ELSEVIER

Analytica Chimica Acta 446 (2001) 223–232

ANALYTICA CHIMICA ACTA

www.elsevier.com/locate/aca

Decompositive classification models for electronic noses

M. Pardo [a,*], G. Sberveglieri [a], A. Taroni [b], F. Masulli [c], G. Valentini [c]

[a] *INFM & University of Brescia, Dept. of Chemistry and Physics, Via Valotti 9-25123 Brescia, Italy*
[b] *INFM & University of Brescia, Dept. of Electronics for Automation, Via Branze 38-25123 Brescia, Italy*
[c] *INFM & University of Genova, Dept. of Computer and Information Sciences, Via Dodecaneso 35, 16146 Genova, Italy*

Received 7 November 2000; received in revised form 30 January 2001; accepted 13 February 2001

Abstract

The classification of 242 measurements in 14 classes is attempted using two different classification approaches. Measurements have been performed with a commercial electronic nose (EN) comprising 11 chemical sensors on extra-virgin olive oils of 14 different geographical provenances. As we deal with a relatively small data set and a big number of classes, the classification task is quite demanding. We first tackled the global classification task using a single multilayer perceptron (MLP), which gave a misclassification rate of 25%. In order to improve the performance, we studied two different approaches based on ensembles of learning machines, which decompose the classification in subtasks. In the first approach, a classification tree was constructed using a priori knowledge (geographical origin) for the formation of sensible superclasses (union of single classes). At each classification node we both used MLPs and SIMCA (soft independent modeling of class analogy). The second approach applies a learning machine called parallel nonlinear dichotomizers (PND) that is based on the decomposition of a K-class classification problem in a set of two-class tasks. A binary codeword is assigned to each class and each bit is learned by a dichotomizer (implemented by a dedicated MLP). In the reconstruction stage, a pattern is assigned to the class whose codeword is most similar (e.g. in L_1 norm) to the output of the set of dichotomizers. We achieved the best results (misclassification error rate of about 10%) using a decomposition based on error correcting output codes (ECOC).
© 2001 Elsevier Science B.V. All rights reserved.

Keywords: Pattern recognition; Ensemble methods; Classification tree; Output coding; Multilayer perceptron; Electronic nose

1. Introduction

In the last decade several methods for constructing ensembles (committees) of learning machines have been developed, which improved the recognition accuracy with respect to single classifiers [1,2]. For example, neural networks show different results with different initializations due to the randomness inherent in the training procedure. Instead of selecting the best network and discarding the others, one can combine the networks. Ensemble methods encompass a wide range of techniques such as ensemble averaging [3,4], where the outputs of different predictors are linearly combined to produce an overall output; boosting [5–7] and bagging [8], where the same learning algorithms are used with subsets of the training examples; mixture of (local) experts [9,10] where the outputs of the different predictors are nonlinearly combined through a gating network; ensemble

* Corresponding author.
E-mail address: pardo@tflab.ing.unibs.it (M. Pardo).

constructed by subsets of input features [11], where each predictor selects a group of the input features.

Different taxonomies have been proposed to help understanding the rationale of these methods. For example, Jain et al. [2] group methods into three main categories according to their architecture: parallel, cascading and hierarchical. In the parallel architecture — which is the most common one — all individual classifiers are invoked independently, and their results combined. In the cascading architecture, individual classifiers are invoked in a linear sequence. In the hierarchical architecture individual classifiers are combined into a structure, which is similar to that of a decision tree. In [1], Dietterich further distinguishes between parallel ensembles according to the way the component classifiers are built. Ensembles can be formed by training the component classifiers on different training subsets, on different subsets of features or different output codings.

The two methods presented in this paper share the property of decomposing the classification in subtasks assigned to different learning machines. In engineering this is also known as divide-and-conquer strategy [12]. *The first method is a tree classifier*, i.e. it establishes a hierarchy of classifiers. In this way the space of input data is repeatedly partitioned by a sequence of splits. The second method — *classification by output coding (OC) decomposition* — uses a parallel architecture. It splits a multiclass problem (*polychotomy*) in a set of independent two-class subproblems (*dichotomies*) and then recomposes the original classification problem using the outputs of the *dichotomizers* trained on each different dichotomy. This is a proper machine learning approach [13–16]. In both cases we will use multilayer perceptrons (MLP) as component classifiers.

The procedure for growing a decision tree has been studied (and automated) in machine learning by Breiman et al. [17] and Quinlan [18] giving rise, respectively, to the CART (classification and regression trees) and to the C4.5 algorithms. Here we will design the tree using a more empirical procedure: the various splits have been designed by hand using a priori information and graphical aid. The definite advantages of trees lie in their interpretability and in the possibility of incorporating a priori knowledge.

The (binary) output coding strategy is a form of voting among multiple hypotheses. The effectiveness of OC decomposition depends mainly on the correlation among output errors and on the learning machine implementing the method [19]. This permits to reduce both the bias and the variance with respect to a single classifier [20].

We will apply these two methods to a dataset produced with an electronic nose (EN). ENs, in the broadest meaning, are instruments that analyze gaseous mixtures for discriminating between different (but similar) mixtures or, in the case of simple mixtures, quantifying the concentration of the constituents. ENs consist of an array of sensors (based on diverse functioning mechanisms, e.g. thin or thick film semiconductor sensors which change their conductance when exposed to gases), electronic circuitry, a sampling system and data analysis software [21]. While the analytical methods developed in the last 25 years usually first perform a separation of the mixture which constitutes the aroma in its components and then identify the components by comparing them with a standard, ENs recognize a fingerprint, that is a global information, of the samples to be classified. EN are still in a developmental phase, though several ENs are already on the market.

ENs have already been successfully applied to food evaluation [22]. In particular, applications to the quality control of edible oil (degradation and rancidity control) are reported in [23–25]. In the present contribution we classify 14 different types of virgin olive oil which differ in their geographic provenance, some of them originating from the same region. The resolving power of the nose is therefore thoroughly tested.

In presentations dealing with ENs quite often PCA score plots with nicely clustered data are shown in order to demonstrate the classification ability of the nose. Since PCA, in itself, is an unsupervised technique [26, p. 10 and 411], which in this case just serves to visualize the data, it is assumed that in these easy cases any classification algorithm, such as nearest neighbors clustering or linear discriminant analysis (even without preprocessing) could do the real classification job. In the last decade various kinds of neural networks (NN) have been used for classifying chemical sensor array data, the most common being MLP [27,28] and self organizing maps (SOM) followed by a supervised labeling stage [29,30]. A detailed account about the use of neural networks in chemical practice is given by [31].

2. Decomposition methods

2.1. Hierarchical classification

A hierarchical structure is easily represented as a tree, where by convention the first root node is displayed on top, connected by successive directional links or branches to the other nodes. These are similar connected until we reach the terminal or leaf nodes, each of which is attributed to a different class. As an example, the tree used for the classification of the 14 oils is shown in Fig. 1. Each rectangle represents a node with its associated question while the ellipses are the leaves with the associated class labels. Near to every link between nodes we see the corresponding superclass.

In the basic implementations of classification trees binary splits are considered, every split is executed considering the value of just one feature (therefore decision boundaries are hyperplanes parallel to the feature axes) and the split is performed which minimizes a cost function, usually the entropy impurity [32].

Here we followed a more experimental path to the tree construction. We allowed multiway splits. At each node *we first decided the split, i.e. the classification subtask*. Then, starting from all the features, we performed feature reduction with PCA and then trained an MLP. Regarding the choice of the split, we propose two types of criteria. The first one makes use of a priori information about the classes and possibly expresses some ranking in the classification interest. In our example one could decide to split the classes according to a geographical criterion: e.g. we decided to discriminate Italian oils from the foreign ones in the first step. This criterion will in general not be optimal in respect to the complete classification. A second type of criterion lets the data tell what the best groupings are. For example one draws the PCA score and looks at the tendencies in the groupings. A simple example of this is presented in the next section. This does not rely on a priori knowledge, that means that it is less straightforward and will require some trial and error, in the spirit of the supervised cost minimization technique which is normally used.

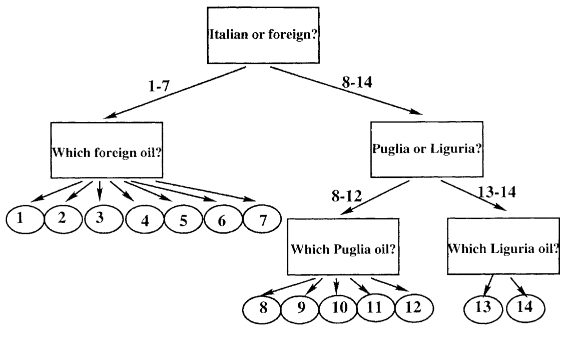

Fig. 1. The three level tree used for the oil classification.

2.2. Classification by output coding decomposition

Let there be K classes polychotomy (or *K-polychotomy*) $\mathcal{P} : \boldsymbol{X} \to \{C_1, \dots , C_k\}$, where $\boldsymbol{X}$ is the multidimensional space of the features, and $C_1, \dots , C_K$ are the labels of the classes. The output coding methods for classification consist of two parts.

First, the *decomposition* of the K-polychotomy generates a set of say L dichotomies $f_1, \dots , f_L$. Each dichotomy f_i subdivides the input patterns in two complementary superclasses $\mathcal{C}_i^+$ and $\mathcal{C}_i^-$ (i.e. $\mathcal{C}_i^+ \bigcap \mathcal{C}_i^- = \emptyset, \forall i$), each of them aggregating one or more classes of the K-polychotomy. By this approach the K classes to be discriminated are univocally labeled through K binary strings called *codewords*.

In Table 1 we show as example the coding scheme for a 4-polychotomy which makes use of the *minimal decomposition* that generates $I = \lceil \log_2 K \rceil$ dichotomies.

A *decomposition matrix* $D = [d_{ik}]$, of dimension $L \times K$, represents the decomposition, and connects classes $C_1, \dots , C_k$ to the superclasses $\mathcal{C}_i^+$ and $\mathcal{C}_i^-$ identified by each dichotomy. An element of D is defined as

$$d_{ik} = \begin{cases} +1 & \text{if } C_k \subseteq \mathcal{C}_i^+ \\ -1 & \text{if } C_k \subseteq \mathcal{C}_i^- \end{cases}$$

Table 1
Minimal decomposition scheme for a 4-class classification problem

Dichotomies	Class codewords			
	C_1	C_2	C_3	C_4
f_1	+1	+1	−1	−1
f_2	+1	−1	+1	−1

In a decomposition matrix, rows correspond to dichotomizers tasks and columns to classes. In this way, each class is univocally determined by its specific codeword. The decomposition matrix associated to the minimal decomposition scheme for a 4-polychotomy shown in Table 1 is

$$\begin{pmatrix} +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \end{pmatrix} \tag{1}$$

The second part of the method is the *decision or recomposition* that assigns the pattern to the class whose codeword is most similar to the output of the set of dichotomizers

$$\text{class} = \arg\max_k(\text{sim}(\hat{\boldsymbol{f}}(x), c_k)), \forall k = 1, \dots, K \tag{2}$$

where c_k is the codeword of class C_k and sim is a similarity measure than can be based on the Hamming distance for dichotomizers with discrete valued outputs, and on inner product, L_1, or L_2 norms for dichotomizers with continuous valued outputs.

For simplicity, let us think at learning machines with discrete outputs ± 1. In the case of minimal decomposition, then, the pattern is trivially assigned to the class whose codeword is *exactly equal* to the output of the dichotomizers. The drawback of this coding scheme is that as soon as a dichotomizer gives a wrong output the class is wrongly predicted. To reduce this possibility longer (redundant) encodings were introduced. In fact, the *error recovering ability* of a decomposition scheme is linked to the maximum column (codewords) separation. The maximal number of errors (due to wrong dichotomies) that can be corrected is, in the worst case (all errors on differing bits of the nearest codewords)

$$\max NE = \left\lfloor \frac{\Delta_D - 1}{2} \right\rfloor \tag{3}$$

where Δ_D is the minimal Hamming distance between pair of columns in the decomposition matrix D. But one cannot simply add bits to the output codes: apart from computational limitations which arise for too long codes, it is not difficult to see that for K classes there can be at most be $2^{K-1} - 1$ different dichotomies [14].

There are many possibilities for decomposing a polychotomy into dichotomies [14,15]. The more popular is the one-per-class (OPC) decomposition scheme [33]: each dichotomizer f_i has to separate a single class from all the others. As a consequence, for K classes we will use K dichotomizers. The Hamming distance between any two codewords is two, so the one-per-class encoding of any number of classes cannot correct any errors. In passing, note that this coding scheme is usually implemented with MLPs. Apart from the a priori bad error recovering capability of the code, in the case of MLP the dichotomizers are not independent (parallel). Therefore one does not have the main benefit of ensembles of learning machines.

In the experiments presented in this paper, we compared two efficient decomposition schemes, namely the *correcting classifiers* (CC) that is a variant of the *pairwise coupling* (PWC) decomposition scheme [34] and the (ECOC) decomposition scheme [14]. The *correcting classifiers* (CC) scheme is defined by the decomposition matrix

$$d_{ik} = \begin{cases} +1 & \text{if } C_k \subset \mathcal{C}_i^+ \\ -1 & \text{if } C_k \subset \mathcal{C}_i^- \end{cases}$$

where the superclasses $\mathcal{C}_i^+$ are composed by pairs of classes and $\mathcal{C}_i^-$ are their complementary superclasses, i.e. $\mathcal{C}_i^- = \mathcal{C} - \mathcal{C}_i^+$, where $\mathcal{C}$ is the set of all classes. Hence we have $(K(K-1))/2$ different dichotomies.

The ECOC decomposition scheme is based on the application of error correcting output codes in the generation of classes codewords. ECOC decomposition tries to maximize error recovering capabilities through the maximization of the minimum distance between each couple of codewords. Dietterich and Bakiri [14] employed different methods to construct error correcting output codes depending on the number of classes k.

3. Experimental

3.1. The electronic nose measurements

In this work a Fox3000 (Alpha M.O.S., Toulouse, France) has been employed. This is a bench-top instrument comprising arrays of 12 MOS sensors placed in two chambers which are connected in series. For this application seven sensors from Alpha M.O.S. and five from Figaro have been selected. The arrays are modular plug-in devices and can be exchanged depending on the type of vapor to be sensed.

Samples of virgin olive oil were prepared by adsorbing 100 ml of each olive oil on filter paper, then sealed into vials for 8 h before headspace analysis. To increase the volatility of the aromatic compound, samples were thermostated at 37°C before the measurement.

The headspace of the olive oil to be analysed is stripped from the sample vial by the carrier gas and drawn over the sensors at a controlled rate. A flow of 300 ml/min of saturated humidified (17 ± 1 g/m^3) chromatographic air at 25°C was employed. The response to a step of headspace is recorded. The injection time was set to be as short as 5 s, in order to contaminate the sensor the least possible, and acquisition time to 120 s, followed by a 400 s equilibration period before the following sample.

Virgin olive oil samples of 14 different cultivars from different geographic zone were analysed: seven Italian (between them five from Puglia and two from Liguria) and seven foreign.[1] A total of 242 measurements were performed and calibrated against a standard to counteract drift.

3.2. Implementation of the decompositive methods

First we selected the features from the sensors' temporal response. We restricted ourselves to two choices: the relative change in resistance (difference between maximum response and baseline value over baseline value) and a set of five parameters characterizing the dynamic response of the sensor (such as the maximum derivative obtained after smoothing the derivative). These two choices resulted in comparable performances and therefore we used the relative change feature. Further we did not notice any significant difference between scaling to unit variance, to the interval $[-1, 1]$ or not scaling at all.

To reduce the noise and the collinearity present in 12 features we made use of PCA [35,36], i.e. the new variables which serve as input to the ANN are the projections of the original data on the first PCs. The data were divided in training and validation set prior to the preprocessing — consisting of normalization and PCA — since the preprocessing itself is part of the training procedure.

[1] The samples were provided by "ISMAR Chimica Genova".

Circa 1/5 of patterns have been used for testing the classifiers. The same division in training and test data has been used for the hierarchical and for the output coding approach to permit the comparison of the results.

For the classification tree approach a Matlab toolbox has been developed which permits to test the dependence of the error on the following points (and consequently to minimize the error).

1. The number of inputs, i.e. the number of principal components on which to project the data.
2. The learning procedure. To eliminate overfitting we tried both early stopping — dividing the data set in training, validation and test — and weight decay regularization [37]. In both cases the Levenberg–Marquardt algorithm was used to minimize the error function. Results were comparable, the early stopping training being faster. In early stopping there was no significant difference between the validation error and the test error. This is not obvious because the validation set is used to stop the training and therefore the validation error could underestimate the generalization error as calculated on the test set. In this case therefore the partitioning of the data set in three sets was too conservative.
3. The classification criterion of the network outputs. The "winner takes all" criterion was adopted because there had not been any advantage in defining a doubt class.

Ten initializations are used for every topology and the best net is held.

It can be worth noting that there does not seem to be a need to constrain the number of weights if one is only interested in having a good misclassification error on the test set. This despite the frequent use of heuristics constraining the ratio between network weights and available training data. The learning procedures proposed in point 2 already takes care of avoiding overfitting. After an optimal minimal network complexity has been reached, for both learning methods the test error does not change by increasing the network size [38,39]. Sarle [38] claims that the best way to use early stopping — at least as far as overfitting is concerned (training times will grow) — is to choose a number of weights much bigger than the number of patterns; the early stopping procedure prevents the overfitting of the

Table 2
Results of the hierarchical procedure

Subtasks (total number of patterns)	Best net	Correctly vs. misclassified patterns (NN)	Correctly vs. misclassified patterns (SIMCA)
Italian–foreign (242)	7–8–2	47:2	42:7
Intra foreign (131)	7–10–7	26:3	24:5
Liguria–Puglia (111)	5–8–2	23:0	20:3
Lig1–Lig2 (32)	3–4–2	7:0	4:3
Intra Puglia (79)	5–4–5	15:3	13:5

training data. In fact, in the application proposed in the next chapter it happens that the smallest misclassification is achieved (on the test set) when the number of weights is bigger (but not much bigger) than the number of measurements, see e.g. second line in Table 2.

As for the output coding method, we have tackled the overall classification problem using an ensemble of multiple input single output (MISO) MLPs that we call *parallel nonlinear dichotomizers* (PND) [19,16]. The decomposition unit of the PND is implemented by a set of one hidden layer MLP, considering two decomposition schemes, *correcting classifiers* (CC) [34], and *error correcting output coding* (ECOC) [14], and the decision unit uses an L_1 norm distance for selecting the class. All the experimentation on the overall data set has been performed using *NEURObjects* [40], a set of library classes for neural networks development. The generation of ECOC codes was performed using a modification of the BCH algorithm [41], in order to adapt it to classification tasks. For this experiment, model selection has been performed varying the number of hidden units and the learning rate of the backpropagation algorithm. As stopping criteria of the learning algorithm we have used different levels of the normalized RMS error approximating the misclassification error and the number of iterations. For each learning machine the training and the test stage has been repeated five times with different pseudo-random initialization of the weights.

4. Results and discussion

In order to get a preliminary idea of the distribution of the data, a PCA score plot of all 14 oils was drawn. As can be seen in Fig. 2 the groups are confused; no nice spontaneous clustering is there to help in the classification.

First a “brute force” classification method was tried: all the 14 oils were considered as distinct outputs of the MLP. That is the oils were directly partitioned in 14 classes. This causes the network to have a big number of weights and consequently long training times, more than 1 h on a Pentium II. The classification procedure still consists in PCA + MLP. We found out the best net incorrectly classified 12 out 49 test patterns (75.5% classification rate). With SIMCA this worsened to 18 out of 49.

Since these results were not satisfying, the second approach was building a classification tree leading from coarse to fine resolution as described in Fig 1. The results are summarized in Table 2. The tree has not been optimized to be the best possible tree (i.e. the one giving smallest misclassification error); in fact looking at the scores distribution in Fig. 2 a different first level division would seem more effective. We also tried to divide all the Italian oils at a time (as for the foreign ones) but the results were indeed not good. The tree testing does not give the overall classification rate but just the rate for the subtasks. A (pessimistic) estimate of the classification rate for every class is simply derived by multiplying the classification rates as the pattern descends the tree. For example, for an oil from Liguria we have 47/49 (Italian–foreign step) $\times$ 23/23 (Liguria–Puglia step) $\times$ 32/32 (Lig1–Lig2 step) = 96%.

Some comments to the results of Table 2 may be appropriate.

1. PCA + ANN outperform the more simple SIMCA. There is no surprise since the first method is more flexible. One should decide if simplicity of interpretation is more important than a certain error

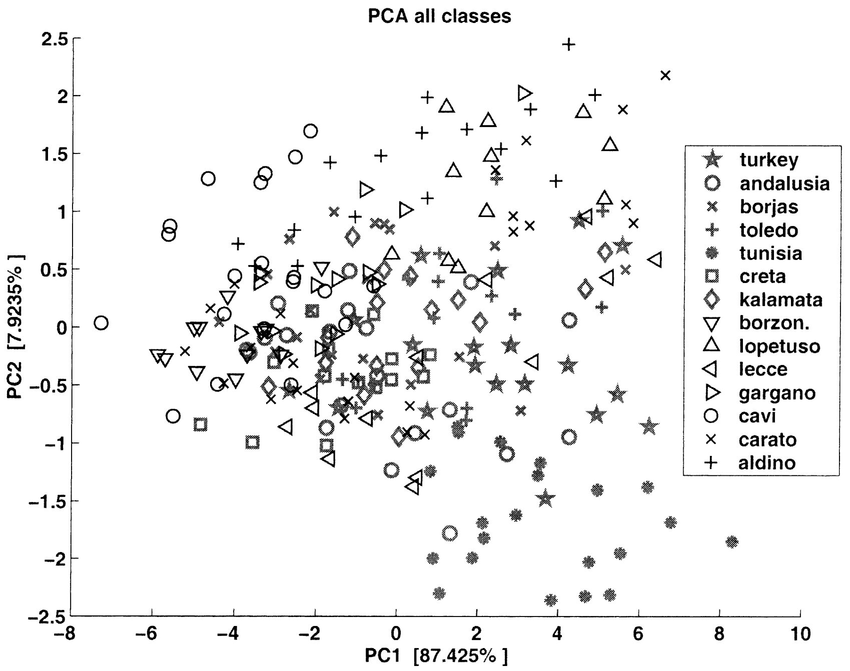

Fig. 2. Score plot considering all 14 oils at a time. The thickness of the marks divides Italian from foreign oils.

reduction. Nothing hinders using both methods and establishing a criterion to solve conflicts.

2. The classification performance for Puglia oils is not as good as the others. In cases like this, one can decide whether to accept the error or further subdivide the problem. In Fig. 3 we see that there is a certain distinction between oils 1,3,5 on the upper side and 2,4 on the lower. With these two superclasses as output of the MLP we had a correct classification of 17 test data out of 17 with a 3–7–2 network, both with early stopping and a division in three sets and with regularization.
3. Although three PCs usually contain more then 95% of the variance, we see that in some cases it is better to hold more components. One possible reason is that the first PC, in spite of accounting for a great part of the variability in the data set is not important for classification. In fact in Fig. 3 we see that it is the second component which distinguishes between the two superclasses; the variance in the first PC could be due to drift with the same effect on all the classes. In this cases the first PC could be left out from the MLP inputs. This point shows that in complex problems PCA, being linear and not supervised, should be used only to remove noise and redundancy (correlation).

Fig. 4 shows the performances of PND CC and PND ECOC on the considered data set. The error bars (standard deviation) and the average refer to the distribution over the weight initializations. PND CC achieves a predicted average error rate of about 20%, with a lower variance and with a reduced number of hidden units (for every PND component classifier) with respect to MLP. PND ECOC shows the best results with only four hidden units with an average misclassification error of about 12%.

We can observe that PND ECOC largely outperforms MLP and PND CC. The error on the best classifier drops from 25% of a single MLP to 10%. The confusion matrix obtained for the best ECOC classifier (15 bit code) is displayed in Table 3. The error recovering capabilities of ECOC codes [14,19] and the independence among the dichotomizers [16] explain the good performances of PND ECOC ensembles of learning machines.

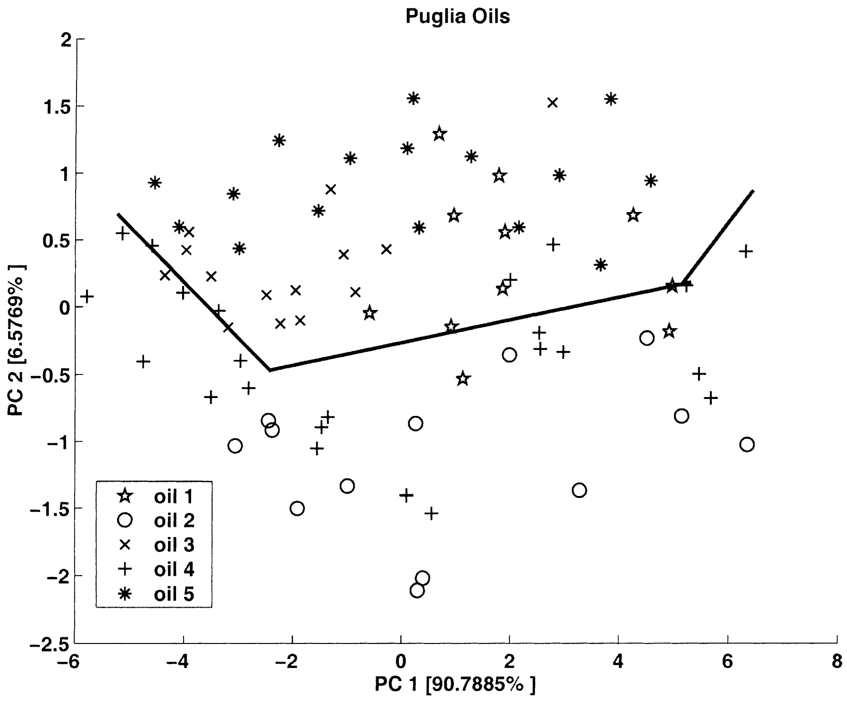

Fig. 3. Score plot of the Puglia oils: the second PC distinguishes between a class formed by two oils and one formed by three. A hand drawn boundary is shown.

From Table 3 the results on specific subtasks can be extracted by adding the number of misclassified patterns for every class contained in a certain superclass. In Table 4 we present the results obtained for the same subtasks performed by the classification tree, see Table 2 for comparison. On these subtasks the two decomposition methods give similar results. Note anyway that the PND ECOC has been trained over the complete classification task and would have reasonably obtained even better results on the subtasks if it were directly trained on them.

Table 3
Confusion matrix for the best ECOC PND classifier

3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0	1	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0	0	0	0
1	0	1	3	0	0	0	0	0	0	1	0	0	0
0	0	0	0	4	0	0	0	0	0	0	0	0	0
0	0	1	0	0	5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3

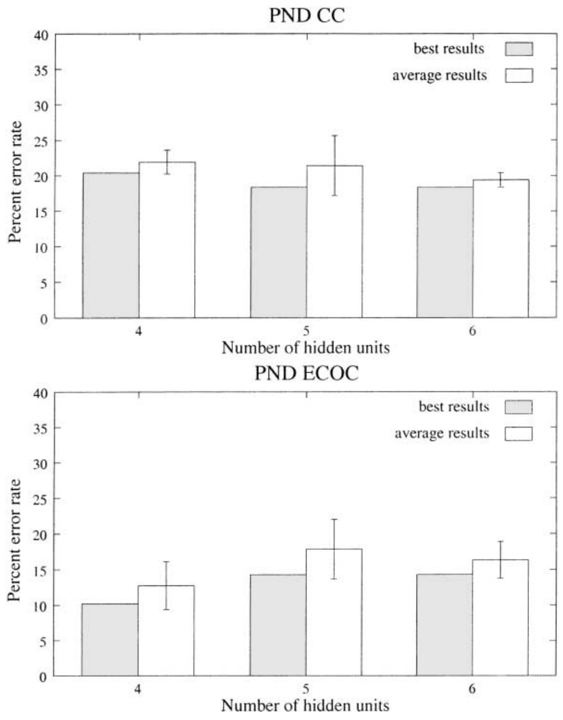

Fig. 4. Compared error rates among PND CC (top) and PND ECOC (bottom) learning machines on the overall data set.

Table 4
Results on specific subtasks obtained with the best ECOC PND classifier (for comparison with Table 2)

Subtasks	Correctly vs. mis-classified patterns	Percentage error
Italian–foreign	48:1	2.0
Intra foreign	26:3	11.1
Liguria–Puglia	21:2	9.2
Intra Liguria	7:0	0.0
Intra Puglia	16:2	12.5

5. Conclusions

A commercially relevant problem has been tackled with an EN. This represented a hard classification problem given the big number of classes and the relatively small number of examples. PCA and MLP are not effective in this case. The classification rate has been enhanced using two ensembles of classifiers, both based on the decomposition of the classification in subtasks. The tree, being interpretable and having the possibility of incorporating a priori knowledge, is more profitable in experimental fields. The recent machine learning literature studies ensembles of parallel classifiers which have been shown to improve the recognition accuracy, as it was the case in this paper. The error on the best classifier drops from 25% of a single MLP to circa 10% with these methods.

More datasets are indeed needed to validate this classification methods. Still other ensembles methods (e.g. boosting) can be explored to enhance the accuracy of chemical data classification.

Acknowledgements

Dr. E. Dalcanale is gratefully acknowledged for providing the data. This work has been partially supported by the Italian National Research Council (CNR) in the frame of the project MADESS II.

References

[1] T.G. Dietterich, Ensemble methods in machine learning, Multiple Classifier Systems, in: First International Workshop, MCS2000, Cagliari, Italy, Springer, 2000, pp. 1–15.
[2] A. Jain, R. Duin, J. Mao, IEEE Trans. Pattern Anal. Mach. Int. 22 (2000) 4–37.
[3] M.P. Perrone, L.N. Cooper, When networks disagree: ensemble methods for hybrid neural networks, in: R.J. Mammone (Ed.), Artificial Neural Networks for Speech and Vision, Chapman & Hall, London, 1993, pp. 126–142.
[4] S. Hashem, Neural Comput. 10 (1997) 599–614.
[5] R. Schapire, A brief introduction to boosting, in: T. Dean (Ed.), 16th International Joint Conference on Artificial Intelligence, 1999, pp. 1401–1406.
[6] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the 13th International Conference on Machine Learning, Morgan Kauffman, 1996, pp. 148–156.
[7] R.E. Schapire, Y. Freund, P. Bartlett, W. Lee, Ann. Statist. 26 (3) (1998) 1651–1686.
[8] L. Breiman, Machine Learning 24 (2) (1996) 123–140.
[9] M.I. Jordan, R.A. Jacobs, Neural Comput. 6 (1994) 181–214.
[10] R.A. Jacobs, Neural Comput. 7 (1995) 867–888.
[11] C.K.J., Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks, in: C.P. (Ed.), Working notes of the AAAI Workshop on Integrating Multiple Learned Models, 1996, pp. 15–21.
[12] S. Haykin, Neural Networks, Prentice Hall, 1999.
[13] E. Mayoraz, M. Moreira, On the decomposition of polychotomies into dichotomies, in: The XIV International

Conference on Machine Learning, Nashville, TN, 1997, pp. 219–226.
[14] T. Dietterich, G. Bakiri, J. Artif. Intell. Res. (2) (1995) 263–286.
[15] F. Masulli, G. Valentini, Comparing decomposition methods for classification, in: KES'2000, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, Brighton, England, in press.
[16] F. Masulli, G. Valentini, Parallel Nonlinear Dichotomizers, in: IJCNN2000, The IEEE-INNS-ENNS International Joint Conference on Neural Networks, Como, Italy, 2000.
[17] L. Breiman, G.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.
[18] Q.J.R., C4.5 Programs for Machine Learning, Morgan Kauffman, 1993.
[19] F. Masulli, G. Valentini, Effectiveness of error correcting output codes in multiclass learning problems, Multiple Classifier Systems, in: First International Workshop, MCS 2000, Cagliari, Italy, Springer, 2000, pp. 107–116.
[20] E. Kong, T. Dietterich, Error-correcting output coding correct bias and variance, in: The XII International Conference on Machine Learning, Morgan Kauffman, San Francisco, CA, 1995, pp. 313–321.
[21] J. Gardner, P. Bartlett, Electronic Noses, Oxford University Press, 1999.
[22] P.N. Bartlett, T. Elliot, J. Barcher, Food Technol. 51 (1997) 44–48.
[23] S. Bazzo, F. Loubet, T. Tan, J. Hewitt-Jones, C. Engelen-Cornax, J. Quadt, Semin. Food Anal. 3 (1998) 15–26.
[24] G. Shiers, M. Adechy, Semin. Food Anal. 3 (1998) 43–52.
[25] M. Schweizer-Berberich, M. Franck, H. Ulmer, J. Mitrovics, U. Weimar, W. Gopel, Quality tests of electronic noses: the influence of sample dilution and sensor drifts on the pattern recognition for selected case studies, in: Proceedings of the 7th International Meeting on Chemical Sensors, Beijing, China, 1998, pp. 139–141.
[26] D. Massart, B. Vandeginste, S. Deming, Y. Michotte, L. Kaufmann, Chemometrics: A Textbook, Elsevier, 1988.
[27] M. Pardo, G. Faglia, G. Sberveglieri, M. Corte, F. Masulli, M. Riani, Sens. Actuators B 65 (2000) 267–269.
[28] M. Pardo, G. Faglia, G. Sberveglieri, M. Corte, F. Masulli, M. Riani, Sens. Actuators B 67 (2000) 128–133.
[29] S. Marco, A. Ortega, A. Pardo, J. Samitier, IEEE Trans. Instrum. Meas. 47 (1998) 316–321.
[30] C. Di Natale, A. D'Amico, Modelling and data analysis of multisensor systems with the self organizing map: application to the electronic nose, in: Proceedings of the First International Workshop on Self Organizing Maps, Espoo, Finland, 1997.
[31] F. Despagne, L. Massart, The Analyst 123 (1998) 157–178.
[32] R. Duda, P. Hart, D. Stork, Pattern classification and scene analysis, 2nd Edition, Wiley, New York.
[33] R. Anand, G. Mehrotra, C. Mohan, S. Ranka, IEEE Trans. Neural Networks 6 (1995) 117–124.
[34] M. Moreira, E. Mayoraz, Improved pairwise coupling classifiers with correcting classifiers, in: Tenth European Conference on Machine Learning, Chemnitz, Germany, 1998.
[35] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[36] R. Poppi, D. Massart, Anal. Chim. Acta 375 (1998) 187–195.
[37] D.J.C. MacKay, Neural Comput. 4 (1992) 415–447.
[38] W. Sarle, Stopped training and other remedies for overfitting, in: Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, ftp://ftp.sas.com/pub/neural/inter95.ps.Z, 1995, pp. 352–360.
[39] H. Demuth, M. Beale, Manual of the Neural Network Toolbox, Version 3, The MathWorks, Inc., 1998.
[40] G. Valentini, F. Masulli, NEURObjects, a set of library classes for neural networks development, in: Proceedings of the third International ICSC Symposia on Intelligent Industrial Automation (IIA'99) and Soft Computing (SOCO'99), ICSC Academic Press, Millet, Canada, 1999, pp. 184–190.
[41] R. Bose, D. Ray-Chauduri, Information Control (3) (1960) 68–79.