ELSEVIER

J. Chromatogr. A, 739 (1996) 15–24

JOURNAL OF CHROMATOGRAPHY A

Teaching a computer ion chromatography from a database of published methods

M. Mulholland[a,*], P. Preston[b], D.B. Hibbert[a], P.R. Haddad[c], P. Compton[b]

[a]*Department of Analytical Chemistry, University of New South Wales, Sydney, Australia*

[b]*Artificial Intelligence Laboratory, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia*

[c]*Department of Chemistry, University of Tasmania, Hobart, Australia*

Abstract

As ion chromatography (IC) has matured as an analytical technique it has become more automated. Most instrument control and data handling is now handled by computers. However, IC has not seen the abundance of automated method optimisation techniques which are provided to conventional chromatography. To a certain extent this was because IC differed greatly in the approach required to optimise selectivity and sensitivity. There was quite a diverse range of chemistries (or separation mechanisms) applicable to IC, such as ion exchange, ion interaction, etc.

This paper describes an effort to fill this gap by developing an expert system which can give comprehensive advise on suitable method conditions for a variety of IC mechanisms. To build this system we applied an approach known as induction by machine learning, which was developed within the field of artificial intelligence (AI). A database of over 4000 published methods using IC, where the sample information and the chromatographic conditions were recorded, was used to train an expert system (ES).

Both induction and a neural network model were applied to this task and an expert system which can advise on the following IC method conditions: mobile phase, column, pH, mechanism, post-column reactors, suppressor use and gradient applicability, was successfully developed. This paper presents a summary of the most pertinent conclusions from this study.

A test set of different methods was extracted from the database and they were not applied in the training of the expert system. These were used to test the expert system and different amounts of information were used as inputs. The resulting outputs of the expert system were evaluated by the expert, who decided whether the method would work or not and if it was a good method or the ideal method for the application. Over 85% of methods were found to work and almost 62% of the methods were considered ideal. These were acceptable results when one considers the limitations of using a database of published methods as a learning set and the time saved by the use of machine learning.

Keywords: Induction by machine learning; Expert systems; Optimization; Ion chromatography optimization; Machine learning; Anions; Cations

1. Introduction

Ion chromatography (IC) can be carried out using one of several different separation mechanisms, these include ion interaction, ion exchange and ion exclusion [1]. Each of these mechanisms requires a column–mobile phase combination which can produce the chemistry necessary to effect the separation of the ions. Once the ions are separated they need to be detected, and there are many different detectors available, together with detection enhance-

*Corresponding author.

0021-9673/96/$15.00
PII S0021-9673(96)00045-3

ment devices such as post-column reactors or suppressors.

The selection of a combination of these method conditions is a configuration task, where the chromatographer configures the instrument for a given application. This differs from classification problems, which are normally solved using machine learning, and thus several modifications to both the application of machine learning and the final implementation of the expert system (ES) [2] were required. All of the examples of machine learning in the chemical literature deal with classification-type problems [3–9].

Over the last decade much progress has been made in the area of machine learning, including genetic algorithms [3] and neural networks. Most attention was given to neural networks which attempt to simulate the workings of the brain [4]. However, another commonly used technique, known as induction, operates on the principles of inductive logic.

Many algorithms have been developed within artificial intelligence (AI) research which perform classification based on an induction method. Induction is simply the informal logical process of inducing rules from examples. Induction can provide many advantages over neural networks, including the development of legible rules rather than an incomprehensible network of weightings. These techniques have been successfully used to teach a computer how to make decisions in a number of expert applications, some of which tackle chemical problems.

The US Environmental Protection Agency applied induction to building an ES for the estimation of molecular masses and the identification of toxic and other volatile organic compounds using mass spectra [5–7]. The resultant ES was in the form of a decision tree. The ES was developed using a training set of 78 mass spectra of 8 groups of compounds: benzenes, alkanes/alkenes, nonhalobenzenes, chlorobenzenes, chloro alka(e)nes, bromoalka(e)nes, mono- and dichloroalka(e)nes, tri-, tetra- and pentachloroalka(e)nes. The rules developed were then tested against the training set and found to give 93 to 100% correct identification. This compared with 62–98% using a SIMCA pattern recognition algorithm. The chloroalka(e)nes were the only group from the training set that failed to achieve 100% prediction accuracy.

This ES was later improved both by allowing the use of low-concentration spectra [6] and by improving the molecular mass prediction method [7]. The ES has been evaluated using over 100 spectra and has shown an average classification accuracy of 90%.

Massart and coworkers [8,9] performed several evaluations of the induction methods provided with the commercial systems EX-TRAN and TIMM, and compared their performance with classical pattern recognition systems. These systems were compared with k-nearest-neighbour and linear discrimination analysis. A set of 100 olive oil classifications was used to train the system. The data included information on numerical results of the oil analysis and four possible conclusions for the district of origin. The use of numerical rather than symbolic inputs is perhaps not the best way to evaluate induction techniques, as they use fairly low level algorithms to handle such continuous data. Despite this the results showed that these methods were as good as (sometimes better than) the traditional pattern recognition systems.

The only previous research found that tackled a chromatography problem was the work of Appel et al. [10]. This group applied machine learning to classify two-dimensional gel electrophoresis patterns. The system was applied to biological samples such as a liver biopsy. The work of Gelernter et al. [11] was interesting in that they used chemical structures as a training attribute to classify various reaction schemes. The data from the machine learning were then further refined using a modified version of Cameo, a program developed specifically for generating descriptions of organic chemistry mechanisms. No data on the success of the system were given. Röse and Gasteiger [12] describe a similar application but specifically applied to the Diels–Alder reaction. They did not supply any information on the algorithm or any results of the system evaluation. All machine learning is based on learning by examples, and for this reason it is essential to have a large number of example data and the conclusions elicited for that data.

1.1. Classification

Classification can be defined as the reduction of a large number of observations into discrete categories

based on some commonality between class members. The following are some examples of classification problems.

Diagnosis: One of the first classification tasks tackled in AI was the diagnosis of diseases. The multitude of illnesses and their symptoms can be reduced to some well-understood ailments by recognising common symptoms.

Naming organic compounds: Naming organic compounds requires several levels of classification based on the various functional groups. For instance, organic acids, alcohols, sugars and carbohydrates are just a few of the possible classes of compounds which need to be recognised by features of the molecular make-up.

Identification of spectra: Various qualitative methods of spectroscopy allow the classification of spectra according to features arising from the presence of functional groups in the molecules being studied. For instance, distinctive bands are observed in IR spectra which are indicative of the presence of a double-bonded oxygen. It is thus feasible to build up a set of rules by which you can identify (or classify) compounds by examining their IR spectra.

In summary, most machine learning tools and expert system technology can be applied to any problem which can be framed as a classification. (Where there is a set of examples which list the various properties and the resultant classification.) Strictly speaking the configuration of an IC method is not a classification problem but rather a configuration problem; however, the problem could be framed in terms of several interdependent classifications.

1.2. Induction

This paper describes the application of a commonly used type of machine learning algorithms which are based on the method of induction. Inductive logic is best known as a scientific method of acquiring knowledge from the collection and examination of raw data. When one performs or collects the results of experiments and extracts a general rule, the logic is referred to as inductive.

This paper describes the application of an induction technique known as INDUCT, which was based on a probability theory algorithm [13,14]. Most inductive methods used the highest inductive statistic to choose rules, i.e. they make rules based on a high rate of occurrence in the database. This meant that cases which were not well represented in the database may be ignored, and hence needed to be added later.

One of the limitations of conventional expert systems was that they were very difficult to modify. For this reason, a technique known as ripple down rules (RDRs), developed over the past few years by workers at our AI Laboratory, was selected as the implementation for the IC ES. RDRs were specifically designed to cope with the requirement of ES maintenance and several medical expert systems have been built using this system [15,16].

2. Theory

2.1. Ripple down rules

RDRs are a formalised structure of rules used for building ESs. The technique was developed as an alternative to traditional ES methods and was an attempt to solve many problems existing in the engineering and maintenance of knowledge. Traditional ESs did not arrange the rules as they were added to system [17]. This resulted in an explosion in the number of potential pathways which can use a particular rule, hence if a new rule is added or an old rule modified all these pathways need to be revalidated. In other words, it becomes very difficult to maintain an ES and to control the effects of additional knowledge. RDRs were developed to combat these problems and they employ a rigid binary tree structure such that any new rule added has a predictable effect. This format not only enhances the ease of maintenance of an expert system but also provides a method for knowledge acquisition that allows the experts to easily build their own system without the need of software support.

The RDR rules are formatted in a tree-like structure where each rule is allowed two branches, a true branch and a false branch. In order to clearly explain the process used to build an RDR tree, a simple example can be given. RDR trees are built using a selection of typical cases together with their conclusions. For this example we will use the selection of a detector for an ion chromatography method.

Table 1
Example cases with which to build an RDR tree

Sample	UV visibility	Post-column reactor	Suppressor	Detector applied for this case
Organic acid	yes	no	no	UV detection
Chloride	sample has both UV and non-UV absorbing solutes	no	yes	Conductivity
Transition metal	no	yes	no	UV detection
Iodide	yes	no	yes	Conductivity
Bromide	yes	no	no	Amperometry
Cyanide	no	no	no	Amperometry

Table 1 shows the cases used in the development of the tree, these are purely speculative and were chosen simply to illustrate the process. The first case concerns the assay of organic acids and is used to create rule 1 in Fig. 1. As this case was used to create the rule it is saved as the *cornerstone case* for that rule. This case can be used later to define the context in which rule 1 was made.

The second case in Table 1 is now presented to the RDR tree. As the case does not fall within the premise of rule 1 the system cannot provide a solution and another rule needs to be added to the system. To be consistent with a philosophy of knowledge in context the user is presented with a list of differences between the cornerstone case of rule 1 and the current case. One or more of these differences can be selected as the premise(s) of rule 2. As rule 1 evaluated to false for this case the new rule is added on the false branch of that rule. Consider case 4: the RDR tree suggests the use of UV detection for this case as rule 1 evaluates to true. However, this is an incorrect result. The user needs to add a new rule and is presented with a difference list from which to select new premises. This rule is added to the true branch of rule 1. In this way the RDR tree begins to grow.

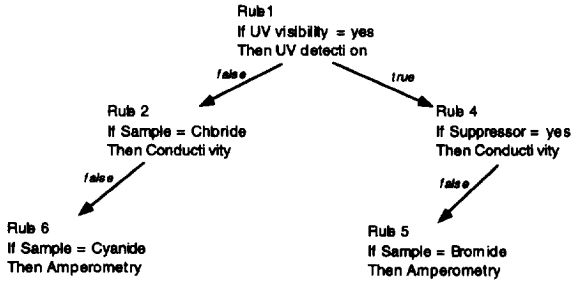

Fig. 1. An example RDR tree for the selection of an IC detector.

Unlike other ES methods, RDR trees do not distinguish between the development and maintenance stages. New rules can be added as required, and because they are added in a controlled way, there is no need to validate the system with each new addition. Each new rule is embedded within the context of the cornerstone case and is not applicable outside this context.

2.2. Induction of the rules

Three different machine learning methods were first evaluated using a reduced set of the database of examples. They were applied to the induction of rules for the selection of an IC detector [18,19]. INDUCT, developed by Gaines, was selected as the most suitable induction method for IC.

3. Application of machine learning and RDRs to the configuration of an IC method

3.1. Reducing the IC method to several classifications

Firstly, the IC method was divided into several individual classifications for the eight method conditions, shown in Table 2. The classes defined for columns and eluents were based on the divisions presented in Paul Haddad's book [1]. Table 2 shows a complete set of classifications for all the conditions except the detectors and the eluents. These attributes had larger numbers of possible values (e.g., the eluent has a total of 48 possible values).

The selection of suitable eluent classifications proved a major task. The methods detailed in the

Table 2
Method conditions and some example values for an IC method

Mechanism	Column	Mobile phase	Gradient	pH	Suppressor	Post-column reactor	Detector
Ion interaction (dynamic)	cation exchange resin	water	yes	acidic pH	yes	yes	conductivity
Ion interaction (permanent)	cation exchange silica	aliphatic acid ions	no	neutral pH	no	no	indirect conductivity
Ion exchange	anion exchange resin	inorganic ions	?	basic pH	?	?	amperometry
Ion exclusion	anion exchange silica	organic base ions		unspecified pH			potentiometry
	neutral silica	silver ions					UV–Vis spectrometry
	neutral resin	hydroxide ions					indirect UV–Vis spectrometry
	crown ether	phenolate ions					refractive index
	carbonate buffer						fluorescence
	HCl						atomic emission spectrometry

database contained descriptions of the eluent which were different for almost every method. This meant that there were over 2000 eluent descriptions; it would not have been useful to induce rules using these. Firstly, because there would be insufficient examples for each eluent and, secondly, the machine learning algorithm could not determine when eluents were similar and hence had similar properties. Thus it was decided to rationalise the eluent into 48 possible values. These values were chosen by using the breakdown of eluent types described by Haddad [1]. Several classes of eluents for each separation mechanism were chosen and these are shown in Table 3. The database was searched for examples of these classes of mobile phases and replaced with the labels shown in Table 3. The UNIX tools SED and AWK were used for this purpose.

Having decided the relevant values for each method condition it was now necessary to define which features of an application (attributes) could influence the selection of an appropriate method. It is clear that the nature of the sample, and the application of the method will play a role in a given choice, hence the following attributes were extracted for each method described in the database.

1. *Solute*: this is the name of the main solute of interest.
2. *Ion class*: the ionic nature of the sample.
3. *Halides*: this defines whether any halides are to be assayed and the nature of the halide.
4. *Sulphates*: this defines whether sulphate or sulphite ions are to be assayed.
5. *Nitrates*: this defines whether nitrate or nitrite ions are to be assayed.
6. *UV absorbance*: if the sample contains just UV absorbing ions or a mixture of both absorbing and non-absorbing ions is defined by this attribute.
7. *No. of solutes*: the number of ions which need to be separated and assayed by the method.
8. *Application*: the application domain is defined by this attribute.

These attributes were developed iteratively by applying machine learning then extending the range of attributes to better define the domain.

Having defined a range of good sample and application attributes, a decision was required as to which attributes should be used for the classifications of each method condition. There were two options available. Firstly, a fixed order for the selection of

Table 3
The classes of eluents in IC

Ion-exchange (suppressed)	Ion-exchange (non-suppressed)	Ion-exclusion	Ion-interaction
Hydrogen ions	aliphatic acid ions	water	tetrabutyl ammonium
Hydrogen and silver ions	inorganic ions	hydrochloric acid	aliphatic sulphonic acids
Silver ions	inorganic acid ions	HIBA	PIC reagents
Hydroxide ions	organic base ions	sulphuric acid	
Phenate ions	EDTA ions	weak acid	
Carbonate buffer	UV absorbing inorganic ions	nitric acid	
Amino acid ions	aromatic acid ions		
Carbonate ions	copper ions		
Borate ions	borate ions		
Phosphate ions	sulphonic acid ions		
Hydroxide ions	UV absorbing organic base ions		
Barium ions			
Lead ions			
Zinc and hydrogen ions			
Bicarbonate ions			
Nitrate ions			
EDTA			
EDDA			

Table 4
Information given to the machine learning algorithm

Mechanism	Column	Eluent
Solute	mechanism	column
Ion class	solute	mechanism
Halides	ion class	solute
Sulphates	halides	ion class
Nitrates	sulphates	halides
UV absorbance	nitrates	sulphates
No of solutes	UV absorbance	nitrates
Application	no of solutes	UV absorbance
	application	no of solutes
		application

the method conditions could be described and only attributes previously defined are used by INDUCT to classify the next method condition. For instance, if the order was specified as follows

1. mechanism,
2. column,
3. eluent,

INDUCT would then be given sample and application information on which to select the mechanism. After the mechanism was chosen, the added information on the sample, application and the mechanism could be used to define a column, and sample, application, mechanism and column could be used to select a eluent and so on until the method was fully defined. This is shown in Table 4. If the machine was forced to learn in this way, then the final expert system would be forced to use that defined order to select conditions. However, this method was the only way machine learning could currently be applied to configuration-type problems without modification. It was easy to see how much information was lost by using this technique and it certainly did not reflect the way in which a typical chromatographer performed this task.

To understand the problem with developing RDR trees in the determinate fashion described above, consider the following example. If the analyst wished to use a specific column, the ES would contain no rules on the selection of a mechanism based on this column. In other words, there was no way to express higher-order relationships between the method conditions. Couple this problem to the fact that the order of selecting various conditions could vary from sample to sample and imposing a fixed order would therefore limit the scope of the expert system.

For these reasons another approach was chosen which allowed more flexibility but required the development of a special iterative technique of evaluating the individual RDR trees. This approach required all method conditions to be used as attributes for the determination of each individual condition, thus eight RDR trees were developed, each using 15 attributes, with the INDUCT algorithm.

Table 5 shows the number of rules together with the errors observed for each of these trees. The training set initially consisted of over 4000 methods taken from the published literature over a ten-year period up to 1990. As induction could only deal with single-solute cases this was expanded to provide a separate method for each solute described, resulting in a final set of 12 000 examples. A random 10% sample was removed to use as a test set. The database was also cross-validated extensively to

Table 5
Results for the evaluation of the knowledge bases

Knowledge base	Number of rules	Errors in database (%)	Errors in test set (%)
Column	440	6.83	9.65
Mechanism	89	0.09	0.37
Detector	407	6.8	8.79
Suppressor	67	3.79	4.34
Eluent	598	11.69	16.96
Gradient	113	1.83	3.05
pH	321	8.61	10.85
Post-column	58	0.56	0.78

check for homogeneity [20]. Table 5 shows the accuracy of prediction for the overall training set together with that for the test set.

All of the trees showed higher errors for the test set and this is to be expected. The training set was used to generate the rules and thus the trees should contain knowledge on these cases.

Most of the trees showed errors of less than 10% with the exception of the selection of the mobile phase. The mobile phase was poorly defined in the original database and hence could not be expected to provide rules with low errors. This point was also demonstrated by the fact that there was not such a large difference in the errors between the training and test data, suggesting that the information in both sets was unable to produce completely satisfactory rules.

The trees were also examined to find out if they used rules which were dependent on other method conditions. Without exception all trees used rules which required other method information. Table 6 shows the number of rules using each method condition for the selection of an IC column, and Table 7 shows the same for the selection of the mobile phase. It was clear that the IC method conditions were interdependent and thus justified the decision to use interdependent trees.

3.2. Configuration of the IC expert system

Having developed a set of interdependent RDR trees, it was necessary to design a method of consulting these. Since an IC method was indeterminate by nature it was necessary to develop a tech-

Table 6
The results of the evaluation of the complete expert system

	Sample	Sample+detector	Sample+column	Sample+suppressor	Totals
% no good	9.3	18.6	11.6	14.0	15
% work	30.2	27.9	27.9	20.9	22.4
% good	14.0	7.5	20.9	16.2	16.4
% ideal	46.5	46.5	39.5	48.8	46.1
Total % work	90.7	81.4	88.4	86.0	85.0
Total % good or ideal	60.4	53.5	60.5	65.1	62.6

Table 7
The % rules using method and sample information

IC method condition	% rules for the column	% rules for the mobile phase
UV absorbance	1.6	3.5
Suppressor	3.4	5.6
Sulfate present	4.6	2.8
Solute number	7.3	7.5
Solute name	11.6	13.2
Post-column reaction	3.4	1.8
pH	7.6	8.0
Nitrate present	3.3	5.3
Mobile phase	13.0	–
Mechanism	4.5	1.1
Ion class	8.2	9.0
Halides present	6.7	7.7
Gradient	3.8	1.7
Detector	13.1	10.7
Application	8.9	15.3
Column	–	6.8
Total method information	48.1	35.7
Total sample information	51.9	64.3

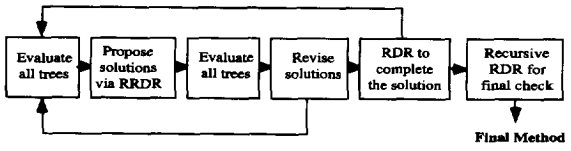

Fig. 2. The propose and revise method.

nique which was completely independent of the order of consultation of the trees.

To do this a propose and revise method was developed, all trees first were consulted and the conclusions examined and revised by further consultations. This was an iterative process that could lend itself to parallel processing. This technique was known as recursive RDR (RRDR)

The next step was to perform a standard RDR consultation on each of the trees. This had the dual effect of forcing a decision for attributes not yet decided and also checking for any remaining inconsistencies within the standard RDR philosophy. Any revisions proposed by the RDR consultation were then checked against recursive RDR, to ensure any RDR updates were consistent with each other, and if all the proposed conclusions evaluated to true then the process was complete. This process is illustrated in Fig. 2. A full explanation of the method was published elsewhere [20].

3.3. Implementation

An RDR shell was developed using the C programming language and the HyperCard program available on Apple Macintosh computers. All of the functions of the package were available to the programming language (Hyper Talk) that forms part of the application.

In the case of the IC expert system, minimal modifications were required to the basic shell to initiate development, and maintain the early stages of exploration. Once the basic algorithms had been proven, the system was provided with a simple and intuitive graphical front end, again developed in HyperCard. The system has been applied to several different-performance Macintosh machines.

4. Results

The accuracy of the overall system was evaluated by presenting 50 cases, all of which were excluded from the training set, to the ES. Seven consultations were performed for each of these 50 cases, each giving a different set of information to the ES (e.g. sometimes only the sample information was given and other starting information included the column or the separation mechanism, etc.).

The methods suggested by the ES were then presented to the expert (Haddad), who judged them to be 'no good', 'workable', 'good' or 'ideal'. The results of this evaluation are presented in Table 5.

All results were expressed as a percentage of the total number of cases studied and each column of Table 5 gives the result achieved when the column heading data was given to the system. Overall, 85% of the methods proposed would work and 62% were good or ideal. Surprisingly, the best results were observed when the minimum amount of data was given to the expert system (see the results for sample information only). This showed that the most successful rules were developed using the sample information. This was probably due to the fact that there were many conflicting cases defined by method conditions. These cases resulted from published papers that attempted new techniques which did not achieve subsequent acceptance.

5. Conclusions

In conclusion, the machine learning program, INDUCT, together with the ES development tool, RDR, were successfully applied to create an ES for IC. Several problems were encountered due to the nature of the IC domain and solutions were developed which performed well.

The main advantage of this research was that it showed that it is possible to build ESs automatically without the need for the lengthy process of knowledge engineering. Knowledge could be extracted from previous examples and gave accuracy results of up to 90% in the development of IC methods.

The next stage of the work is to evaluate the ES in a laboratory environment and independently assess the validity of the IC methods proposed.

References

[1] P.R. Haddad and P.E. Jackson, Ion Chromatography, Principles and Applications, Journal of Chromatography Library, Vol. 46, Elsevier, Amsterdam, 1990.
[2] M. Mulholland, P. Preston, C. Sammut, B. Hibbert and P. Compton, The proceedings of IEA/AIE-93 Edinburgh, June 1993.
[3] D.B. Hibbert, Chemometrics Intelligent Lab. Syst., 19 (1993) 277–293.
[4] J. Zupan and J. Gasteiger, Anal. Chim. Acta, 248(1) (1991) 1–30.
[5] D.R. Scott, Anal. Chim. Acta, 265 (1992) 43–54.
[6] D.R. Scott, Chemometrics Intelligent Lab. Syst., 12 (1991) 189–200.
[7] D.R. Scott, Anal. Chim. Acta, 211 (1988) 11–29.
[8] M.P. Derde, L. Buydens, C. Guns, D.L. Massart and P.K. Hopke, Anal. Chem., 59 (14) (1987) 1868–1871.
[9] L. Buydens, D.L. Massart and P.K. Hopke, Chemometrics Intelligent Lab. Syst., 3 (1988) 199–204.
[10] R. Appel, D. Hochstrasser, C. Roch, M. Funk, A.F. Muller and C. Pellegrini, Electrophoresis, 9(3) (1988) 136–42.
[11] H. Gelernter, J. Rose, C. Royce, Chyouhwa, J. Chem. Inf. Comput. Sci., 30(4) (1990) 492–504.
[12] P. Röse and J. Gasteiger, in J. Gasteiger (Editor), Software Dev. Chem. 4, Proc. Workshop Comput. Chem., 4th, Meeting-Date 1989, Springer, Berlin, 1990, pp. 275–288.
[13] B.R. Gaines and P. Compton, Proceedings of AAAI-92, July 12–17, 1992, San Jose, CA.
[14] B. Gaines and P. Compton, AI '92 Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, Hobart, Tasmania, World Scientific, Singapore, 1992, pp. 349–354.
[15] P. Compton and R. Jansen, Knowledge Acquisition, 2 (1990) 241–257.
[16] P. Compton, A. Srinivasan, G. Edwards, R. Malor and L. Lazarus, First World Congress on Expert Systems, Florida, December 1991.
[17] M. Mulholland, in C. Bryce (Editor), Microcomputers and their Application in Biochemistry, Oxford University Press, Oxford, 1992, Ch. 8, pp. 243–265.
[18] M. Mulholland, D.B. Hibbert, P.R. Haddad and C. Sammut, Chemometrics Intelligent Lab. Syst., 27 (1994) 95–104.
[19] M. Mulholland, D.B. Hibbert, P.R. Haddad and P. Parslov, Chemometrics Intelligent Lab. Syst., (1995) accepted for publication.
[20] M. Mulholland, P. Preston, C. Sammut, B. Hibbert and P. Compton, Presented at IEA/AIE-93 Edinburgh, June 1993, Published as a long paper in the proceedings.