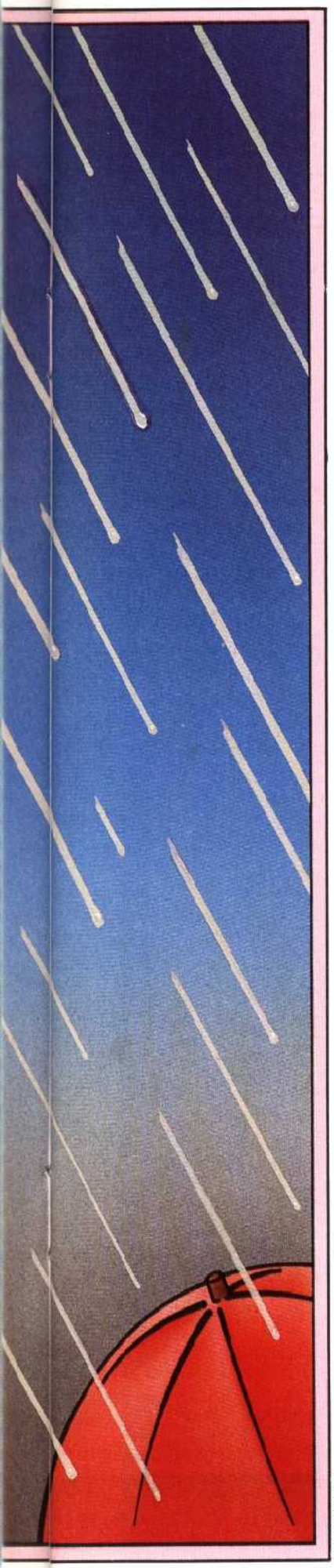

MIKE BROWNLOW

drawing inferences: 'forward chaining' and 'backward chaining'. Broadly speaking, forward chaining involves examining data in order to form hypotheses, while backward chaining attempts to find data to prove or disprove an already formed hypothesis. Pure forward chaining leads to unfocused 'What if...' type questioning of the system, whereas pure backward chaining tends to be rather relentless in its goal-directed questioning.

Most successful systems use a mixture of both. Whether an inferencing procedure works primarily backwards or forwards, it will have to deal with uncertain data. Computer specialists have tried to force the world we live in into the rigid confines of the computer, and it has never been a comfortable fit. Now ES research has given us means of dealing precisely with uncertainty — in other words with the real world rather than some idealised abstraction that our data system forces us to use.

Indeed we have too many ways of dealing with uncertainty. There are Fuzzy Logic, Bayesian Logic, Multi-Valued Logic and Certainty Factors to name only four: such logics replace the certainty of 'IF X IS TRUE THEN Y IS TRUE' with the cautious statistical inference, 'IF X IS TRUE 65% OF THE TIME THEN Y IS BETWEEN 50% AND 70% LIKELY'. All sorts of schemes have been tried, and the odd thing is that they all seem to work. A possible explanation of this state of affairs is that the organisation of knowledge matters more than the numeric values attached to it. Most knowledge bases incorporate redundancy to allow the expert system to reach correct conclusions by several different routes. The numbers measuring degree of belief are important only as a way of choosing between one value set and another.

Software packages designed to facilitate the knowledge-based approach to systems design are now coming onto the market. So far the only one within the budget of the home computer enthusiast is the HULK (Helps Uncover Latent Knowledge), on sale from Brainstorm Computer Solutions at around £25. It runs only on the BBC Model B and Torch computers, although a QL version is promised.

HULK enables the user to build up and test a set of decision rules, which can later be used for prediction or classification. You will find it useful if you have a set of examples or cases each measured on several variables and wish to discover patterns and regularities for predictive purposes.

For instance, suppose a farmer kept detailed measurements on a home computer of the height, leaf colour and so on of several hundred sugar beet plants, and recorded those that became diseased before harvest time and those that remained healthy. The system could be used to help develop rules relating the characteristics of the plant to its state of health. Later those rules could identify plants at risk and thereby improve prospects for the following year's crop. This farmer might never know what those rules were, but the system could apply them to the plant data as it was supplied.

Alternatively, you might hold information on a season's football results and want to develop a way of categorising games as likely wins, draws or losses on the basis of various indicators known before kick-off. The necessary data for making decisions, such as the home team's recent performance, previous results of this fixture, relative positions of the teams in the league table, is all readily available. But it's very time consuming to organise and difficult to correlate without the aid of an ES. There are plenty of applications for the HULK — all you need is a set of data.

The HULK package consists of two main programs: LOOK (Logical Organiser Of Knowledge) and LEAP (Likelihood Estimator And Predictor). These allow the user to develop a set of rules from a data file of observations held on disk (for example, results of past matches and any other decision factors), and then to apply that rule set to another data file of incomplete observations (for example, the decision factors for next Saturday's matches) in order to make predictions about them. The rule set is the knowledge base: it expresses the system's knowledge about the data. In HULK, it consists of probabilistic decision rules that can be used for classification or forecasting.

The knowledge base grows through the interaction of the user and the computer: rules are proposed by the user and tested by the computer; the user discards those that do not improve the overall performance of the system.

To use LOOK you need a data set, or preferably two (a large data set can be split into two parts for this purpose). One is called the 'training set' for trying out rules, and the other is called the 'test set' for confirming their usefulness on unseen data

Once the data is on file, LOOK is used to test your hunches by proposing new rules, one at a time. It runs each rule over the training set and tells you how much, if at all, it improves the prediction score of the rules already present. Then it recommends whether to keep or discard the new rule. The final decision is left to the user.

LOOK is not a true learning program, but a kind of filter that allows only useful guesses or conjectures through. In a sense, it is a learning program without a learning algorithm; the user does the learning, while the machine does the clerical work of comparing, evaluating, and assessing the data set.

Having created the rules with LOOK, LEAP can be used to apply them to unknown samples. The rules are combined to give a single probability estimate for every sample in the test data set. LEAP's output includes a list of samples ranked according to their likely outcome. What this outcome might be (SCORE DRAW, or HIGH YIELD FROM THIS VINE) depends on the nature of the sample data, and what the user was trying to predict from it. HULK will supply the acquisition module and knowledge base facility, leaving the user in control of the inference engine.