Chemometrics and Intelligent Laboratory Systems 227 (2022) 104614

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemometrics

Simultaneous class-modelling in chemometrics: A generalization of Partial Least Squares class modelling for more than two classes by using error correcting output code matrices

O. Valencia [a], M.C. Ortiz [b,*], S. Ruiz [a], M.S. Sánchez [a], L.A. Sarabia [a]

[a] Departamento de Matemáticas y Computación, Facultad de Ciencias, Universidad de Burgos, Plaza Misael Bañuelos s/n, 09001, Burgos, Spain
[b] Departamento de Química Facultad de Ciencias, Universidad de Burgos, Plaza Misael Bañuelos s/n, 09001, Burgos, Spain

ARTICLE INFO

Keywords:
Compliant class-models
Error-correcting output code (ECOC)
Partial least squares multi-response regression (PLS2)
Sensitivity
Specificity
Diagonal modified confusion entropy (DMCEN)

ABSTRACT

The paper presents a new methodology within the framework of the so-called compliant class-models, PLS2-CM, designed with the purpose of improving the performance of class-modelling in a setting with more than two classes. The improvement in the class-models is achieved through the use of multi-response PLS models with the classes encoded via Error-Correcting Output Codes (ECOC), instead of the traditional class indicator variables used in chemometrics.

The proposed PLS2-CM entails a decomposition of a class-modelling problem into a series of binary learners, based on a family of code matrices with different code length, which are evaluated to obtain simultaneous compliant class-models with the best performance.

The methodology develops both a new encoding system, based on multi-criteria optimization to search for optimal coding matrices, and a new decoding system, based on probability thresholds to assign objects to class-models. The whole procedure implies that the characteristics of the dataset at hand affect the final selection of the coding matrix and therefore of built class-models, thus giving rise to a data-driven strategy.

The application of PLS2-CM to a variety of cases (controlled data, experimental data and repository datasets) results in an enhanced class-modelling performance by means of the suggested procedure, as measured by the DMCEN (Diagonal Modified Confusion Entropy) index and by sensitivity-specificity matrices. The predictive ability of the compliant class-models has been evaluated.

1. Introduction

Class prediction in a multiclass context is a recurrent topic in chemometrics and, even more, in machine learning. In the 2-class case, a series of binary classification algorithms can be applied to this task, some of which can be naturally extended for a multiclass setting, such as neural networks, decision trees, K-Nearest Neighbor (KNN), Naive Bayes, or Support Vector Machines (SVM). Nevertheless, when the number of categories or classes, k, is greater than two, class prediction is a more complex problem due to class overlapping or class imbalance, and the performance of different competing classifiers is not easy to measure and compare.

An alternative, broadly spread, approach to manage the problem is based on decomposing the multiclass classification problem into multiple binary classifiers. A wide range of procedures have been suggested for such a decomposition: in One- versus All (OVA), a binary classifier compares each class to all others, so k binary learners are used [1], whereas in All versus All (AVA) or pairwise coupling [2], a binary classifier discriminates between each pair of classes discarding the remaining ones, so $k(k-1)/2$ binary classifiers are required. Decomposing a k-class problem into k class-models is the proposal in Ref. [3] with an ensemble of One-Class Classifiers (OCC) building each learner with the objects of a single class, setting aside the examples from the other class (or classes). Moreover, there are procedures inspired in those of DOE (Design of Experiments) such as Hadamard matrices [4] or symmetric factorial designs [5].

The design of these binary classifiers has been largely enhanced with an improved encoding strategy known as the Error Correcting Output Codes (ECOC) [6,7]. The ECOC framework has been useful in the engineering field and applied to different research areas.

* Corresponding author.
E-mail address: mcortiz@ubu.es (M.C. Ortiz).

https://doi.org/10.1016/j.chemolab.2022.104614
Received 18 February 2022; Received in revised form 25 May 2022; Accepted 24 June 2022
Available online 30 June 2022

According to the pioneer work by Dietterich in 1995 [6], a multiclass learning problem, a class-modelling problem for $k > 2$ in the present paper, can be regarded as a communication problem: the information about the actual class for a new object is transmitted over a channel, which is made up of some predictor variables (a training dataset) and a prediction model. As there may be flaws along the channel, whether it is an unrepresentative training set, insufficient predictors, or an inadequate prediction model, errors may arise in the predicted class for some objects.

Using an error-correcting code, the information of the k classes is encoded into a series of *binary learners*, which can allow the channel to recover from errors. This kind of code is defined by a matrix **M** of binary values (−1 and +1), with k rows or classes, and c columns or binary learners. The i-th class (row) is then represented by a *codeword* $\mathbf{r_i}$ of length c, whereas every binary learner (column), $f_1, f_2, \ldots, f_c$, corresponds to a partition of the k classes into two subsets ('minus ones' versus 'plus ones').

Considering that two columns that only differ in the sign (i.e., $f_i = -f_j$) result in the same binary learner and that constant vectors (all 1 or all −1) are discarded, the 'binary complete' ECOC matrix is the one with k rows and all the different 2^{k-1}-1 binary learners as columns. Consequently, the design of a proper code for a given problem consists of selecting the length c of the codeword, that is, to choose among the complete ECOC matrix a suitable subset of c columns to construct **M**.

The implicit idea in OVA, namely that each learner is binary, is generalized in Ref. [7] by introducing a code with three labels {-1, 0, +1}, in such a way that the class labelled as 0 does not intervene in the corresponding ternary learner. This provides a unifying framework including different types of encoding: OVA, pairwise, complete ECOC, dense random coding DRC (all binary codes), and sparse random coding (ternary codes). However, the ternary codes introduce two different biases, giving rise to variations in decoding [8] and the need of defining a new ternary ECOC distance [9].

To overcome these types of predefined output codes, Crammer and Singer [10] published a general discussion on the design of output codes for a given multiclass problem, showing that looking for an optimal code is a *NP*-complete problem. They also argue that continuous codes may be better than traditional discrete ones and provide an algorithm that is two orders of magnitude faster than standard quadratic programming.

Traditionally, ECOC encoding has been performed through data-independent algorithms where the coding matrix **M** is predefined regardless data characteristics. More recently, research on ECOC has been focused on data-dependent algorithms, with **M** based on the different data distributions among classes and intended to search for optimal class reassignments schemes. Several algorithms for this task are shown in Ref. [11], such as DECOC, Forest-ECOC, and ECOC-ONE. Setting aside a part of the training set to be used as a prediction/validation set to estimate the misclassification error, there are algorithms that use evolutionary computation [12] or the Receiver Operating Characteristic (ROC) curve [13].

Conventionally, the ECOC framework adopts a static ensemble strategy, in which, once the coding matrix **M** is defined, the corresponding ECOC ensemble is fixed for the predictions of all new objects.

Some research is focused on implementing a dynamic ensemble selection strategy (DES) [14], where each binary learner (column of the coding matrix **M**) is matched with a set of feature subsets obtained by several feature selection methods. Therefore, more columns may be involved in the classification of tough classes or minority classes or even variable-length codewords (VL-ECOC) are used [15], longer for the though classes and shorter for the easy ones, at the expense of higher computational costs [16].

Krawczyk [3] suggests the use of dynamic ensemble selection to discard non-competent binary learners. In a k-class problem, a KNN algorithm is used to pre-select the classes in a large neighborhood: when a new object must be modeled, the 3 nearest neighbors are considered and just the classes presented in this neighborhood will determine the binary learners selected. Moreover, if certain classes are sparsely represented in the neighborhood, because of outliers or noisy data, they are also removed, thus improving class boundaries, and reducing misallocations.

In [17] the performance of six differently based learners (neural networks, decision trees, support vector machines with a linear and a Gaussian kernel, logistic regression, and naive Bayes classifiers) is studied. However, as the influence of the kind of binary learners used has not been studied in detail, the encoding process is usually learner independent.

In any case, when a new object **x** should be assessed, the c binary learners are evaluated at **x**, and a vector with the outputs $\mathbf{y}=(y_1, y_2, \ldots, y_c)$ or predicted codeword is obtained. The stage of decoding usually consists of comparing that predicted codeword **y** to the codeword $\mathbf{r_i}$ of each of the k classes, thus assigning the object to the closest one in terms of a given similarity measure, such as the Hamming distance in Ref. [6], or to the class that minimizes some loss function, as in Ref. [7]. In fact, there are several ECOC decoding strategies, which can be grouped into three types: i) those based on a distance between the output code (predicted codeword **y**) and a class codeword, ii) those based on a probability of belonging to a class, and iii) those that compute a pattern space transformation via cluster analysis of the output of the learner. A detailed revision can be found in Refs. [8,18]. Ref. [11] contains code for nine decoding designs of type i) and ii). Research on the matter is still open, some advances can be seen in Ref. [14] where the decoding process selects an optimal feature subset from the candidate subsets based on the data complexity theory. The study uses different binary learners and various ECOC algorithms to check their proposal and find a better performance particularly in the minority classes. Liu et al. [16] introduce the concept of soft recoding strategies designed by replacing elements in code matrices with the mean values or the intervals of learner outputs.

Whatever the decomposition of the multiclass classification problem into several binary classifiers, two perspectives can be adopted: discrimination or class-modelling.

Unlike discrimination, the class-modelling methods aim at capturing the unique properties of the target class, so each class model relies on training objects of the target or positive class (+1), while (−1) stands for an object which does not satisfy the description of the target class. This implies modelling the categories independently to one another, reason why in chemometrics they are called one-class classifiers [19].

However, when more than one class is being studied, a distinction can be made [20] between 'rigorous' (equivalent to one-class classifiers) and 'compliant' class-modelling methods. The rigorous concept arises when only objects of the modeled class are considered whereas the use of the objects of the other classes for building the class-models make the technique a compliant class-modelling method. Alternative denominations distinguish between soft or hard models [21,22] depending on whether intersection between class models is allowed.

In the present work, a k-class problem is tackled using k class-models that are built together in a compliant class-modelling situation.

Different proposals are found in literature for class-model building, such as density-based methods, reconstruction-based methods, boundary-based methods, and ensemble-based methods (see Ref. [23] for further detail). Authors emphasize some advantages in the class-modelling approach, provided a suitable algorithm is used, such as avoiding overfitting, being more robust to class imbalance, or dealing with noisy data in the target class that might cause a too large decision boundary more likely to overlap others.

The class modelling approach is preferable in chemometrics, where overlapping classes are likely to arise, due to the very nature of the classes, or to their description by predictors without sufficient information, and/or to an inadequate prediction procedure. So, instead of a forced discrimination, a more realistic approach should include chances of assigning an object to more than one class or even not assigning it at all. A further discussion can be found in Refs. [24,25].

If an object (sample) is not identified as a member of any of the modeled classes, it can be considered as a sample from an unknown class, a counterfeit, or a sample of low quality, reason why class-modelling in

chemometrics is widely applied in food authentication [26,27] or drug authentication [28], for assessing PDO (Protected Designation of Origin) [29], or for process control [25,30,31].

Examples of class-modelling techniques in chemometrics include SIMCA (soft independent models of class analogy [32]), SVM (support vector machine) [33] or UNEQ (unequal class models or unequal dispersed classes [34]). Khan [35] proposes a taxonomy of class-modelling techniques or one-class classifiers. A review of class-modelling can be found in Refs. [20,21,36,37]. More recently, Malyjurek [38] reviews the performance of SIMCA, One-Class Partial Least Squares (OCPLS) [31,39], and other techniques such as kernel-PCA [40], Support Vector Domain Description (SVDD) [41] and Potential Functions Method (PFM) [42].

Some developments on compliant class-modelling via PLS (Partial Least Squares) regression in chemometrics have been conducted in Refs. [43–47], the so-called PLS-CM (Partial Least Squares for Class-Modelling). Its suitability has been proved in this field provided that typical data of chemical instruments entail a substantial number of correlated variables, such as spectra.

When $k = 2$, PLS-CM means using just a binary learner, and the definition of the class model is well determined from the probability distributions fitted to the predicted values of the response, separately for each class, given a critical value or threshold. However, in a multiclass context with more than two classes, $k > 2$, a larger number of binary learners is required, which increase the number of responses to be fitted with PLS. Consequently, PLS2 will be selected as a prediction model to generalize PLS-CM as a compliant class-modelling method for multiclass settings.

In the field of chemometrics, only three articles have been found that apply coding using ECOC matrices though all of them for purely discrimination tasks, whether to detect defects in potatoes with hyperspectral images and SVM [48], to classify extra-virgin olive oils of fourteen different geographical origin with a commercial electronic nose and multilayer perceptron [49], or to accurately distinguish the geographical origins of wolfberry fruit with NIR data and a modified SVM as binary learner [50].

No previous discussion about the ECOC coding strategy for PLS2 has been found in literature. Usually, the default strategy for multiclass problems is encoding them following OVA, meaning that the length of the codeword is just defined as the number of classes and there are k response variables. However, in PLS2, orthogonality is not required for the response variables, so the prediction model can benefit from a more efficient encoding within the ECOC framework.

The present work studies the effect of using ECOC matrices coupled with PLS2 prediction models for class-modelling tasks with k classes ($k > 2$). As far as the authors know, neither ECOC codes have been used for class-modelling techniques nor PLS2 regression has been used as a binary learner.

2. PLS2-CM strategy: using ECOC matrices for designing class-models for more than two classes

To use PLS2 together with error correcting codes as a class-modelling technique, several aspects should be taken into account. All of them are itemized in the present section to make it easier to understand the integration of all the elements of the proposal.

As the PLS2 algorithm is well known in chemometrics, it suffices to say that a matrix of predictor variables $\mathbf{X}$ is assumed, with N objects belonging to k conceptually different classes. The following subsections describe the procedure of computing the matrix of responses to be fitted with PLS2 by using an ECOC matrix $\mathbf{M}$ (precisely, the rows of response matrix $\mathbf{Y}$ will be the corresponding rows in the coding matrix $\mathbf{M}$).

2.1. Error-correcting code design

Given a problem of class modelling with k classes and a given c, length of the codeword, an ECOC matrix is a $k \times c$ matrix (k rows and c columns) of the form $\mathbf{M} = (m_{ij})$ with $m_{ij} \in \{-1, 1\}$. We will denote as $\mathbf{r}_i = (m_{i1}, m_{i2}, \ldots, m_{ic})$, $i = 1, \ldots, k$, the rows in $\mathbf{M}$ and by $\mathbf{c}_j = (m_{1j}, m_{2j}, \ldots, m_{kj})^T$, $j = 1, \ldots, c$, its column vectors, each one representing the corresponding f_j binary learner or binary classifier.

As an example, Table 1 shows three different code matrices for $k = 4$ classes. Thus, they all have four rows but different length of the codeword c. Matrix $\mathbf{M_1}$ has $c = 7$ binary learners, $\mathbf{M_2}$ has $c = 4$ columns for coding the classes, and $\mathbf{M_3}$ is a coding matrix with a codeword length of $c = 3$.

As mentioned, the *complete code* is the exhaustive code for k classes, made up of binary leaners which correspond to all the non-trivial partitions of the set of k classes into two subsets (positive and negative). After removing columns that only differ in sign and the constant vectors (all -1 or all $+1$ columns), this coding system produces at most (2^{k-1}-1) useable columns (binary learners).

As $k = 4$ in Table 1, the complete code will be the matrix with $c = 2^3–1 = 7$ columns, that is, the 4×7 matrix named $\mathbf{M_1}$. If an object belongs to the ith class ($i = 1, \ldots, 4$) the corresponding codeword (response vector for fitting with PLS2) would be $\mathbf{r}_i$, the ith row of the corresponding coding matrix. For example, an object of class 2 is encoded as vector (-1, -1, -1, -1, $+1$, $+1$, $+1$), which is the second row of $\mathbf{M_1}$, but as (-1, $+1$, -1, -1) if we were to use $\mathbf{M_2}$.

On the other hand, looking at the coding matrix by columns, each binary learner f_i is associated with a partition of the four classes into two subsets (kind of *super classes*) $\mathbb{A}$ and $\mathbb{B}$, which comprise the objects with code -1 and $+1$, respectively. For example, for f_4 in $\mathbf{M_1}$, $\mathbb{A}$ is composed only for class 2 whereas $\mathbb{B}$ is the union of classes 1, 3, and 4, whereas f_5 on its part opposes classes 3 and 4 in $\mathbb{A}$ to classes 1 and 2 in $\mathbb{B}$.

In that sense, in both $\mathbf{M_2}$ and $\mathbf{M_3}$ in Table 1, superclass $\mathbb{B}$ only contains one class at a time: $\mathbf{M_2}$ has $c = 4$ binary learners (f_1 to f_4), each one with a single $+1$ as against the remaining -1, it is the usual OVA code. A small variation of it is $\mathbf{M_3}$, it needs only three binary classifiers as f_4 of $\mathbf{M_2}$ is discarded, but the common f_1, f_2, and f_3 are one versus all classifiers, hence the notation (OVAv) [22,51].

Finally, as $\mathbf{M_1}$ is the complete code, it contains all binary learners in $\mathbf{M_2}$ (and $\mathbf{M_3}$): f_1 is the same, f_2, f_3, and f_4 in $\mathbf{M_2}$ are, respectively f_4, f_6, and f_7 in $\mathbf{M_1}$ (only a change of sign). Therefore, one would probably tend to think that it is better to always use the complete code. However, taking into account that c determines the dimension of the response space to be fitted with PLS2 and that 2^{k-1} increases very rapidly with k, the question that arises is whether it is possible to obtain the same, or even better, performance by using shorter codewords. In other words, in the example in Table 1, is it worth to increase from three to seven responses when using $\mathbf{M_1}$ instead of $\mathbf{M_3}$? If the answer is no, the question that still remains is why this particular three columns were selected, and if any other three would provide the same performance.

In summary, if a code with length $c < 2^{k-1}$-1 is desired, it must be decided how to choose the c columns, i.e., which particular c columns should be selected from the complete code.

With this goal, in Refs. [6,52] different criteria are described, that is, different properties of the $\mathbf{M}$ matrix that *a priori* improve the performance of a code. Basically, there are five criteria, detailed in the following paragraphs, and related to the fact that to achieve the capability of correcting errors, the design of such a $\mathbf{M}$ matrix should satisfy some conditions, like that the codeword for each class (different rows in $\mathbf{M}$) should be well separated from those used for each of the other classes, and also the different binary learners (columns in $\mathbf{M}$) should define different enough $\mathbb{A}$ and $\mathbb{B}$ super classes.

In what follows, $\mathbf{M}$ is a $k \times c$ coding matrix and the metric to compare rows and columns in $\mathbf{M}$ will be the Hamming distance. In general, for n-dimensional binary vectors, the Hamming distance is defined in Eq. (1), where *card* refers to the cardinal number of the corresponding set.

$$dH(\mathbf{u}, \mathbf{v}) = dH((u_1, \ldots, u_n), (v_1, \ldots, v_n)) = card\{i | u_i \neq v_i\} \quad (1)$$

Notice that dH is always a natural number, which is the number of coordinates by which $\mathbf{u}$ and $\mathbf{v}$ differ.

Table 1
Different code matrices for four classes, and their quantification in terms of the five criteria in Eqs. (2)–(6).

Name	Coding matrix	$crit_1$	$crit_2$	$crit_3$	$crit_4$	$crit_5$
M_1 Complete code	$f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7$ $\begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{pmatrix}$	4	4	0	1	3
M_2 OVA (one vs all)	$f_1\ f_2\ f_3\ f_4$ $\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix}$	2	2	0	2	2
M_3 OVAv	$f_1\ f_2\ f_3$ $\begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & -1 & -1 \end{pmatrix}$	1	2	0	2	2

Regarding codeword for class separation (rows in **M**), the key feature of a code to be able to correct errors is related to the minimum of the Hamming distance between any pair of codewords. This is because, if this minimum is equal to d, the code can correct at least $(d-1)/2$ single errors, since the nearest codeword to the one predicted for a new object will still be the correct codeword.

2.1.1. Criterion 1

Minimum of the Hamming distances, dH in eq. (1), between any pair of rows $\mathbf{r}_i$ and $\mathbf{r}_j$ in **M**, which is related to the ability of the code **M** to correct errors, that is, to distinguish one class from another.

Intuitively, one code is better than another if the former has a larger minimum than the latter. Furthermore, it is shown in Ref. [7] that a great minimum distance improves the bound related to the generalization error of the classifier.

Therefore, the first criterion, $crit_1$ as precisely defined in Eq. (2), should be maximized when looking for an optimal coding matrix **M** of a given length c.

$$crit_1(\mathbf{M}) = \min_{i,j,i\neq j}\{dH(\mathbf{r}_i, \mathbf{r}_j)\} \tag{2}$$

2.1.2. Criterion 2

Closely related to the previous one, the second criterion in Eq. (3) is the maximum of the Hamming distances between pairs of rows, since the larger the maximum, the better the coding matrix.

$$crit_2(\mathbf{M}) = \max_{i,j,i\neq j}\{dH(\mathbf{r}_i, \mathbf{r}_j)\} \tag{3}$$

Like the first one, $crit_2$ should be maximized.

2.1.3. Criterion 3

The effectiveness of a code improves if the Hamming distance between pairs of rows shows little variation [52]. Consequently, the distribution of the $k\times(k-1)/2$ Hamming distances between rows in **M** is compared to a uniform distribution. This is done by summing up the square of the differences between the vector of frequencies of the Hamming distances in **M** and those expected under a uniform distribution, as defined in Eq. (4),

$$crit_3(\mathbf{M}) = \sum_{i=a}^{b}\left(fr_i - \frac{k(k-1)}{2(b-a+1)}\right)^2 \tag{4}$$

where $a = crit_1(\mathbf{M})$, $b = crit_2(\mathbf{M})$ and fr_i is the frequency of the i-th Hamming distance in **M**. When $a = b$, $crit_3$ is set to zero. To obtain a distribution as close to the uniform as possible, $crit_3$ should be minimized.

Regarding the columns in **M**, it is intended to obtain sufficient separation between the binary learners (provided that they are not identical or differing in sign) to avoid simultaneous errors. In the ECOC framework, the misclassification errors can be due either to the decision rule or to the inability of the subproblems, solved by each binary learner, to represent the global modelling. The subproblems, represented by each column, will contribute better to the global modelling the greater the Hamming distance between all columns $\mathbf{c}_i$ and $\mathbf{c}_j$ of the coding matrix **M**. This issue is quantified with the criteria 4 and 5 in Eqs. (5) and (6).

2.1.4. Criterion 4

The fourth criterion defined in Eq. (5) is the minimum distance between any pair of columns (binary learners) in **M**.

$$crit_4(\mathbf{M}) = \min_{i,j,i\neq j}\{dH(\mathbf{c}_i, \mathbf{c}_j)\} \tag{5}$$

As explained, $crit_4$ should be maximized for optimal code matrices.

2.1.5. Criterion 5

Like with the rows, the maximum Hamming distance between columns should be measured. This is the $crit_5$ in Eq. (6), which should also be maximized.

$$crit_5(\mathbf{M}) = \max_{i,j,i\neq j}\{dH(\mathbf{c}_i, \mathbf{c}_j)\} \tag{6}$$

Table 1 contains the values of the five criteria for the different code matrices. They are not directly comparable to each other when the code matrices have different c.

If $k = 3$, the complete code with $2^2-1 = 3$ binary learners f_1, f_2, and f_3 (three columns) coincides with the OVA code, that is, a 3×3 matrix with ones in the main diagonal and -1 elsewhere. For $c = 2$, there are three different code matrices, namely those with f_1 and f_2, f_1 and f_3, or f_2 and f_3. All of them differ in the same number of coordinates and for both rows and columns, so the three matrices have the same value in all five criteria.

When $k > 3$, the optimum value for the five criteria can no longer be achieved in a single coding matrix **M**. For example, with $k = 5$ and $c = 9$, there are 5005 different ECOC matrices. Ten of them achieve the maximum of $crit_1$ (which is 5), while the maximum of $crit_2$ (which is 8) is achieved in 70 matrices, all different from the previous ten. Analogously, 80 matrices achieve the minimum in $crit_3$, but none of them reach the maximum in either $crit_1$ or $crit_2$.

Consequently, a systematic procedure for selection of one or several ECOC matrices is needed. Up to $k = 5$, it is feasible to compute all possible combinations of c columns among the total 2^{k-1} -1 and, thus, the complete population of criteria values. However, for six or more classes, and $c > 5$, this is not viable, and a sample of matrices from the whole population should be selected.

2.1.6. Column selection for k-classes (3 < k < 6)

Starting with less than six classes (3 < k < 6), the procedure consists of the following steps:

Step 1. - For a value of c, all the possible $k \times c$ code matrices **M** are computed with c binary learners (c columns) extracted from the complete code. The total number of matrices is $q = (2^{k-1} - 1)!/(c!(2^{k-1} - 1 - c)!)$. For example, $q = 3003$ with $k = 5$ classes and $c = 5$ binary learners for the coding matrix, only one of them corresponds to the usual OVA code.

Step 2. - For each of the q code matrices **M**, the five criteria in Eqs. (2)–(6) are computed. In this way, there are q vectors of five values that qualify the corresponding code in **M**. Among the q values for each criterion, the maximum and minimum value is known, which are the target values for the different criteria.

As the criteria behave oppositely, the desirability function approach [53], widely used for multiresponse experimental optimization [54], is selected to reach a compromise among them.

In short, it starts with the definition of individual desirability functions to map the individual values of every criterion into [0, 1], in such a way that the value 0 is assigned for inadmissible values of the corresponding criterion and 1 for the target values. In this way, the possible effect of each criterion being on a different scale is avoided.

The precise definition of the four individual desirability functions d_i for the criteria to be maximized (i = 1, 2, 4, and 5), namely $crit_1$, $crit_2$, $crit_4$, and $crit_5$, is in eq. (7).

$$\begin{aligned} d_i(crit_i) &= 1.00 \text{ if } crit_i \text{ takes the maximum among the } q \text{ values} \\ d_i(crit_i) &= 0.01 \text{ if } crit_i \text{ takes the minimum, among the } q \text{ values} \\ d_i(crit_i) &\text{ linear between } 0.01 \text{ and } 1.00, \text{ for the remaining values of } crit_i \end{aligned} \tag{7}$$

Eq. (8) corresponds to the individual desirability function d_3 for $crit_3$ that must be minimized.

$$\begin{aligned} d_3(crit_3) &= 1.00 \text{ if } crit_3 \text{ takes the minimum among the } q \text{ values} \\ d_3(crit_3) &= 0.01 \text{ if } crit_3 \text{ takes the maximum of the } q \text{ corresponding values} \\ d_3(crit_3) &\text{ linear between } 0.01 \text{ and } 1.00, \text{ otherwise} \end{aligned} \tag{8}$$

Finally, the global desirability function, D, assigns a weighted geometric mean of functions d_i, i = 1, 2, …, 5, to each **M** according to eq. (9).

$$D(crit_1, crit_2, \ldots, crit_5) = d_1^{w_1} d_2^{w_2} d_3^{w_3} d_4^{w_4} d_5^{w_5} \tag{9}$$

with $0 \le w_i \le 1$, $i = 1, 2, \ldots, 5$ and $\sum_{i=1}^{5} w_i = 1$. The weights w_i allow to give different importance to each individual criterion.

The coding matrices of interest are those with maximum value of D, ideally 1. Usually, the maximum of the global desirability function D in eq. (9) is reached for several code matrices **M**.

Step 3. Each of these optimal coding matrices is used to define **Y** and build k-class-models with PLS2. Finally, the one with the best performance in the class-modelling task is selected.

As it can be noticed, the procedure described in *step1-step3* is different from the customary procedure of considering a single coding matrix **M**, selected for a prefixed pair (k, c), i.e., with a single predefined c (usually 1, k, or k −1).

It is also noteworthy that a "k-class-model" comprises the overall set of the k different class-models built for each class, which are computed together and evaluated against one another. In that sense, it is conceptually different from just k independent one-class classifiers, in other words, it is a compliant class-modelling method which is evaluated as such.

2.1.7. Column selection for k-classes (k ≥ 6)

The approach in the previous section is no viable with $k \ge 6$. For example, if $k = 6$, the complete code has 31 binary learners, and the possible code matrices with $c = 15$ or 16 would be 300540195 ≈ 3.0×10^8 and increasing with the number of classes.

For this reason, instead of computing the five criteria for the whole family of possible code matrices, a subset of m matrices with c columns is randomly selected and the procedure in section 2.1.1 is applied to the m matrices. In this situation, it is expected that the maximum of the desirability function is reached in a single matrix **M**, reason why the procedure is repeated several times (taking samples of m matrices each time).

2.2. Decoding. Making a k-class-model via PLS2

Step 3 in section 2.1 (irrespective of k) involves the computation of k-class-models from the predicted responses of a PLS2 model fitted with the corresponding **Y**, whose rows are selected from the optimal coding matrices, according to the matching class.

There are bibliographic references of use of ECOC matrices to discriminate among k classes by using c binary/ternary learners, that is, the discriminant rule assigns all the objects to one, and only one, class. There is not object without a predicted class or any ambiguity in the assignation. The class-modelling strategy followed in the present paper, widely spread in chemometrics (particularly in food authentication), essentially differs from this approach because there can be objects outside all class-models and objects that belong to more than one class-model.

As it has been pointed out, the use of PLS as binary class-modelling based on a hypothesis test on the distribution of the predicted responses in the two classes is very versatile and efficient [55–58].

The possibility of fitting several responses with a PLS model, the so-called PLS2 model which finds a compromise between the covariance of predictors and responses, makes the study of the "hyphenation" PLS2-ECOC interesting given that all the found approaches of using PLS2 as a classifier always use OVA for encoding classes in the response matrix (or the OVAv to avoid collinearity among responses). With the criteria in Eqs. (2)–(6), these two options would be poorly valued (see Table 1). Moreover, their codeword length c is directly determined by the number of classes, that is, c is either k or k-1.

An advantage of PLS models is the definition of a kind of *domain* for the model, the PLS-box [59], which is the region in the space of the latent variables bounded by the critical values for T and Q^2 statistics at a given confidence level. As all the responses are fitted together, PLS2 provides the additional advantage of having a common PLS-box for the c binary learners.

Specifically, for a given problem, as usual, $\mathbf{X} = (x_{ij})$, $i = 1, \ldots, N$, $j = 1, \ldots, v$ will denote the predictor data matrix, so that, x_{ij} is the value that the j-th predictor takes on the i-th object.

Each object belongs to one of k classes, and following the steps in section 2.1, there is a coding matrix **M** with c columns (length of the codeword for these k classes).

The response matrix on its part, $\mathbf{Y} = (y_{ij})$ $i = 1, \ldots, N$, $j = 1, \ldots, c$ with $y_{ij} \in \{-1, 1\}$, is the matrix whose rows are copies of the corresponding row in **M**, according to the class the object belongs to.

The procedure for building a k-class-model that follows is also itemized to facilitate its comprehension.

Stage 1.- Fitting a prediction model.

X and **Y** are used to fit a PLS2 model with a given number of latent variables, namely a linear function $f : \mathbb{R}^v \to \mathbb{R}^c$. Provided that an object $(x_1, \ldots, x_v)$ is in the PLS-box, its predicted values in eq. (10) are those that correspond to each single classifier f_i up to the c binary learners.

$$f(x_1, \ldots, x_v) = (f_1(x_1, \ldots, x_v), f_2(x_1, \ldots, x_v), \ldots, f_c(x_1, \ldots, x_v)) = (\hat{y}_1, \ldots, \hat{y}_c) \tag{10}$$

Then, considering the corresponding *super classes* $\mathbb{A}$ and $\mathbb{B}$ for each binary learner, there are two subsets of predicted values, f_i ($\mathbb{A}$) and f_i ($\mathbb{B}$).

Stage 2.- Fitting probability distributions to the predicted responses.

It cannot be assumed that the values f_i $(\mathbb{A})$ and f_i $(\mathbb{B})$ follow a pre-stablished probability distribution (e.g., normal) [60]. Therefore, for each binary classifier f_i, a univariate kernel density [61] is estimated, separately for f_i $(\mathbb{A})$ and f_i $(\mathbb{B})$, by a normal kernel smoothing function and an optimal bandwidth for estimating normal densities.

Stage 3.- Class-models for $\mathbb{A}$ and $\mathbb{B}$ for each f_i , $i = 1, \ldots, c$.

For given probabilities α_i and β_i (that can be different for each binary learner), the fitted distribution in stage 2 is used to compute critical values $CV_i(\mathbb{A})$ and $CV_i(\mathbb{B})$ with the conditions established in Eqs. (11) and (12).

$$P\{\hat{y}_i \in f_i(\mathbb{A}) \mid \hat{y}_i \leq CV_i(\mathbb{A})\} = \alpha_i \tag{11}$$

$$P\{\hat{y}_i \in f_i(\mathbb{B}) \mid \hat{y}_i \leq CV_i(\mathbb{B})\} = \beta_i \tag{12}$$

Notice that the definitions in Eqs. (11) and (12) imply that α_i would be a large value, say 0.95, whereas β_i would be close to zero, e.g., 0.10.

Stage 4.- Decoding.

For each object $\mathbf{x} = (x_1, \ldots, x_v)$ in the PLS-box, if $\hat{y}_i = f_i(x_1, \ldots, x_v) \leq CV_i(\mathbb{A})$, then the i-th coordinate on the decoding vector will be -1, and if $\hat{y}_i = f_i(x_1, \ldots, x_v) > CV_i(\mathbb{B})$, the coordinate will be $+1$.

For each binary learner, the objects whose predicted values are between the corresponding critical values present two possibilities: i) if $CV_i(\mathbb{A}) < CV_i(\mathbb{B})$, the object will not be assigned, neither $+1$ nor -1, and ii) when $CV_i(\mathbb{A}) > CV_i(\mathbb{B})$, the object is assigned to both $+1$ and -1 (intersection).

By applying the rule with all classifiers, each object will be related to zero, one, or more decoding vectors (vectors of length c with the assignations).

Stage 5.- Assignation of objects to class-models.

Finally, the object $\mathbf{x} = (x_1, \ldots, x_v)$ is inside the i-th class-model if the codeword of the i-th class in $\mathbf{M}$ is one of the decoding vectors related to it.

Depending on the number of decoding vectors associated to the object in stage 4, $\mathbf{x}$ can be inside one or several class-models, or even outside all of them. In other words, we have a k-class-model.

2.3. Model assessment

The characteristics of class-models are usually evaluated in term of sensitivity and specificity. Sensitivity refers to the capacity of the class-model to contain its own objects and specificity is related to the proper rejection of foreign objects. They are estimated, respectively, as the rate of objects of the class correctly inside the class-model and the rate of objects outside the class that are also outside the class-model. Consequently, both values vary between 0 and 1, the nearer to 1 the better the model, though in general sensitivity and specificity behave oppositely.

With k classes, there is k values of sensitivity and specificity. However, it is more informative to break down the specificity of each class-model into specificities computed between pairs of classes. By arranging sensitivities and pair-wise specificities in matrix form, each k-class-model can be evaluated in terms of a sensitivity-specificity matrix $\mathbf{S} = (s_{ij})$, which is a $k \times k$ square matrix with $s_{ij} \in [0, 1]$. In s_{ij} the first subindex (i) refers to the real class (C_i) an object belongs to and the second (j) to whether it is inside the class-model constructed for C_j. Consequently, the main diagonal contains sensitivities, and the off-diagonal terms, pairwise specificities, with the different specificities for each individual class-models in the columns of matrix $\mathbf{S}$.

The evaluation of the performance of the k-class-models for Step 3 in section 2.1 will be based on the sensitivity-specificity matrices, whose information will be summarized by an index grounded in the notion of entropy. This performance measure, DMCEN (Diagonal Modified Confusion Entropy), has been largely discussed in Ref. [24] and its code is available through MATLAB Central File Exchange [62]. Although the index also varies between 0 and 1 for every matrix $\mathbf{S}$, contrary to sensitivity and specificity, the lower the DMCEN value, the better the k-class-model related to $\mathbf{S}$.

In the PLS2-CM strategy, a family of code matrices with different code length c for each k is used to obtain the k-class-model with the best performance. As a consequence, the characteristics of the dataset at hand influence the final selection of the coding matrix and hence of the k-class-model. So, the procedure developed is a data-driven strategy.

3. Materials and methods

3.1. Simulated datasets

To better understand the behavior of PLS2-CM, some data are simulated as FTIR spectra, built from spectra of toluene, xylene, and naphthalene (solvent and concentrations) recorded in the spectral region from 650 to 4000 cm^{-1} using an Agilent DialPath module for the Agilent Cary 630 FTIR. Figure S1 in the supplementary material shows the three experimental spectra, as well as the experimental details to obtain them.

The procedure to generate the simulated spectra, similar to the one in Ref. [63], is as follows.

For each class, n spectra of length l ($l = 1798$ in the present case) are simulated. They form matrix $\mathbf{X}$ of size $n \times l$. The assumption is that every spectrum is the sum of two contributions, according to equation (13).

$$\mathbf{X} = \mathbf{X}_u + \mathbf{X}_d \tag{13}$$

where $\mathbf{X}_u$ represents the part of the spectra that is common to all the classes, and $\mathbf{X}_d$ the part specific to each class.

To build the spectra of every type, the common part is computed as in equation (14).

$$\mathbf{X}_u = \mathbf{t}_u \mathbf{p}_{napht}^T \tag{14}$$

with $\mathbf{t}_u$ a vector of scores, with n random numbers following a folded-normal distribution with parameters (0.6, 0.1), which are the mean and standard deviation of the raw normal distribution [64], and $\mathbf{p}_{napht}^T$ is the row vector with the FTIR spectrum of naphthalene. The specific part is defined in equation (15).

$$\mathbf{X}_d = \mathbf{X}_{d1} + \mathbf{X}_{d2} = \mathbf{t}_{d1} \mathbf{p}_{tol}^T + \mathbf{t}_{d2} (\mathbf{1} + \mathbf{t}_{d1}) \mathbf{p}_{xyl}^T \tag{15}$$

Being $\mathbf{t}_{d1}$ and $\mathbf{t}_{d2}$ vectors with also n scores extracted from two folded-normal distributions with parameters (m_1, s_1) and (m_2, s_2), respectively.

For the four cases that will be studied in section 4.1, scores $\mathbf{t}_u$ $\mathbf{t}_{d1}$, and $\mathbf{t}_{d2}$ are generated independently for each class and simulated case, while the standard deviations s_1 and s_2 are kept at 0.05, so that the classes only differ in the means m_1 and m_2.

Table S1 in the supplementary material shows the different values of (m_1, m_2) defined for the four cases studied. For cases 1 and 2, the means of the four classes are located at the vertices of a square of side 0.2 and 0.1, respectively, as it is illustrated in Fig. 1a) for side 0.2. Cases 3 and 4 have five classes whose means are at the vertices of a regular pentagon with side 0.2 and 0.1, respectively, centered at (0.5, 0.5). Fig. 1b) also depicts the geometric layout of the points (m_1, m_2) for case 3.

The same structure is observed in Fig. 2 that graphs the scores of a Principal Component Analysis (PCA) of the generated spectra, for cases 1 to 4, Fig. 2a) to 2d) respectively. Comparing to Fig. 1, the projections of the different classes maintain the relative position of the points (m_1, m_2) defining the classes. It is also observed that the confusion among classes increases when reducing the distance between vectors (m_1, m_2) when passing from Case 1 to 2, or from Case 3 to 4, and also among the classes of the same case.

Fig. 3 shows the 400 spectra of the four classes simulated for case 1, where it is difficult to observe the differences among classes, which are depicted separately per class in Figure S2 in the supplementary material.

3.2. Non-simulated datasets

Table 2 summarizes the numerical characteristics of the different

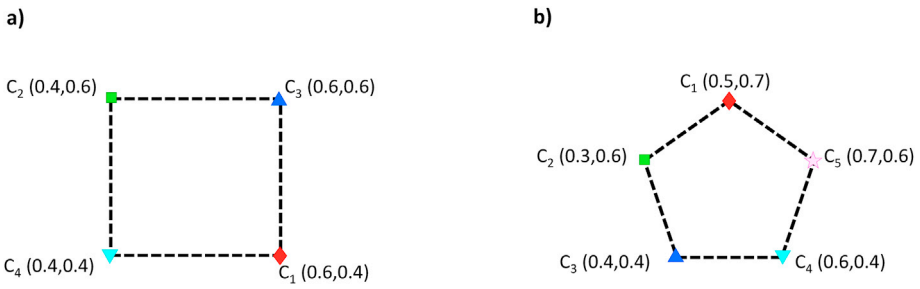

Fig. 1. Schema of (m_1,m_2) for: a) Case1, b) Case 3, of section 3.1.

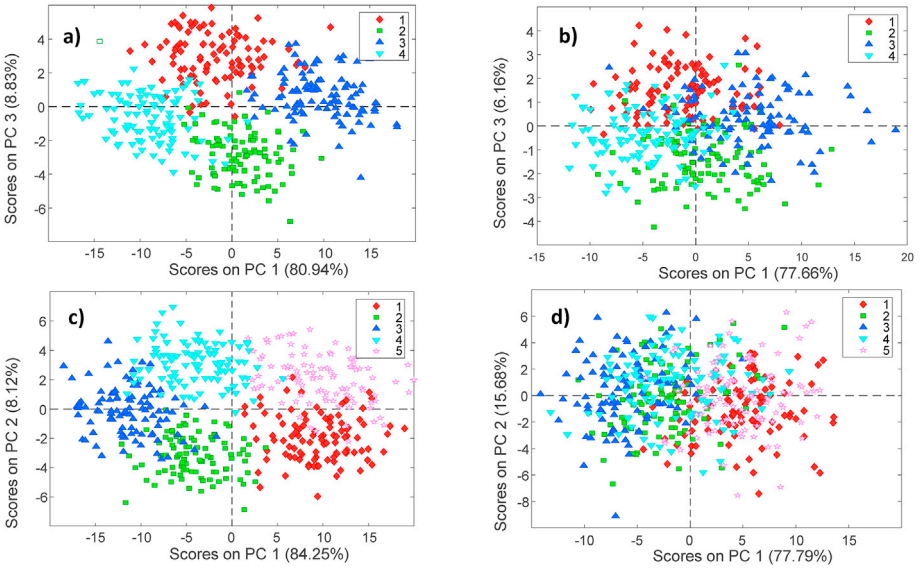

Fig. 2. PCA scores of the simulated spectra, section 3.1. a) Case 1; b) Case 2; c) Case 3; d) Case 4.

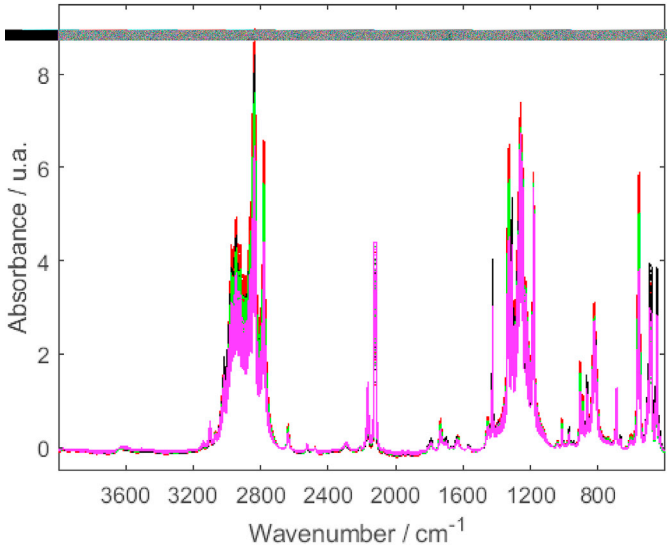

Fig. 3. Four hundred simulated spectra in Case 1. Class 1 in black, class 2 in green, class 3 in red, and class 4 in magenta. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

datasets studied in this work. The *Nutrient Monitoring* dataset has 4 classes. The dataset has 414 objects and 11 variables. The first 5 variables are macronutrients (Mg, Ca, K, P, N) and the last 6 micronutrients (Na, B, Cu, Zn, Mn, Fe), all of them measured on two organs of the plant over two periods of time. These differences constitute the 4 classes, namely C_1 (organ 1-period 1), C_2 (organ 2-period 1), C_3 (organ 1-period 2), and C_4 (organ 2-period 2).

Table 2
Description of the datasets used in the analysis.

Dataset	No. objects	No. Variables	No. classes	No. objects per class
Nutrient monitoring	414	11	4	84, 84, 122, 122
GranaPadano1	95	11	5	11, 21, 18, 19, 18
GranaPadano2	95	18	5	11, 21, 18, 19, 18
GranaPadano3	95	29	5	11, 21, 18, 19, 18
Thyroid	2642	5	4	17, 33, 25, 2567
Dermatology	366	33	6	112, 61, 72, 49, 52, 20
Glass	214	9	6	70, 76, 17, 13, 9, 29
Olives	572	8	9	25, 56, 206, 36, 65, 33, 50, 50, 51

Grana Padano's datasets refer to samples of Grana Padano cheese (registered as European Union Protected Designation of Origin) on which casein fractions have been obtained from casein hydrolysis, measured by two different electrophoresis techniques [65]. These 3 datasets have 5 classes (C_1, C_2, C_3, C_4, and C_5) that correspond to different months of cheese curing (4, 7, 9, 12, and 15 months, respectively) being 12 months the minimum curing required for this cheese. The number of objects per class can be seen in the 5th column of Table 2. There are 95 samples with different number of variables: The *GranaPadano1* dataset contains the 11 variables obtained by means of Poly-Acrylamide Gel Electrophoresis (PAGE), whereas *GranaPadano2* has 18 variables obtained when

characterizing the cheese by means of Poly-Acrylamide Gel Isoelectric Focusing (PAGIF). *GranaPadano3* contains 29 variables resulting from the fusion of the different measurements in the two previous datasets.

The *Thyroid* dataset [66] consists of 2642 patients distributed in four classes: C_1, replacement therapy; C_2, underreplacement; C_3, overreplacement; and C_4, negative. The five continuous variables have been selected as predictor variables: TSH, thyroid stimulating hormone; T3, triiodothyronine; TT4, total L-thyroxine; T4U, thyroxine uptake; and FTI, free thyroxine index.

The following datasets, *Dermatology* and *Glass* with the characteristics in Table 2, have also been taken from the UCI Machine Learning Repository [65].

Finally, some methods of multivariate data analysis have been applied to the recognition of the geographical origin of food. As the research on samples of Mediterranean olive oils has shown [67], the chemometric methods can be very useful to control the geographical origin of olive oil samples. The *Olives* dataset used here contains the percentages of eight fatty acids (palmitic, palmitoleic, stearic, oleic, linoleic, eicosanoic, linolenic, and eicosenoic) found in the lipid fraction of 572 Italian olive oils, from 9 growing regions: 4 from southern Italy (North and South Apulia, Calabria, Sicily), two from Sardinia (Inland and Coastal) and 3 from northern Italy (Umbria, East and West Liguria).

In summary, datasets with different characteristics are used to check the proposal, the number of classes ranging from 4 to 9. In terms of the number of objects, the largest dataset (*Thyroid*) with thousands of objects presents a noticeable class imbalance, whereas classes within the remaining datasets (tens or hundreds of objects) are generally more balanced. In addition, the number of predictors is also quite diverse (ranging from only 5 to 33).

3.3. Application of the proposed procedure

In all the analyses conducted in the search for the optimal ECOC matrices (Step 2, section 2.1), the global desirability D of the q or m candidate ECOC matrices has been evaluated with the weights (3/10, 3/10, 2/10, 1/10, 1/10) in equation (9).

When k is equal to 4 and 5, Table 3 shows the number of different ECOC matrices for each combination of number of classes (k) and code length, c. From them, the number of ECOC matrices where the maximum D is reached is in brackets.

For example, for $k = 5$ classes and code length equal to 8, there are 6435 different ECOC matrices, 16 of which reach the maximum of overall desirability. Each of these 16 matrices are used to encode the classes defining different $\mathbf{Y}$ matrices and, thus, different PLS2 models.

When $k = 6$, if c is 4 or 5, it is still possible to perform an exhaustive search within the space of all possible ECOC matrices, which are 31,465 and 169,911, with 360 and 60 ECOC matrices with maximum D, respectively. Beyond that, a manageable sample of matrices must be taken.

For a particular problem, k is known. In the studies conducted for the present paper, when k is equal to 4 or 5, c ranges from k-1 to the number of columns of the binary complete ECOC matrix (2^{k-1}-1) and similarly, when $k = 6$ and c is equal to 4 or 5.

When $k > 6$ and $c > 5$, the search for an ECOC matrix with maximum D is performed with the procedure described in section 2.1.2. This process is repeated 20 times, so 20 different coding matrices are available to solve the modelling problem.

For each ECOC matrix with maximum D, the class each object of $\mathbf{X}$ belongs to is encoded, thus obtaining the matrix $\mathbf{Y}$ with which the PLS2 models are constructed, also varying the number of latent variables from 1 to min{9, v-1}, with v being the number of predictor variables of the $\mathbf{X}$ dataset. In all cases, the corresponding PLS-box has been constructed at 0.99 confidence level for T^2 and Q statistics.

In the modelling of each superclass, $\mathbb{A}$ and $\mathbb{B}$, probabilities $\alpha_i = 0.99$ and $\beta_i = 0.01$ are defined for all binary learners (see equations (11) and (12)).

Finally, the calculation of the DMCEN of all the ECOC matrices built is conducted and the matrix with minimum DMCEN is selected.

4. Results and discussion

This section shows the impact of using encodings other than OVA, which is the usual one in chemometrics. Therefore, the results of using OVA are compared with the ones obtained with the procedure explained in the previous sections, PLS2-CM, which includes the selection of the code length c, coding matrix $\mathbf{M}$, and the number of latent variables of the PLS2 model.

The results obtained with PLS2 and OVA and with PLS2-CM are compared on the basis of the reached DMCEN value as well as on the sensitivity-specificity matrices of the k-class-models.

4.1. Simulated datasets

The results of modelling the four simulated datasets with the proposed procedure is shown in Table 4 and discussed below.

The sensitivity-specificity matrices for the 4-class-models computed for the dataset with four classes at the vertices of a square and more distant (Case 1) show that the 4-class-models result in large sensitivities and perfect specificities between non-adjacent classes in Fig. 1a) as $s_{12} = s_{21} = 1$ and $s_{34} = s_{43} = 1$. However, specificities from adjacent classes are lower as expected. While they do not exceed 0.90 with the OVA encoding, the ECOC approach using 5 binary learners raises them mostly well above 0.90, thus getting a better class-modelling as the decrease in DMCEN (from 0.27 to 0.17) reflects.

When the four classes are closer, as in Case 2 illustrated in Fig. 2b), sensitivities remain high, but specificities decrease markedly as expected. With the OVA encoding, they decrease not only for the adjacent classes (whose s_{ij} fall below 0.30) but also for the non-adjacent ones, whose specificities, $s_{12} = 0.37$, $s_{21} = 0.65$ and $s_{34} = 0.59$, $s_{43} = 0.49$, are not as close to 1 as in the previous case. Although specificities increase overall with the ECOC encoding, which happens to use the same number of binary learners as OVA, i.e. 4, this lack of specificity is not fully solved due to the actual closeness between the simulated classes.

Case 3, where the means are located at the vertices of a regular pentagon centered on (0.5, 0.5) with larger side (0.2), Fig. 1b), shows a similar pattern as Case 1. Sensitivities are high but the problem of specificity focuses on adjacent classes: the values for s_{12}, s_{21}, s_{23}, s_{32}, s_{34}, s_{43}, and s_{54}, s_{45} range from 0.50 to 0.75. Fortunately, the ECOC proposal

Table 3
Number of ECOC matrices for each number of classes, k, and code length, c. In brackets, the number of ECOC matrices in which the maximum value of the overall desirability is reached with weights (3/10, 3/10, 2/10, 1/10, 1/10) for the five criteria used.

	c												
	3	4	5	6	7	8	9	10	11	12	13	14	15
$k = 4$	35 (12)	35 (3)	35 (12)	6 (4)	1 (1								
$k = 5$	455 (12)	1365 (3)	3003 (3)	5005 (7)	6435 (36)	6435 (16)	5005 (180)	3003 (30)	1365 (5)	455 (30)	105 (20)	15 (15)	1 (1)
$k = 6$	4495 (120)	31,465 (360)	169,911 (60)	7×10^5	3×10^6	8×10^6	2×10^7	4×10^7	8×10^7	1×10^8	2×10^8	3×10^8	3×10^8

Table 4
Class-models for the simulated spectral datasets. Codification: OVA, One versus All; ECOC, Error Correcting Output Code; *c*, code length. Measure of model performance: DMCEN, Diagonal Modified Confusion Entropy.

	OVA		ECOC	
	DMCEN (*c*)	Sensitivity-specificity matrix	DMCEN (*c*)	Sensitivity-specificity matrix
Case1	0.2654 (4)	$\begin{pmatrix} 0.96 & 1.00 & 0.87 & 0.82 \\ 1.00 & 0.99 & 0.80 & 0.78 \\ 0.88 & 0.90 & 0.99 & 1.00 \\ 0.83 & 0.85 & 1.00 & 0.98 \end{pmatrix}$	0.1725 (5)	$\begin{pmatrix} 0.97 & 1.00 & 0.98 & 0.93 \\ 1.00 & 0.99 & 0.96 & 0.91 \\ 0.94 & 0.96 & 0.97 & 1.00 \\ 0.91 & 0.94 & 1.00 & 0.98 \end{pmatrix}$
Case 2	0.4615 (4)	$\begin{pmatrix} 1.00 & 0.37 & 0.24 & 0.26 \\ 0.65 & 0.97 & 0.26 & 0.19 \\ 0.30 & 0.25 & 0.96 & 0.59 \\ 0.25 & 0.20 & 0.49 & 0.97 \end{pmatrix}$	0.4486 (4)	$\begin{pmatrix} 0.97 & 0.41 & 0.33 & 0.39 \\ 0.75 & 0.96 & 0.44 & 0.40 \\ 0.45 & 0.35 & 0.97 & 0.72 \\ 0.38 & 0.24 & 0.59 & 0.99 \end{pmatrix}$
Case 3	0.2818 (5)	$\begin{pmatrix} 0.98 & 0.66 & 1.00 & 1.00 & 0.59 \\ 0.70 & 1.00 & 0.58 & 1.00 & 1.00 \\ 1.00 & 0.58 & 0.96 & 0.53 & 1.00 \\ 1.00 & 1.00 & 0.74 & 0.99 & 0.74 \\ 0.69 & 1.00 & 1.00 & 0.72 & 0.98 \end{pmatrix}$	0.1397 (9)	$\begin{pmatrix} 0.96 & 0.96 & 1.00 & 1.00 & 0.95 \\ 0.97 & 0.99 & 0.92 & 1.00 & 1.00 \\ 1.00 & 0.97 & 0.98 & 0.93 & 1.00 \\ 1.00 & 1.00 & 0.94 & 0.99 & 0.96 \\ 0.95 & 1.00 & 1.00 & 0.95 & 0.98 \end{pmatrix}$
Case 4	0.4411 (5)	$\begin{pmatrix} 0.98 & 0.29 & 0.79 & 0.80 & 0.25 \\ 0.30 & 0.98 & 0.27 & 0.74 & 0.74 \\ 0.82 & 0.20 & 0.94 & 0.20 & 0.62 \\ 0.74 & 0.70 & 0.23 & 0.97 & 0.21 \\ 0.24 & 0.74 & 0.67 & 0.20 & 0.96 \end{pmatrix}$	0.4295 (9)	$\begin{pmatrix} 0.98 & 0.36 & 0.83 & 0.85 & 0.44 \\ 0.44 & 0.98 & 0.44 & 0.81 & 0.76 \\ 0.77 & 0.35 & 0.95 & 0.37 & 0.81 \\ 0.79 & 0.80 & 0.29 & 0.95 & 0.46 \\ 0.41 & 0.74 & 0.77 & 0.40 & 0.95 \end{pmatrix}$

with 9 binary learners notably improves the 5-class-model (DMCEN 0.1397 vs 0.2818 in OVA), achieving specificities well above 0.90 even for the adjacent classes.

Finally, when the five classes are closer (Case 4 depicted in Fig. 2d)), the 5-class-model has lower quality (compared to Case 3), regardless the encoding system (DMCEN above 0.40 in both systems). Sensitivities remain high, but the lack of specificity between adjacent classes is more obvious (s_{ij}<0.30 for OVA encoding) even if an ECOC approach is applied, as specificities increase but those of adjacent classes do not reach 0.50.

In summary, through these datasets, whose degree of overlap between classes is under control, it is shown that the PLS2 with ECOC encoding procedure reflects both the overlap caused by the proximity between the classes of spectra (comparing the distance 0.2 and 0.1) and the one caused by the relative position of the classes, contiguous classes vs distant classes, when the distance remains constant. In all cases, the use of the ECOC encoding improves 4 (5)-class-models.

4.2. Non-simulated datasets

The results of the application of the suggested procedure to datasets listed in Table 2 are shown in Table 5 and discussed below individually.

Nutrient monitoring: The quality of the 4-class-model for the *Nutrient monitoring* dataset with OVA encoding is reflected by a low DMCEN (0.21), high sensitivities for all the classes but low specificity between several classes. The poorest results in specificity have to do with classes C_1 versus C_3 ($s_{31} = 0.55$, $s_{13} = 0.73$) as well as classes C_2 versus C_4 ($s_{24} = 0.46$, $s_{42} = 0.69$).

The improvement brought by the ECOC encoding ($c = 5$) on these specificities leads to a better overall model (DMCEN = 0.16), where all specificities exceed 0.70 at the cost of a minimal fall of sensitivities.

GranaPadano1: 5-class-models for the *GranaPadano1* dataset have high sensitivities for all the classes but very bad specificities for some 'contiguous' (similar cheese curing period) classes, reaching the worst possible values in $s_{32} = 0$ and $s_{34} = 0$ with OVA coding.

This problem is slightly improved with the ECOC encoding using the 6 suggested binary learners, thus achieving greater specificities overall, even in the worst cases mentioned which become $s_{32} = 0.61$ and $s_{34} = 0.28$.

GranaPadano2: A quick look at the sensitivity-specificity matrix for *GranaPadano2* dataset allows us to conclude that variables measured by means of PAGIF provide less sensitive but more specific class-models than those observed with PAGE in *GranaPadano*1.

DMCEN improves with ECOC (from 0.3519 to 0.2919) due to the improved specificity of contiguous classes (similar curing period of cheese) without losing sensitivity with respect to OVA encoding.

GranaPadano3: According to the results just shown, it is not surprising that the 5-class-models for the *GranaPadano3* dataset shows that the fusion of data greatly improves the class-models with the two encodings, with low values of DMCEN (0.0913 with OVA and 0.0707 with ECOC). It is noteworthy that the class-model of C_4 (12-months), of particular interest in this context, has a value of 1 for all pairwise specificities (in both rows and columns except for $s_{45} = 0.95$ with OVA), and sensitivity 0.95 with OVA-encoding, whereas using the ECOC proposal this class becomes perfectly defined, with the sensitivities of the remaining classes barely affected.

Thyroid: Under the OVA encoding, the 4-class-model for the *Thyroid* dataset shows two class-models with sensitivity below 0.90 (C_1 and C_3), but its main flaw is the lack of specificity affecting all the class-models, mainly the one for C_4, where low values in s_{14}, s_{24} and s_{34} imply that the class-model of C_4 accepts patients from all other classes. It is worth mentioning that the class-modelling has to deal with three minority classes of patients, C_1 (replacement therapy), C_2 (under-replacement) and C_3 (over-replacement), whose sizes do not exceed a few tens, against a majority class C_4 (size 2567).

The problems are not solved by the proposed ECOC strategy, but the overall picture slightly improves due to the larger sensitivities of C_1 and C_3 and a better specificity in relation to C_1 and C_2 (higher values of s_{21} and s_{12}). Although specificity problems remain, 3 binary learners are enough to achieve even a better result than the four binary learners usually applied.

Dermatology: The high quality of the 6-class-models to determine the type of Erythematous-Squamous Disease (*Dermatology* dataset) is reflected by a low DMCEN of approximately 0.07 with both encodings. The OVA encoding provides a sensitivity-specificity matrix with just one specificity below 0.90. As $s_{24} = 0.82$, the class-model for C_4 (pityriasis rosea) is including patients actually having seborrheic dermatitis, C_2. Similarly, the class-model for C_2 is also accepting like 8% of patients really having C_4 ($s_{42} = 0.92$) so some confusion occurs between these two diseases.

Although there is obviously not much room for improvement, the ECOC approach involves the same number of binary learners here (from $c = 4$ to 25, the smallest DMCEN is reached at $c = 6$). The effect seems to be that it rebalances the sensitivities and specificities of the two classes involved ($s_{22} = 0.95$, $s_{44} = 0.94$, $s_{24} = 0.84$ and $s_{42} = 0.84$), but does not overcome the confusion mentioned. However, it slightly enhances the overall class-modelling, increasing the sensitivities of most class-models (around 0.95 or more) and keeping the remaining specificities at their maximum value.

Glass: Regarding the class-modelling of six types of glasses (*glass*

Table 5
Class-models for datasets in Table 2. Codification: OVA, One Versus All; ECOC, Error Correcting Output Code. Measure of model performance: DMCEN, Diagonal Modified Confusion Entropy. *c* is the codeword length.

	OVA DMCEN (*c*)	Sensitivity-specificity matrix	ECOC DMCEN (*c*)	Sensitivity-specificity matrix
Nutrient monitoring	0.21059 (4)	$\begin{pmatrix} 0.94 & 1.00 & 0.73 & 1.00 \\ 1.00 & 0.95 & 1.00 & 0.46 \\ 0.55 & 1.00 & 0.98 & 1.00 \\ 1.00 & 0.69 & 0.99 & 1.00 \end{pmatrix}$	0.16481 (5)	$\begin{pmatrix} 0.94 & 1.00 & 0.96 & 1.00 \\ 1.00 & 0.90 & 1.00 & 0.77 \\ 0.94 & 1.00 & 0.97 & 1.00 \\ 1.00 & 0.73 & 1.00 & 0.98 \end{pmatrix}$
Grana-Padano1	0.3918 (5)	$\begin{pmatrix} 1.00 & 0.89 & 0.95 & 0.95 & 1.00 \\ 0.57 & 1.00 & 0.52 & 0.05 & 0.38 \\ 1.00 & 0.00 & 1.00 & 0.00 & 0.78 \\ 0.95 & 0.42 & 0.26 & 0.95 & 0.11 \\ 1.00 & 0.83 & 0.83 & 0.72 & 1.00 \end{pmatrix}$	0.3806 (6)	$\begin{pmatrix} 1.00 & 0.84 & 0.95 & 1.00 & 1.00 \\ 0.67 & 1.00 & 0.52 & 0.38 & 0.67 \\ 1.00 & 0.61 & 0.94 & 0.28 & 0.72 \\ 1.00 & 0.63 & 0.53 & 0.95 & 0.32 \\ 1.00 & 0.83 & 0.83 & 0.56 & 1.00 \end{pmatrix}$
Grana-Padano2	0.3519 (5)	$\begin{pmatrix} 0.89 & 0.89 & 0.89 & 0.95 & 0.89 \\ 0.90 & 0.95 & 0.76 & 0.76 & 0.76 \\ 0.89 & 0.83 & 0.94 & 1.00 & 0.83 \\ 0.95 & 0.95 & 0.95 & 1.00 & 0.95 \\ 0.95 & 0.83 & 0.83 & 0.83 & 0.94 \end{pmatrix}$	0.2919 (7)	$\begin{pmatrix} 0.89 & 0.63 & 0.84 & 1.00 & 1.00 \\ 0.90 & 0.95 & 0.57 & 0.86 & 0.90 \\ 1.00 & 1.00 & 0.94 & 1.00 & 0.89 \\ 0.95 & 0.95 & 0.95 & 1.00 & 0.95 \\ 1.00 & 0.89 & 0.89 & 0.67 & 0.94 \end{pmatrix}$
Grana-Padano3	0.0913 (5)	$\begin{pmatrix} 0.95 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.95 & 0.95 & 1.00 & 0.90 \\ 1.00 & 1.00 & 0.94 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 0.95 & 0.95 \\ 1.00 & 0.94 & 1.00 & 1.00 & 1.00 \end{pmatrix}$	0.0707 (7)	$\begin{pmatrix} 0.95 & 0.95 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.95 & 0.95 & 1.00 & 1.00 \\ 1.00 & 1.00 & 0.94 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.94 & 1.00 & 1.00 & 0.94 \end{pmatrix}$
Thyroid	0.4259 (4)	$\begin{pmatrix} 0.88 & 0.65 & 0.18 & 0.12 \\ 0.88 & 1.00 & 1.00 & 0.00 \\ 0.16 & 1.00 & 0.84 & 0.16 \\ 0.81 & 0.08 & 0.93 & 0.97 \end{pmatrix}$	0.3764 (3)	$\begin{pmatrix} 0.94 & 0.71 & 0.18 & 0.06 \\ 0.91 & 1.00 & 1.00 & 0.00 \\ 0.16 & 1.00 & 0.96 & 0.16 \\ 0.83 & 0.07 & 0.92 & 0.97 \end{pmatrix}$
Dermatology	0.0725 (6)	$\begin{pmatrix} 0.97 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.93 & 1.00 & 0.82 & 1.00 & 1.00 \\ 1.00 & 1.00 & 0.99 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.92 & 1.00 & 0.96 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 0.94 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.90 \end{pmatrix}$	0.0690 (6)	$\begin{pmatrix} 0.98 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.95 & 1.00 & 0.84 & 1.00 & 1.00 \\ 1.00 & 1.00 & 0.99 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.84 & 1.00 & 0.94 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.95 \end{pmatrix}$
Glass	0.4621 (6)	$\begin{pmatrix} 1.00 & 0.06 & 0.27 & 1.00 & 0.19 & 0.23 \\ 0.21 & 0.95 & 0.36 & 0.89 & 0.16 & 0.16 \\ 0.12 & 0.00 & 1.00 & 1.00 & 0.06 & 0.18 \\ 1.00 & 0.23 & 1.00 & 0.85 & 0.46 & 0.23 \\ 0.89 & 0.33 & 0.56 & 0.89 & 0.89 & 0.33 \\ 0.86 & 0.62 & 0.93 & 0.72 & 0.59 & 0.93 \end{pmatrix}$	0.3934 (4)	$\begin{pmatrix} 0.99 & 0.10 & 0.01 & 1.00 & 0.76 & 1.00 \\ 0.21 & 0.96 & 0.22 & 0.70 & 0.58 & 0.78 \\ 0.00 & 0.18 & 1.00 & 1.00 & 0.94 & 1.00 \\ 1.00 & 0.23 & 1.00 & 0.92 & 0.54 & 0.46 \\ 0.89 & 0.56 & 0.89 & 0.56 & 0.89 & 0.00 \\ 0.93 & 0.93 & 0.97 & 0.90 & 0.86 & 1.00 \end{pmatrix}$
Olives	0.3343 (9)	$\begin{pmatrix} 0.80 & 0.72 & 1.00 & 0.56 & 1.00 & 1.00 & 0.96 & 1.00 & 1.00 \\ 0.75 & 0.98 & 0.75 & 0.07 & 1.00 & 1.00 & 0.70 & 1.00 & 0.98 \\ 0.98 & 0.94 & 1.00 & 0.89 & 1.00 & 1.00 & 0.98 & 1.00 & 1.00 \\ 0.61 & 0.44 & 0.69 & 0.67 & 1.00 & 1.00 & 0.75 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 0.91 & 0.97 & 0.80 & 0.65 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 0.94 & 0.94 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.84 & 1.00 & 0.92 & 0.96 & 1.00 & 0.82 & 0.94 & 0.84 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.98 & 1.00 & 0.94 \end{pmatrix}$	0.2627 (16)	$\begin{pmatrix} 0.92 & 0.92 & 0.92 & 0.84 & 0.92 & 0.92 & 0.96 & 0.92 & 0.92 \\ 1.00 & 0.89 & 0.11 & 0.09 & 1.00 & 0.98 & 0.89 & 0.98 & 1.00 \\ 1.00 & 0.00 & 1.00 & 0.45 & 0.99 & 1.00 & 0.99 & 0.97 & 1.00 \\ 0.97 & 0.75 & 0.75 & 0.81 & 1.00 & 1.00 & 0.97 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.77 & 0.80 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 0.21 & 0.94 & 1.00 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.90 & 0.96 & 0.90 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.96 & 0.98 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.90 \end{pmatrix}$

dataset) according to their oxide content, the traditional OVA encoding leads to poor specificities throughout the whole matrix, particularly in the last two columns, i.e. from glasses coming from tableware, C_5, and headlights, C_6. Values of specificities s_{15}, s_{16}, s_{25}, s_{26}, s_{35}, s_{36} and s_{46} (all of them below 0.25) indicate that the class-models for these two types of glasses (C_5 and C_6) are including glasses from building-windows, C_1 and C_2, vehicle windows C_3, or even containers, C_4. When the ECOC strategy is implemented, specificities involving classes C_5 and C_6 largely rise, while keeping or increasing the sensitivities of almost all the class-models above 0.90. Besides the fall in DMCEN (from 0.46 to 0.39), the class C_6, headlight glasses, becomes much better modeled as figures on both the sixth row and column of the sensitivity-specificity matrix show. Besides, this general improvement is achieved with an ECOC coding matrix of just four binary learners instead of the six used with OVA.

Olives: The last dataset aims to jointly model olive oils of 9 Italian regions from their percentage composition of fatty acids. Starting with OVA encoding, an overall assessment according to DMCEN (0.3343) allows to say that the 9-class-model shows high sensitivities for olive oils of most of the regions, except for C_4, Sicilian oils, which present the lowest sensitivity (0.67), as well as specificity problems related to southern regions, particularly South Apulia (C_2), and to a lesser extent Apulia (C_1). Oils from the remaining areas seem to be properly modeled, notably those collected in the northern Italy (West Liguria, C_8, and Umbria, C_9), with large sensitivities and almost all specificities at their maximal values, as the two last rows and columns of the sensitivity-specificity matrix point out.

Once the suggested ECOC encoding is introduced, the overall performance improves (DMCEN = 0.2627). Concerning Sicilian oils, C_4, specificities generally increase (except for s_{34}), which implies better delimitation regarding southern olive oils and greater sensitivity for this class-model. With respect to C_1, with greater sensitivity (0.92) and specificities mostly exceeding 0.90, a better class-model is achieved so North Apulia olive oils become better delimited. On the contrary, confusion between olive oils from South Apulia (C_2) and Calabria (C_3) is observed (low values of s_{32} and s_{23}). Finally, olive oils from the northern Italy regions of West Liguria, C_8, and Umbria, C_9, remain properly modeled while the class-model for oils of East Liguria, C_7, attains larger sensitivity and specificities.

In the last case, where 9 classes have to be modeled, the ECOC encoding suggested comprises 16 binary learners (virtually without computational cost) which achieve better performance than the 9 in OVA.

This is the most frequent situation observed: the ECOC strategy leads to a larger number of binary learners than the one used in more traditional encoding methods such as OVA.

However, the *Thyroid* and *Glass* datasets share an interesting property: the codeword length is lower in ECOC than in OVA, thus meaning that the number of binary learners with the ECOC approach suggested is

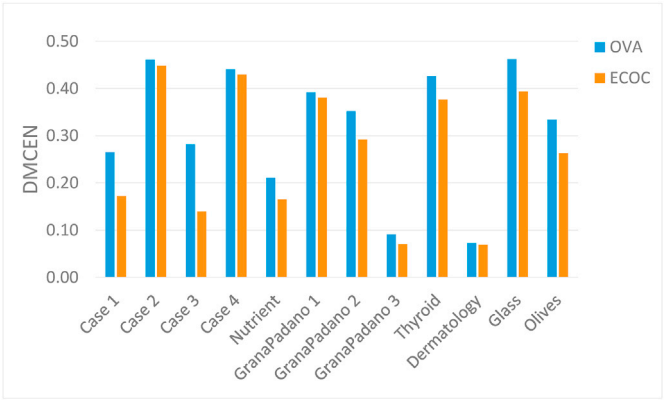

Fig. 4. DMCEN of the different models in the twelve datasets. Blue bars are for OVA encoding, orange ones for ECOC. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

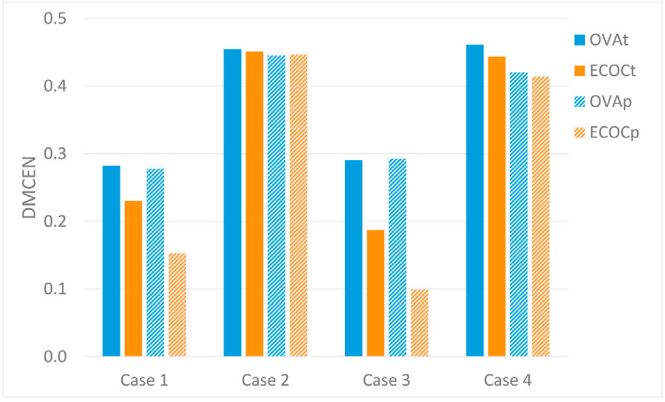

Fig. 5. DMCEN of the different models in the four cases in the simulated spectra. Blue bars are for OVA encoding, orange ones for ECOC. Final t (solid colours) means results in training, p means those in prediction (dashed bars). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

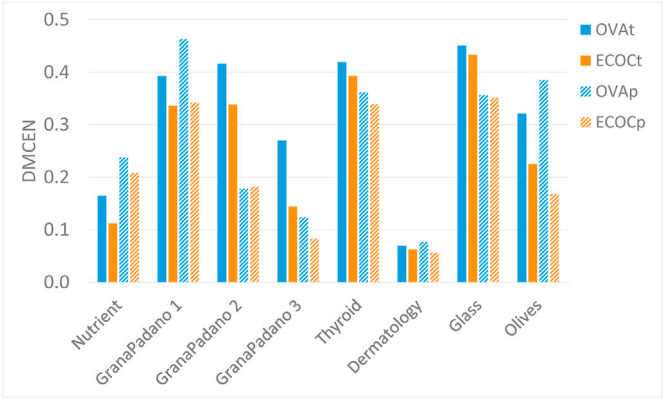

Fig. 6. DMCEN of the different models in the eight non-simulated datasets. Blue bars are for OVA encoding, orange ones for ECOC. Final t (solid colours) means results in training, p means those in prediction (dashed bars). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.

not necessarily higher than the number of classes (see also that in the *Dermatology* dataset this number turns out to be the same).

Overall, the analysis of this series of datasets, with such different characteristics among them, is summarized in Fig. 4 that contains the DMCEN values for all the models, and for both encodings: ECOC matrices, computed by optimizing the five suggested criteria, and the usual OVA. It is clear that the former improves in all cases the quality of the built class-models in relation to the latter.

4.3. Comparing ECOC versus OVA encodings in prediction

The previous section shows how ECOC encoding improves the performance of the *k*-PLS2-CM compliant model. The analysis has been conducted in training, so it is interesting to check whether the behavior is maintained in prediction. With this aim, datasets have been split into two subsets, denoted as TR for training and TS for the external test. As usual, the *k*-PLS2-CM model is built with TR and applied to predict the samples in TS.

Although there are several methods in the literature to obtain TR and TS, those based on distances between objects, such as K–S algorithm by Kennard-Stone [68] or DUPLEX [69], are the most usual.

In chemometrics, the comparative analysis of K–S, which searches for a uniform distribution of the objects, and the methods based on clustering objects [70] are relevant. The recent review by Xu and Goodacre [71] updates the analysis of splitting methods.

To obtain TR and TS, the Kennard-Stone method has been used in the present work. In each class *C*, the method sequentially selects a preset percentage of samples to be in TR in such a way that they are as uniformly distributed as possible in the class, including samples at the boundary as well. The remaining samples (not selected) are kept in the test set, TS.

For each of the 12 sets (four with simulated spectra in section 3.1 and the eight described in Table 2), the training set TR is made with 70% of the objects of each class, independently selected according to K–S algorithm. With the TR, the *k*-PLS2CM compliant model is computed as in section 3.3, and subsequently applied to the 30% of objects per class set aside in the corresponding TS.

The resulting DMCEN in both training (OVAt and ECOCt) and prediction (OVAp and ECOCp) are depicted in Fig. 5 for the simulated spectra and in Fig. 6 for the non-simulated datasets. The corresponding sensitivity-specificity matrices are in the supplementary material, tables S2 and S3.

For the simulated spectra, DMCEN in cases 1 and 3 follows the same behavior as when the models are built with all the objects, section 4.1 and Fig. 4. There are larger differences between OVA and ECOC encodings in case 3 (five classes) than in case 1 (four classes), whereas the differences are shorter when the classes are closer to each other (cases 2 and 4). The results in prediction are similar to those in training with lower DMCEN for ECOC than for OVA encoding.

Similar comments apply for the 8 non-simulated datasets analysed in section 4.2, whose DMCEN values in Fig. 6 maintains the same pattern as that in Fig. 4. Overall, the k-PLS2 compliant class-models are stable in prediction, with lower (better) DMCEN values for the models computed with the ECOC matrices in training. Regarding the DMCEN values computed in prediction (dashed bars in Fig. 6), the same behavior is observed, ECOC encoding improves models as against the OVA ones.

For Grana Padano 2, Grana Padano 3, Thyroid and Glass datasets, irrespective of encoding, the DMCEN values in prediction are lower than in training. This is likely due to the presence of small classes (see Table 2), affecting the selection made by the K–S algorithm, especially when no guarantee exists of objects being evenly distributed within a class.

5. Conclusions

The coupling of the use of optimal ECOC matrices to build PLS2-based class-models shows that response encoding for PLS2 modelling is important and should not be limited to OVA (or OVAv).

In that sense, multi-criteria optimization to obtain proper ECOC matrices is efficient and has not been previously considered in the literature on the subject.

The possibility of constructing ECOC matrices of a prefixed code length and the evaluation of the obtained class-models in terms of sensitivity and specificity by means of DMCEN *inserts* the characteristics of the dataset into the construction of the ECOC matrix itself.

The procedure is flexible enough to incorporate, in a specific class-modelling problem, characteristics of interest in the construction of the optimal ECOC matrix, for example, maintaining the greatest specificity between two groups of classes. To do it, it suffices to define restrictions on the Hamming distances, which would then be included in the search for the optimal desirability.

The assignment of objects to classes defined by probability thresholds is a noteworthy aspect, since it directly links the length of the code with an assignment to the most demanding class.

In all the cases studied, the *k*-class-model obtained has shown an improvement over the model based on class indicator variables (OVA). Moreover, when the structure of the classes is known, like in the simulated spectra, the resulting sensitivity-specificity matrices reflect this structure.

The predictive ability of the compliant class-models is evaluated by setting aside 30% of objects per class by means of the Kennard-Stone algorithm. The results show stable models, with ECOC outperforming OVA encoding.

Author statement

O. Valencia: Conceptualization, Formal analysis, Methodology, Writing - original draft, Writing - review & editing.

M.C. Ortiz: Formal analysis, Conceptualization, Methodology, Supervision, Writing - review & editing, Funding acquisition.

S.Ruiz Formal analysis, Methodology, Supervision Writing - review & editing.

M.S. Sánchez: Formal analysis, Software; Methodology, Supervision, Writing - original draft, Writing - review & editing.

L.A. Sarabia: Formal analysis, Conceptualization, Methodology, Software, Supervision Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is part of the project with reference BU052P20 financed by Junta de Castilla y León, Conserjería de Educación with the aid of European Regional Development Funds.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.chemolab.2022.104614.

References

[1] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[2] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Ann. Stat. 26 (1998) 451–471.
[3] B. Krawczyk, M. Woźniak, F. Herrera, On the usefulness of one-class classifier ensembles for decomposition of multi-class problems, Pattern Recogn. 48 (2015) 3969–3982.
[4] A.-R. Yin, X. Xiang, K. Jingming, Using Hadamard ECOC in Multi-Class Problems Based on SVM. Ninth European Conference on Speech Communication and Technology, 2005.
[5] R.C. Bose, D.K. Ray-Chaudhuri, On A class of error correcting binary group codes, Inf. Control 3 (1960) 68–79.
[6] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.
[7] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, J. Mach. Learn. Res. 1 (2002) 113–141.
[8] S. Escalera, O. Pujol, P. Radeva, On the decoding process in ternary error-correcting output codes, IEE Transactions on pattern Analysis and Machine Intelligence 32 (2010) 120–134.
[9] S. Escalera, O. Pujol, P. Radeva, Separability of Ternary Error-Correcting Output Codes, 2008 19th International Conference on Pattern Recognition.
[10] K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems, Mach. Learn. 47 (2002) 201–233.
[11] S. Escalera, O. Pujol, P. Radeva, Error-correcting output codes library, J. Mach. Learn. Res. 11 (2010) 661–664.
[12] M.A. Bautista, S. Escalera, X. Baró, P. Radeva, J. Vitriá, O. Pujol, Minimal design of error-correcting output codes, Pattern Recogn. Lett. 33 (2012) 693–702.
[13] L. Lei, Y. Song, X. Luo, A new re-encoding ECOC using reject option, Appl. Intell. 50 (2020) 3090–3100.
[14] J.Y. Zou, M.X. Sun, K.H. Liu, Q.Q. Wu, The design of dynamic ensemble selection strategy for the error-correcting output codes family, Inf. Sci. 571 (2021) 1–23.
[15] K.-J. Feng, S.-T. Liong, K.-H. Liu, The design of variable length coding matrix for improving error correcting output codes, Inf. Sci. 534 (2020) 192–217.
[16] K.H. Liu, X.N. Ye, H.Z. Guo, Q.-Q. Wu, Q.Q. Hong, The Design of Soft Recoding-Based Strategies for Improving Error-Correcting Output Codes, Applied Intelligence, 2021, pp. 1–18.
[17] N. García-Pedrajas, D. Ortiz-Boyer, An empirical study of binary classifier fusion methods for multiclass classification, Inf. Fusion 12 (2011) 111–130.
[18] R.S. Smith, T. Windeatt, in: N.C. Oza, et al. (Eds.), Decoding Rules for Error Correcting Output Code Ensembles in Multiple Classifier Systems, Springer, 2005.
[19] R.G. Brereton, Chemometrics for Pattern Recognition, John Wiley and Sons, Ltd, United Kingdom, 2009.
[20] O.Y. Rodionova, P. Oliveri, A.L. Pomerantsev, Rigorous and compliant approaches to one-class classification, Chemometr. Intell. Lab. Syst. 159 (2016) 89–96, https://doi.org/10.1016/j.chemolab.2016.10.002.
[21] R.G. Brereton, One-class classifiers, J. chemometrics 25 (2011) 225–246, https://doi.org/10.1002/cem.1397.
[22] A.L. Pomerantsev, O.Y. Rodionova, Multiclass partial least squares discriminant analysis: taking the right way—a critical tutorial, J. chemometrics (2018), e3030, https://doi.org/10.1002/cem.3030.
[23] B. Krawczyk, M. Galar, M. Woźniak, H. Bustince, F. Herrera, Dynamic ensemble selection for multi-class classification with one-class classifiers, Pattern Recogn. 83 (2018) 34–51, https://doi.org/10.1016/j.patcog.2018.05.01.
[24] O. Valencia, M.C. Ortiz, M.S. Sánchez, L.A. Sarabia, A modified entropy-based performance criterion for class-modelling with multiple classes, Chemometr. Intell. Lab. Syst. 217 (2021), 104423, https://doi.org/10.1016/j.chemolab.2021.104423.
[25] S. Ruiz, L.A. Sarabia, M.S. Sánchez, M.C. Ortiz, Handling variables, via inversion of Partial Least Squares Models for Class-Modelling, to bring defective items to non-defective ones, Front. Chem. 9 (2021), 681958, https://doi.org/10.3389/fchem.2021.681958.
[26] P. Oliveri, Class-modelling in food analytical chemistry: Development, sampling, optimisation and validation issues - a tutorial, Anal. Chim. Acta 982 (2017) 9–19.
[27] L. Zhang, P. Li, X. Sun, J. Mao, F. Ma, X. Ding, Q. Zhang, One-class classification-based authentication of peanut oils by fatty acid profiles, RSC Adv. 5 (2015) 85046–85051.
[28] O.Y. Rodionova, K.S. Balyklova, A.V. Titova, A.L. Pomerantsev, Quantitative risk assessment in classification of drugs with identical API content, J. Pharm. Biomed. 98 (2014) 186–192.
[29] A. Biancolillo, S. DeLuca, S. Bassi, L. Roudier, R. Bucci, A.D. Magrì, F. Marini, Authentication of an Italian PDO hazelnut ("Nocciola romana") by NIR spectroscopy, Environ. Sci. Pollut. Res. 25 (2018) 28780–28786.
[30] S. Kittiwachana, D.L.S. Ferreira, G.R. Lloyd, L.A. Fido, D.R. Thompson, R.E.A. Escott, R.G. Brereton, One class classifiers for process monitoring illustrated by the application to online HPLC of a continuous process, J. Chemometr. 24 (2010) 96–110, https://doi.org/10.1002/cem.1281.
[31] L. Xu, S.-M. Yan, C.-B. Cai, X.-P. Yu, One-class partial least squares (OCPLS) classifier, Chemometr. Intell. Lab. Syst. 126 (2013) 1–5.
[32] S. Wold, Pattern recognition by means of disjoint principal components models, Pattern Recogn. 8 (1976) 127–139, https://doi.org/10.1016/0031-3203(76)90014-5.
[33] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
[34] M.P. Derde, D.L. Massart, UNEQ: a disjoint modelling technique for pattern recognition based on normal distribution, Anal. Chim. Acta 184 (1986) 33–51, https://doi.org/10.1016/S0003-2670(00)86468-5.
[35] S.S. Khan, M.G. Madden, One-class classification: taxonomy of study and review of techniques, Knowl. Eng. Rev. 29 (2014) 345–374.
[36] M. Forina, P. Oliveri, S. Lanteri, M. Casale, Class-modelling techniques, classic and new, for old and new problems, Chemometr. Intell. Lab. Syst. 93 (2008) 132–148, https://doi.org/10.1016/j.chemolab.2008.05.003.
[37] R.G. Brereton, Pattern recognition in chemometrics, Chemometr. Intell. Lab. Syst. 149 (2015) 90–96, https://doi.org/10.1016/j.chemolab.2015.06.012.

[38] Z. Małyjurek, B. Walczak, The scope of applicability of the selected class-modelling methods, Chemometr. Intell. Lab. Syst. 218 (2021), 104427.
[39] L. Xu, M. Goodarzi, W. Shi, C.-B. Cai, J.-H. Jiang, A MATLAB toolbox for class modelling using one-class partial least squares (OCPLS) classifiers, Chemometr. Intell. Lab. Syst. 139 (2014) 58–63.
[40] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[41] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (2004) 45–66, https://doi.org/10.1023/B:MACH.0000008084.60811.49.
[42] M. Forina, C. Armanino, R. Leardi, G. Drava, A class-modelling technique based on potential functions, J. Chemom. 5 (1991) 435–453.
[43] M.C. Ortiz, L.A. Sarabia, Caracterización de vinos de Toro mediante técnicas quimiométricas de análisis multivariante, Free download at, in: Anuario 1992', Instituto de Estudios Zamoranos C.S.I.C., 1992, pp. 397–460. Last visit: 14/02/2022, https://iezfloriandeocampo.com/anuarios/1992/.
[44] M.C. Ortiz, J.A. Saez, J. López Palacios, Typification of alcoholic distillates by multivariate techniques using data from chromatographic analyses, Analyst 118 (1993) 801–805.
[45] M.C. Ortiz, L.A. Sarabia, R. García-Rey, M.D. Luque de Castro, Sensitivity and specificity of PLS-class modelling for five sensory characteristics of dry-cured ham using visible and near infrared spectroscopy, Anal. Chim. Acta 558 (2006), 125–13.
[46] M.S. Sánchez, M.C. Ortiz, L.A. Sarabia, V. Busto, Class-modelling techniques that optimize the probabilities of false noncompliance and false compliance, Chemometr. Intell. Lab. Syst. 103 (2010) 25–42.
[47] M.C. Ortiz, L.A. Sarabia, M.S. Sánchez, Tutorial on evaluation of type I and type II errors in chemical analyses: from the analytical detection to authentication of products and process control, Anal. Chim. Acta 674 (2010) 123–142.
[48] T. Huang, X.-Y. Li, R. Jin, J. Ku, S.-M. Xu, M.-L. Xu, Z.-Z. Wu, D.-G. Kong, Multi-target recognition of internal and external defects of potato by semi-transmission hyperspectral imaging and manifold learning algorithm, Spectrosc. Spectr. Anal. 35 (2015) 992–996, https://doi.org/10.3964/j.issn.1000-0593(2022)02-0333-08.
[49] M. Pardo, G. Sberveglieri, A. Taroni, F. Masulli, G. Valentini, Decompositive classification models for electronic noses, Anal. Chim. Acta 446 (2001) 221–230.
[50] Z. Lin, S. Jia, G. Luo, X. Dai, B. Xu, Z. Wu, X. Shi, Y. Qiao, Dealing with heterogeneous classification problem in the framework of multi-instance learning, Talanta 132 (2015) 175–181.
[51] M. Barker, W. Rayens, Partial least squares for discrimination, J. chemometrics 17 (2003) 166–173, https://doi.org/10.1002/cem.785.
[52] T. Windeatt, R. Ghaderi, Coding and decoding strategies for multi-class learning problems, Inf. Fusion 4 (2003) 11–21.
[53] G. Derringer, R. Suich, Simultaneous optimization of several response variables, J. Qual. Technol. 12 (1980) 214–219.
[54] L.A. Sarabia, M.C. Ortiz, M.S. Sánchez, Response surface methodology, in: Comprehensive Chemometrics: Chemical and Biochemical Data Analysis, Elsevier, Amsterdam, Holland, 2020, pp. 287–326.
[55] N. Rodríguez, M.C. Ortiz, L.A. Sarabia, A. Herrero, A multivariate multianalyte screening method for sulfonamides in milk based on front-face fluorescence spectroscopy, Anal. Chim. Acta 657 (2010) 136–146, https://doi.org/10.1016/j.aca.2009.10.048.
[56] B. Álvarez-Sánchez, F. Priego-Capote, J. García-Olmo, M.C. Ortiz-Fernández, L.A. Sarabia-Peinador, M.D. Luque de Castro, Near-infrared spectroscopy and partial least squares-class modeling (PLS-CM) for metabolomics fingerprinting discrimination of intervention breakfasts ingested by obese individuals, J. chemometrics 27 (2013) 221–232, https://doi.org/10.1002/cem.2526.
[57] M. Casale, B. Pasquini, M. Hooshyari, S. Orlandini, E. Mustorgi, C. Malegori, F. Turrini, M.C. Ortiz, L.A. Sarabia, S. Furlanetto, Combining excitation-emission matrix fluorescence spectroscopy, Parallel Factor Analysis, cyclodextrin-modified micellar electrokinetic chromatography and Partial Least Squares Class-Modelling for green tea characterization, J. Pharmaceut. Biomed. Anal. 159 (2018) 311–317.
[58] J.M. Benito, M.C. Ortiz, A. León, L.A. Sarabia, J.M. Ligos, M. Montoya, M. García, E. Ruiz-Mateos, R. Palacios, A. Cabello, C. Restrepo, C. Rodríguez, J. del Romero, M. Leal, M.A. Muñoz-Fernández, J. Alcamí, F. García, M. Górgolas, N. Rallón, Class-modelling analysis reveals T-cell homeostasis disturbances involved in loss of immune control in elite controllers, BMC Med. 16 (2018) 30, https://doi.org/10.1186/s12916-018-1026-6.
[59] S. Ruiz, L.A. Sarabia, M.C. Ortiz, M.S. Sánchez, Residual spaces in latent variables model inversion and their impact in the design space for given quality characteristics, chemometrics Intell. Lab. Syst. 203 (2020), 104040, https://doi.org/10.1016/j.chemolab.2020.104040.
[60] R. Díez, L. Sarabia, M.C. Ortiz, Rapid determination of sulfonamides in milk samples using fluorescence spectroscopy and class modelling with n-way partial least squares, Anal. Chim. Acta 585 (2007) 350–360.
[61] M. P Wand, M.C. Jones, Kernel Smoothing, Monographs on Statistical an Applied Probability, vol. 60, Springer-Science-Business Media, 1995.
[62] M.S. Sánchez, O. Valencia, S. Ruiz, M.C. Ortiz, L.A. Sarabia, DMCEN a MATLAB function to evaluate the entropy improvement provided by a multivariate k-class-model, Last visit: 25/05/2022, https://www.mathworks.com/matlabcentral/fileexchange/112175-dmcen, 2022.
[63] M. Metz, A. Biancolillo, M. Lesnoff, J.M. Roger, A note on spectral data simulation, Chemometr. Intell. Lab. Syst. 200 (2020), 103979.
[64] F.C. Leone, L.S. Nelson, R.B. Nottingham, The folded normal distribution, Technometrics 3 (1961) 5453–5550.
[65] E. Rueda, Bachelor Thesis, Universidad de Burgos, April 1997.
[66] D. Dua, D, C. Graff, C, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2019. Last visit: 11/02/2022, http://archive.ics.uci.edu/ml.
[67] M. Forina, C. Armanino, S. Lanteri, E. Tiscornia, Classification of olive oils from their fatty acid composition, in: H. Martens, H. Russwurm Jr. (Eds.), Food Research and Data Analysis, 1983, pp. 189–214.
[68] R.W. Kennard, L.A. Stone, Computer aided design of experiments, Technometrics 11 (1969) 137–148.
[69] R.D. Snee, Validation of regression models: methods and examples, Technometrics 19 (1977) 415–428.
[70] M. Daszykowski, B. Walczak, D.L. Massart, Representative subset selection, Anal. Chim. Acta 468 (2002) 91–103, https://doi.org/10.1016/S0003-2670(02)00651-7.
[71] Y. Xu, · R. Goodacre, On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning, Journal of Analysis and Testing 2 (2018) 249–262, https://doi.org/10.1007/s41664-018-0068-2.