ELSEVIER

Analytica Chimica Acta 348 (1997) 389–407

ANALYTICA CHIMICA ACTA

Classification of pyrolysis mass spectra by fuzzy multivariate rule induction-comparison with regression, *K*-nearest neighbour, neural and decision-tree methods

B.K. Alsberg[a,*], R. Goodacre[a], J.J. Rowland[b], D.B. Kell[a]

[a]*Institute of Biological Sciences, University of Wales, Aberystwyth, Ceredigion SY23 3DA, UK*
[b]*Department of Computer Science, University of Wales, Aberystwyth, Ceredigion SY23 3DA, UK*

Received in revised form 18 December 1996; accepted 31 December 1996

Abstract

The fuzzy multivariate rule building expert system (FuRES) is applied to solve classification problems using two pyrolysis mass spectral data sets. The first data set contains three types of milk (from cow, goat and ewe) and the second data set contains two types of olive oils (adulterated and extra virgin). The performance of FuRES is compared with a selection of well-known classification algorithms: backpropagation artificial neural networks (ANNs), canonical variates analysis (CVA), classification and regression trees (CART), the *K*-nearest neighbour method (KNN) and discriminant partial least squares (DPLS). In terms of percent correct classification the DPLS and ANNs were best since all test set objects in both data sets were correctly classified. FuRES was second best with 100% correct classification for the milk data set and 91% correct classification for the olive oil data set, while the KNN method showed 100% and 61% for the two data sets. CVA had a 100% correct classification for the milk data set, but failed to form a model for the olive oil data set. The percent correct classifications for the CART method were 92% and 74%, respectively. When both model interpretation and predictive ability are taken into consideration, we consider that the ranking of these methods on the basis of these two data sets is in order of decreasing utility: DPLS, FuRES, ANNs, CART, CVA and KNN.

Keywords: Rule induction; Canonical variate analysis; Discriminant partial least squares; PLS2; *K*-nearest neighbour method; Fuzzy rule building expert system (FuRES); Artificial neural networks; Classification and regression trees (CART); Pyrolysis mass spectrometry (PyMS)

1. Introduction

The application of pattern recognition methodology within chemistry and biology is becoming more important [1–10]. Spectroscopic and chromatographic profiles as feature vectors have been used to solve a multitude of classification problems. We are here interested in the performance of a variety of classification methods as applied to pyrolysis mass spectrometry (PyMS) profiles. PyMS is a rapid method that has been shown to be effective in producing high-resolution feature vector profiles in important biological and chemical classification problems [11–17].

Classification is important both in applied and fundamental research. From an industrial perspective,

*Corresponding author: Tel.: +44 1970 622353; fax: +44 1970 622354; e-mail: bka@aber.ac.uk.

PII S0003-2670(97)00064-0

accurate classification of spectroscopic profiles is central to e.g. quality monitoring and diagnosis. Classification is in general interpreted as partitioning of the feature space according to the individual sample memberships to the different classes. The classification methods differ in how they partition the feature space. Even though the various supervised classification methods appear to be very different they all try to find a consensus or correspondence between the inherent structures in the distribution of objects in feature space and the space of membership values assigned to each object. In this context, the membership information in a training set provides a set of *constraints* that are used to form decision surfaces of various complexity. In addition to the empirical information given in the feature and membership space, assumptions about model parsimony are used to form the final decision surface(s).

Linear methods such as the linear discriminant analysis (LDA) [18,19] attempt to find lines or hyperplanes such that the different classes occupy different positions relative to the decision planes. Non-linear methods like artificial neural networks [2,8,20,21] can create non-linear hypersurfaces with various degree of smoothness and curvature. Decision hyperplanes may also be interpreted as *rules* which for these purposes means a statement that can be written in the form IF..THEN...ELSE. In this article we are interested in methods that construct such rules directly from data by the use of induction. The language of *set theory* [22] is well suited to discuss the induction of rules and space partitions. The membership of an object in a set is defined in terms of its feature space properties. We want to create sets that contain as few different types of classes as possible, preferably just one. Assume we have two sets labelled A and B where the points in the sets can be members of two classes 1 and 2. If both sets between them contain all members of class 1 and no other members, then class 1 is the *union* of A and B, i.e. $A \cup B$. All set relationships can be translated into logical operators within a rule; for the above example: IF an object is in set A *or* in set B, THEN it belongs to class 1. If on the other hand all class 2 objects are found in the *intersection* between A *and* B, i.e. $A \cap B$, the rule would be: IF an object is in set A and in set B, THEN it belongs to class 2.

In general, the rule induction process can be divided into two major parts:

1. Building sets that contain objects based on feature space characteristics.
2. Creating logical rules on the basis of class membership and set interactions.

In the simplest case we have *univariate* rules that depend on one original variable axis only. Multivariate statistical classification methods are often more effective because they utilise linear combinations of *all* variables rather than just individual variables. Unfortunately, these methods in general lack the IF..THEN..ELSE level of representation of the final decision model as seen in rule induction. These types of decision models are very desirable because they make interpretation of the decision models much easier. To accomplish this in situations where a multivariate approach is unavoidable, *multivariate rule induction* has been developed [23–25]. This is an attempt to incorporate the best from univariate rule induction and multivariate statistics: each rule which is responsible for the partition of the feature space is multivariate and the different multivariate rules are arranged into a tree hierarchy which represents the IF..THEN..ELSE structure. In this article we compare one such multivariate rule induction method; the fuzzy rule-building expert system (FuRES) [42], with a selected set of more traditional classification methods. We investigate whether FuRES has better classification ability and whether the final FuRES classification model is easier to interpret compared with other models.

2. Experimental

Two different experimental data sets, referred to as Data set 1 and 2, are used to compare the different classification methods. Both data sets use PyMS as feature profiles.

2.1. Preparation of data set 1

Mixtures of the three milks from cow, goat and ewe were prepared which differed in their fat content as described previously [26]. The samples were then split into training (calibration) and test sets as detailed in Table 1.

Table 1
Training and test sets for calibrations based on milk type

	Percentage fat in milk mixtures		
	Cow	Goat	Ewe
Training set	0	0	0
	1.9	2.15	2.7
	3.8	4.3	5.4
Test set	0.38	0.43	0.54
	0.76	0.86	1.08
	1.14	1.29	1.62
	1.52	1.72	2.16
	2.28	2.58	3.24
	2.66	3.01	3.78
	3.04	3.44	4.32
	3.42	3.87	4.86

2.2. Preparation of data set 2

Two sets of samples of virgin and adulterated olive oils were prepared in Italy by Prof. Giorgio Bianchi (Istituto Sperimentale per la Elaiotecnica, Contrada 'Fonte Umano' no. 37, 65013 Città S. Angelo, Pescara, Italy), each consisting of 12 samples of various extra virgin olive oils plus 12 samples variously adulterated with 5–50% of Soya, sunflower, peanut, corn or rectified olive oils. Full details of oil collection have been given previously [27,28]. Details of the training and test sets are shown in Table 2.

2.3. Pyrolysis mass spectrometry (PyMS)

2.5 μl of the milks (data set 1) and the oils (data set 2) were evenly applied onto iron–nickel foils to give a thin uniform surface coating. Prior to pyrolysis the samples were oven-dried at 50°C for 30 min. Each sample was analysed in triplicate. The pyrolysis mass spectrometer used was the Horizon Instruments PYMS-200X (Horizon Instruments); for full operational procedures see [13,14,29]. For analysis of the milks the sample tube carrying the foil was heated, prior to pyrolysis, at 100°C for 5 s; for the olive oil experiment the tube heater was left off so as to minimise volatilisation of lower molecular weight oil. Curie-point pyrolysis was at 530°C for 3 s, with a temperature rise time of 0.5 s. The data from PyMS were collected over the *m/z* range 51–200 and may be displayed as quantitative pyrolysis mass spectra (e.g.

Table 2
Training and tests sets for calibrations based on virginity or adulteration of olive oils

	Number	Code name	Satus
Training set	1	Lucia	virgin
	2	Alfomosa	virgin
	3	Mario	virgin
	4	Leonardo	virgin
	5	Mara	virgin
	6	Gabriella	virgin
	7	Ugo	virgin
	8	Giorgio	virgin
	9	Walter	virgin
	10	Ezilde	virgin
	11	Vanda	virgin
	12	Catia	virgin
	13	Pietro	adulterated
	14	Rosa	adulterated
	15	Sandra	adulterated
	16	Giuseppe	adulterated
	17	Mira	adulterated
	18	Anna	adulterated
	19	Augusto	adulterated
	20	Luca	adulterated
	21	Giulia	adulterated
	22	Paola	adulterated
	23	Claudia	adulterated
	24	Patrizia	adulterated
Test set	25	Perugia	virgin
	26	Lecce	virgin
	27	Urbino	virgin
	28	Rimini	adulterated
	29	Taormina	adulterated
	30	Napoli	virgin
	31	Milano	virgin
	32	Trieste	virgin
	33	Torino	adulterated
	34	Cagliari	virgin
	35	Bolzamo	virgin
	36	Venezia	adulterated
	37	Roma	adulterated
	38	Genova	virgin
	39	Bari	virgin
	40	Pescara	adulterated
	41	Padova	adulterated
	42	Palermo	adulterated
	43	Firenze	virgin
	44	Ancona	virgin
	45	Siena	adulterated
	46	Messina	adulterated
	47	Bologna	adulterated

as in Figs. 1–3). The abscissa represents the *m/z* ratio whilst the ordinate contains information on the ion count for any particular *m/z* value ranging from 51 to 200. Data were normalised as a percentage of total ion count to remove the influence of sample size per se.

2.3.1. Characteristicity

Characteristicity is closely related to the Fisher (F) ratio [15] and has been used to select relevant masses in PyMS spectra for multivariate analysis [30]:

$$F = \frac{M_1}{M_2}, \tag{1}$$

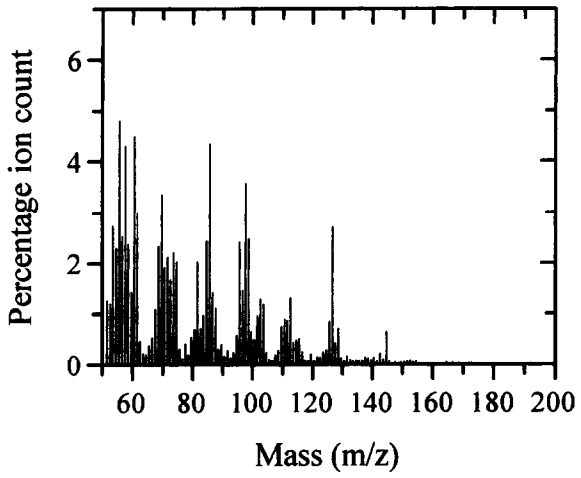

Fig. 1. Pyrolysis mass spectrum of pure cows' milk.

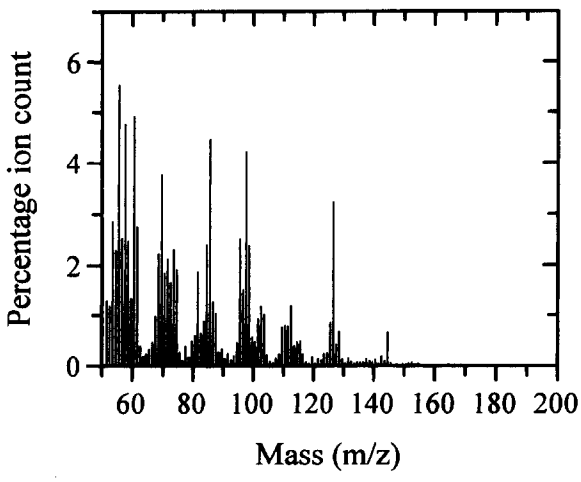

Fig. 2. Pyrolysis mass spectrum of pure goats' milk.

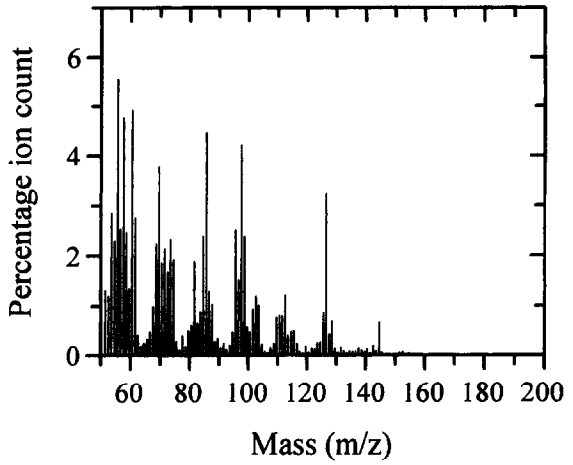

Fig. 3. Pyrolysis mass spectrum of pure ewes' milk.

where $M_1=B/(k-1)$ and $M_2=W/(n-k)$ and $B=T-W$. n is the total number of objects and k is the number of classes. M_1 and M_2 are related to the between and within class variance, respectively. The general expressions for T and W are presented in Eq. (15).

Characteristicity is calculated as described by Eshius et al. [31]. The first stage is the calculation of the following expressions:

(1) inner variance or reproducibility (r_i) [32]:

$$r_i = \frac{1}{k-1}\left[\sum_{j=1}^{k} \nu_{(i,j)}\right], \tag{2}$$

where n is the number of classes and $\nu_{(i,j)}$ the variance of peak i in class j

(2) outer variance or specificity (s_i) [33]:

$$s_i = \frac{1}{k}\left[\sum_{j=1}^{k} \left(m_{(i,j)} - m_i\right)^2\right], \tag{3}$$

where k is the number of classes, $m_{(i,j)}$ the mean of peak i in class j and m_i is the mean of $m_{(i,j)}$ (i.e. mean for all classes of peak i).The 'characteristicity' (c_i) is then calculated by the following [31]:

$$c_i = \frac{s_i}{r_i}. \tag{4}$$

The mass intensities can then be ranked in order of their characterisities; large values are more important, smaller ones less so.

3. Description of classification methods

The two data sets have been analysed using the following six classification methods:

- Classification and regression trees (CART)
- Fuzzy multivariate rule building expert system (FuRES)
- Canonical variates analysis (CVA)
- Artificial neural networks with backpropagation (ANNs)
- *K*-nearest neighbour method (KNN)
- Discriminant partial least squares (DPLS)

Unless otherwise specified, all program implementations of the methods described run under the Microsoft Windows NT operating system on IBM compatible personal computers.

Below we give a brief introduction to the different methods; since rule induction is a central part of the FuRES algorithm, a more general introduction to the subject is presented.

3.1. Rule induction (CART and FuRES)

Rule induction is a way of partitioning the space of sample objects into regions of single class memberships [23]. Let each of the N objects be members of the set X and let the set $C=\{1,2,\ldots,J\}$ contain the identification indices to each of the J different classes in the data set. A classification problem can in general be regarded as finding a *mapping* M from the input space into the space of classes:

$$M: X \rightarrow C.$$

This mapping is also referred to as a *rule* which is derived from the training set. In rule induction the data set is *recursively split* into smaller subsets where each subset contains objects belonging to as few different classes as possible. The 'purity' of a subset, i.e. the distribution among the classes of the objects within the set, can be measured by using the concept of *entropy* [34]. For each subset there is a set of fractions $P=[p_1, p_2,\ldots, p_J]$, of the objects belonging to the J different classes. The fraction p_i is computed as $p_i = n_i^{(s)}/n$ where $n_i^{(s)}$ is the number of objects belonging to class i in subset s and n is the total number of objects in subset s. We can also interpret such fractions as the *probability* of finding an object belonging to class i in the subset.

Intuitively, a subset consisting of objects from one class only will have the highest possible 'purity' and the vector P of probabilities will have a structure $P_{\min}=[0, 0,\ldots,1,\ldots,0]$. The most impure vector P will correspond to the case where there are equal fractions of each class, i.e.

$$P_{\max} = \left[\frac{1}{J},\ldots,\frac{1}{J}\right]. \tag{6}$$

The entropy of P

$$H(P) = -\sum_{i=1}^{J} P_i \log(p_i) \tag{7}$$

has properties in accordance with our intuitive understanding of 'impurity': $H_{\min}(P)=0$ and $H_{\max}(P)=\log_2(J)$ when $p_i=1/J$. Thus, ensuring the highest purity in a subset corresponds to *minimising* $H(P)$ by selecting an optimal partitioning. To see the various strategies for finding the best split we shall distinguish between *univariate* and *multivariate* rule induction. In the univariate case we want to find the single variable x_i at each recursion step that gives rise to the purest subsets (i.e. those that have minimum entropy). In this article we will look only at numerical variables, but categorical variables can also be used [25]. In univariate rule induction a split or a partitioning of the input feature space can be formulated as a question like 'Is $x_i < c$?' where c is some value chosen from the *finite set* of values variable x_i has among the N calibration objects. All the objects that satisfy the question are grouped into one subset and those that do not into another. Let a_k be the different outcomes of a test on variable x_i. For the numerical tests discussed here we have only two outcomes (a_1='yes' and a_2='no'). The entropy in a given subset of objects will thus be:

$$H(C|a_k) = -\sum_{i=1}^{J} p(c_i|a_k)\log(p(c_i|a_k)), \tag{8}$$

where $p(c_i|a_k)$ is the probability or fraction of the objects satisfying the outcome a_k and belonging to class i. $H(C|a_k)$ is read as the entropy of all the classes in C given the outcome a_k for variable x_i. Often $p(c_i|a_k)$ is computed as the number of objects belonging to class i that has outcome a_k divided by the

total number of objects satisfying a_k in the subset. Assuming that the number of outcomes is two, we get two entropies calculated for each variable tested: $H(C|\text{'yes'})$ and $H(C|\text{'no'})$. A measurement of the total impurity (entropy) for the variable x_i selected in the split will be related to the sum of the two individual entropies $H(C|\text{'yes'})$ and $H(C|\text{'no'})$. We cannot, however, use a simple addition of entropies because the two subsets usually contain different number of objects and are therefore not directly comparable. To incorporate this, we multiply each entropy with the fraction of objects ($p(a_i)$) that is present in the current subset, relative to the previous subset. If we started out with 100 objects that are split into one subset containing 30 objects (which answered 'yes' to the split question) and a second containing 70 (which answered 'no' to the split question), we multiply the corresponding entropies with $p(a_1=\text{'yes'})=30/100$ and $p(a_2=\text{'no'})=70/100$, respectively. In general we have:

$$H(C|A) = \sum_{j=1}^{m} p(a_j)H(C|a_j), \tag{9}$$

where we have $m=2$ for analyses discussed in this article. The symbol A means the set of possible outcomes to a decision question. The general algorithmic structure for univariate rule induction programs is thus:

Do the following recursively for each subset until you have reached terminal subset (i.e. a subset either containing one object or one class only):

begin

  For each variable **do**:
  begin

    Find the split question
    split the data into *m* groups
    compute the weighted entropies for the *m* groups and generate $H(C|A)$

  end
  Select the variable with the lowest entropy and generate two subsets.

end;

In realistic situations the number of rules in the a tree can get very large unless some kind of *pruning* of the tree is performed [25]. This means that the splitting of the feature space to generate higher purity of the subsets is stopped when a less than optimal purity is reached.

CART (classification and regression trees) [23] is an example of a univariate rule induction method which we have chosen to apply to our two data sets. We use here the SCAN program [35] implementation of the CART algorithm.

In *multivariate rule induction* we find a partition of the input feature space that depends on a *linear combination* of all the variables instead of just using one variable. We can formulate this as a question like: 'Is $\sum_{j=1}^{n} w_j x_j \leq c$?'. This type of partitioning of the data space is particularly useful if there are any collinearities between the variables. Our aim now is not to find the single best variable, but to find a *vector* $\mathbf{w}$ which best separates the data set into pure subsets. Given such a vector we project the object vectors stored as rows in the matrix $\mathbf{X}$ onto it:

$$\mathbf{t} = \mathbf{X}\mathbf{w}/(\mathbf{w}^{\mathrm{T}}\mathbf{w}) - b, \tag{10}$$

where b is a bias (scalar) and $\mathbf{t}$ is a score vector where the sign of each score determines whether that object is to be classified as class 1 ('yes') or 2 ('no'). This is similar to the paradigm used in the multivariate rule building expert system (MuRES) approach by Harrington [24], the oblique rule induction by Murthy et al. [36] and the linear machine decision trees by Utgoff and coworkers [37–39]. The FuRES approach is a modification of the MuRES algorithm where the central change is in how the $p(c_j|a_k)$ probability is calculated. As can be seen from the projections onto $\mathbf{w}$, it may be difficult to say on which side of the hyperplane an object really is. When an object can be on one side (membership value=1) or the other (membership value=0) of the hyperplane only, we refer to this as *crisp classification*. To describe intermediate positions relative to the hyperplane we use *fuzzy set theory* [40] where the degree of position is described by a continuous value between 0 and 1. This means that an object may exist on both sides of the hyperplane because the plane itself is fuzzy. To obtain a 'fuzzyness' in the object position relative to $\mathbf{w}$, a *logistic function* on the score value t_i

can be used [41,42]:

$$h = \frac{1}{1 + \exp(-t_i/T)}, \tag{11}$$

$$h^c = \frac{1}{1 + \exp(t_i/T)}, \tag{12}$$

where T is defined to be a *temperature* which is similar to that used in simulated annealing [43] and certain neural networks. When $T \rightarrow 0$ the logistic function will approach crisp classification and when $T \rightarrow \infty$ we approach the maximum entropy result, i.e. no partition.

When we take into consideration several rules, this will correspond to decision hyperplanes that intersect and increasingly isolate the region in feature space containing a homogenous class distribution of objects. This means that when we test for the position of an object in relation to K rules, the result is not determined just in terms of which side of a single decision plane the object is located, but in terms of the object position relative to all hyperplanes taken together. Therefore, instead of referring to a particular *side* with respect to decision plane $\boldsymbol{w}$, it is better to talk about the *interior* and *exterior* that describe the position of an object with respect to all the decision planes. Assume we have a matrix $\boldsymbol{H}$ where each row i corresponds to an object and each column j corresponds to the logistic function applied to the score of Rule j. Let $\boldsymbol{H}_k$ be the kth row vector in $\boldsymbol{H}$. All the $\boldsymbol{H}_k$ row vectors that correspond to objects belonging to class u are collected in the matrix $\boldsymbol{H}_k^{(u)}$. We can now use this matrix in the definition of the probabilities $p(c_j|a_k)$ which have to be defined in terms of the ratio of the total fuzzy membership of objects belonging to a certain class *inside the region of interest* to the total fuzzy membership of objects in the region. In CART we saw that only the number of objects in the subset belonging to class i divided by the total number of objects in the subset was used to calculate $p(c_j|a_k)$. Here, the a_k now refers to either 'inside' or 'outside' of a region and we thus calculate these conditional probabilities as:

$$p(c_i|\text{'inside'}) = \frac{\sum_{k=1}^{n_i} \min(\boldsymbol{H}_k^{(i)})}{\sum_{k=1}^{N} \min(\boldsymbol{H}_k)}, \tag{13}$$

where n_i is the number of objects in class i and N is the total number of objects.

For the outside of the region we have:

$$p(c_i|\text{'outside'}) = \frac{\sum_{k=1}^{n_i} \max(\boldsymbol{H}_k^{(i)})}{\sum_{k=1}^{N} \max(\boldsymbol{H}_k)}. \tag{14}$$

To retain consistency with fuzzy set theory, the interior of a region is defined to be the minimum logistic values and the outside as the maximum value.

The use of the logistic function makes FuRES share some similarities with the feedforward multilayer perceptron architecture commonly associated with artificial neural networks. In fact, the FuRES classifier has been referred to as a minimal neural network (MNN) [41]. It should be noted, however, that the current version of the FuRES algorithm produces *linear* decision planes only whereas ANNs can give rise to non-linear decision surfaces which can solve more difficult classification problems. It is, however, possible to approximate non-linear decision surfaces with FuRES by applying the linear planes in a sequential manner.

For multivariate rule induction we used the FuRES program provided by Prof. Peter de B. Harrington. This program runs under MS DOS 6.2.

3.2. Canonical variates analysis

Canonical variates analysis (CVA) (also referred to as Discriminant function analysis (DFA)) is a multivariate statistical technique that separates objects (samples) into groups or classes by minimising the within-class variance and maximising the between-class variance [16,30,44].

The general principle of CVA is similar to that of principal components analysis (PCA), but the objective of CVA is to find latent variables that maximise the ratio of the between-class to within-class variance, rather than maximising the between-object variance. To find the canonical variates (CV) direction we first compute the within-sample matrix of sums of squares and cross products, **W**, and the total sample matrix of sums of squares and cross products, **T**. The between-class matrix is computed as: $\mathbf{B}=\mathbf{T}-\mathbf{W}$ and the eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$ correspond to the CVs onto which we project our data objects. The matrices **W** and **T** can be expressed in matrix algebra

as follows:

$$\mathbf{T} = \sum(\mathbf{X}_j - \mathbf{M})^{\mathrm{T}}(\mathbf{X}_j - \mathbf{M}),$$
$$\mathbf{W} = \sum(\mathbf{X}_j - \mathbf{M})^{\mathrm{T}}(\mathbf{X}_j - \mathbf{M}_j), \quad (15)$$

where $\mathbf{M} = \mathbf{1}\bar{\mathbf{x}}^{\mathrm{T}}$ ($\mathbf{1}$ is a vector of ones and $\bar{\mathbf{x}}^{\mathrm{T}}$ is the transposed total mean vector). $\mathbf{M}_j = 1\bar{\mathbf{x}}_j^{\mathrm{T}}$ ($\bar{\mathbf{x}}_j^{\mathrm{T}}$ is the transposed mean vector for each class j).

In the CVA reported here, PCA is first used to reduce the dimensionality of the data where only those principal components (PCs) whose eigenvalues accounted for more than 0.1% of the total variance are used. After the first few PCs, the axes generated will usually be due to random 'noise' in the data; these PCs can be ignored without significantly reducing the amount of useful information representing the data, since each PC is now independent of (uncorrelated with) any other PC.

CVA was here carried out using the program GENSTAT [45] running under MS DOS 6.2.

3.3. Artificial neural networks (ANNs)

For excellent introductory texts to ANNs see [2,8,20,21,46–48].

All ANN analyses were carried out with a user-friendly neural network simulation program, NeuDesk version 2.1 (Neural Computer Sciences, Totton, Southampton, Hants), which runs under Microsoft Windows NT on an IBM-compatible PC.

In-depth descriptions of the modus operandi of this type of ANN analysis are given elsewhere [13,14,29].

The algorithm used was standard backpropagation in a fully-interconnected multilayer perceptron architecture [46]. This employs processing nodes (neurones or units) linked by abstract interconnections (connections or synapses). Connections each have an associated real value, termed the weight, that scaled signals passing through them. Nodes summed the signals feeding to them and output this sum to each driven connection scaled by a logistic function.

For training ANNs for the analysis of the two data sets, each of the inputs was the normalised triplicate pyrolysis mass spectrum derived from the training set and was paired with each of the desired outputs. In data set 1 these were binary encoded such that cows' milk was coded as [1,0,0], goats' milk as [0,1,0], and ewes' milk as [0,0,1]; details of training and test set are given in Table 1. These 27 training pairs collectively made up the training set. In data set 2 a single output node was used and coded such that a virgin olive oil was referred to as 1 and an adulterated olive oil was referred to as 0. The input was applied to the network, which was allowed to run until an output was produced at each output node. The differences between the actual and the desired output, taken over the entire training set, were fed back through the network in the reverse direction to signal flow (hence backpropagation) modifying the weights as they want. This process was repeated until an acceptable level of error was achieved.

3.4. K-nearest neighbour method

KNN [49–51] is a non-parametric classification method that does not explicitly form a separate model from the calibration data set. In KNN the 'classification model' is the whole of the calibration set. The classification is performed as follows: a test set object is placed in the same multidimensional hyperspace as the calibration set. The K nearest neighbours from the new test set object to the calibration objects are computed. By 'nearest' we mean the minimum distance using a chosen norm. In all the applications discussed here we use the standard Euclidean norm. In the general case, other norms like the Mahalanobis or the Manhattan distances may be more suitable [30].

In the simplest KNN method, which will be used here, the fractions of the different classes among the K neighbours are recorded. The class prediction for the new object is the class that has the largest number of objects among the K neighbours. Note that K is always an odd number to ensure that a majority vote is obtained locally.

The next problem is how to determine the optimal number of nearest neighbours. In this article we have solved this problem by performing a cross validation on the calibration set [52]. The cross validation is started by assuming $K=1$. Each object in the calibration set is then taken out and interpreted as a test set object and a classification is based on the scheme described above for the that single object. The classification for all the objects in the calibration set with K nearest neighbours is recorded and the percentage misclassification is calculated. This procedure is performed for all possible values of $K=\{1,2,\ldots,N-1\}$

where N is the number of objects in the calibration set. The K is selected that has the lowest percentage misclassification.

A MATLAB 4.2 (The MathWorks, USA) program was written to implement the KNN algorithm with cross validation.

3.5. Discriminant PLS (DPLS)

The theory and properties of the partial least squares algorithms PLS1 (with one dependent (Y) variable) and PLS2 (with several dependent Y-variables) have been extensively studied and reported in the literature [53–67]. We will therefore give only a short description of the DPLS method which is the PLS2 algorithm applied to classification problems. The central point in the PLS paradigm is to find latent variables in the feature space which have a maximum covariance with the Y-variable(s). Thus, linear combinations of the feature space variables are found that are tilted to have maximum prediction ability for the Y-variable(s). In PLS2 one also uses linear combinations of the Y-space variables. PLS2 therefore has an iterative stage in each of the PLS2 factor calculations. One common way to use PLS2 in classification problems is to introduce a *coding* where each column in the $\mathbf{Y}$ matrix (which contains all the dependent variables) corresponds to a class. If e.g. three classes need to be coded, each sample is associated with one of the three following vectors: [1,0,0] (class no. 1), [0,1,0] (class no. 2) and [0,0,1] (class no. 3). This is similar to the crisp coding described above. A fuzzy coding scheme can of course be applied if necessary because PLS is inherently quantitative.

The final PLS can be formulated as a regression equation:

$$\mathbf{Y} = \mathbf{XB}, \tag{16}$$

where the estimated regression coefficients $\mathbf{B}$ are:

$$\mathbf{B} = \mathbf{X}^{+}\mathbf{Y}, \tag{17}$$

$\mathbf{X}^{+}$ is a generalised inverse provided by the PLS algorithm. In order to obtain a prediction from the PLS model it is sufficient to use Eq. (17). In this article we compute the regression matrix $\mathbf{B}$ as demonstrated by Martens and Naes [68]:

$$\mathbf{B} = \mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{Q}^{\mathrm{T}}, \tag{18}$$

where $\mathbf{W}$ is the matrix of weights of the X space, $\mathbf{Q}$ is the weights matrix for the Y space and $\mathbf{P}$ is the X space loadings matrix.

The prediction of dependent variables on a new set of objects is now done by:

$$\mathbf{Y}_{\mathrm{test}} = \mathbf{X}_{\mathrm{test}}\mathbf{B}. \tag{19}$$

Since each of the predicted rows in $\mathbf{Y}_{\mathrm{test}}$ usually will not have the form [0,0,...,1,...,0,0], a re-coding is performed by finding the index of the maximum column in $\mathbf{Y}_{\mathrm{test}}$. This index is used as the identity of the predicted class.

4. Results and discussion

4.1. Data set 1 – Classification of three milk types

4.1.1. CVA

PCA was first applied to the data set and the first two score vectors are shown in Fig. 4. No clear separation of the three milk types is apparent in this plot.

A dimension reduction using PCA was employed where nine PCs were extracted and the resulting score vectors were subsequently used as inputs to the CVA algorithm. CVA was very effective in discriminating the objects for this data set. The results for the calibration objects together with the test set is shown

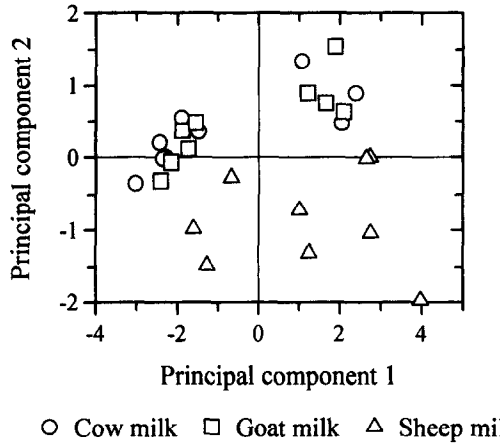

Fig. 4. PCA biplot based on PyMS data analysed by GENSTAT showing the relationship between the three types of milk with varying fat content. The first and second PCs accounted for 76.7% and 10.8% (87.5% total) of the total variance, respectively.

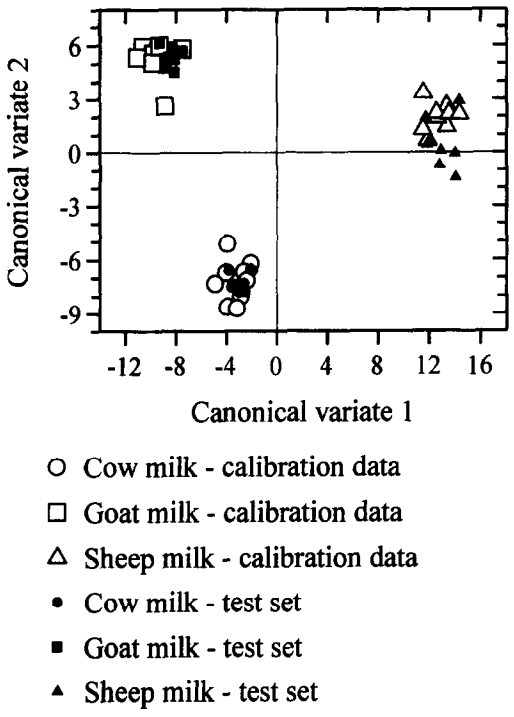

Fig. 5. CVA biplot based on PyMS data analysed by GENSTAT showing the relationship between the three types of milk. CVA was given the *a priori* information (training set; open large symbols) according to the milk type in Table 1 and trained with the first nine PCs (accounting for 99.73% variance); the replicates are shown. The test set (closed small symbols) are projected into the CV space; the CV means for the replicates are shown.

in Fig. 5. The test set objects were first projected onto the nine principal components of the calibration set to obtain comparable score vectors and further projected onto the final CVA model. The projection of calibration and test set objects together is shown in Fig. 5 where it can be seen that all samples are well separated and correctly identified. The mass loading biplot shown in Fig. 6 presents the original variables (masses) together with the three milk type classes. From this we can see, for instance, that mass 97 is important for the GOAT milks, mass 84 is important for the EWE milks and mass 98 is important for the COW milks.

4.1.2. KNN

Autoscaling on both calibration and test set was performed. The cross validation analysis of the calibration set showed that an autoscaling would improve the classification. The optimal number of nearest neighbours (K) was estimated on the calibration set only using the full cross validation approach described above. Here the optimal value is $K=1$. The predictions of the class memberships of the objects in the validation set were perfect. When it comes to interpretation of the model, KNN unfortunately gives no information about which variables would be important for prediction of a certain class.

4.1.3. CART

CART produced a rule set which is shown in Fig. 7 as a classification tree. Written out the rule is: IF mass 81 is less than 1.70 THEN the sample object is EWE milk ELSE IF mass 97 is less than 3.93 THEN the sample object is COW milk, ELSE the sample object is GOAT milk. This result is in agreement with the ranking based on the characteristicities shown in Table 4 where EWE vs. COW and GOAT has *m/z* value 81 as rank 1, and GOAT vs. COW and EWE has *m/z* 97 as rank 1. Using this rule on the test set we found that six objects (objects 10,13,14,45,46 and 47) out of 72 (8.3%) were misclassified. The most important variable in the CART rule is in agreement with CVA where mass 97 was found to be important for the discrimination between GOAT and (COW and EWE) milks.

4.1.4. ANN

The structure of the ANN used in this study to analyse pyrolysis mass spectra was as follows: 150 input nodes, 3 output nodes (one for each milk type), and one 'hidden' layer containing 8 nodes (i.e., a 150-8-3 architecture). Before training commenced, the values applied to the input and output nodes were normalised between 0 and 1 for each mass, and the connection weights were set to small random values [47]. Each epoch represented 1235 connection weight updatings and a recalculation of the root mean squared (RMS) error between the true and desired outputs over the entire training set. A plot of the RMS error vs. the number of epochs represents the 'learning curve', and was used to estimate the extent of training (not shown). Finally, after training to an RMS error of 0.001, all 72 test set pyrolysis mass spectra were used to test that the ANN had learnt the three difference classes of milk; the network then calculated its estimate and for each test set sample and the winning node

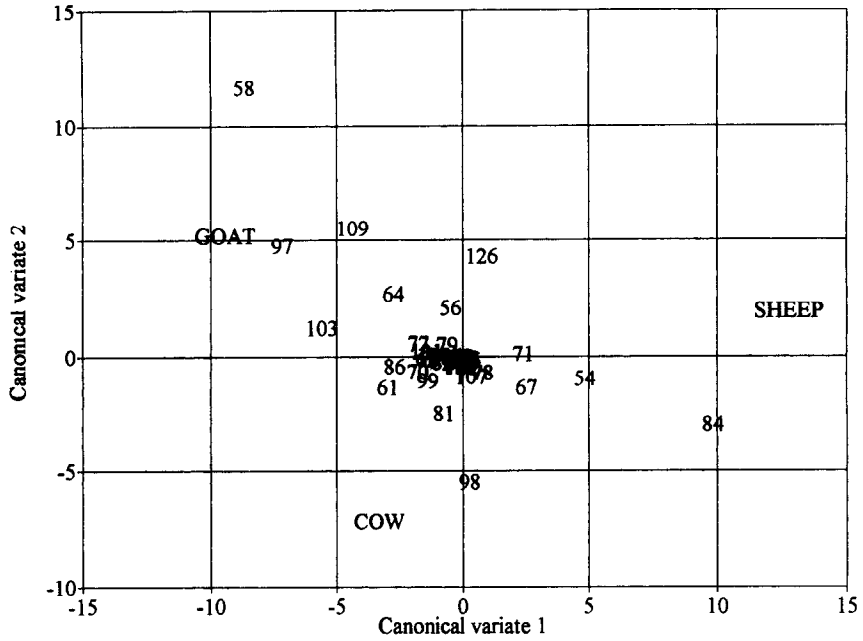

Fig. 6. CVA scores and mass loadings biplot based on PyMS data analysed by GENSTAT. The relationship between the three types of milk are as those presented in Fig. 3. The 50 most significant masses as judged by their characteristicities are shown.

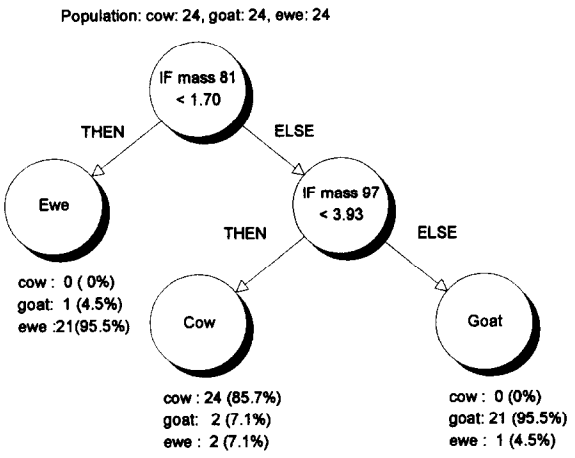

Fig. 7. CART analysis of Data set 1. This is the resulting classification tree which has a percent misclassification on the test set of 8.3% (6 out of 72).

in the output layer was taken as its identity. All 72 were correctly identified (and see Goodacre et al., in preparation).

In an attempt to observe which masses were important in discriminating the three milk types the next stage was to 'turn-on' each input node in turn; this was achieved by interrogating the trained ANN with the 150 by 150 identity matrix, the influence of each mass could be ascertained in terms of which of the three nodes in the output layer 'won', see Fig. 8. One should be careful when interpreting plots like these. The ten highest peaks for the cow milk were 61, 70, 62, 81, 74, 66, 103, 75, 196 and 63. For the goat milk the ten most significant peaks are 97, 64, 109, 69, 125, 51, 96, 127, 123 and 52. For the ewe milk the following masses were found to be important: 84, 60, 112, 54, 142, 71, 111, 55, 106 and 56. Similar to the other methods, the ANNs have found masses 81,84 and 97 to be important. The results are also in agreement with the results from the FuRES analysis (see below) where the following masses were found to be important: 55, 61, 69, 84, 103 and 112.

4.1.5. DPLS

The percent misclassification for each PLS2 factor for the calibration and test set has a minimum of zero for 3 factors. Note that the misclassification error is computed by finding the number of correct classifications relative to the total number of objects. The

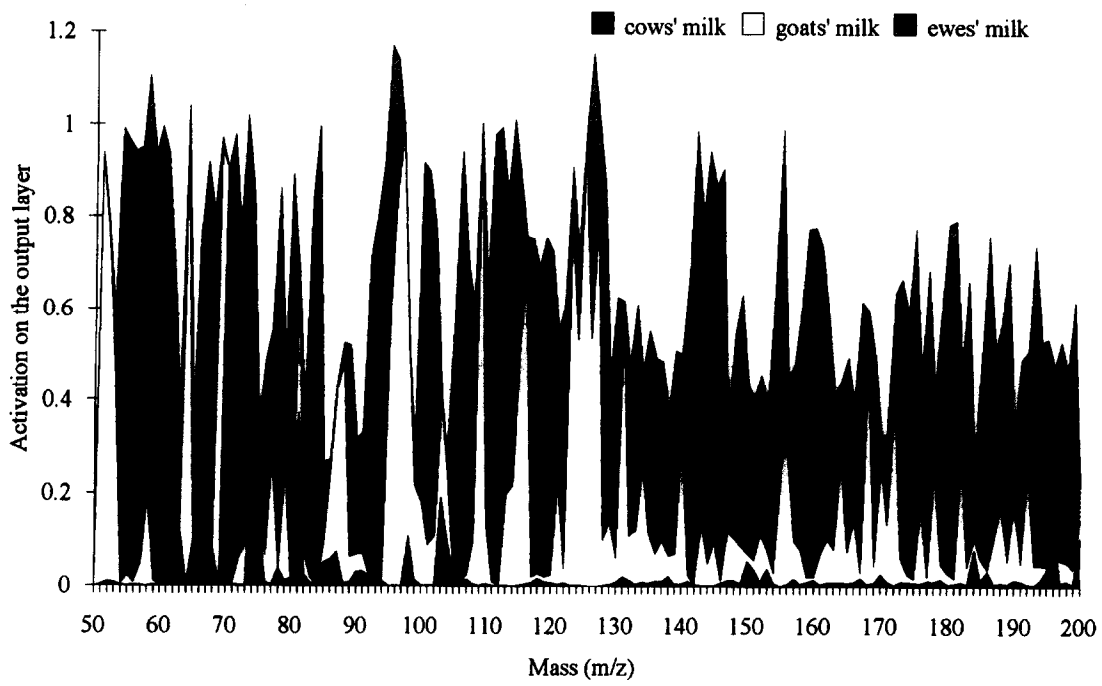

Fig. 8. Results of the estimates of trained 150-8-3 ANNs for the identity matrix. Three ANNs were interrogated after the RMS error was 0.001, this took between 400–800 epochs. The results shown are the averages of the three ANNs.

predictions were calculated by multiplying each sample vector with the regression matrix **B**. Each of the three column vectors in **B** are shown in Fig. 9. A 100% correct classification of the test set was achieved. The ten most significant masses in the first regression vector are (sorted in descending order of importance): 126, 97, 61, 85, 81, 144, 73, 96, 98, 95. The second regression vector has the following ten most important masses: 97, 126, 69, 61, 51, 53, 96, 95, 55, 60 and the third has: 55, 69, 97, 84, 81, 126, 53, 85, 112, 51. We note in particular the masses 97, 126, 84 and 61 which were also found to be important using CVA, FuRES (see below) and ANNs.

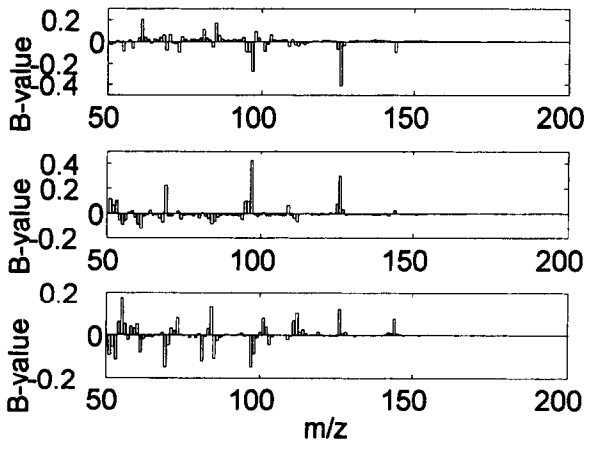

Fig. 9. The three regression coefficient vectors (in the **B** matrix) used in the PLS2 prediction for data set 1.

4.1.6. FuRES

The FuRES algorithm produced two rules for this data set, see the decision tree in Fig. 11. The two vectors corresponding to the two rules are shown in Fig. 10. FuRES accomplished 100% correct predictions on the test data set. The first multivariate rule is illustrated by its $\boldsymbol{w}$ vector (as in Eq. (10)) which defines the decision hyperplane, see the upper part of Fig. 10. This rule is responsible for separating COW's milk from GOAT and EWE's milks. This fact can be read from the decision tree in Fig. 11. Similarly, the second rule is responsible for discriminating between GOAT and EWE milk. The following ten most important masses in Rule 1 are (sorted with

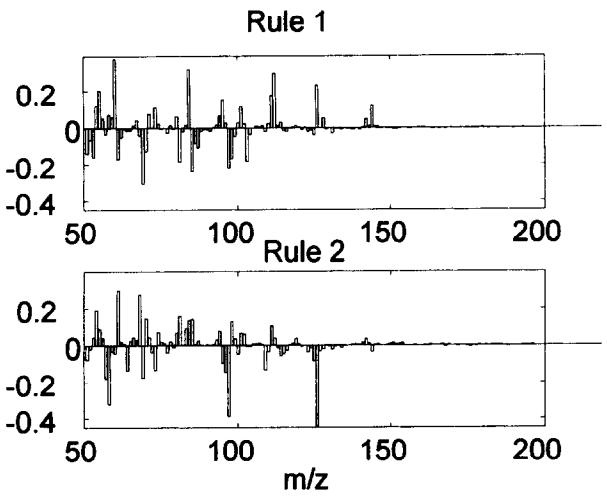

Fig. 10. Each of the two FuRES rules for data set 1 are shown. Note that these two rules can be interpreted as a kind of PyMS spectra. High amplitude means that a certain *m/z* is important.

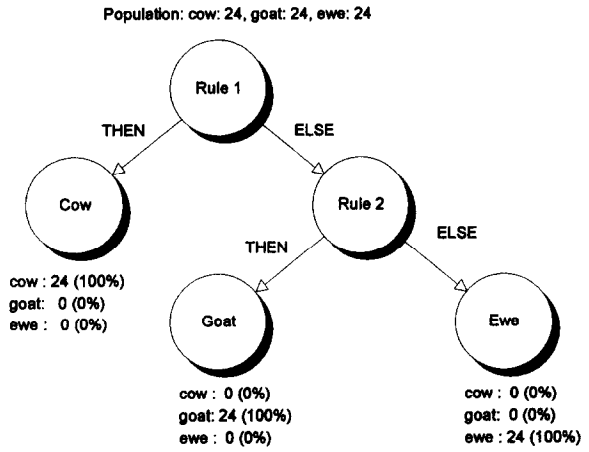

Fig. 11. The FuRES rule induction tree for data set 1. Each rule here is a vector onto which each sample is projected. The positions of the object relative to such a vector (on the positive or negative side) determine which class it belongs.

respect to absolute height of peak, in descending order): 60, 84, 69, 112, 85, 126, 97, 55, 81, and 103. The ten most dominating masses for Rule 2 are 126, 97, 58, 61, 68, 54, 57, 69, 81, and 96. It is of interest to compare these important masses with the mass-ranking shown in Table 3 (based on single spectra subtractions) and Table 4 (based on the calculations of characteristicities). The ten largest FuRES

Table 3
m/z values ranked in order based on subtraction spectra

Rank	Cow[1]	Goat[2]	Ewe[3]	Rank	Cow	Goat	Ewe
1	**61**	**97**	**55**	26	128	104	199
2	98	**69**	**60**	27	124	88	195
3	**81**	**126**	**57**	28	131	168	183
4	**103**	51	**84**	29	107	77	186
5	**85**	**96**	**54**	30	137	155	193
6	86	53	**126**	31	115	134	187
7	87	95	56	32	88	176	181
8	70	125	**68**	33	93	172	200
9	99	109	**112**	34	102	192	196
10	110	**81**	73	35	138	188	155
11	74	52	**58**	36	123	156	173
12	82	98	144	37	63	182	189
13	94	127	101	38	121	190	188
14	62	**103**	111	39	78	198	191
15	80	64	71	40	90	178	184
16	67	123	59	41	100	170	159
17	77	87	72	42	116	200	160
18	83	115	142	43	140	197	180
19	104	70	119	44	122	164	174
20	91	79	83	45	92	187	172
21	79	86	114	46	129	189	169
22	89	110	145	47	136	184	143
23	108	124	146	48	139	173	179
24	75	116	197	49	152	191	158
25	65	131	198	50	151	199	194

[1] Calculated on training set; (average of cow spectra) – (average of goat and ewe spectra).
[2] Calculated on training set; (average of goat spectra) – (average of cow and ewe spectra).
[3] Calculated on training set; (average of ewe spectra) – (average of cow and goat spectra).
Bold indicates important masses given by the FuRES method.

masses are indicated in the tables as boldface. We see that in Table 3 almost all of the top five ranked masses are among the ten important masses found by the FuRES method. In contrast, several of the important FuRES masses have low ranking in Table 4, which suggests that the ranking based on calculation of characteristicities is here less effective than the simple straightforward subtraction of spectra.

4.2. Data set 2 – Adulteration of olive oils

4.2.1. CVA

Even though the CVA analysis indicated the presence of groups they did not correspond to adulterated and virgin olive oils and thus the CVA method failed to

Table 4
m/z values ranked in order based on their characteristicities. Characteristicities calculated as given in the text

Rank	All against one another	Cow vs. Goat and Ewe	Goat vs. Cow and Ewe	Ewe vs. Cow and Goat
1	62	**126**	**97**	**81**
2	**81**	62	64	**84**
3	**84**	**61**	**84**	98
4	**97**	63	109	62
5	98	65	51	70
6	70	200	78	71
7	**126**	70	95	**103**
8	**61**	196	**96**	**54**
9	63	92	67	63
10	65	78	69	56
11	200	184	71	79
12	64	75	126	77
13	71	190	125	104
14	**103**	98	142	65
15	190	**81**	112	64
16	77	67	94	190
17	184	186	80	200
18	104	77	106	109
19	78	91	52	**58**
20	196	170	186	131
21	79	150	59	86
22	170	105	93	170
23	75	76	56	164
24	92	153	146	**61**
25	86	195	54	178
26	**54**	107	111	168
27	**58**	104	127	188
28	164	174	160	123
29	109	164	74	134
30	178	82	81	55
31	67	147	118	99
32	91	151	61	184
33	56	178	159	182
34	131	74	195	156
35	147	**103**	120	147
36	188	163	196	176
37	150	86	83	75
38	82	138	92	**112**
39	186	136	66	91
40	134	137	107	121
41	76	169	**68**	192
42	99	175	117	88
43	105	148	167	82
44	153	188	181	172
45	182	191	98	**97**
46	156	**58**	**60**	87
47	174	152	175	73
48	163	194	177	**126**
49	168	99	179	124
50	107	79	180	140

Bold indicates important masses given by the FuRES method.

separate the classes in this data set. This was also confirmed by the group CV means which were separated by less than the χ^2 95% confidence limits [28]. It was seen that the clustering observed mainly corresponded to the oil's cultivars [28]; the results from this experiment are detailed elsewhere [27,28].

4.2.2. KNN

In this case it was found that autoscaling reduced the predictive ability of the method and it was therefore not used here. The cross validation results for the calibration set indicated that $K=3$ is optimal. When applied to the test set, 9 out of 23 (39.1%) objects were misclassified.

4.2.3. CART

The decision tree produced by CART on data set 2 is shown in Fig. 12. The rules written out are: IF mass $177 \geq 0.02$ THEN adulterated ELSE IF mass $149 \geq 0.14$ THEN virgin ELSE IF mass $53 \geq 1.77$ THEN adulterated ELSE virgin. Using this rule on the validation set it was found that six out of 23 objects (26.1%) were misclassified. The misclassified objects were 3,4,7,8,10 and 22 where four of the virgin oils were wrongly predicted to be adulterated. CART appears to be better than both the CVA and KNN for this data set.

4.2.4. ANN

The detailed experimental set-up for the ANNs analyses has been presented elsewhere [27,28]. Similarly to the ANN analysis of Data set 1 we have a three layered network with 150 input nodes, eight nodes in the 'hidden' layer and one output node in the third 'output' layer. The training was performed using the standard backpropagation algorithm. ANNs were trained to 0.001 RMS error and the network was able to predict all the test set samples correctly in an average of five trainings using randomised starting connection weights [28]. There was a minor uncertainty compared to the other predictions for sample no. 10 (codename Cagliari) of virgin quality in that one of the five ANNs classified that object as adulterated. ANNs without any hidden layers, i.e. perceptrons which can only make linear classification hyperplanes, failed for this (no. 10) and other samples, for details see [28].

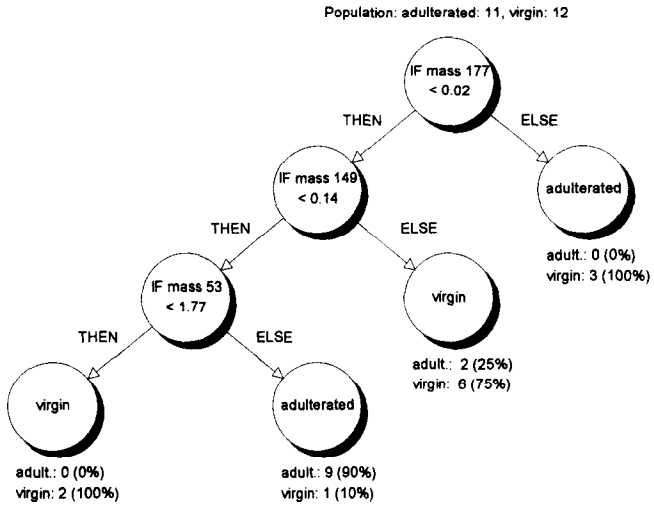

Fig. 12. The CART decision tree for data set 2.

4.2.5. DPLS

Since we have only two possible classes PLS1 instead of PLS2 can be used. The reason for this is that the degree of freedom is one since the probability of membership sums to one. The class membership of an object is determined by whether the predicted *Y*-variable (i.e. class variable) minus the *Y* calibration mean is above or below zero. This is similar to the class assignment procedure described above for multivariate rule induction without the use of fuzzy set theory.

The percent misclassification for each PLS1 factor for the calibration and test sets is shown in Fig. 13. The upper part of the figure shows the calibration model predicting the whole of the calibration set and the lower figure shows the prediction of the test set. As can be seen from the lower part of this figure, the predictions of the test set are sensitive to the selection of the number of PLS factors. Using traditional cross validation techniques with various splitting criteria (not shown) on the calibration set does not produce the optimal number of PLS factors. From the figure we see that the use of only three PLS factors produces a perfect prediction: 3,5, and 7. Analysis of the calibration set alone suggested 7 PLS factors as optimal, which was used in the final PLS model. The PLS score plot of the two first factors in the calibration set are shown in Fig. 14. It seems that PLS factor no. 2 is better than factor no. 1 in

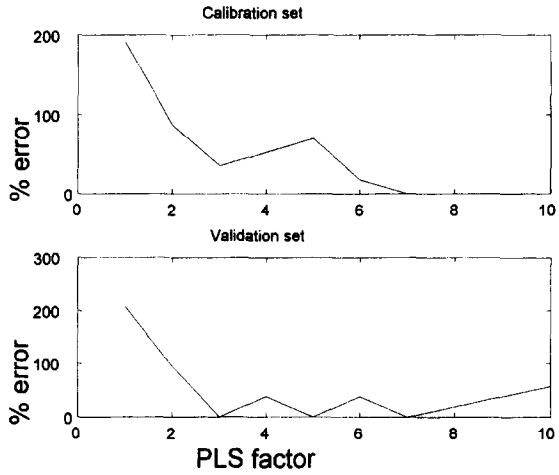

Fig. 13. The percentage misclassification versus the different number of PLS1 factors is shown for the calibration (upper part) and test set (bottom part) for data set 2.

discriminating between adulterated and virgin olive oils.

The predictions of the *Y*-values were calculated by multiplying each sample vector with the regression vector **B**. This vector is shown in Fig. 15. A 100% correct classification for the test set was achieved.

The ten most important masses in **B** are (sorted in descending order of importance): 83, 88, 85, 69, 73, 60, 91, 74, 104, 101. Of these masses, 60, 69, 73, 85,

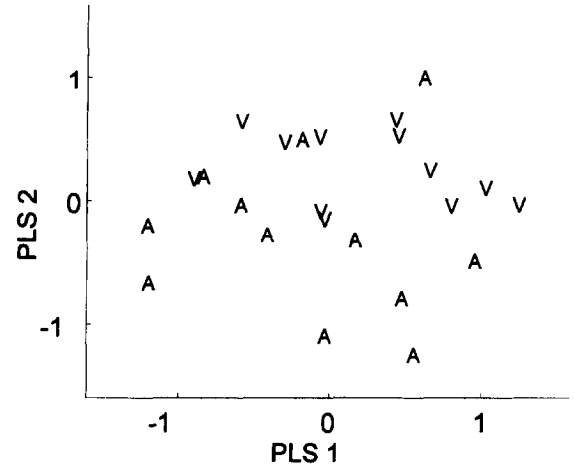

Fig. 14. PLS score plot for data set 2.

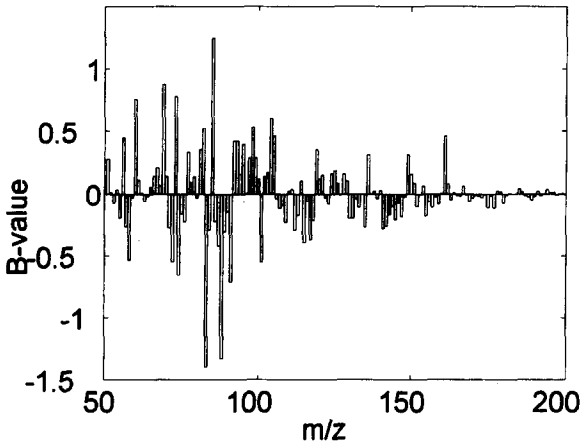

Fig. 15. The regression coefficient vector **B** used in the PLS1 prediction for data set 2.

88 and 91 were also found to be important using FuRES (see below). None of these masses were found by the CART method.

4.2.6. FuRES

The FuRES program generated two rules for data set 2, see the decision tree in Fig. 16. The first rule separates the majority of the adulterated oils from the virgin oils and is plotted as a PyMS like spectrum in Fig. 17. The ten most important variables for this rule (sorted in descending order) are 57, 72, 60, 73, 85, 91, 88, 69, 78, and 97.

The second rule, however, is needed to separate only one adulterated oil from the rest of the virgin oils.

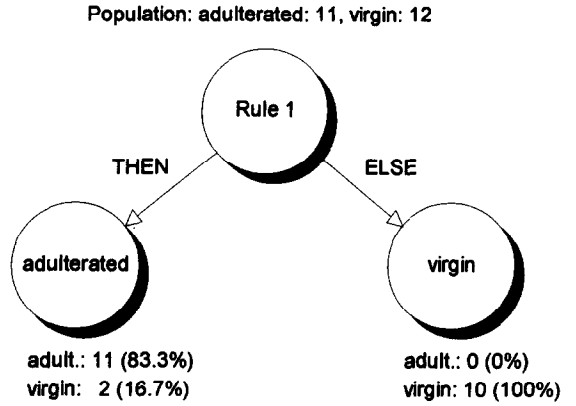

Fig. 16. The FuRES rule induction tree for data set 2. Note that Rule 2 is not significant since it is being used to explain one object only (and is therefore omitted from this decision tree diagram). By using the test set it is possible to show that Rule 2 does not generalise to any other object.

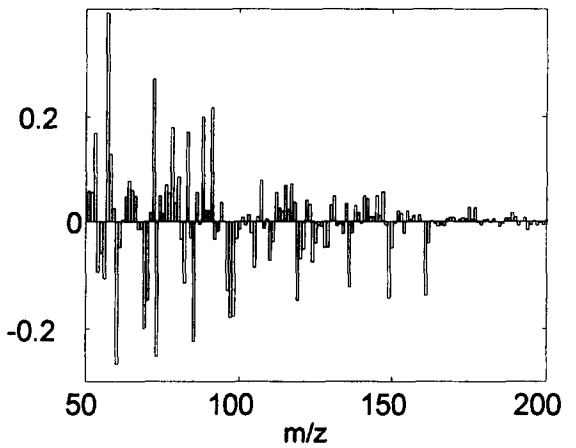

Fig. 17. Here the only significant FuRES rule for data set 2 is shown.

This object was assumed to be an outlier in this analysis, see Fig. 18 lower part. This was further verified when the two rules were used on the validation set, see Fig. 19 lower part. Rule 2 did not generalise and could therefore not improve the prediction for any of the objects in the validation set. Two out of 23 oils (8.7%) were misclassified using Rule 1 only. It is interesting to see that the FuRES misclassified object no. 10 which was also indicated as a problematic sample in the ANN analyses. In particular, the ANN without a hidden layer (the 150-1 net) was not able to classify this sample correctly either. Both

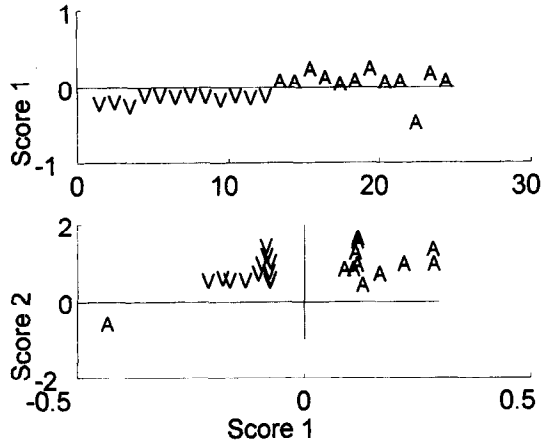

Fig. 18. The FuRES rules 1 and 2 have been applied to the calibration set of data set 2. The upper part of the figure shows the use of Rule 1 only, and the lower part shows the use of both rules. Note that Rule 2 is only used to classify a single object.

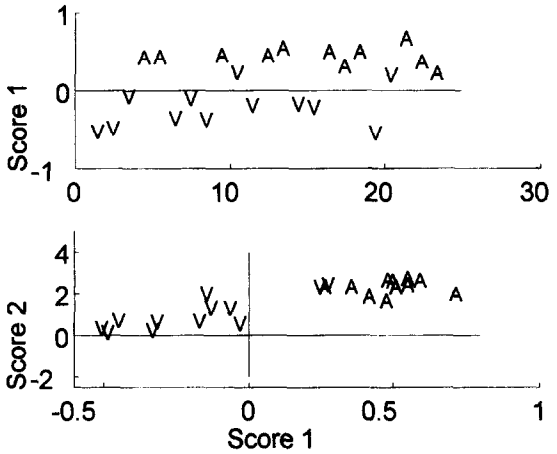

Fig. 19. The FuRES rules 1 and 2 have been applied to the validation set of data set 2. The upper part of the figure shows the use of Rule 1 only, and the lower part shows the use of both rules. Note that Rule 2 does not improve the classification of the test set objects and has therefore been excluded from the final model.

FuRES and the 150-1 network in addition misclassified the virgin oil *Ancona* (no. 20) as being adulterated. It should be noted that the 150-0-1 network was worse than FuRES since it misclassified the virgin oils *Urbino* (no. 3) and *Trieste* (no. 8) also, whereas FuRES classified these test set objects correctly. One possible explanation for the difference in prediction between FuRES and the ANN with a hidden node layer is that in order to obtain a perfect classification for this data set, the classification hypersurface should be curved and not linear. This version of FuRES produces linear hyperplanes only.

Table 5
A summary of the results: The two data sets have been analysed by six different classification methods and are here compared in relation to percentage misclassification of the test set objects and whether it is easy to interpret the results from the classification model

Method	Data set 1 Misclassification	Data set 2 Misclassification	Ease of interpretation
FuRES	0%	8.7%	Easy
CVA	0%	Failed[a]	Easy
DPLS	0%	0%	Easy
ANN (150-1)	0%	17.4%	Difficult
ANN (150-8-1)	0%	0%	Difficult
CART	8.3%	26.1%	Easy
KNN	0%	39.1%	Very difficult

[a]Failed to make a model on the training set. CV means were $< \chi^2$ 95% distances (see text and [28]).

A summary of the performances of the different classification method on these two data sets is presented in Table 5.

5. Conclusion

Based on these two data sets only, it seems that the DPLS method is the best of those considered here. In data set 2, however, we saw that it was difficult to obtain the correct number of PLS factors. Using any other than 3,5 or 7 factors, we would have obtained at best 10% misclassification (with four or six PLS factors). In spite of this the DPLS method is both fast and easy to interpret and therefore should be one of the first classification methods applied to a data set.

FuRES is also a powerful classification algorithm both in terms of the number of correct classifications and the fact that it is possible to extract meaningful information from the classification rules. The tree structure of the different multivariate rules provides an effective tool for the investigator to interpret his or her model using a priori information. In order to improve the classification ability of FuRES it may be possible to extend the method to include polynomial or non-linear decision hypersurfaces.

It should be noted that the DPLS method can, in theory, also be extended to include polynomials or spline decision hypersurfaces.

ANNs with hidden layers are better as classifiers for problems involving non-linear decision hypersurfaces, but are much harder to interpret. There are, however, techniques that try to improve the interpretation of trained ANNs by e.g. extracting IF..THEN..ELSE rules [69–79]. Such techniques may in future become standard tools in ANN analyses.

Acknowledgements

R.G. is indebted to the Wellcome Trust for financial support (grant number 042615/Z/94/Z). The authors thank Prof. P. de B. Harrington for kindly providing the FuRES program. B.K.A., J.J.R. and D.B.K. thank the Chemicals and Pharmaceuticals Directorate of the UK BBSRC, Glaxo Wellcome and Bruker/Spectrospin for financial support.

References

[1] Y. Anzai, Pattern Recognition and Machine Learning, Academic Press, New York, 1989.
[2] C.M. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.
[3] R.G. Brereton, Multivariate Pattern Recognition in Chemometrics., Elsevier, Amsterdam, 1992.
[4] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1992.
[5] D. Helm, H. Labischinski and D. Naumann, J. Microbiological Methods, 14 (1991) 127–142.
[6] G.J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, John Wiley, London, 1992.
[7] G. Montague and J. Morris, Trends Biotechnol., 12 (1994) 312–324.
[8] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
[9] R. Zamora, J.L. Navarro and F.J. Hidalgo, J. American Oil Chemists Soc., 71 (1994) 361–364.
[10] D.B. Kell and B. Sonnleitner, Trends Biotechnol., 13 (1995) 481–492.
[11] C.S. Gutteridge, L. Vallis and H.J.H. MacFie, in M. Goodfellow, D. Jones and F. Priest (Eds.), Computer-assisted Bacterial Systematics, Academic Press, London, 1985, pp. 369–401.
[12] C.S. Gutteridge, Methods in Microbiology, 19 (1987) 227–272.
[13] R. Goodacre, M.J. Neal and D.B. Kell, Anal. Chem., 66 (1994) 1070–1085.
[14] R. Goodacre, M.J. Neal, D.B. Kell, L.W. Greenham, W.C. Noble and R.G. Harvey, J. Appl. Bacteriology, 76 (1994) 124–134.
[15] H.L.C. Meuzelaar, J. Haverkamp and F.D. Hileman, Pyrolysis Mass Spectrometry of Recent and Fossil Biomaterials, Elsevier, Amsterdam, 1982.
[16] W. Windig, H.L.C. Meuzelaar, B.A. Haws, W.F. Campbell and K.H. Asay, J. Anal. Appl. Pyrolysis, 5 (1983) 183–198.
[17] W.J. Irwin, Analytical Pyrolysis: A Comprehensive Guide, Marcel Dekker, New York, 1982.
[18] D. Coomans, D.L. Massart and L. Kaufman, Anal. Chim. Acta, 112 (1979) 97.
[19] P.A. Lachenbruch, Discriminant Analysis, Haffner Press, New York, 1975.
[20] S. Haykin, Neural Networks, Macmillan, New York, 1994.
[21] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Massachusetts, 1990.
[22] J.M. Henle, An Outline of Set Theory, Springer-Verlag, New York, 1986.
[23] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, Classification and Regression Trees, Wadsworth, Pacific Grove, California, 1984.
[24] P.d.B. Harrington and K.J. Voorhees, Anal. Chem., 62 (1990) 729–734.
[25] J.R. Quinlan, Mach. Learning, 1 (1986) 81–106.
[26] R. Goodacre, Applied Spectroscopy, in press.
[27] R. Goodacre, D.B. Kell and G. Bianchi, Nature, 359 (1992) 594.
[28] R. Goodacre, D.B. Kell and G. Bianchi, J. Science of Food and Agriculture, 63 (1993) 297–307.
[29] R. Goodacre, S. Trew, C. Wrigley-Jones, G. Saunders, M.J. Neal, N. Porter and D.B. Kell, Anal. Chim. Acta, 313 (1995) 25–43.
[30] B.F.J. Manly, Multivariate Statistical Methods: A Primer, Chapman and Hall, London, 1994.
[31] W. Eshuis, P.G. Kistemaker and H.L.C. Meuzelaar, in C.E.R. Jones and C.A. Cramers (Eds.), Analytical Pyrolysis, Elsevier, Amsterdam, 1977, pp. 151–156.
[32] J.B. Kruskal, Psychometrika, 29 (1964) 1–27.
[33] J.B. Kruskal, Psychometrika, 29 (1964) 115–129.
[34] C.E. Shannon, Bell System Technical J., 379(1948) 379–423, 623–656.
[35] I.E.D. Minitab, State College, PA 16801–3008, SCAN Software for Chemometric Analysis, Release 1 for Windows ed., USA, 1995.
[36] S.K. Murthy, S. Kasif and S. Salzberg, J. Artificial Intell. Research, 2 (1994) 1–32.
[37] P.E. Utgoff and C.E. Brodley, Linear Machine Decision Trees, University of Massachusetts, Amherst, 1991.
[38] B.A. Draper, C.E. Brodley and P.E. Utgoff, IEEE Tran. on Pattern Anal. and Machine Intell., 16(1994) 888–893.
[39] C.E. Brodley and P.E. Utgoff, Mach. Learning, 19 (1995) 45–77.
[40] L.A. Zadeh, Fuzzy sets, fuzzy logic, and fuzzy systems: Selected papers, World Scientific, River Edge, N.J., 1996.
[41] P.d.B. Harrington, Chemometrics and Intell. Lab. Sys., 18 (1993) 157–170.

[42] P.d.B. Harrington, J. Chemometrics, 5 (1991) 467–486.
[43] J. Pannetier, Institute of Physics Conference Series, (1990) 23–44.
[44] H.J.H. MacFie, C.S. Gutteridge and J.R. Norris, J. Gen. Microbiol., 104 (1978) 67–74.
[45] J.A. Nelder, Genstat Reference Manual, Scientific and Social Service Program Library, University of Edinburgh, 1979.
[46] D.E. Rumelhart, J.L. McClelland and The PDP Research Group, Parallel distributed processing, Experiments in the Microstructure of Cognition, MIT Press, Cambridge, 1986.
[47] P.D. Wasserman, Neural Computing: Theory and Practice, Van Nostrand Reinhold, New York, 1989.
[48] P.J. Werbos, The Roots of Back-propagation: From Ordered Derivatives to Neural Networks and Political Forecasting, John Wiley, Chichester, 1994.
[49] J.C. Bezdek and S.K. Chuah, Fuzzy Sets and Sys., 18 (1986) 237–256.
[50] D. Coomans and D.L. Massart, Anal. Chim. Acta, 138 (1982) 15.
[51] J. Kittler and P.A. DeVijver, Pattern Recognition, 13 (1981) 245–249.
[52] C. Schaffer, Mach. Learning, 13 (1993) 135–143.
[53] S. Wold, N. Kettanehwold and B. Skagerberg, Chemometrics and Intell. Lab. Sys., 7 (1989) 53–65.
[54] S. Wold, C. Albano, W.J. Dunn, K. Esbensen, S. Hellberg, E. Johansson, W. Lindberg and M. Sjostrom, Analusis, 12 (1984) 477–485.
[55] S. Wold, A. Ruhe, H. Wold and W.J. Dunn, Siam J. Scientific and Statistical Computing, 5 (1984) 735–743.
[56] S. Wold, Technometrics, 35 (1993) 136–139.
[57] H. Martens and T. Næs, Multivariate Calibration, John Wiley, Chichester, 1989.
[58] H. Martens, L. Izquierdo, M. Thomassen and M. Martens, Anal. Chim. Acta, 191 (1986) 133–148.
[59] M. Martens, H. Martens and S. Wold, J. Science of Food and Agriculture, 34 (1983) 715–724.
[60] R. Manne, Chemometrics and Intell. Lab. Sys., 2 (1987) 187–197.
[61] A. Höskuldsson, J. Chemometrics, 6 (1992) 307–334.
[62] A. Höskuldsson, J. Chemometrics, 9 (1995) 91–123.
[63] T.R. Holcomb and M. Morari, Computers and Chem. Eng., 16 (1992) 393–411.
[64] P. Geladi and B.R. Kowalski, Anal. Chim. Acta, 185 (1986) 1–17.
[65] I.E. Frank, Chemometrics and Intell. Lab. Sys., 8 (1990) 109–119.
[66] S. DeJong, J. Chemometrics, 9 (1995) 323–326.
[67] S. DeJong, J. Chemometrics, 7 (1993) 551–557.
[68] H. Martens and T. Naes, Multivariate Calibration, John Wiley and Sons, Chichester, 1989.
[69] R. Andrews, J. Diederich and A.B. Tickle, Knowledge-based Syst., 8 (1995) 373–389.
[70] S. Abe and M.S. Lan, IEEE Trans. Fuzzy Sys., 3 (1995) 18–28.
[71] J. Downs, R.F. Harrison and S.S. Cross, Lecture Notes Artificial Intell., 934 (1995) 239–250.
[72] D.L. Hudson, M.E. Cohen and M.F. Anderson, Int. J. Intell. Sys., 6 (1991) 213–223.
[73] T. Ichimura, E. Tazaki and K. Yoshida, Int. J. Bio-Medical Computing, 40 (1995) 139–146.
[74] H. Narazaki, T. Watanabe and M. Yamamoto, Ieee Trans. Systems Man and Cybernet. Part B-Cybernet., 26 (1996) 107–117.
[75] C.W. Omlin and C.L. Giles, IEEE Trans. Knowledge and Data Eng., 8 (1996) 183–188.
[76] C.W. Omlin and C.L. Giles, Neural Networks, 9 (1996) 41–52.
[77] R. Setiono, Artificial Intell. in Medicine, 8 (1996) 37–51.
[78] H.L. Viktor and I. Cloete, Lecture Notes Computer Sci., 930 (1995) 611–618.
[79] G.G. Towell and J.W. Shavlik, Mach. Learning, 13 (1993) 71–101.