} (g[d, t] + (x[d, s] - \mu[d, t])^2/2(\sigma[d, t])^2). \quad (14)$$

During the classify process, performed by a routine '*classify*', each of the S things is treated in turn. No notice is taken of the previous classification of the thing. The thing is assigned to the class giving the least cost as calculated by *samprob*. If two or more classes give the same cost, the thing is assigned to that class to which the fewest things have so far been assigned. If there is still a tie between two or more classes, the assignment is made randomly to one of these classes.

As *classify* assigns things to classes, it accumulates for all classes and dimensions the quantities $n[t]$, $n[d, t]$ and $\sum_{\text{in } t} x[d, s]$ and $\sum_{\text{in } t} (x[d, s])^2$, or $n[m, d, t]$ which are required by the procedure *adjust*.

Splitting

This tactic enables the number of classes to be increased. A class can be split into two and at the same time the optimum choice of the class distribution parameters is made for the two new classes.

Distribution parameters are tabulated for each class, and also for each of two subclasses of each class. The procedure *classify*, having decided to assign a thing to a particular class then uses *samprob* to assign it to one of the two subclasses of that class. Sums of variables, sums of squares, and numbers in states are accumulated for subclasses as for classes.

The procedure *adjust* uses the accumulated sums for subclasses to optimise the subclass distribution parameters exactly as for classes, and calculates and tabulates the $F[t]$ function for the subclasses.

When a class is split by the splitting tactic, it is replaced by what were its subclasses. The split affects the class dictionary term $H(t)$, and the term $F[t]$ for the class split.

The decrease in message length if class t is split into its two subclasses u and v is given by

$$B(t) = H(T) - H(T+1) + F[t] - (F[u] + F[v]) \quad (15)$$

In the program a procedure *split* finds the class yielding the largest value of the function $B(t)$ and if $B(t) > 0$ and there is still storage space to accommodate more classes, the split is carried out. All parameters of class t are replaced by those of one subclass and the other subclass parameters are transferred to class $(T+1)$ which was previously vacant.

The two new classes now have no subclasses so as a starting point the subclass parameters are set equal to those of their parents. This causes a tie for subclass membership during the next classify step, resulting in two pairs of random subclasses being set up. Further reclassifying should improve these.

Merging

This tactic is the exact opposite of splitting: two classes are combined into one, thus reducing the total number of classes by one. The optimum choice of parameters is made for the new class. The decrease in message length if classes u and v are combined to form class t is given by:

$$A(u, v) = H(T) - H(T-1) + F[u] + F[v] - F[t]. \quad (16)$$

A routine *merge* (u, v) evaluates $A(u, v)$. The term $F[t]$ is evaluated by *typprob*(t) which requires the terms:

(*a*) $n[t]$;
(*b*) for each multistate attribute $n[m, d, t]$;
(*c*) for each continuous attribute $\sum_{\text{in } t} x[d, s]$ and $\sum_{\text{in } t} (x[d, s])^2$.

These are formed by *merge* (u, v) each as the sum of the corresponding pair of terms for classes u and v.

After combining two classes their parameters are retained as those of the subclasses of the new class.

Swapping

This tactic forms two new classes from two existing classes by transferring a subclass from one to the other. There are four ways in which this may be done, but the total number of classes remains unchanged. The optimum choice of class parameters is made for the new classes.

The decrease in message length when class u transfers a subclass to class v to produce classes u' and v' is given by

$$W(u, v) = F[u] + F[v] - (F[u'] + F[v']) \quad (17)$$

The routine *swap*, which calculates $W(u, v)$ uses *typprob*(t) in the same way as *merge* does for $F[t]$, to evaluate $F[v']$; $F[u']$ is already known as it was a subclass of u.

After swapping, the swapped subclasses of u and the old v are kept as subclasses of v'. The new class u' has its subclass parameters set equal to its own as is done after a split.

In the program merging and swapping are both carried out together by a procedure *combine*. Operations yielding the greatest decrease in message length are carried out first. *Combine* uses *swap* (u, v) and *merge* (u, v) to find the largest of the terms $W(u, v)$ and $A(u, v)$ for all combinations of u and v. If worthwhile, then the merge or swap operation is carried out. This procedure is repeated until there are no more swappings or combinings worthwhile.

If there is no spare class storage, which is possible if nothing has been combined, then spare storage for one class and subclasses is made available by combining the pair of classes with maximum (negative) $A(u, v)$. At this point the compaction procedure is again used to remove any classes with no members and form a compact set numbered 1 to T.

Programmed minimisation strategy

An iteration

The four procedures incorporating the five minimisation tactics are combined into a single iteration. This iteration can be split into two stages. The first using *classify* and *adjust* will by itself improve the information measure by altering the class membership, but is unable to increase the number of classes. The second stage utilising *split* and *combine* allows the number of classes to change and will also reshuffle subclasses to help prevent the first stage from becoming trapped in a false minimum. It can be shown that repeated iterations will converge (not necessarily to the true minimum) after a finite number of iterations.

Start up

The iteration requires as a starting point at least one class with values of all the distribution parameters for it and its two subclasses.

If Snob is required to assess and attempt to modify a known classification, then the classes of this given classification provide the starting point. These classes may be specified to the program either by listing the class distribution parameters or by listing the class membership of each thing. For the later case, the distribution parameters are obtained by first accumulating the necessary quantities (in a dummy *classify* phase) and then using *adjust*. All subclass parameters are set equal to those of their parents which causes the initial allocation to subclasses to be random.

If no initial classification is supplied then a classification consisting of a number of random classes is generated. This is achieved by setting the distribution parameters of the required number of classes and their subclasses all equal. The initial *classify* stage will allocate things to the classes and subclasses randomly as a result of the ties due to identical parameters. For the first few iterations combining is inhibited to stop the random classification from collapsing and to allow the classes to stabilise.

Possible improvements to an iteration

It has been found in practice that the split-combine stage does not usually find any worthwhile changes to perform every iteration. It thus need not be applied during every iteration.

It is also found that modification of subclass membership generally continues for a number of iterations after the class membership has become constant. Reclassifying of subclasses is continued until the subclasses stabilise, as a *split* or *swap* may become worthwhile as they improve. Computation time can be saved at this stage by applying *classify-adjust* only to subclasses once the class membership has become constant. To achieve this saving we must store the current class membership of each thing so that we know which two subclasses it can belong to.

Computer considerations

Storage

The bulk of the data is the matrix of $D \times S$ attribute values. As this is only accessed when each thing is being reclassified, only one column containing the D values of the thing being classified need be held internally at any time. The whole matrix can thus be kept on magnetic tape making the number of things that can be classified not limited by internal storage considerations.

The information measure is minimised with respect to the number of classes. However, due to storage considerations an upper limit to their number must be set. Except for very large problems this limit Tm is usually greater than the number of classes in the final solution. It can, however, be intentionally set lower, in which case the optimum solution of not more than $(Tm - 1)$ classes will be found.

The value of Tm imposed by storage considerations is approximately given by:

$$Tm = \left(\frac{A - B - 2D}{2P + 3} - 2\right)\Big/3$$

where A is the available data storage, B is the size of the segment of the data matrix $x[d, s]$ actually held in core and $P = \sum_{d=1}^{D} p[d]$ where $p[d] = 3$ for d continuous, and $p[d] = M[d]$ for d multitstate.

Input

The following information is input to Snob.

(*a*) The type of each attribute, that is, multistate or continuous.
(*b*) For each continuous attribute $\epsilon[d]$.
(*c*) For each multistate attribute $M[d]$.
(*d*) The data matrix $x[d, s]$.

For multistate attributes the states are arbitrarily numbered from 1 to $M[d]$. There is no need to normalise continuous attribute scales as the range of values does not affect the weight attached to an attribute. Missing attribute values are indicated by an extreme value of 10^{+9}.

We use another program to record the data matrix on magnetic tape. It has facilities for excluding any combination of attributes and for replacing a group of a number of binary attributes by a single equivalent multistate attribute. This is done when M-state data has been coded in terms of M mutually exclusive binary attributes because of the inability of some process to deal with it in the multistate form.

(*e*) Tm, that is, the maximum number of classes plus one.
(*f*) A value to set up the random number generator used for resolving ties.

Different initial settings of the random number generator will often result in final classifications with slightly different information measures, and slightly different class memberships.

Output

The optimum classification may be specified either by the class membership of each thing or by the set of distribution parameters of each class. Both of these specifications are output by Snob. The distribution parameters are always held in core, however the class membership need not be so held. It can, however, be generated by reclassifying.

To indicate the diagnostic efficiency of different attributes the following additional information is output for each:

(*a*) For a continuous attribute the population is dichotomised in T different ways, one per class. The dichotomy for class t is between class t and the rest of the population. The mean and standard deviation of class t and of the rest of the population are output.
(*b*) For multistate attributes T, $M[d] \times 2$ contingency tables are constructed, one for each class. For class t the $M[d] \times 2$ table compares the incidence of the attribute in class t with its

incidence outside class t. For each such table a value of chi-square is output.

Computation time

The computations during *classify* as the computer scans the data matrix on tape are quite simple. For each thing $T + 2$ *samprob* calculations are made which involve only table lookup operations and the evaluation of squared differences. The more extensive calculations of the *adjust* and *combine* procedures are carried out during tape rewind and the time they take is independent of S.

About 10 iterations are normally required, depending on the size of the problem and how well the starting point leads to the final solution.

For a problem involving 200 things and 80 binary attributes about 30 minutes of computation time were required when starting from one random class. We would expect quite a large reduction in computation time if the program was rewritten in assembly language, as KDF9 ALGOL is rather inefficient when a large amount of sequential array addressing is involved, as is the case here.

Discussion

Snob can be discussed on at least three levels: first, on the merits of what it attempts to do, i.e. on merits of the minimum information measure concept, second, on the merits of what it actually does, i.e. the efficiency and effectiveness of the minimisation tactics, and third, on how it does it, i.e. program structure. We do not intend to discuss the third level, especially as the program is designed for ease of modification and experiment rather than efficiency.

The information measure

Discussion of the first question is difficult in that few other objective criteria for judging classifications, based on a clearly stated aim, have been discovered by us in the literature. However, we may make some modest claims for the information measure.

Firstly, and perhaps most importantly, we hope that its proposal will stimulate others to consider whether or not the excellence of different classifications can be objectively compared, and if so, on what criteria.

Secondly, provided that our premises concerning the expected forms of density distributions of classes are acceptable, a classification which minimises the information measure has the following properties:

(*a*) A probability distribution function in measurement space is provided for each class.
(*b*) The parameters describing the probability distribution for a class are assigned values which are essentially maximum likelihood estimates based on the things assigned to that class.
(*c*) Each thing is assigned to that class most likely to contain a member like that thing.

If the context of the classification problem is such that it is reasonable to expect the classes derived to have a simple and compact internal distribution in measurement space, the notion of a probability distribution function for a class is natural. If the context leads one not to expect a compact internal distribution within each class, no other classification technique known to us has much hope of success.

Given, then, that a distribution function is a reasonable concept, there seems little point in not attempting the best possible estimate of its parameters. The nearly maximum likelihood estimators used by Snob can be expected to yield estimates nearly as good as can be achieved by any estimators. Again, if the concept of a probability distribution is accepted, there seems little point in assigning a thing to a class which, on the basis of the estimated distributions, is not the most likely to contain such a thing.

Note that these remarks are not intended as a justification of our particular choice of model distribution functions, viz. uncorrelated normal and unordered multistate probability functions. However, this choice is not central to the information measure concept, and the mathematical form of the measure can be modified to accommodate different forms of distribution function (e.g. poisonnian, log normal, correlated multivariate normal, etc.) where these are appropriate.

It may be instructive to compare the information measure approach with one recently proposed by G. H. Ball and D. J. Hall (1967). The latter sets up a measure which is the total within-class variance, and seeks to minimise this measure by an iterative technique similar to our own. It may be shown that the resulting classification is a maximum likelihood estimate of the parameters of a complex hypothesis about the data, the elements of the hypothesis being:

(*a*) That each thing belongs to a certain class.
(*b*) That each attribute is distributed normally within each class with a certain mean, and that attributes are uncorrelated within a class.
(*c*) That all classes are equally abundant.
(*d*) That all attribute distributions in all classes have the same (unspecified) standard deviation.

By contrast, the information measure approach gives estimates of the parameters of an hypothesis containing elements (*a*) and (*b*), but replacing (*c*) and (*d*) by separate estimates for each class of its relative abundance, and within each class, of the standard deviation of each continuous attribute.

The inclusion of class description information in our measure is perhaps more contentious. If the aim of providing an efficient encoding is accepted, the class description information is undoubtedly an essential part of the total message length. If, however, Snob is viewed as producing maximum likelihood estimates, the class description information plays the role of an *a priori* probability, and has the effect of yielding an optimum number of classes. (By contrast, the number of classes is controlled by Ball and Hall by imposing arbitrary upper and lower limits to the variance within a class. If the upper limit is exceeded, the class if split. If the total variance of two classes falls below the lower limit, they are combined.)

Work is at present proceeding to relate the prior probabilities in our measure to tests of the statistical significance of the distinction between classes. Although this work is at an early stage, it appears that at least in some simple cases, the cost of class splitting introduced

by the class description terms is of the same order as the expected benefit in likelihood resulting from the most advantageous split of a random sample from single-class population.

The minimisation tactics

The tactics used by Snob to find the minimum information measure are deficient in that the minimum is often not found. The problem of false minima is inherent in the iterative approach to the solution, in which at each stage we seek to improve an existing class structure by relatively minor adjustments. Locally optimum structures can occur which cannot be improved by any minor adjustment, but which differ grossly, e.g. in the number of classes, from the optimum.

The tactics of splitting, merging, and swapping are certainly useful, but even they make adjustments to at most two classes simultaneously. No doubt tactics which considered restructuring groups of three or more classes simultaneously would reduce the number of inescapable false minima. The number of possible groups, and the number of possible rearrangements of each group, are large and would present a formidable computing task unless some simple technique were found for selecting for examination those regroupings most likely to be profitable.

Even within the armoury of tactics we employ, there is room for considerable variation in strategy, which has not yet been fully explored. However, there are indications that if no known classification is to be used as the starting point, it is usually wiser to begin with a large number of classes than with one. If a large number of classes is set up randomly, and merging initially inhibited, the classes will rapidly distribute themselves throughout measurement space in a rough correspondence with the sample distribution. After the first few iterations, merging is permitted and the classes rapidly combine, with refinement of their parameters and membership, to a stable situation.

If one starts with a single class, it will split up, but only after its subclasses have been somewhat refined. The subclass mechanism always underestimates the benefit to be obtained from splitting a class, and our experience has been that if the process is started from a single class, the number of classes found will often be less than the optimum.

Acknowledgements

This work was carried out whilst the authors were in the Basser Computing Department, School of Physics, University of Sydney. We are grateful for the use of the KDF9 computer and other facilities of the School of Physics. Whilst engaged in this work one of us (D. M. Boulton) was in receipt of a University of Sydney Post-Graduate Studentship.

References

BALL, G. H., and HALL, D. J. (1967). A Clustering Technique for Summarizing Multivariate Data, *Behavioural Science*, Vol. 12, No. 2.

BOULTON, D. M., and WALLACE, C. S. (1969). The Information Content of a Multistate Distribution, *Jnl. of Theoretical Biology*, Vol. 23, p. 269.

OLIVER, B. M. Efficient Coding, *Bell System Tech. Jnl.*, Vol. 31, pp. 724–750.

SHANNON, G. E. (1948). A Mathematical Theory of Communication, *Bell System Tech. Jnl.*, Vol. 27, p. 379 and p. 623.

SOKAL, R. R., and SNEATH, P. H. A. (1963). *Numerical Taxonomy*, W. H. Freeman & Co., San Francisco and London.

WALLACE, C. S., and BOULTON, B. M. (1968). An Information Measure for Classification, *The Computer Journal*, Vol. 11, p. 185.

WILLIAMS, W. T., and DALE, M. B. (1965). Fundamental Problems in Numerical Taxonomy, in Advances in Botanical Research 2.