Pattern Recognition Letters 32 (2011) 458–467

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Online error correcting output codes

Sergio Escalera [a,c,*], David Masip [b,c], Eloi Puertas [a], Petia Radeva [a,c], Oriol Pujol [a,c]

[a] Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona 585, Edifici Històric, 08007 Barcelona, Spain
[b] Universitat Oberta de Catalunya, Rambla del Poblenou 156, 08018 Barcelona, Spain
[c] Computer Vision Center (CVC), Edifici O, Campus Universitat Autònoma de Barcelona, 08193 Barcelona, Spain

ARTICLE INFO

Article history:
Received 24 December 2009
Available online 11 November 2010
Communicated by F. Roli

Keywords:
Online learning
Error correcting output codes
Machine vision

ABSTRACT

This article proposes a general extension of the error correcting output codes framework to the online learning scenario. As a result, the final classifier handles the addition of new classes independently of the base classifier used. In particular, this extension supports the use of both online example incremental and batch classifiers as base learners. The extension of the traditional problem independent codings one-versus-all and one-versus-one is introduced. Furthermore, two new codings are proposed, unbalanced online ECOC and a problem dependent online ECOC. This last online coding technique takes advantage of the problem data for minimizing the number of dichotomizers used in the ECOC framework while preserving a high accuracy. These techniques are validated on an online setting of 11 data sets from UCI database and applied to two real machine vision applications: traffic sign recognition and face recognition. As a result, the online ECOC techniques proposed provide a feasible and robust way for handling new classes using any base classifier.

1. Introduction

Traditional visual machine learning problems are solved training robust statistical classifiers (Vapnik, 1997), using as input a predefined set of training examples. This learning process is usually computationally demanding, and it needs the availability of the whole training set. However, machine vision applications are constantly evolving. The entities from which training samples are extracted change their appearance, or even the structure of the classification problem must change due to the inclusion of new classes. This fact required the development of strong online classifiers that can deal with the variability of the data. Given a classification task, the goal of online learning is to model the classifiers parameters using an initial training set, being able to incrementally evolve this model parameters as new data or classes become available.

The online learning objectives are clearly differentiable in comparison to classic batch learning. However, the "online" term involves different levels of behavior associated to the classifier. One can clearly distinguish two behaviors at different level of abstraction: the first one, from the point of view of data examples – one expects the classifier to adapt to new data from previously seen classes. The second abstraction layer, from the point of view of classes – one expects the classifier to adapt to new classes without retraining the complete classifier. Fig. 1 shows the relationship between these two layers. The bottom layer deals only with examples; in this layer, we can distinguish between data incremental/decremental classifiers and batch classifiers, depending on the un/capability of the method to adapt to new data examples. On a higher level, we find the class incremental/decremental behavior. This layer deals with the addition/removal of classes. Observe that this level is independent of the previous one. Consider the following examples: in our society, it is usual to change employees privileges to resources or access to the company building or to certain restricted areas; consider a biometric verification application in which at a certain point we want to grant access to those resources to a new subject. Or, an object recognition problem where some object categories were not foreseen and want to be added to the full system. In those applications, we barely desire to retrain the complete problem due to scalability or complexity issues. However, an adjustment must be made to accommodate this new data. In these examples, layer 2 behavior is only needed, disregarding on which the actual base learning strategy is.

Up to now, literature has considered the first and second layers as one. However, few are the methods that allow both behaviors at the same time, and most of literature is focussed on the first layer. This second level of abstraction is where our proposal, online ECOC, is defined. As a meta-learning strategy it can accommodate either example incremental online classifiers of the first layer or batch classifiers.

* Corresponding author at: Computer Vision Center (CVC), Edifici O, Campus Universitat Autònoma de Barcelona, 08193 Barcelona, Spain. Tel.: +34 687291957; fax: +34 93 581 16 70.

E-mail addresses: sergio@maia.ub.es, sescalera@cvc.uab.es (S. Escalera), dmasipr@uoc.edu (D. Masip), eloi@maia.ub.es (E. Puertas), petia@maia.ub.es (P. Radeva), oriol@maia.ub.es (O. Pujol).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.11.005

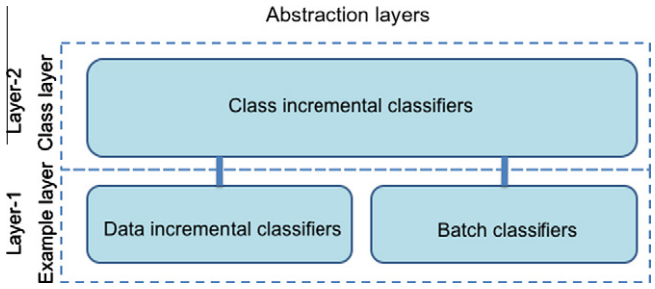

Fig. 1. Abstraction layers of the online behavior.

In this paper, we study the suitability of the error correcting output codes (ECOC) (Kong and Dietterich, 1995) framework to adapt to the online learning scenario. In particular, we focus on layer-2 using ECOC coding schemes, which incrementally allow new classes to be added to the original problem independently of the base classifier used. The addition of new unseen samples in the online ECOC algorithm becomes straightforward using an online (layer-1) base binary classification algorithm. However, the addition of new classes involves the addition of new codewords. Moreover, new dichotomizers (columns in the encoding matrix) could be needed to maintain the classification performances and isolate the new classes. Depending on how these new code words and dichotomizers are generated, different online ECOC strategies are defined. We study and compare four encoding matrix generation algorithms: (i) the online extension of the classic one-versus-all strategy, (ii) the inclusion of ternary encoding to improve the performance obtaining unbalanced online ECOCs, (iii) the online extension of the one-versus-one technique, and (iv) finally a more efficient problem dependent online ECOC encoding. This last algorithm solves class incremental problems minimizing the length of the encoding table (i.e. the computational needs) while maintaining significant classification rates. Notice that the online ECOC scheme does not necessarily involve the use of online layer-1 base classifiers. Some applications could require the use of specific binary base classifiers that can not be extended to the online case. In this sense, this article also provides a solution to this problem.

The paper is organized as follows: in the next section an overview of the state-of-the-art online learning is shown. Section 3 describes the basic ECOC technique and, in particular, it explains the weighing decoding step. Section 4 proposes the extension of the ECOC framework to the layer-2 online case. Section 5 shows the validation performed on the UCI database and the experiments in two independent machine vision problems: a traffic sign recognition and face recognition. Finally, Section 6 concludes this work.

2. Online learning

Online learning is the area of machine learning concerned with learning from examples on the fly. In this framework, each example is trained once and it is never examined again. Online learning is necessary when data continuously arrives and it is not possible to perform an efficient storing for batch learning, or when retraining the whole framework becomes computationally unfeasible. This causes that the training phase can be done in different episodes or rounds. Hence, the main flow of an online algorithm uses to following scheme:

(a) Train the model using an initial data set.
(b) Add the additional data set and adjust the model parameters.
(c) Repeat (b) while new data arrives.

Recent online learning approaches are based on extending classical classifiers to the online case, typical examples are decision trees (Utgoff et al., 1997), online Support Vector Machines (SVM) (Katagiri and Abe, 2006, 2000,,) or online ensemble of classifiers (Oza, 2000). Nevertheless, all these previous works have focused on two class problems. The general approach for dealing with the multiclass case is based on reducing a multiclass problem to multiple binary problems. Katagiri and Abe (2006) in their own online SVM implementation use the one-versus-all scheme when using it in multiclass data. An n-class problem is converted into n two-class problems and for the ith two-class problem, class i is separated from the remaining classes. Another strategy for online multiclass classification is the one proposed by Crammer and Singer (2001) in their called "ultraconservative" algorithms. In their approach, one prototype vector is maintained for each class. Given a new instance, each prototype is compared to the instance by computing the similarity-score between the instance and each of the prototypes for the different classes. The class predicted is the one which achieves the highest similarity score. After the algorithm makes a prediction, it receives the correct label of the instance and updates the set of prototypes. In this context, online feature extraction has also been studied in the prototype based classification. Incremental Principal Component Analysis (iPCA) was the first online extension of the well known PCA unsupervised feature extraction technique (Artac et al., 2002; Hall et al., 1998). More recently Pang et al. (2005) proposed an extension of its supervised counterpart based on Fisher Linear Discriminant Analysis criterion. In addition, Polikar et al. designed the Learn++ algorithm (Polikar et al., 2001; Muhlbaier et al., 2009) where an ensemble of weak NN classifiers is generated to incrementally accommodate new samples to a previously learnt classifier.

3. Error correcting output codes

Error correcting output codes (ECOC) is a metalearning strategy that allows to extend any binary classifier to the multiclass case. The classic ECOC meta learning algorithm (Dietterich and Bakiri, 1995) has two phases: in the learning step, an ECOC encoding matrix is constructed in order to define the combination of the M binary classifiers that allow full multi-class classification. In the testing phase (decoding step), the new sample $\boldsymbol{x}$ is classified according to the M binary classifiers set. The decoding algorithm finds the most suitable class label for the test sample using the output of this binary set of classifiers.

Briefly, given a set of N training samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where each $\boldsymbol{x}_i$ belongs to the class $C_i \in \{C_1, \ldots, C_K\}$, an ECOC encoding consists on constructing M binary problems (called dichotomizers h_j) from the original K classes. At each dichotomizer, the class set is split into the binary classes $\{+1, -1\}$, forming a $K \times M$ encoding ECOC matrix $\boldsymbol{T}$. When a new sample must be classified, the outputs of the dichotomizers (columns of the matrix $\boldsymbol{T}$) are used to construct the codeword that is compared with each row of the matrix $\boldsymbol{T}$. The class with the minimum distance is selected as the classifier output. In this binary grouping setting, the total number of possible splits is $2^{K-1} - 1$, being the efficient construction of the ECOC matrix $\boldsymbol{T}$ the key issue in the training step. The encoding step was extended by Allwein et al. (2002) to include a new symbol – symbol 0 – so that, a class can be omitted in the training of a particular dichotomizer. Classical multi-class classification strategies, such as the one-vs-all or one-vs-one (when ternary representations are used) can be easily represented as an ECOC matrix. Nevertheless, more sophisticated problem dependent encodings have been shown to outperform classical approaches (Pujol et al., 2006; Escalera et al., 2008), without a significant increment of the codeword length.

The most simple strategy for decoding is to use the Hamming distance between the output of the dichotomizers on the new test sample and each codeword (row) of the encoding matrix. However, other strategies, such as Euclidean decoding, Probabilistic decoding or Loss-based decoding can be used in this step.

3.1. Weighted decoding

In this section we detail the decoding strategy that is used throughout this article. This strategy has been shown (Escalera et al., 2008) to be generally better than most of the state of the art decoding measures. The weighted decoding strategy decomposes the decoding step of the ECOC technique in two parts: a weighting factor for each code position and any decoding strategy. The weighting methodology is designed to fulfill two properties that allow a better behavior of the binary and ternary decoding. In (Escalera et al., 2006), the authors show that for a decoding strategy in a ternary ECOC to be successful, the bias induced by the zero symbol should be zero. Additionally, the dynamic range of the decoding strategy must be constant for all the codewords. Then, the goal of the weighting matrix is twofold: first, it isolates the decoding strategy from the commented properties. Second, it allows to encode the confidence on the prediction of each class.

Definition. We define the metaclass relative accuracy (r-value) of class k on the set S given the definition of the metaclasses ρ as,

$$r_k(S,\rho,i) = \frac{\#\text{elements of class } k \text{ classified as metaclass } i \text{ in the set } S}{\#\text{elements belonging to class } k \text{ in the set } S}, \quad (1)$$

where ρ defines which classes belong to which metaclass. Using the ECOC notation, this is defined by a column of the matrix T (e.g. for column j, $\rho = T(\cdot,j)$). Observe that for obtaining this value for all classes, we just have to test once the classifier trained for dichotomy j on the set S.

In order to encode the prediction of each class, one can use the r-values associated to each dichotomizer in the following way,

$$w(i,j) = \begin{cases} r_i(S,T(\cdot,j),T(i,j)) & T(i,j) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The zero value ensures that the bias due to the zero symbol is nullified at the decoding step. Moreover, to fulfill the second requirement (same dynamic range), each row of the matrix is normalized,

$$\sum_{j=1}^{M} w(i,j) = 1, \quad \forall i \in \{1 \ldots K\}.$$

The second part of the weighting decoding depends on the base decoding strategy. In this article, we chose to use Linear Loss-based decoding as base strategy decoding. Linear Loss-based decoding was introduced by Allwein et al. (2002) and is defined in the following way: given the input sample x and the binary code y result of applying all the dichotomizers ($h_1, h_2, \ldots, h_M$) to the input test sample, the decoding value is defined as follows:

$$d(y,T(i,\cdot)) = \sum_{j=1}^{M} \mathcal{L}(T(i,j) \cdot h_j(x)),$$

where $T(i,\cdot)$ denotes the codeword for class i, $h_j(x)$ is the prediction value for dichotomizer j, and $\mathcal{L}$ is a loss function that represents the penalty due to the misclassification. In the case of Linear Loss-based decoding $\mathcal{L}(\rho) = -\rho$.

The complete decoding strategy weights the contribution of the decoding at each position of the codeword by the elements of W. As such, the final decoding strategy is defined as,

$$d(y,T(i,\cdot)) = \sum_{j=1}^{M} W(i,j) \cdot \mathcal{L}(T(i,j) \cdot h_j(x)).$$

From the point of view of online processes, this decoding strategy is directly applicable to new classes. Observe that the weighting matrix for a class depends only on the elements of that class, disregarding the rest of the data. Given that, each time a new class is introduced in the framework only the information concerning this new class needs to be computed.

4. Online ECOC coding

In general, the addition of a new class in the ECOC matrix reshapes it in two ways: first, a new code must be defined for the new class. Second, new dichotomies could be needed to discriminate the new class. Additionally, we have a restriction, any knowledge in the ECOC matrix should be preserved, this means that it is undesirable to retrain previously learnt dichotomizers.

In this section, we introduce the extension of classic ECOC strategies – one-versus-all, one-versus-one strategies – to the layer-2 online case and propose a novel problem dependent layer-2 online ECOC coding that takes advantage of the particularities of the dataset for defining a more efficient ECOC coding in terms of number of classifiers and performance. With the exception of the first proposal, the rest of the techniques are independent of the base classifier and can accommodate both layer-1 online and batch learners.

4.1. Problem independent online ECOC designs

4.1.1. One-versus-all online ECOC (1vsAllo)

The simplest approach for extending the ECOC framework to the online case is to take advantage of layer-1 base online learners. Recall that layer-1 base online learners are incremental learners that adapt to new samples without the need of retraining the full classifier on the whole data set. Thus, it is straight forward to think that if the new data belongs to a new class, one only needs to train this data for each dichotomizer according to the metaclass membership defined by the ECOC matrix.

The first possible extension of ECOC to online ECOC is the online one-versus-all ECOC. The algorithm takes as input the $K \times M$ encoding ECOC matrix $\boldsymbol{T}$, where each $T(i,j) \in \{+1,-1\}$ represents the binary metaclass membership of the class i in the dichotomizer j. In the online 1vsAll encoding strategy, the addition of samples of a new class C_{K+1} generates a new dichotomizer h_{M+1} and a new codeword C_{K+1}, where,

$$T(i,M+1) = \begin{cases} -1 & i \neq K+1, \\ +1 & \text{otherwise.} \end{cases} \quad (2)$$

Additionally, all positions of the new codeword will be assigned to the metaclass -1, except for the position $M+1$,

$$T(K+1,i) = \begin{cases} -1 & i \neq M+1, \\ +1 & \text{otherwise.} \end{cases} \quad (3)$$

Fig. 2. 1vsAll encoding strategy.

The process is exemplified in Fig. 2, where a black square represents a -1 symbol in the matrix T, and a white square corresponds to the symbol +1. Notice that the addition of the new class increases in one row and column the characteristic identity matrix of the one-versus-all online approach.

4.1.2. Unbalanced online ECOC (UNBo)

If we want to extend the online layer-2 behavior to batch base classifiers, we must be sure not to retrain the classifiers whenever the metaclass definition is changed by the fact of adding a new codeword. This objective can be achieved by ignoring new classes in previously trained classifiers. This leads to a degenerate matrix which we call unbalanced online ECOC[1] (UNBo).

Analogous to the previously defined method, the online unbalanced ECOC requires of a new dichotomizer h_{M+1} and a new codeword C_{K+1}. The new dichotomizer is defined by the one-versus-all strategy as shown in Eq. (2). On the other hand, since we cannot afford to retrain any previously learnt dichotomizer, the new codeword preserves the original encodings by adding the 'don't care' symbol to all positions of the new codeword except for the position $M+1$, in the following way,

$$T(K+1,i) = \begin{cases} 0 & i \neq M+1, \\ +1 & \text{otherwise.} \end{cases}$$

The process is exemplified in Fig. 3(a), where a black square represents the -1 symbol in the matrix $\boldsymbol{T}$, a grey square represents the zero symbol and a white square the +1 symbol. In this case, the incremental addition of new classes generates binary dichotomizers as in the 1vsAll case. However, the codewords for the new classes are selected in such a way that do not distort the previously learned dichotomizers. From the perspective of tree embedding into ECOC matrices (Pujol et al., 2006) this kind of codification comes from an unbalanced classification tree. Fig. 3(b) shows the unbalanced tree that generates the ECOC matrix of Fig. 3(a).

4.1.3. One-versus-one online strategy (1vs1o)

Another widely used ECOC coding strategy is one-versus-one. The definition of this technique allows to extend its behavior to the online case in a simple way independently of the type of base classifier used. Observe that in this scheme a new class introduces a new codeword for class $K+1$ and K dichotomizers corresponding to all pairs of classes involving class $K+1$.

Given the $K \times M$ encoding ECOC matrix $\boldsymbol{T}$, and using the ternary representation, $T(i,j) \in \{+1, 0, -1\}$, the online one-versus-one method requires the definition of a new codeword of size $(K+1)K/2$ as follows:

$$T(K+1,i) = \begin{cases} 0 & i \leqslant K, \\ +1 & \text{otherwise.} \end{cases}$$

Observe that the new class it is not considered in the previously trained classifiers. On the other hand, a new class will need of K new classifiers,

$$T(i,M+j) = \begin{cases} -1 & i = j, \\ +1 & i = K+1, \\ 0 & \text{otherwise.} \end{cases} \quad \forall i,j \in \{1 \ldots K\}.$$

The main drawback of this technique is that the addition of a new class makes the ECOC matrix to grow linearly, resulting in quadratic codewords with respect to the number of classes. This makes this approach unfeasible if the expected number of classes is high – as it is usual in machine vision applications.

[1] Note that, though this method is motivated by the necessity of extending the one-versus-one technique when the base dichotomizers belong to the batch classifiers family, this strategy supports both layer-1 online and batch learners.

4.2. General problem dependent online ECOC (PDo)

In the previous algorithms, the grouping properties of the encoding matrix $\boldsymbol{T}$ are predefined at design time. Nevertheless, it seems that more efficient grouping decisions could be performed when the specific nature of the problem is taken into account. A problem dependent approach can select the proper values of $T(i,j)$ using a data driven criteria, such as the training error on a validation subset. For this purpose, we take advantage of the previously defined weighted decoding, which allows to take into account the metaclass relative accuracy (r-value) as defined in Eq. (1). The new algorithm is independent of the base classifier applied.

Algorithm 1. General algorithm for the creation of the problem dependent layer-2 online ECOC matrix.

Input: Set of data points $S = \{(x_i, C_i) | x_i \in \boldsymbol{X} \wedge C_i \in \mathbf{C}\}$ divided in a training set $S_t \subset S$ and a validation set $S_v \subset S$ so that, $S_t \bigcup S_v = S$ and $S_t \bigcap S_v = \varnothing$

Input: ECOC matrix $\boldsymbol{T}$ of size $K \times M$, $K = |\mathbf{C}|$

Input: Set of new training instances $S_o = \{x_{N+1}, \ldots, x_{N+U}\}$ from a new class C_{K+1}

Input: Parameters ϵ and α

Output: Expanded ECOC matrix $\widetilde{T}$

begin step 1: *Vertical expansion*

for each column/dichotomy $j \in \{1, \ldots, M\}$ **do**

/* Find the weight associated to that class for dichotomy j as the maximum metaclass relative accuracy for all possible codes */

$W(K+1,j) = max(\alpha r_{K+1}(S_t, T(\cdot,j), l) + (1-\alpha)\, r_{K+1}(S_v, T(\cdot,j), l)) \quad \forall\, l \in \{1, -1\}$

if $W(K+1,j) < \epsilon$ **then**

$W(K+1, j) = 0$; $\widetilde{T}(K+1,j) = 0$

else

/* Fill the ECOC matrix with the code value that maximized the weight */

$\widetilde{T}(K+1,j) = argmax_l(\alpha r_{K+1}(S_t, T(\cdot,j), l) + (1-\alpha) r_{K+1}(S_v, T(\cdot,j), l)) \quad \forall l \in \{1, -1\}$

end

begin step 2: *Base horizontal expansion*

$\widetilde{T}(K+1, M+1) = -1$

$\widetilde{T}(j, M+1) = 1 \quad \forall j \in \{1 \ldots K\}$

$w(K+1, M+1) = \alpha r_{K+1}(S_t, T(\cdot, M+1), -1) + (1-\alpha)\, r_{K+1}(S_v, T(\cdot, M+1), -1)$

end

begin step 3: *Problem dependent horizontal expansion*

while $w(K+1, M+1) \leqslant \epsilon$ **and** $|\{\widetilde{T}(j, M+1) = 1, \quad \forall j \in \{1 \ldots K\}\}| > 1$ **do**

Calculate the confusion vector with respect to class C_{K+1}

Select the class C_e with maximum error

$\widetilde{T}(e, M+1) = 0$

Add a new column at position $s = length(\widetilde{T}) + 1$ so that,

$$\widetilde{T}(j,s) = \begin{cases} 1 & j = e \\ -1 & j = K+1 \\ 0 & \text{otherwise} \end{cases}$$

Find the new weights according to the new dichotomy definitions

$w(j, M+1) = \alpha r_j(S_t, T(\cdot, M+1), T(j, M+1)) + (1-\alpha) r_j(S_v, T(\cdot, M+1), T(j, M+1))$

and $w(j,s) = \alpha r_j(S_t, T(\cdot,s), T(j,s)) + (1-\alpha) r_j(S_v, T(\cdot,s), T(j,s)) \quad \forall j \in \{1, \ldots, K+1\}$

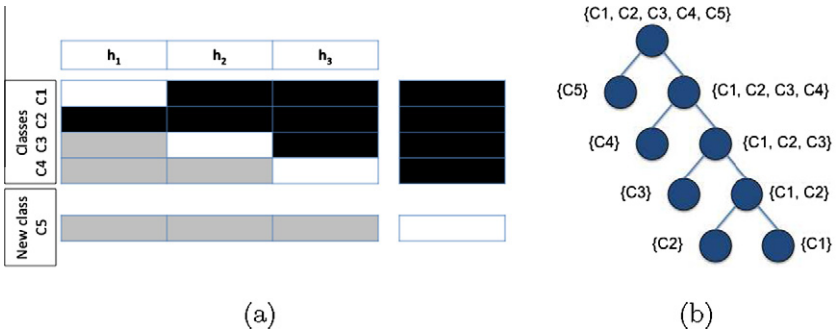

Fig. 3. Unbalanced encoding strategy. (a) Coding matrix and (b) unbalanced tree associated.

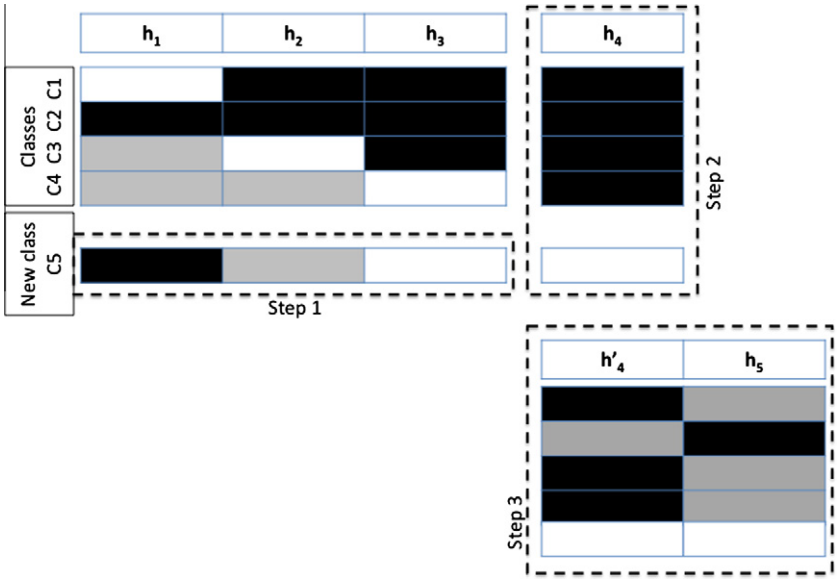

Fig. 4. Problem dependent online strategy.

The problem dependent online ECOC coding is built in three steps:

(1) *Vertical extension:* This step consists of creating a new codeword for the new class. As such, a new row is added to the matrix. The values of this codeword depend on how well each dichotomy models the new class. The metaclass relative performance is a measure that reflects how well the dichotomizer splits the new class when it is considered as belonging to the metaclass +1 or −1. Then, the new code will be the value of the metaclass that maximizes the metaclass relative performance. Note that the maximum of this value lies between [0.5...1]. Additionally, the value is stored in the weighting matrix as a measure of the confidence in that code position. Observe that if one considers the *r*-value to be too small, one has the option of setting the value to zero meaning that we 'do not consider' that dichotomizer in the new codeword. We reflect this issue by allowing the user to define a minimum confidence represented by ϵ ($\epsilon > 0.5$). In Fig. 4, the first step tests dichotomizers (h_1, h_2, h_3) with the samples of the new class, assuming that they belong to the metaclasses {+1, −1}. The value of the metaclass with higher *r*-value is assigned to the new code. In the example, the new code corresponds to {−1, 0, +1}. As we commented, the *r*-value obtained for the new class by h_2 is not higher than *epsilon* and, thus, the result of the dichotomizer is not considered in the decoding step.

(2) *Base horizontal extension:* In general, a new class requires of an specialized dichotomizer. The second step uses a one-versus-all configuration as basic horizontal extension to define the new dichotomizer. This step requires the training of a new classifier and the adjustment of the corresponding *r*-values in the weighting matrix. In Fig. 4, the second step sets class 5 to +1 and the rest to −1. Again the *r*-value for each class must be computed and stored in the weighting matrix *W*.

(3) *Problem dependent horizontal refinement:* This step proposes an *r*-value driven variable codeword expansion. If the *r*-value for the new class in the specialized dichotomizer

Table 1
UCI repository data sets characteristics.

Problem	#Training samples	#Features	#Classes
Dermathology	366	34	6
Iris	150	4	3
Ecoli	336	8	8
Vehicle	846	18	4
Wine	178	13	3
Segmentation	2310	19	7
Glass	214	9	7
Thyroid	215	5	3
Vowel	990	10	11
Balance	625	4	3
Yeast	1484	8	10

Table 2
UCI classification results using online classifiers.

DB	1vs1o	iLDA	1vsAllo	UNBo	PDo	Batch PDo	Learn++	SVM
Balance	**0.97 (0.02)***	0.84 (0.03)	0.91 (0.01)	**0.97 (0.02)***	**0.97 (0.02)***	**0.97 (0.02)***	**0.94 (0.02)**	0.97 (0.02)
Wine	0.61 (0.04)	**0.74 (0.06)***	0.51 (0.10)	0.62 (0.03)	**0.65 (0.05)**	0.64 (0.04)	**0.71 (0.04)**	0.61 (0.04)
Thyroid	**0.95 (0.03)**	0.72 (0.03)	**0.90 (0.04)**	**0.95 (0.03)**	**0.96 (0.03)***	**0.96 (0.03)***	0.90 (0.02)	0.95 (0.03)
IRIS	**0.97 (0.04)***	**0.97 (0.03)***	**0.97 (0.04)***	**0.97 (0.04)***	**0.95 (0.04)**	**0.95 (0.04)**	**0.93 (0.02)**	0.97 (0.04)
Glass	**0.46 (0.04)**	**0.53 (0.08)***	**0.46 (0.02)**	0.36 (0.03)	**0.50 (0.04)**	**0.49 (0.04)**	**0.50 (0.03)**	0.46 (0.04)
Ecoli	**0.86 (0.02)***	**0.82 (0.04)**	0.77 (0.03)	**0.79 (0.04)**	**0.84 (0.03)**	**0.86 (0.02)**	**0.85 (0.02)**	0.85 (0.02)
Yeast	**0.59 (0.02)***	0.52 (0.02)	0.46 (0.02)	**0.56 (0.03)**	**0.58 (0.03)**	**0.57 (0.03)**	0.51 (0.03)	0.59 (0.02)
Vowel	0.55 (0.04)	**0.74 (0.04)***	0.39 (0.09)	0.45 (0.04)	0.54 (0.05)	0.50 (0.03)	**0.72 (0.01)**	0.53 (0.04)
Derma.	**0.96 (0.01)***	0.88 (0.03)	0.81 (0.03)	0.68 (0.01)	**0.96 (0.01)***	**0.96 (0.01)***	**0.95 (0.01)**	0.96 (0.01)
Vehicle	**0.72 (0.02)**	0.38 (0.04)	**0.70 (0.01)**	**0.73 (0.02)***	**0.73 (0.02)***	**0.73 (0.03)***	**0.73 (0.02)***	0.72 (0.02)
Segmen.	**0.95 (0.02)***	0.60 (0.01)	0.87 (0.02)	0.86 (0.02)	**0.95 (0.02)***	**0.91 (0.03)**	**0.90 (0.03)**	0.95 (0.02)
Rank	1.86	3.68	5.13	3.77	1.50	2.04	2.86	
Dichot.	16.73	–	5.64	5.64	8.52	8.52		

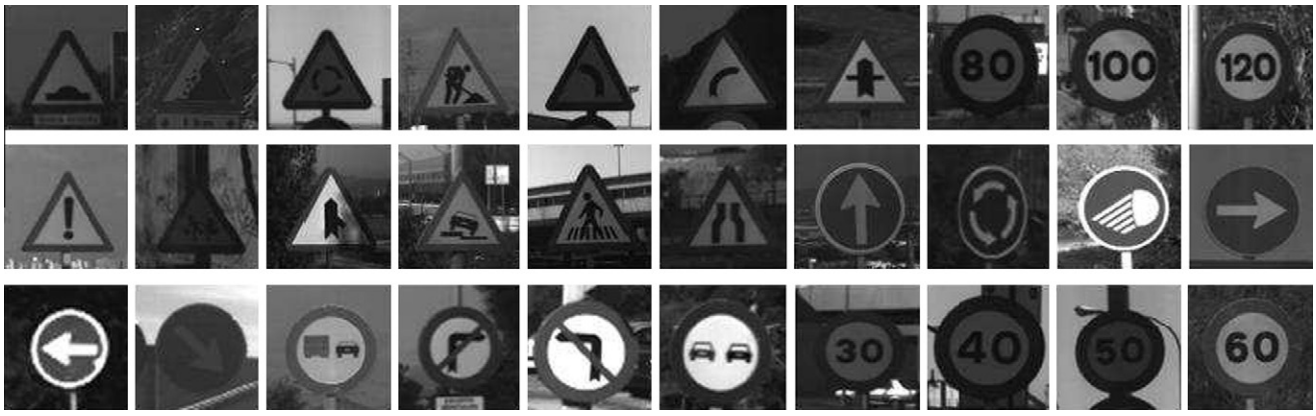

Fig. 5. Traffic sign classes.

obtained at step 2 is too small, we can look for a more suitable dichotomy configuration by designing and adding a new set of columns in order to obtain the desired *r*-values for the new class. To perform this step, we look for the class that reduces the metaclass relative performance of the new class the most, by means of the confusion matrix. Once this value is located, we remove this class by setting the value in the new dichotomizer to zero. We create an specialized dichotomizer that trains the new class against the removed one. In Fig. 4, C_2 is the class with the highest error in the confusion vector for the new class. Thus, we set the value corresponding to that class to zero in dichotomizer 4. Furthermore, we create a new dichotomizer specialized on distinguishing C_2 from C_5.

Algorithm 1 shows the detailed procedure for coding a problem dependent online ECOC. Note that the algorithm splits the training set in two subsets: the validation and training sets. The validation set is used as an external oracle and allows to model the generalization of the method avoiding or delaying overfitting in the matrix designing process.

Although this algorithm is general with respect of the base classifier, few considerations could be taken into account to take advantage of the full capabilities of the base classifier:

- *The base classifier belongs to the batch family:* In this case, the *r*-values come from the evaluation of the dichotomizer assuming that the new items belong either to +1 or −1. This means that we have to test each classifier on both training and validation sets for all the elements of that class and obtain and combine the *r*-values. This requires to run each dichotomizer once on the subset for the new class.
- The base classifier is layer-1 online: In this case, we can take advantage of the online behavior of the classifier and we can choose either to treat it as a batch classifier or to incrementally train each dichotomizer with the data of the new class in order to obtain the best codification. In the same way, as in the other case, the *r*-values for the new data must be obtained and combined.

5. Results

In order to present the results, first, we discuss the data, methods, measurements, and experimental settings of the experiments.

- *Data*: The first data used for the experiments consists of 11 multi-class data sets from the UCI Machine Learning Repository database (Asuncion and Newman, 2007). The number of training samples, features, and classes per data set are shown in Table 1. Then, we apply the online classification methodology in two challenging computer vision categorization problems. First, we use the video sequences obtained from a Mobile Mapping System (Casacuberta et al., 2004) to test the methods in a real traffic sign categorization problem consisting of 36 traffic sign classes. And second, 30 classes from the ARFaces (Martinez and Benavente, 1998) data set are classified using the present methodology.
- *Methods*: For the experimental evaluation, we test the online and batch classifiers presented in this paper. Concerning the batch classifiers, we test the multi-class OSU implementation of Support Vector Machines with Radial Basis Function kernel (Vapnik, 1995; Osu-svm-toolbox, xxxx). In the case of the online classifiers, we compare the *iLDA* with one-nearest neighbor classifier and our ECOC online methodologies: online one-

versus-all ECOC (*1vsAllo*), online unbalanced ECOC (*UNBo*), online one-versus-one ECOC (*1vs1o*), the Problem Dependent online ECOC (*PDo*), and the online Problem Dependent ECOC with batch base classifier (batch *PDo*). For all the ECOC strategies, the decoding is performed with the weighted decoding with Linear Loss-based function, as described in Section 3.1, to test the methods performances in the same conditions. We also provide a comparative with the incremental learn++ algorithm of Muhlbaier et al. (2009).

- *Measurements*: To measure the performance of the different strategies, we apply stratified tenfold cross-validation and test for the confidence interval with a two-tailed *t*-test. We also use the Friedman and Nemenyi test to look for significant statistical differences between the methods' performances (Demsar, 2006).
- *Experimental settings*: All the data used in the experiments is normalized to a hypercube with side length of one. The *RBF SVM* classifier is tuned via cross-validation, where the sigma and regularization parameters are tested from 0.05 increasing per 0.05 up to one and from one increasing per five up to 150, respectively. Once the multi-class *SVM* has been optimized, the optimal parameters for each data set are shared for all the classification strategies that use *SVM* as the base classifier and for the online version of *SVM* used in the *PDo* approach. The optimization for the *PDo* classifiers is done via cross-validation of α on the validation set from 0.2 up increasing per 0.2 up to 0.8 and ϵ is estimated from 0.05% increasing per 0.05 up to 0.4. All online multi-class experiments are solved by considering an initial 2-class problem and progressively increasing the number of classes by one.

5.1. Validation over UCI data sets

In order to understand the behavior of the different ECOC online strategies on different feature spaces, we classify the 11 UCI data sets of Table 1 with the online classifiers. The average accuracy and rankings are shown in Table 2. The asterisks mark the best performance and the values in bold correspond to the methods which fall within the 95% confidence interval of the best result. The rankings are obtained estimating each relative rank r_i^j for each data set i and each classification strategy j, and computing the mean ranking R for each classifier as $R_j = \frac{1}{I}\sum_{i=1}^{I} r_i^j$, where I is the total number of data sets. Thus, the smaller the ranking value, the best results the method achieves. The numbers in parenthesis correspond to the confidence interval computed using a two-tailed *t*-test. For comparison purposes, the last column in the table shows the SVM results trained as a multi-class off-line classifier, which are not used in the computation of the ranking and confidence interval. Notice that the best online method is the *PDo*, followed by the 1*vs*1*o* and the batch*PDo*. In addition, *PDo* compares favorably with the reference off-line SVM trained with the full set of classes, being their performance not statistically distinguishable in any dataset. In order to further analyze the results, we test for statistical significance applying the Nemenyi test (Demsar, 2006): two techniques are significantly different if the corresponding average ranks differ by at least the critical difference value (CD):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{4}$$

where q_α is based on the Studentized range statistic divided by $\sqrt{2}$. In our case, when comparing seven methods ($k = 7$) in 11 experiments ($N = 11$ data sets) with a confidence value $\alpha = 0.10$, $q_{0.10} = 1.41$. Substituting in Eq. (4), we obtain a critical difference value of 1.29. Observing the ranks of each classifier in the global rank row of Table 2, one can see that the only combination of methods for which the difference is smaller than the critical value of 1.29 is achieved when comparing the *PDo* approach, the batch *PDo* and online one-versus-one ECOC against the rest. Therefore, we can argue that these strategies are significantly better than the rest in the present experiments. Observe also that the mean number of

Table 3
Traffic signs data set classification results.

Strategy	1vs1o	1vsALL	SVM	iLDA	Learn++
Performance	**0.98 (0.01)***	**0.95 (0.02)**	**0.97 (0.01)**	0.90(0.01)	0.83(0.04)
Rank	1.9	3.5	2.7	7.3	8.2
Strategy	1vsAllo	UNBo	PDo	batch PDo	
Performance	0.94 (0.01)	0.92 (0.03)	**0.95 (0.02)**	0.93 (0.03)	
Rank	5.4	6.2	3.4	4.9	

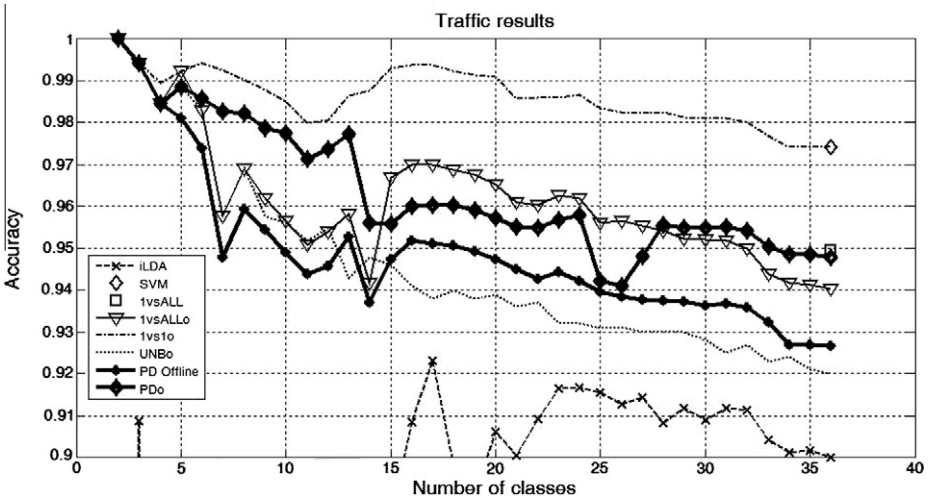

Fig. 6. Traffic sign data set performance results.

dichotomizers in *PDo* and batch *PDo* is half the number of dichotomizers of the 1*vs*1*o* approach. We argue that this fact is important when considering computational resources. On the other hand, the mean ranks for the *iLDA* and Learn++ classifiers show that they offer better results than the online one-versus-all and unbalanced approaches for these data sets.

5.2. Computer vision applications

In this section, we test the ECOC online methodology on two real challenging Computer Vision problems: traffic sign and face categorization.

5.2.1. Traffic sign categorization

For this first computer vision experiment, we use the video sequences obtained from the Mobile Mapping System of Casacuberta et al. (2004) to test the online ECOC methodology on a real traffic sign categorization problem. In this system, the position and orientation of the different traffic signs are measured with video cameras fixed on a moving vehicle. The system has a stereo pair of calibrated cameras, which are synchronized with a GPS/INS system. The result of the acquisition step is a set of stereo-pairs of images with their position and orientation information. From this system, a set of 36 circular and triangular traffic sign classes are obtained. Some categories from this data set are shown in Fig. 5. The data set contains a total of 3481 samples of size 32×32, filtered using the Weickert anisotropic filter, masked to exclude the background pixels, and equalized to prevent the effects of illumination changes. These feature vectors are then projected into a 100 feature vector by means of PCA. More details of this system can be found in Escalera et al. (2010).

The classification results of the traffic sign data set considering all the previous classification strategies and ranks are numerically and graphically shown in Table 3 and Fig. 6, respectively. The ranks are computed taking into account each iteration of the 10-fold evaluation as a different experiment. On can see that the *PDo* and batch *PDo* strategies are close to the online one-versus-one and batch *SVM* strategies, meanwhile the online one-versus-all and unbalanced ECOC approaches are similar to the batch one-versus-all ECOC design.

In order to analyze if the difference between methods ranks are statistically significant, we apply the Friedman and Nemenyi tests. In order to reject the null hypothesis that the measured ranks differ from the mean rank, and that the ranks are affected by randomness in the results, we use the Friedman test. The Friedman statistic value is computed as follows:

$$X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]. \quad (5)$$

In our case, with $k = 9$ designs to compare, $X_F^2 = 29.13$. Since this value is undesirable conservative, Iman and Davenport proposed a corrected statistic:

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2}. \quad (6)$$

Applying this correction we obtain $F_F = 5.15$. With nine methods and ten experiments, F_F is distributed according to the F distribution with eight and 72 degrees of freedom. The critical value of $F(8,72)$ for 0.05 is 2.02. As the value of F_F is higher than 2.02 we can reject the null hypothesis. Once we have checked for the non-randomness of the results, we can perform a post hoc test to check if one of the techniques can be singled out. For this purpose we use the previously commented Nemenyi test. In our case, when comparing nine methods with a confidence value $\alpha = 0.10$, $q_{0.10} = 1.38$. Substituting in Eq. (4), we obtain a critical difference value of 1.69. Considering this *CD* value, only the 1vs1o, SVM, 1vsAll, and PDo methods can be singled out as the best methods in the traffic sign recognition experiment.

5.2.2. Face classification

We applied the online ECOC methods described in this paper to a class-incremental face classification problem using the public AR Face database. The AR Face database (Martinez and Benavente, 1998) is composed of 26 face images from 126 different subjects (70 men and 56 women). The images have uniform white background. The database has from each person two sets of images, acquired in two different sessions, with the following structure: one sample of neutral frontal images, three samples with strong changes in the illumination, two samples with occlusions (scarf and glasses), four images combining occlusions and illumination changes, and three samples with gesture effects. One example of each type is plotted in Fig. 7. For this experiment, we selected all the samples from 30 different categories (persons). The classification results and ranks applying the batch and online base classifiers are shown in Table 4 and Fig. 8, respectively. The ranks are computed taking into account each iteration of the 10-fold evaluation as a different experiment. The differences among strategies are similar to the previous cases. The better results are obtained by the online one-versus-one ECOC and batch *SVM* approaches, followed by the *PDo* and batch *PDo* strategies, respectively. Finally, the *iLDA* approach offers the less performance in this problem.

In order to analyze if the difference between methods ranks are statistically significant, we apply the previously described Friedman and Nemenyi tests. In our case, with $k = 9$ designs to compare, $X_F^2 = 26.04$. Applying the corrected statistic of Iman and Davenport, we obtain $F_F = 4.34$. With nine methods and ten experiments, F_F is distributed according to the F distribution with eight and 72 degrees of freedom. The critical value of $F(8,72)$ for 0.05 is 2.02.

Fig. 7. ARFaces data set classes. Examples from a category with neutral, smile, anger, scream expressions, wearing sun glasses, wearing sunglasses and left light on, wearing sun glasses and right light on, wearing scarf, wearing scarf and left light on, and wearing scarf and right light on.

Table 4
AR faces data set classification results.

Strategy	1vs1o	1vsALL	SVM	iLDA	Learn++
Performance	**0.88 (0.06)***	0.69 (0.10)	**0.88 (0.06)***	0.49 (0.09)	0.72 (0.08)
Rank	1.3	6.8	1.4	7.5	5.2
Strategy	1vsAllo	UNBo	PDo	batch PDo	
Performance	0.68 (0.10)	0.55 (0.08)	**0.83 (0.07)**	**0.74 (0.09)**	
Rank	6.9	6.7	2.1	3.8	

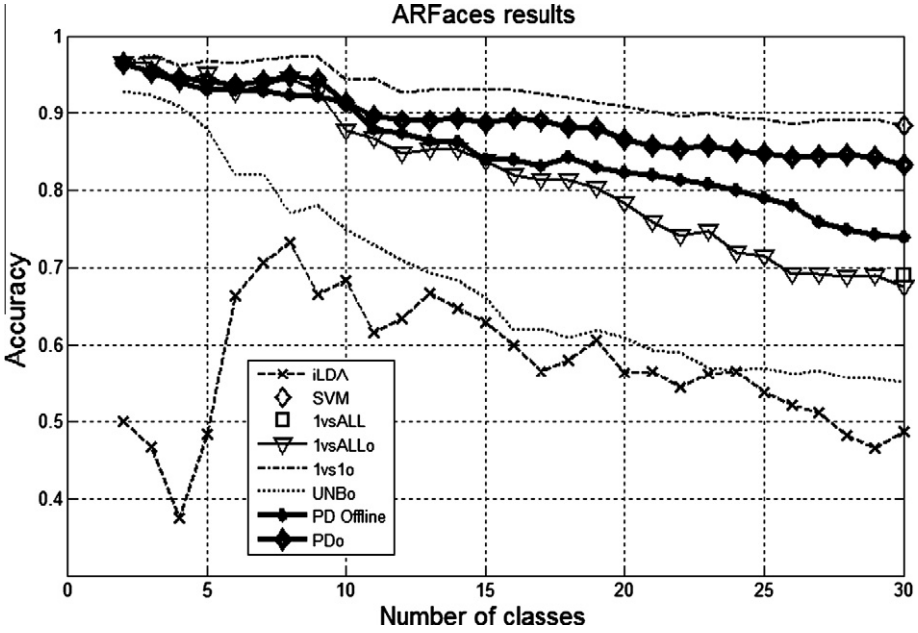

Fig. 8. ARFaces data set performance results.

As the value of F_F is higher than 2.02 we can reject the null hypothesis. Once we have checked for the non-randomness of the results, we use the Nemenyi test. In this case, when comparing nine methods with a confidence value $\alpha = 0.10$, $q_{0.10} = 1.38$. Substituting in Eq. (4), we obtain a critical difference value of 1.69. Considering this *CD* value, as in the previous computer vision experiment, only the 1vs1o, SVM, and PDo approaches can be singled out as the best methods in the ARFace categorization experiment. This result is specially significant in the face classification field, given that both 1vs1o and PDo achieve similar accuracies as the state-of-the-art batch SVM technique, where all data has been made available in training time. The contribution is twofold: (i) face recognition applications can usually manage large amounts of training samples that can only be faced using online learning strategies, and (ii) often only a small portion of the face recognition problem is known at the first training stage (limiting the availability of data), therefore we seek for online learning techniques that can accommodate new classes and fare equivalently to solving the whole classification problem.

In addition, we show in Fig. 8 the evolution of the mean classification accuracies with respect to the number of classes. Notice that the more classes are included in the problem, the less accuracy is obtained (large class classification problems are inherently more complex). Nevertheless, the proposed ECOC-based methodologies show increased stability as a function of the number of classes in comparison with the classic Feature Extraction + Nearest Neighbor approach (iLDA in the figure), where a decrease of a 20% in the accuracy is observed as the number of classes increases. We conjecture that the error induced by the addition of new samples can be better isolated in a ECOC approach than a NN classifier, where a few set of outliers in the training set can miss lead the classification of multiple samples.

6. Conclusions

In this paper, we proposed an online version for the Error Correcting Output Codes framework. Different alternatives to design online ECOC matrices for both online and batch base classifiers have been proposed. We have shown different applications where the online methodology can be applied. In the different scenarios, multi-class problems are solved by considering an initial 2-class problem and progressively increasing the number of classes. Moreover, these online results have been compared with multi-class batch classifiers by learning directly the whole set of classes. We have proposed and compared four online ECOC approaches. The two first approaches, the problem independent online one-versus-all (1vsAllo) and unbalanced ECOC (UNBo), do not take into account the knowledge of the problem domain, so the analysis of the data before the ECOC matrix is designed is not required. Moreover, these two strategies yield a relatively small length codewords, since only *K* classifiers are required for an *K*-class problem. The 1vsAllo approach requires the use of an online base classifier, whereas the UNBo approach can work with either online and batch base classifiers. In the experimental section, we found that in most cases these two approaches tend to the results of the batch one-versus-all approach. A third problem-independent online ECOC approach has been proposed, the one-versus-one (1vs1o). In this case, the 1vs1o provides a successful way to use either batch and online

base classifiers reaching the same performance that its batch counterpart. However, in this case, a quadratic number of dichotomizers is required to model each problem. Finally, the last online ECOC approach corresponds to a problem dependent design (PDo). In this case, the knowledge of the problem domain is exploited so that for both online and batch base classifiers, the coding matrix adapts to the different distributions of the data, obtaining a robust trade off between performance and codeword length. Although in the present work we only consider the inclusion of new classes in the online ECOC framework, the addition of new samples to previously learnt classes results straightforward by using the online base classifiers, without modifying the coding matrix. The evaluation of the novel methodology has been performed on 11 data sets from the UCI machine learning repository and two real computer vision problems, traffic sign categorization and face recognition, showing robust results when compared to batch classifiers.

We plan as a future work to develop semi-supervised extensions of the proposed online ECOC methodology, and the use of decremental learning strategies for class/sample forgetting.

Acknowledgments

This work has been supported in part by a research grant from Projects TIN2009-14404-C02, FISPI061290 and CONSOLIDER INGENIO 2010 (CSD2007-00018). Authors would like to thank Robi Polikar for providing the source code of Learn++ algorithm used in the experiments.

References

Allwein, E., Schapire, R., Singer, Y., 2002. Reducing multiclass to binary: A unifying approach for margin classifiers. In: JMLR, vol. 1, pp. 113–141.

Artac, M., Jogan, M., Leonardis, A., 2002. Incremental pca or on-line visual learning and recognition. In: ICPR, vol. 3, pp. 781–784.

Asuncion, A., Newman, D., 2007. In: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. <http://mlearn.ics.uci.edu/MLRepository.html>.

Casacuberta, J., Miranda, J., Pla, M., Sanchez, S., Serra, A., Talaya, J., 2004. On the accuracy and performance of the GeoMobil system. In: Internat. Society for Photogrammetry and Remote Sensing.

Cauwenberghs, G., Poggio, T., 2000. Incremental and decremental support vector machine learning. In: Leen, T.K., Dietterich, T.G., Tresp, V. (Eds.), NIPS. MIT Press, pp. 409–415.

Crammer, K., Singer, Y., 2001. Ultraconservative online algorithms for multiclass problems. In: COLT '01/EuroCOLT '01: Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory. Springer-Verlag, London, UK, pp. 99–115.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. In: JMLR, vol. 7, pp. 1–30.

Dietterich, T., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. J. Artifi. Intell. Res. 2, 263–282.

Escalera, S., Pujol, O., Radeva, P., 2006. Decoding of ternary error correcting output codes. In: CIARP, vol. 4225, pp. 753–763.

Escalera, S., Tax, D.M.J., Pujol, O., Radeva, P., Duin, R.P.W., 2008. Sub-class problem-dependent design for error-correcting output codes. IEEE Trans. Pattern Anal. Machine Intell. 30 (6), 1041–1054.

Escalera, S., Pujol, O., Radeva, P., 2010. Traffic sign recognition system with β-correction. Mach. Vision. Appl. 21 (2), 99–112.

Hall, P.M., Marshall, A.D., Martin, R.R., 1998. Incremental eigenanalysis for classification. In: BMVC.

Katagiri, S., Abe, S., 2006. Incremental training of support vector machines using hyperspheres. Pattern Recognition Lett. 27 (13), 1495–1507.

Kong, E.B., Dietterich, T.G., 1995. Error-correcting output coding corrects bias and variance. In: ICML, pp. 313–321.

Martinez, A., Benavente, R., 1998. The ar face database. In: Computer Vision Center Technical Report #24.

Mitra, P., Murthy, C.A., Pal, S.K., 2000. Data condensation in large databases by incremental learning with support vector machines. In: ICPR, pp. 2708–2711.

Muhlbaier, M., Topalis, A., Polikar, R., 2009. Learn++.nc: Combining ensemble of classifiers combined with dynamically weighted consult-and-vote for efficient incremental learning of new classes. IEEE Trans. Neural Networks 20 (1), 152–168.

Osu-svm-toolbox, xxxx. <http://svm.sourceforge.net>.

Oza, N.C., 2000. Online ensemble learning. In: AAAI/IAAI. AAAI Press/The MIT Press, p. 1109.

Pang, S., Ozawa, S., Kasabov, N., 2005. Incremental linear discriminant analysis for classification of data streams. IEEE Trans. Systems Man Cybernet. – Part B 35 (5), 905–914.

Pedroso, J.P., Murata, N., 2000. Optimization on support vector machines. In: IJCNN, vol. 6, pp. 399–404.

Polikar, R., Upda, L., Upda, S., Honavar, V., 2001. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Trans. Systems Man Cybernet. Part C 31 (4), 497–508.

Pujol, O., Radeva, P., Vitrià, J., 2006. Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. In: PAMI, vol. 28, pp. 1001–1007.

Utgoff, P.E., Berkman, N.C., Clouse, J.A., 1997. Decision tree induction based on efficient tree restructuring. Machine Learn. 29 (1), 5–44.

Vapnik, V., 1995. In: The Nature of Statistical Learning Theory. Springer.

Vapnik, V., 1997. The support vector method. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (Eds.), ICANN, Lecture Notes in Computer Science, vol. 1327. Springer, pp. 263–271.