Defence against the flood

Douglas Kell suggests that genetic computing can offer a solution to the data mining and predictive modelling challenges of today

The post-genomic era is generating such torrents of data that researchers now face the question of how to build a working data-flood defence system. Existing software is just not coping and the problem is to find new ways of extracting knowledge from all these data.

'Genomic computing' is a new and valuable strategy for analysing very complex datasets of high dimensionality by evolving simple, intelligible rules with high explanatory power.

Currently the majority of packages require the user to write a lot of code as part of their operation. Few packages have a user interface good enough to enable relative novices to use these techniques smoothly and easily. However, genomic computing's potential for ploughing rapidly through vast amounts of data to evolve simple, human-readable solutions, would appear to make it the ideal approach. The reason for its lack of wider recognition probably stems from the fact that the bulk of the real expertise still resides in the form of 'hair shirt' programs within university departments. Until the mainstream software vendors become fluent in this branch of computer science, the market is still open to any resourceful software development group. The Welsh software company Aber Genomic Computing has now made available a genomic computing package (gmax-bio) that is designed to address data mining and predictive modelling problems encountered in the life sciences and elsewhere.

The reason data analysis is becoming ever more complex is that it has to deal with large numbers of data objects, each represented by tens, hundreds, or thousands of variables, as in DNA microarray analysis. The combinatorial explosion of solutions to be evaluated can thwart conventional attempts at empirical interpretation.

Consider a predictive model with only 100 variables (e.g. levels of metabolite, antigen, gene expression etc). The simple problem of deciding whether or not (a simple 'yes' or 'no') to use each of these 100 variables gives 2^{100} possibilities, which is about 10^{30}. Considering that the lifetime of the universe is 'only' 10^{17} seconds, to find a solution for this comparatively trivial problem by random search would take more than an eternity. Fortunately nature has shown us a way – a process that is incredibly simple and yet phenomenally powerful – natural selection. Computing has an equivalent approach to solving these complex problems: 'evolutionary computing' or the evolution of computer programs by methods of Darwinian selection.

All datasets can be viewed as a spreadsheet or database table (Fig. 1), in which different samples (individuals, objects) appear in different rows while the values or classes of variables (properties) associated with them appear in different columns. It is frequently the case that we wish to account for some properties in terms of combinations of some of the other variables. In data mining, the ones we want to account for are usually termed the dependent variables or *y*-variables or *y*-data while the ones contributing to the explanation are usually called the explanatory variables or *x*-variables or *x*-data.

In a pharmaceutical drug discovery problem, the *x*-variables could be structural and/or physico-chemical attributes of candidate drug molecules and the *y*-variable the potency or binding efficiency of the molecule in an appropriate assay. In predictive medicine, the *x*-variables could be relevant or surrogate metabolic properties, or markers, and the *y*-variable the existence or severity of some disease state. In all cases, the aim of predictive modelling is to find those *x*-variables which, when combined in a stated way (to make a mathematical model), best account for the values or properties of the *y*-variables in a statistically robust fashion (so that the models produce correct predictions for other examples on which they are subsequently tested).

Class assignment from multivariate data

This problem can be set out equivalently in the form given in Fig. 2. In this formulation, it can be restated as a 'class assignment problem', represented in general terms by:

(1) a set of inputs, whose choice one might hope to optimise;

(2) a set of output classes into which one might wish to assign a sample whose

Variables are arrayed in different columns

		Explanatory (*x*-) Variables.......			Dependent (*y*-) Variable(s)		
		XVar1	Xvar2	Xvar3....	Yvar1	Yvar2...	Yvar3...
Objects	Obj1						
(Samples)	Obj2						
Going	Obj3						
Down	Obj4						
In	Obj5						
Different	Obj6						
Rows	Obj7						

Fig. 1: Structure of a database table or spreadsheet matrix.

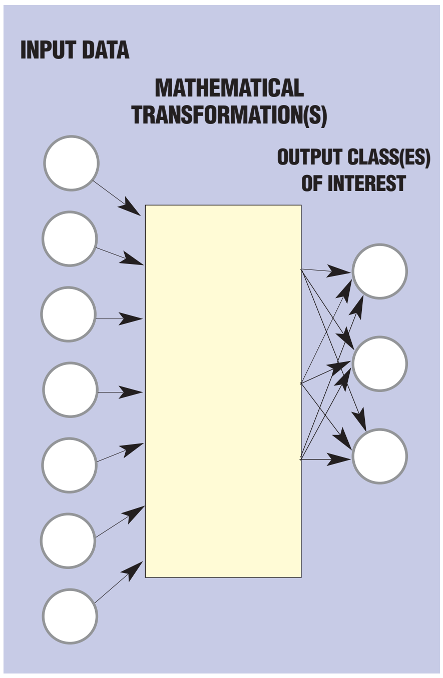


Fig. 2: By restating the spreadsheet formulation of the data mining problem, it can be transformed into an equivalent, class assignment problem and this becomes an issue of pattern recognition.

measured properties are to be used as the inputs; and

(3) a set of mathematical relationships (functions) which use the differential data in the input patterns to establish the correct functional class assignments on the basis of those input patterns.

Data mining is designed to optimise (1) and (3), given (2).

Data mining is thus, in essence, a problem in pattern classification. Indeed, there are many areas of science in which pattern classification methods developed in statistics and artificial intelligence are important, and where the arrangement is exactly as shown in Fig. 2. The goal of pattern recognition is to classify objects of interest that possess particular attributes into a number of categories or classes. Functional genomics, predictive medicine, and other extremely important problems are therefore to be seen, in part, as problems of pattern recognition.

Pattern classification methods of this type can be grouped into two different categories, called 'unsupervised' and 'supervised' learning methods. If a set of multivariate observations is given with the aim of establishing the existence of classes in the input data, with no knowledge or care for an imposed class structure (i.e. we use only the *x*-data as defined above), we will be using clustering or unsupervised learning. Alternatively, there may be a defined class structure (based on knowledge for at least some samples of the *y*-data) and the need is then to establish rules by which new objects are correctly classified into one or more of the existing classes. This supervised learning is often referred to as 'discrimination' or 'multivariate calibration' in the statistical literature, as the class structure is produced on the basis of known, correctly classified objects and their attendant properties.

Because the input data may have hundreds or thousands of variables, getting the correct classification is only part of the answer. We also require our solutions to be intelligible, so simple, explanatory rules are greatly favoured over complex, incomprehensible ones.

Models are made with explanatory variables (and not objects) as the inputs. Historically, datasets might have had many objects and few variables, but now the opposite is true. There are many cases in which we can have a large number of variables on both large and small numbers of objects. The interesting fact is that the number of variables and not the number of objects dominates the mathematical difficulty of a modelling problem.

Imagine that we have one dependent variable in a diagnostics problem (this person does/doesn't have a certain disease such as cancer) but that there are *N* explanatory variables (levels of metabolites, antigens, gene expression profiles, etc).

As discussed above, even if each variable (of the *N*) can take only 2 values – present or absent – the number of models one can make to try and solve the problem is 2^N. If each variable could take 10 values then for 100 variables there are 10^{100} models. In general, obviously, the complexity of a system with *N* variables present at M levels scale as M^N and if $N >> M$ then M^N is always likely to be enormously greater than N^M. Combinatorial difficulty scales exponentially (or worse) with the number of explanatory variables. (Gene microarrays, for examply, normally have several thousand variables….) The combinatorial difficulty scales only linearly with the number of objects. It is the experimental ability to acquire huge numbers of explanatory variables that opens up the opportunity for serious and efficient data mining (or, equivalently, provides competitive advantage to those who can deal with them and competitive disadvantage to those who cannot).

The ideal data mining system has seven key features:

- Automatically identifies the key components and patterns in huge datasets;
- Gives the solution within minutes;
- The solution is simple, intelligible, robust, and maps directly into the measured input variables;
- Has the ability to provide several alternative solutions to a problem;
- Fully transparent, easy to follow solutions – no black box operations or answers;
- Able to incorporate prior knowledge from the database; and
- Produces readily exportable solutions to be applied to new datasets.

Systems such as neural networks can address only part of this profile – they are flexible, non-linear fitting systems – but they are rather poor at explaining how they arrive at their answer since every input tends to contribute. Only rule-based systems can give simple, effective, and intelligible explanations, but traditional rule-based systems (deterministic, tree-based classifiers), in which each variable is selected in the order of its apparent individual importance, often perform poorly on complex datasets.

The ideal method would evolve rules based on an optimal selection of the variables and an optimal selection of the form of the interactions between them, to produce a rule – exactly equivalent to a little computer program – in which the selected variables are applied at the input, the rule followed, and the correct classification arrived at as the output. The generalised methods of data mining known as Genetic Programming or Genomic Computing do exactly this.

Genomic computing is a new development within the more general field known as evolutionary computation, and as such shares its underlying principles (Fig 3).

Its operation is based on the Darwinian ideas of 'survival of the fittest' and natural selection. In evolutionary computing, we have a population of individual computer

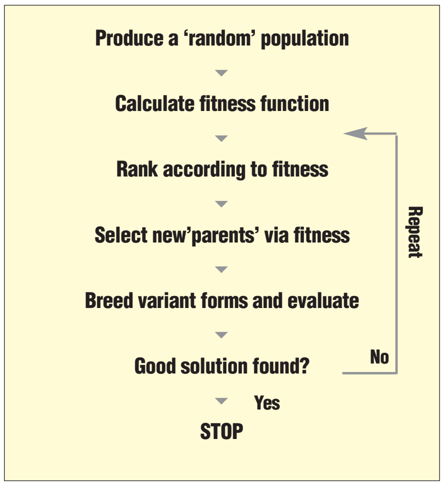

Produce a 'random' population
Calculate fitness function
Rank according to fitness
Select new'parents' via fitness
Breed variant forms and evaluate
Good solution found?
No
Yes
STOP
Repeat

Fig. 3: The basic strategy of evolutionary computing.

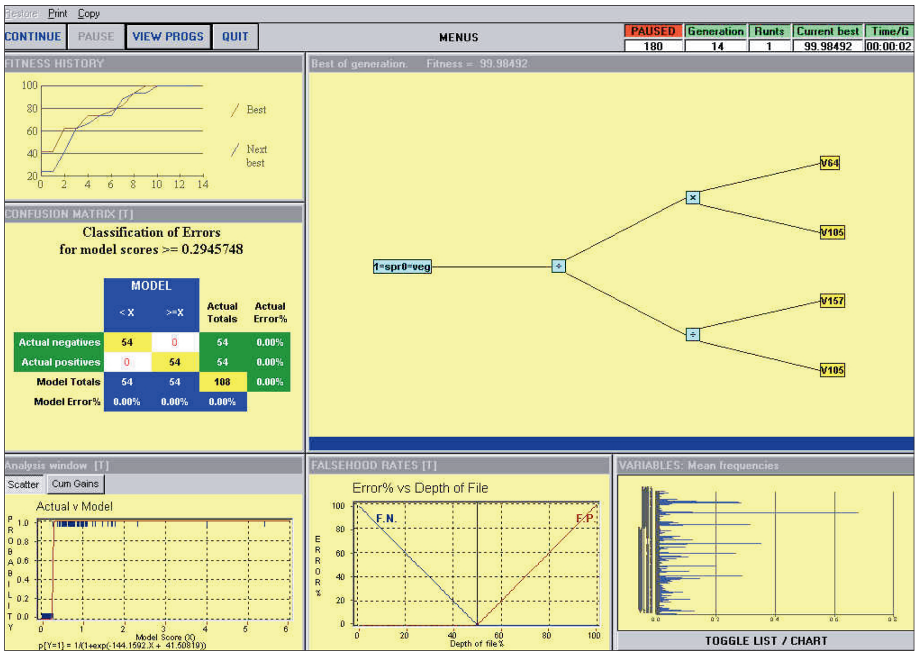
programs or algorithms whose output is a potential solution to a problem (typically a combinatorial optimisation problem).

The process begins with a fitness function and a set of user-selectable mathematical and logical operators. A first generation of as many as a thousand models is randomly generated using the operators available, usually in the form of a tree; the 'fitness' of each model is evaluated using a training data set. These outputs are ranked according to their 'fitness', and the better performing individuals retained.

Some of these individuals are then modified, by mutation ('asexually') or by recombining parts of them from more than one parent ('sexually'), and the process of generating an output, evaluating their fitness and mutating and selecting at each generation is continued until an individual evolves with the ability to solve the problem or reaches a stopping criterion.

Although there can be specific operations that are not known to occur in the natural world, there is an equivalence or mapping between biological evolution and evolutionary computing (Fig. 4).

Although there are many approaches and algorithms within the evolutionary computing field, most (such as genetic algorithms, evolutionary strategies, evolutionary programming) assume knowledge of the basic equation and variables to be used, and simply seek to parameterise them optimally. However, an important alternative approach, popularised by John Koza as genetic programming, seeks not only to extract the most important subset of the variables but also to derive the functional form of the relationship between them that best accounts for the problem at hand. This strategy can automatically and inductively derive new knowledge in the form of simple, explanatory rules, and is implemented in genomic computing. Importantly, it is particularly good at selecting only a small subset of the available variables; this both makes the searching a lot easier and ensures that the rules which are generated are simple and intelligible.



PAUSED	Generation	Runts	Current best	Time/G
180	14	1	99.98492	00:00:02

Classification of Errors
for model scores >= 0.2945748

	MODEL			
	< X	>=X	Actual Totals	Actual Error%
Actual negatives	54	0	54	0.00%
Actual positives	0	54	54	0.00%
Model Totals	54	54	108	0.00%
Model Error%	0.00%	0.00%	0.00%	

Fig. 5: A screenshot showing how gmax-bio identified a simple rule using just 3 variables (out of 150) to correctly classify all entries in a dataset from bacterial physiology. The rule is read 'up' the tree from right to left and is (output = V105*V63*V85/3).

All output functions can be discussed in terms of classification or quantification. However, although there are technical reasons why one might treat them differently, quantification is really just a special form of classification ('this sample is in the class whose values lie in the range x to (x+∂x)'). Problems that have exploited these rule-generating methods of evolutionary computing include spectroscopy, searching biological sequences for important motifs, DNA microarray analysis for functional genomics and metabolic profiling.

The gmax-bio genomic computing package, now available from Aber Genomic Computing, addresses data mining and predictive modelling in the life sciences, other natural sciences, and elsewhere. Fig. 5 shows a simple rule which the software has evolved to solve a bacterial biomarker identification problem. Gmax-bio has identified a simple rule, which uses just 3 variables out of 150 and which correctly classifies all examples in a dataset from bacterial physiology.

Professor Douglas B. Kell is at the Institute of Biological Sciences, University of Wales, Aberystwyth, SY23 3DD. DBK@aber.ac.uk

For further information on organisations and products mentioned in this article, please visit http://enquiries.scientific-computing.com

Property	**Biological Evolution**	**Evolutionary computation**
Unit of selection	Individual organism	Individual string, tree, algorithm or computer program
Fitness	Likelihood of surviving long enough to produce offspring	Relative closeness to desired property and hence propensity to be chosen for next generation
Mutation	Change of the nucleotide at a certain position	Change of an individual string, tree, algorithm or computer program by changing the encoding at a certain location
Recombination	Mating between organisms leading to exchange of alleles via recombination	Change of an individual string, tree, algorithm or computer program at a certain location by taking segments of encoding from (usually) 2 or more 'parents' and recombining them to create a new string, tree, algorithm or program

Fig. 4: Some relationships between biological evolution and evolutionary computing.