Information Sciences 223 (2013) 42–55

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Enhancing directed binary trees for multi-class classification

Elena Montañés, Jose Barranquero, Jorge Díez, Juan José del Coz *

Artificial Intelligence Center, University of Oviedo at Gijón, 33204 Asturias, Spain

ARTICLE INFO

Article history:
Received 10 February 2012
Received in revised form 3 August 2012
Accepted 10 October 2012
Available online 25 October 2012

Keywords:
Multi-class classification
Decomposition methods
Support vector machines
Directed binary trees
Generalization error bounds

ABSTRACT

One approach to multi-class classification consists in decomposing the original problem into a collection of binary classification tasks. The outputs of these binary classifiers are combined to produce a single prediction. Winner-takes-all, max-wins and tree voting schemes are the most popular methods for this purpose. However, tree schemes can deliver faster predictions because they need to evaluate less binary models. Despite previous conclusions reported in the literature, this paper shows that their performance depends on the organization of the tree scheme, i.e. the positions where each pairwise classifier is placed on the graph. Different metrics are studied for this purpose, proposing a new one that considers the precision and the complexity of each pairwise model, what makes the method to be classifier-dependent. The study is performed using Support Vector Machines (SVMs) as base classifiers, but it could be extended to other kind of binary classifiers. The proposed method, tested on benchmark data sets and on one real-world application, is able to improve the accuracy of other decomposition multi-class classifiers, producing even faster predictions.

1. Introduction

The multi-class classification problem has been extensively studied in the literature due to the great number of tasks that fall into this category. Two examples of this problem might be: diagnosing which of several illnesses a patient is suffering from according to certain symptoms [38], or classifying web content into several topics [26]. The starting point is a set of instances described by a group of properties and labeled as belonging to a certain class. The multi-class component appears when $m > 2$ different classes are involved, being an extension of the more traditional binary classification task. The goal is the same, i.e. to obtain a classifier able to correctly predict the true class of new unlabeled instances, but with more than two classes within the possibilities to choose from. Multi-class classification is a more complex task than it may seem a priori, as more combinations of classification errors may appear as the number of classes increases.

Although it is possible to extend the proposal of this paper to other classifiers, we will focus here on Support Vector Machines (SVMs), due to the extension of the analysis and to some decisions which depend on the base classifier used. In case of SVMs, two kinds of approaches are commonly adopted to tackle multi-class classification. The first solves a single optimization problem [7,8,29,45], whereas the second decomposes the original problem into a set of binary SVMs that must be combined after having been trained [1,20,37]. Although both are able to solve multi-class classification tasks, a single optimization presents some learning complexities that the combination of binary models is able to reduce [21]. Several attempts at extending binary learning algorithms have been proposed to provide methods of the former kind. However, most of them are unfeasible mainly because of their high computational cost or due to being difficult to handle [36]; if so,

* Corresponding author. Tel.: +34 985182501.
E-mail addresses: elena@aic.uniovi.es (E. Montañés), barranquero@aic.uniovi.es (J. Barranquero), jdiez@aic.uniovi.es (J. Díez), juanjo@aic.uniovi.es (J.J. del Coz).
URL: http://www.aic.uniovi.es (J.J. del Coz).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.10.011

experiments are limited to small data sets [45]. Hence, the decomposition approach seems to be a better choice for SVM multi-class classification [25], since each binary problem is sometimes simpler—in the sense that separability between only two classes is likely to be higher—and smaller in the number of examples, and hence, in the number of constraints of the corresponding optimization problem.

To design a decomposition-based method for multi-class classification, two decisions have to be made, namely: the decomposition technique used—the classes involved in each binary classifier—and the combination strategy applied subsequently, i.e. how to combine their outputs. As regards decomposition techniques, a wide range of studies are available in the literature. A survey of the main decomposition strategies used to solve multi-class problems is presented in [30]. One-vs-all [34], one-vs-one [24] and error-correcting output coding (ECOC) [11,46] are the most widely used. Other alternatives are Data-driven Error Correcting Output Coding (DECOC) [47], which was presented as an improvement of ECOC strategy, or all-and-one strategy [22] that combines both one-vs-one and one-vs-all classifiers. Besides, hierarchical clustering [32] or random strategies [16] have also been studied as a decomposition techniques for building a binary tree.

As combination strategies, several ones have also been proposed. Max-wins [20] employs a voting scheme and is usually applied together with the one-vs-one approach, while the winner-takes-all strategy [31] is a typical approach to combine one-vs-all classifiers. Another combination, which can be applied after some decomposition strategies, consists in arranging the classifiers into a hierarchical structure [5,16,19,27,32,37]. The main advantage of such approaches is that they reduce the time needed to predict the class of a new instance, since a sequence of discriminations takes place. The basic idea is to start at the top of the hierarchy and successively discard certain classes at each level until reaching the bottom of the structure, where only one class remains as candidate and is returned. In order to improve the performance of these methods, some efforts have been made to find a suitable location for each classifier in the graph [6,42]. However, the reported experimental results do not show a clear improvement. Finally, other alternatives employ stacked generalization and clustering [15] to induce a multi-class classifier or a neural network ensemble [34].

This paper focuses on the one-vs-one decomposition strategy and proposes a promising combination method to locate each pairwise model in a binary tree. We shall focus only on SVMs as base classifiers, due to the fact that our approach is based on a metric which depends on measuring the complexity of the pairwise models learned with a binary SVM. This makes, in turn, the method to be classifier-dependent. However, our ideas could be applied to other base classifier, whenever the complexity of the pairwise models induced with this learner can be somehow quantified. The experimental results show that our approach reduces the error rate while at the same time speeding up the prediction process compared to max-wins and other tree-based combination strategies; all of which use the same pairwise classifiers. Therefore, in what follows, only pairwise classifiers are considered. This means that a total of $m(m-1)/2$ binary models must be combined in some way to obtain a multi-class classifier. This paper mainly discusses those approaches based on the use of trees.

Our study does not include any comparison or discussion with other decomposition strategies, such as the one-vs-all or those based on error-correcting output codes. There are some such studies in the literature, the most relevant is perhaps [39], in which Rifkin and Klautau defend the one-vs-all strategy. The main conclusion of their paper is that when binary classifiers are properly tuned, the differences in performance are not significant, contrary to what other authors claim [1,21,25]. However, Rifkin and Klautau admit that the one-vs-one approach can produce more accurate results, especially on small data sets with a large number of classes, in addition to presenting certain advantages, mainly because it requires less training and testing time. Although the one-vs-one strategy has the shortcoming of involving a quadratic number of classifiers in relation to the number of classes, it trains faster than the one-vs-all approach, since the binary problems to learn are easier and smaller on account of only involving two of the original classes [25]. As regards error-correcting output coding methods, which allow different configurations of binary classifications according to a code matrix optimized for error-correction purposes, Allwein et al. [1] point out that, even when such optimization is obtained, many of the binary subproblems generated may be difficult to learn. For this reason, simpler one-vs-one decomposition has shown comparable or superior results to those of error-correcting output coding, as stated by Klautau et al. [28]. The combination of one-vs-all and one-vs-one classifiers, called all-and-one [22], seems to show slightly better performance over the two strategies on their own, but at the cost of training more classifiers. Besides, a recent study [23] of the same authors does not include this strategy in their empirical study, restricting it to one-vs-all, one-vs-one and ECOC techniques. In fact, they corroborates previous results present in the literature, being the main contribution of their paper that the base classifier used has more influence on the performance than the decomposition strategy.

The rest of the paper is organized as follows: Section 2 reviews state-of-the-art methods that use hierarchical structures to combine the predictions of the set of binary classifiers yielded by the one-vs-one decomposition strategy. Section 3 provides an in-depth description of the proposals of this paper aimed at increasing the accuracy of decomposition approaches based on the use of tree prediction schemes. Finally, our experimental results are reported in Section 4 and some conclusions are drawn in Section 5.

2. Related work

Decision Directed Acyclic Graphs (DDAGs) [37] and Adaptive Directed Acyclic Graphs (ADAGs) [27] are hierarchical structures that place a classifier at each internal node and a label at each leaf. Fig. 1 shows an example of these graph-based structures for combining binary classifiers in a four-classes task.

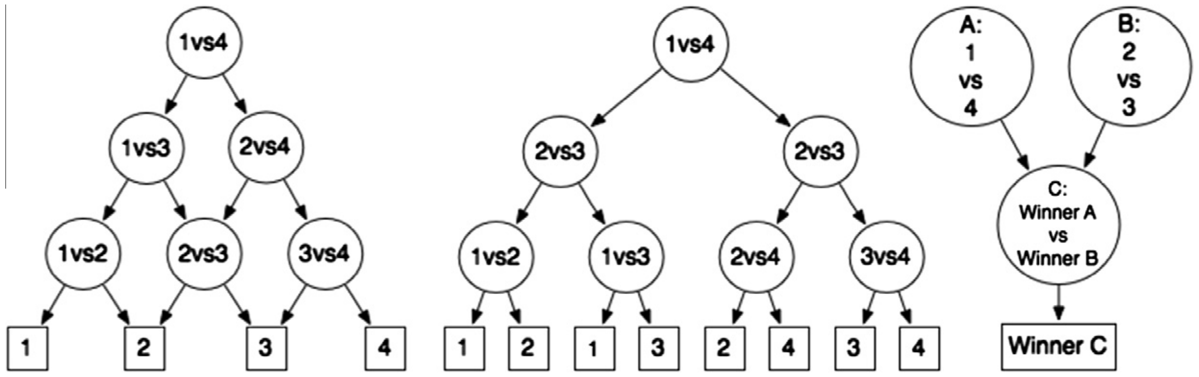


Fig. 1. Examples of a DDAGs (left), a DBTs (center) and an ADAGs (right) in a four-classes task.

A DDAG [37] is a directed graph with no cycles and, at most, two edges pointing to the same node. More precisely, the nodes of a DDAG are arranged in a triangle with a single root node at the top, two nodes in the second layer and so on down to the final layer, which is formed by m leaves. Besides the leaves, each node has an attached model, namely $\mathcal{M}_{i,j}$, the purpose of which is to separate the classes ℓ_i and ℓ_j. It also has two successors which will be two leaves when $i = j - 1$, or two decision nodes with models $\mathcal{M}_{i,j-1}$ and $\mathcal{M}_{i+1,j}$ otherwise. The prediction procedure of a DDAG starts from the root (model $\mathcal{M}_{1,4}$ in Fig. 1) and decides at each node, applying model $\mathcal{M}_{i,j}$, which of the two classes ℓ_i and ℓ_j is preferred for a certain instance. If the former is the winner, then the instance is evaluated over its left child node (model $\mathcal{M}_{i,j-1}$), so class ℓ_j is discarded. Otherwise, the instance is evaluated using the right child node (model $\mathcal{M}_{i+1,j}$), then class ℓ_i is discarded. The process continues until a leaf is reached whose label is returned.

Kijsirikul and Ussivakul [27] proposed ADAG in an attempt to overcome certain shortcomings in DDAG, that will be discussed in Section 3.2. An ADAG is an up-side-down tree that has $\lceil m/2 \rceil$ nodes on the top, in which pairwise classifiers are placed ensuring that all classes appear in that level (if the number of classes is odd, then a node with a single class is added). For classifying a particular instance, the outputs of the classifiers of a certain level determine the classifiers employed in the next layer. This procedure is repeated until the last layer (with a unique classifier) is reached, which gives the final prediction. Hence, a different structure of the graph may be applied depending on each instance, since the classifiers used in all levels but the first one depend on the predictions of previous layers. In the example of Fig. 1 (right), the classifier used at node C depends on the predictions made by the models placed at nodes A and B. Formally, if models $\mathcal{M}_{i,j}$ and $\mathcal{M}_{k,l}$ are employed in the current level for a certain instance $\boldsymbol{x}$, being $\mathcal{M}_{i,j}(\boldsymbol{x}) = \ell_r$ and $\mathcal{M}_{k,l}(\boldsymbol{x}) = \ell_s$, then model $\mathcal{M}_{r,s}$ will be used in the next layer. Notice that there are four possible models for $\mathcal{M}_{r,s} : \mathcal{M}_{i,k}, \mathcal{M}_{i,l}, \mathcal{M}_{j,k}$ and $\mathcal{M}_{j,l}$. The model used at node C in Fig. 1 (right) will be $\mathcal{M}_{1,2}, \mathcal{M}_{1,3}, \mathcal{M}_{2,4}$ or $\mathcal{M}_{3,4}$, depending on the instance being classified.

Once a hierarchical structure is selected, several ways have been proposed for deciding which models are placed at each node. For instance, in [37], the authors defend an arbitrary criterion for a DDAG. An efficient data structure is used in [6] to locate nodes in the graph and an improved decision algorithm is used to perform test predictions. Also, in [42], the authors optimize the structure of a DDAG by placing at the upper nodes the models that have a lower proportion of support vectors. They show that this heuristic performs in a similar way to the arbitrary criterion adopted in [37]. Similarly, Feng et al. [17] employed the Jaakkola–Haussler error bound [4]. With regard to an ADAG, a structure is selected based on the hard margin error in [41].

In [37], the authors hold that the way in which models are attached to the nodes in a DDAG does not affect the accuracy performance of the algorithm. However, they admit that the experimentation they employed was limited: only three different data sets. Furthermore, the experimental results reported in [42] do not show significant differences between their method and a DDAG. However, once again, the experiments were limited to only five data sets and no parameter tuning was carried out. This paper extends the ideas presented in [41,42], arguing that the pairs of classes that are easier to separate must be placed in the upper nodes of the hierarchy. However, we employ a simpler and more accurate method performing a more exhaustive experimentation and finding that there are in fact significant differences.

3. Enhanced directed binary trees

This section describes the main contributions of the paper to improve the performance of multi-class classification decomposition methods based on the use of tree prediction schemes. Firstly, we propose Directed Binary Trees (DBTs) as the best tree structure to arrange the pairwise classifiers obtained when a one-vs-one decomposition strategy is applied. This structure has been used before for multi-class classification [16,32], but not for placing two-class classifiers, otherwise for placing two-group-classes ones. Also, DBTs have been considered for placing pairwise classifiers [5], but again serving as separators of other classes. As we shall prove, DBTs are more general than other tree structures and present a number of advantages that will be exposed in Section 3.2. Secondly, we study different criteria to select the binary model that is attached to each node of the tree. As experimental results will show, the arrangement of pairwise classifiers in the tree has a major influence on the performance of the final multi-class classifier obtained, both in terms of error rate and prediction speed.

3.1. Directed binary trees

In a DBT, each internal node has one input edge and two output edges. Fig. 1 (center) depicts an example of a DBT. The root, whose model is $\mathcal{M}_{i,j}$, has two children. In the left one, a model $\mathcal{M}_{k,l}$ is placed and selected among those not involving class ℓ_j. Similarly, a model $\mathcal{M}_{p,q}$ verifying $p, q \neq i$ is chosen for the right node. These constraints are applicable for all of their respective descendants; for instance, no descendants of model $\mathcal{M}_{k,l}$ can include class ℓ_j. The leaf of a branch is labeled with the only class that was not previously removed by this sequence of consecutive decisions. The prediction procedure of a DBT in this context works like that of a DDAG, one class is discarded at each level and the label of the reached leaf is returned.

One of the differences with respect to a DDAG is that, when $m > 3$, some binary models are attached to more than one node, since the number of decision nodes ($2^{m-1} - 1$) is greater than the number of pairwise classifiers ($m(m-1)/2$). For instance, in Fig. 1, model $\mathcal{M}_{2,3}$ is applied at the two nodes of the second level. Obviously, this does not increase the memory requirements of the final classifier, because it is enough to store each binary model once. Notice also that the label of each class may appear more than once in the leaves of a DBT.

3.2. Properties of DDAGs, ADAGs and DBTs

DBTs are more general than both DDAGs and ADAGs in the sense that it is not possible to find an equivalent DDAG or ADAG for some DBTs, but the opposite is always possible, although it may not always be evident: every DDAG or ADAG has an equivalent DBT. In the case of a DDAG being clearer, it is only necessary to duplicate those nodes pointed to by two ancestor nodes. The equivalent DBT of an ADAG is one in which the first $m/2$ layers house the classifiers of the first layer of the ADAG in such a way that all the paths taken pass through the same classifiers. That is, all the nodes of each of these levels use the same model (see the first two layers of the DBT example in Fig. 1). Hence, there is a branch for each possibility of the second level of the ADAG, and so on. An example of this equivalence is shown in Fig. 1. Both the DBT and ADAG depicted in this figure represent, in fact, the same classifier. Notice that the two first layers of the DBT can be exchanged and the resulting classifier will produce the same predictions.

From the point of view of the prediction procedure, all of these graphs are equivalent in the case of using pairwise models. All require the evaluation of $m - 1$ binary classifiers, since only one class is discarded at each decision node. To correctly classify an instance $\boldsymbol{x}$, the models that involve its true class, called *competent* models, must provide correct predictions. The predictions of the *non-competent* models do not impede the true class from being returned: the discarded class will never be the true one if the model is non-competent. As can be easily proved, any of these graphs misclassify an example $\boldsymbol{x}$ if, and only if, one competent model in the prediction path followed by $\boldsymbol{x}$ fails. This situation allows us to state that the classification performance depends on how the binary classifiers are arranged inside the graph. The fact that a class is discarded at each stage and that a different classifier can carry out this task means that different structures obtain different decision regions.

As originally designed [37], a DDAG can be constructed by ordering (randomly, for instance) the set of classes. This order determines the location of each binary classifier and the order of the classes at the leaves of the tree. Although this structure has provided good results for some multi-class problems, it has certain disadvantages [27]. The main drawback of the DDAG structure is that it imposes severe constraints, especially for the two classes at the extremes of the class order (classes 1 and 4 in the example in Fig. 1). To correctly classify an instance $\boldsymbol{x}$ of one of these classes, *all* its $m - 1$ competent models must provide correct predictions. This condition is more stringent than that required by, for instance, the max-wins strategy, which only requires a majority of its competent models to provide correct predictions. When the number of classes is large, correctly classifying these examples becomes even more difficult. Notice that whenever a competent model of $\boldsymbol{x}$ is reached in a DDAG during its classification process, the path from that node to the leaf labeled with the true class of $\boldsymbol{x}$ only includes its competent models. All of them must make a correct decision in order to correctly predict the class of $\boldsymbol{x}$.

Unlike DDAGs, ADAGs present a very interesting property as regards the arrangement of the competent models for any given instance. In each level of an ADAG, each class has only one competent model. Thus, since the depth of an ADAG is $\lceil \log_2 m \rceil$, $\lceil \log_2 m \rceil$ is also the upper bound of the number of competent models that must be correct in order to correctly classify an instance. In the case of a DDAG, this bound varies between $\lceil \log_2 m \rceil$ (for the class or pair of classes just in the middle of the class order) to $m - 1$ (for the classes at the extremes). Comparing all these bounds intuitively suggests that an ADAG structure reduces the risk of misclassifying examples, especially in multi-class problems in which the number of classes is large.

On the other hand, however, an ADAG is quite imprecise, in the sense that only the binary models at the first layer are known beforehand; the classifiers used in the following levels depend on the results of the previous layers. Actually, an ADAG works just like a tennis tournament, in which only first-round games are known when the draw is made. The games of the next rounds depend on the results of the previous rounds. This is an undesirable property if we want to select the precise binary classifier to be used at each stage of the classification process.

DBTs constitute an alternative to DDAGs and ADAGs, the advantage of which derives from the fact that they do not impose any constraints, in addition to generalizing the other two structures. DBTs are less restrictive than DDAGs, more precise than ADAGs and can be designed to have the same competent models bound that characterizes ADAGs. Interestingly, any layout of the binary classifiers is possible using DBTs. Thus, in what follows, we shall focus on DBTs. The main aim of this paper is to analyze different methods to select the adequate layout in a DBT of the binary classifiers yielded by the one-vs-one decomposition method.

3.3. Main idea

As stated previously, these hierarchical structures discard a class each time a model is evaluated and different examples may take different paths until reaching a leaf. From the point of view of pairwise classifiers, this means that the probability of a model being used to classify an example increases with its proximity to the root. Considering this behavior, it appears intuitively reasonable to assume that the classification ability of a DBT depends on the arrangement of the classifiers. For instance, it may be quite dangerous to locate model $\mathcal{M}_{i,j}$ at the root if classes ℓ_i and ℓ_j are hard to separate, because the model will evaluate all examples of these classes. It is therefore preferable to place models of this kind at lower levels, as they will classify less examples. Consequently, those classes that are easier to separate must tend to be located nearer the root node.

The purpose of this paper is thus to (i) define a metric to estimate the ease of separating a pair of classes and (ii) apply a greedy algorithm based on such a metric to select the appropriate pairwise classifier at each node of a DBT. The metric can be seen as a way of measuring the *distance* between two classes: the greater the distance, the easier it will be to distinguish these classes. In what follows, we shall discuss several alternatives to define this kind of metric.

3.4. Metrics to measure the distance between two classes

A reasonable first criterion may be to consider the classification ability of the binary classifier. If we restrict ourselves to SVMs, the ability to generalize can be explained by their capacity control. The VC dimension of a hyperplane can be bounded by the margin and the diameter of the smallest sphere containing the training set [44], which is the theoretical idea that motivates the maximization of the margin. Thus, the generalization error bounds may be the first choice. In [9], the authors review the main results of VC theory, which places reliable bounds on the generalization of SVMs. In order to apply this criterion in a practical situation, our first proposal is to adopt a soft margin bound. For instance, it can be proved that given a binary classification data set with n examples drawn from a probability distribution with support in a ball of radius R around the origin, there is a constant c such that any hypothesis yielded by a binary soft-margin SVM has an error no greater than

$$\frac{c}{n}\left(\frac{R^2 + \|\xi\|_1^2 \log(1/\gamma)}{\gamma^2}\log^2 n + \log\frac{1}{\delta}\right), \tag{1}$$

with probability $1-\delta, \gamma$ being the margin of the model and ξ the margin slack vector. The main problem with bounds of this kind is that they are sometimes quite pessimistic.

A second possible alternative is to apply an empirical method for estimating the generalization error of every binary classifier; less error implies a greater distance. The leave-one-out procedure gives an almost unbiased estimation of the expected error. This procedure consists in removing one instance from the training data, constructing the model on the basis of the remaining training data and then testing on the removed element. The process is repeated for each training example and, finally, the average of the errors of all the models is computed. The problem is that its computational cost is very high, being in fact impracticable if we take into account the number of models that arise in the one-vs-one decomposition approach. However, a leave-one-out upper bound can be computed using the findings of [4],

$$\sum_{i=1}^{n}\Psi(U_i - 1), \tag{2}$$

where, once again, n is the number of training examples, Ψ is the classical step function and U_i is an upper bound of the difference of prediction for the ith example considering and not considering said example in the training process. Since a bound of the step function is 1 and $U_i = 0$ for the support vectors, then the bound consisting in counting the support vectors with regard to the number of original examples gives an idea of the performance of the classifier [44],

$$\frac{\#sv}{n}, \tag{3}$$

where $\#sv$ is the number of support vectors. This bound was previously used in [42] for optimizing DDAG structures.

Another possibility is to apply the different bounds discussed in [4]. The Jaakkola–Haussler bound, used in [17], only works when the SVM is trained without a threshold. The Opper–Winther bound is suitable for hard margin SVMs without a threshold. The Radius-margin bound works well for SVMs without a threshold and with no training errors. Thus, the more interesting one is the span bound that includes the concept of span of support vectors and works under the assumption that the set of support vectors remains the same during the leave-one-out procedure. However, this bound is quite hard to compute and so will not be considered in the experimentation process.

This paper proposes a new metric for measuring the distance between a pair of classes. In [43], the authors present a multi-class classifier for high-dimensional input spaces that employs a hierarchy of classes to build a decision tree. This hierarchy is obtained employing the complete linkage method, in which the distance between two classes is defined as the margin between them, i.e. $\frac{2}{\|\boldsymbol{w}\|}$, in which $\boldsymbol{w}$ determines the SVM hyperplane for that pair of classes. This distance was adopted to configure an ADAG in [41]. However, this measure is only meaningful when the classes are known to be separable, which is not always the case in practical situations. In [12], the authors present a generalization of this distance to the non-separable case. They propose the following expression to compute the distance between two classes:

$$\frac{1}{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i}, \tag{4}$$

where C is the classical regularization parameter of a soft-margin binary SVM formulation and ξ_i is the slack variable of the ith example. The idea is that when classes are similar, the model will be complex ($\|\boldsymbol{w}\|$ will be high) and/or there will be a lot of misclassified examples or points inside the margin ($\sum\xi_i \gg 0$); in which case, the distance will be small. When classes are very different, the classifier will be a simple model ($\|\boldsymbol{w}\|$ will be low) and/or the number of misclassified examples or points inside the margin will be small ($\sum\xi_i \to 0$); hence, the distance will be high.

It is worth noting that the metrics discussed in this section are classifier-dependent. Specifically, the proposed metric considers two factors: the precision and the complexity of the pairwise models learned with a binary SVM. This last aspect is, indeed, classifier-dependent, because the complexity directly depends on the hypothesis space that the classifier explores. For instance, measuring the complexity of a model based on decision trees or linear classifiers is totally different. It is not trivial to define a classifier-independent metric that considers these two factors. However, this idea can be adapted to be used with other base classifiers whenever the complexity of the induced pairwise models can be quantified.

3.5. Algorithms to select the binary model at each node of a DBT

In this paper we consider two different greedy algorithms to locate the pairwise classifiers in a DBT structure, given any of the metrics described previously. The first algorithm is based on the most intuitive rule: at a given node, the binary model selected will be the one involving the two classes—among the remaining possible ones—that are the easiest to distinguish (with the most distance between them). For instance, the algorithm will place model $\mathcal{M}_{i,j}$ at the root if the distance between classes ℓ_i and ℓ_j is the biggest one. Then, the selected model for the left branch node will be $\mathcal{M}_{k,l}$, verifying that $k, l \neq j$ and the distance between classes ℓ_k and ℓ_l is the biggest one among all the pairs of classes, excluding class ℓ_j. The procedure for the right branch node will be the same, though excluding class ℓ_i. The same rule is applied for the rest of the nodes, always discarding those classes removed by the previous decision nodes of that path.

The problem of this first algorithm is that a class can have more than $\lceil \log_2 m \rceil$ competent models in some branches. As the number of models in a branch is always $m-1$, whenever a class has more than $\lceil \log_2 m \rceil$ competent models in a branch, other classes have less competent models. If one of these latter classes is finally predicted, the confidence of that prediction may be small. This happens, for instance, in the DDAG in Fig. 1 when the predicted class is ℓ_2 following the left-most branch: the class ℓ_2 only has one competent model in that branch, $\mathcal{M}_{1,2}$.

It would be desirable for our DBTs to preserve the competent-models bound of ADAGs. This can be achieved by adding another constraint to the rule of the first algorithm: the selected binary model will be the one involving the two classes with the most distance between them, among those classes with less appearances in the models of that path. This rule ensures that, at any branch of our DBT, a class has no more than $\lceil \log_2 m \rceil$ competent models, and the class at the leaf appears at least $\lceil \log_2 m \rceil - 1$ times in the models of that branch. In other words, the same desirable property of ADAGs. This produces a balanced situation for every class, boosting the confidence of the resulting prediction.

Other more complex algorithms could be designed for this task. However, the problem of arranging the pairwise classifiers in a DBT to obtain the tree *with maximum distance* (given a metric, computed as the sum of distances between the classes of the models of all paths) is NP-hard [33]. Thus, the greedy algorithms described here are simple and provide a good trade-off between computational time and empirical performance, as will be shown in Section 4.

3.6. Generalization error bounds

From the point of view of learning theory, the advantage of decomposition approaches based on the use of tree structures is that generalization error bounds can be obtained. This is not the case with other popular decomposition methods based on different combination strategies, like max-wins or winner-takes-all. Platt et al. [37] present a VC analysis of DDAGs when the node classifiers are hyperplanes, like the models obtained if SVM is used as the base learner. The main contribution is that the resulting bound on the test error depends on the number of classes and on the margin achieved by the hyperplane classifiers, but not on the dimension of the input space.

In fact, the probability of misclassifying an example of class ℓ_j, denoted by ϵ_{ℓ_j}, can be bounded according to the following theorem taken from [37]:

Theorem 1. *Given a DDAG which contains hyperplane classifiers with margins γ_i at decision nodes and is able to correctly classify the instances of class ℓ_j in a random data set with n examples that belong to m classes, then with probability $1-\delta$,*

$$\epsilon_{\ell_j} \leqslant \frac{130R^2}{n}\left(D' \log(4en)\log(4n) + \log\frac{2(2n)^{m-1}}{\delta}\right), \tag{5}$$

where

$$D' = \sum_{i \in CM_j} \frac{1}{\gamma_i^2}, \tag{6}$$

CM_j being the set of $m-1$ competent models for class ℓ_j and R the radius of a ball containing the distribution's support.

This same theorem can also be applied to ADAGs and DBTs. It implies that we can control the capacity of these multi-class classifiers by enlarging the margins of the binary models placed at their decision nodes. Specifically, when the competent models of class ℓ_j have larger margins, the resulting classifier will identify examples of that class with more accuracy. In other words, the bound only depends on the margins of binary classifiers.

A similar bound, considering binary classifiers as black-boxes, was also obtained [13] up to a small constant factor as a result of taking different generalization error bounds for a DDAG. Notice that in this last case an erratum was announced [14] as a consequence of an incorrect application of the union bound, which led to small modifications of the bounds. In addition, a generalization error was obtained for an ADAG [13], also slightly affected by the erratum [14]. Once again, the bound depends on the error of competent models.

However, these bounds may be quite pessimistic: precisely by correctly applying the union bound, the obtained bounds will depend on the margins of *all* the competent models (factor D'; see Eq. (6)). Intuitively, as was pointed out in [37], the only margins that should matter for a given example are those of the classifiers placed in the path of the tree followed by the classification process of that example.

In fact, the ideas proposed in this paper head in that direction. On the one hand, the metric of Eq. (4) allows us to locate the classifiers with a larger margin at the nodes near the root of the tree, which are the nodes shared by more branches. On the other hand, applying the second algorithm from Section 3.5 reduces the number of competent models of any class in any branch from an order of m in a DDAG to, at most, $\log_2 m$. Both proposals taken together hence help decrease the value of Eq. (6), restricted just to the competent models of a particular branch.

4. Experimental results

This section reports the results of the experiments carried out to evaluate the proposed ideas for constructing a DBT multi-class classifier based on pairwise models. The experiments were designed so as to perform an in-depth comparative study of all the different tree structures described throughout the paper. The main hypothesis we aim to prove is that the arrangement of binary models in a tree structure affects both the classification performance and the evaluation time of the resulting classifier. Section 4.1 describes the settings used in the experiments: data sets, learning methods, base learner and the procedures to set parameters. Sections 4.2 and 4.3 respectively discuss the classification performance and the evaluation time of the compared methods on benchmark data sets. Finally, in Section 4.4 a DBT multi-class classifier is used in a real application.

4.1. Experimental settings

The data sets used in the experiments are described in Table 1, most of which were taken from the UCI repository [2]. We selected those data sets having continuous or ordinal attribute values and no more than 11,000 examples, and we excluded those with missing values and with less than 4 classes. We wished to include the three data sets used in [37]. However, finally, only the USPS digit data set was used, since no results were obtained in a reasonable time under our experimental framework (see below) for the other two (Letter recognition and Covertype) due to the high number of examples. We also employed a group of five meteorological weather station data sets[1] with a high number of classes. One of these was finally discarded because, once again, no results were obtained in a reasonable time.

We compared the original DDAGs proposed in [37] and the ADAGs [27] (see Section 2), with several implementations of DBTs (see Section 3). Each different version of a DBT comes from combining one of the two algorithms described in Section 3.5 with one of the metrics from Section 3.4. Recall that both algorithms differ in the rule used for selecting the binary model placed at each node. They will be labeled as:

- DBT1$_x$: the binary model selected is the one between the two classes with the most distance between them.
- DBT2$_x$: the binary model selected involves the two classes with the most distance between them, among those classes with less appearances in the path of that node.

The subindex x will describe the metric used. In the results reported here, we have included only two of the metrics discussed (Section 3.4): Eq. (3) (labeled as *sv*), which is based on the proportion of support vectors, and Eq. (4) (labeled as *d*), which measures the distance between two classes according to the value of the optimization function of their pairwise

[1] http://www.uni-marburg.de/fb12/kebi/research/repository/metstatdata.zip.

Table 1
Statistics for the multi-class data sets used in the experiments, ordered by the number of classes.

Data set	# Classes	# Att.	# Ex.
Soybeansmall	4	35	47
Vehicle	4	18	846
Car evaluation	4	6	1728
Pageblocks	5	10	5473
Glass	6	9	214
Dermatology	6	33	366
Landsat	6	36	6435
Zoo	7	16	101
Image seg.	7	19	2310
Ecoli	8	7	336
Wine	10	13	178
Led1000	10	7	1000
Yeast	10	8	1484
Semeion	10	256	1593
Optdigits	10	64	5620
Usps	10	257	9298
Pendigits	10	16	10,992
Vowel	11	11	990
MetStatLocRST	12	3	336
MetStatLocSunshine	14	12	422
MetStatTemperature	15	12	673
Librasmovement	15	91	360
MetStatLocRainfall	16	12	4748

model. We do not report the results using Eq. (1) because they were significantly worse than those obtained with the other two metrics. Thus, four different DBT versions are compared: $DBT1_{sv}, DBT1_{d}, DBT2_{sv}$ and $DBT2_{d}$.

In order to obtain a meaningful comparison, we added SVM_{ovo}[20], the one-vs-one decomposition approach using the max-wins strategy for combining the outputs of binary classifiers. This selection is justified because SVM_{ovo} also learns the $m(m-1)/2$ pairwise models that are needed to construct DDAGs, ADAGs and DBTs. The base learner to build these pairwise classifiers was SVM; the implementation used was *libsvm* [3] with the linear kernel.

The scores reported are the error rate or 0/1 loss estimated by means of a 5-fold cross-validation repeated 2 times. We did not use the 10-fold procedure, since certain data sets contain too few examples for some classes. The regularization parameter C was established by performing a grid-search over the interval $C \in [10^{-2}, \ldots, 10^{2}]$, optimizing the aforementioned loss estimated through a 2-fold cross validation repeated five times. We proceeded performing two kinds of experiments with regard to parameter C. In the former, the goal is to individually adjust the value of parameter C for each pairwise model yielded by one-vs-one decomposition method, allowing that each of them may take a different value. Once all pairwise classifiers are trained, then the combination stage of each method is performed. In the latter all the pairwise classifiers of each method use the same value of C, selecting the value that minimizes the error rate of the whole multi-class classifier, which includes the combination strategy.

Following [10], a two-step statistical test procedure was carried out. The first step consists of a Friedman test of the null hypothesis that states that all approaches perform equally. Then, in the case of this hypothesis being rejected, a Nemenyi test is performed to compare the methods in a pairwise way. Both tests are based on the ranks average. The comparison includes seven algorithms over 23 data sets, so the critical differences (CDs) in the Nemenyi test are 1.8786 and 1.7155 for significance levels of 5% and 10%, respectively.

4.2. Classification performance

The first experiment consists in learning the $m(m-1)/2$ pairwise models yielded by the one-vs-one decomposition approach, adjusting the value of parameter C to minimize the error rate of each pairwise classifier. Notice that different values of C can be used for different models. These models are then used to build all the compared approaches. This guarantees that all the methods are using exactly the same pairwise classifiers and only differ in the combination strategy. The scores are summarized in Table 2.

As can be seen, the DDAG algorithm is in general not competitive. In terms of the average ranks, all the other algorithms outperform DDAG. This is even clearer in data sets with a larger number of classes (notice that the data sets are ordered by this value). When the number of classes is equal to or greater than 10, DDAG is the worst method seven times out of 12,[2] and in four other cases it is the second worst algorithm. Notice that in such situations, DBT approaches are only the worst method once ($DBT1_{sv}$), while ADAG and SVM_{ovo} are the worst methods two times apiece. Obviously, this suggests that it is in these situations where a more elaborated arrangement of the pairwise models helps to reduce the error rate. When the number of classes is smaller, the differences between tree-based approaches are generally less important.

[2] We do not include the results of the wine data set in this statistic because all the approaches perform equally.

Table 2
Error percentage of compared methods, all of which use the same pairwise classifiers (parameter *C* was established to minimize the error rate of each individual pairwise classifier). The number in parentheses behind the error rate is the rank (in decreasing order) of the method on the corresponding data set. The average rank is the average of the ranks across all data sets.

Data set	SVM_{ovo}		DDAG		$DBT1_{sv}$		$DBT1_d$		ADAG		$DBT2_{sv}$		$DBT2_d$	
	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank
Soybeansmall	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)
Vehicle	19.98	(7)	19.80	(5.5)	19.50	(3)	19.50	(2)	19.45	(1)	19.80	(5.5)	19.74	(4)
Car	14.29	(2)	14.93	(6)	15.39	(7)	14.15	(1)	14.35	(3.5)	14.47	(5)	14.35	(3.5)
Pageblocks	3.53	(1)	3.54	(2)	3.65	(6)	3.57	(4)	3.70	(7)	3.55	(3)	3.59	(5)
Glass	33.43	(4.5)	33.89	(6.5)	33.43	(4.5)	33.20	(2)	33.89	(6.5)	33.19	(1)	33.20	(3)
Dermatology	3.55	(4)	3.55	(4)	3.55	(4)	3.55	(4)	3.55	(4)	3.55	(4)	3.55	(4)
Landsat	13.05	(4)	13.01	(2)	13.12	(6.5)	13.12	(6.5)	12.98	(1)	13.05	(4)	13.05	(4)
Zoo	4.98	(5)	4.98	(5)	4.98	(5)	3.98	(1.5)	4.98	(5)	3.98	(1.5)	4.98	(5)
Image	4.00	(1)	4.09	(4)	4.11	(5.5)	4.11	(5.5)	4.05	(2.5)	4.05	(2.5)	4.13	(7)
Ecoli	11.75	(2)	11.90	(5.5)	11.90	(5.5)	11.75	(2)	11.90	(5.5)	11.75	(2)	11.90	(5.5)
Wine	3.67	(4)	3.67	(4)	3.67	(4)	3.67	(4)	3.67	(4)	3.67	(4)	3.67	(4)
Led1000	27.90	(2)	27.85	(1)	28.85	(7)	28.55	(5)	28.45	(4)	28.80	(6)	28.15	(3)
Yeast	41.54	(7)	41.47	(6)	41.41	(4)	41.44	(5)	41.24	(1.5)	41.31	(3)	41.24	(1.5)
Semeion	6.56	(2)	6.81	(6)	6.53	(1)	6.69	(4)	6.88	(7)	6.69	(3)	6.75	(5)
Optdigits	1.86	(1)	2.04	(7)	1.96	(4)	1.98	(5)	1.90	(2)	2.00	(6)	1.92	(3)
Usps	4.15	(4)	4.23	(7)	4.15	(3)	4.11	(1)	4.16	(5)	4.16	(6)	4.12	(2)
Pendigits	1.76	(1)	1.99	(7)	1.84	(5)	1.84	(4)	1.87	(6)	1.83	(3)	1.80	(2)
Vowel	19.14	(1)	20.46	(7)	19.85	(5)	19.75	(4)	19.55	(2.5)	19.95	(6)	19.55	(2.5)
mSLocRST	61.17	(1)	62.06	(7)	61.61	(4)	61.17	(2)	61.61	(3)	61.91	(5.5)	61.91	(5.5)
mSLocSun.	29.98	(7)	29.98	(6)	29.86	(5)	29.74	(4)	29.27	(1)	29.74	(3)	29.51	(2)
mSTemp.	23.62	(3)	24.29	(6)	24.07	(5)	23.55	(2)	24.37	(7)	23.70	(4)	23.33	(1)
Librasmove.	19.17	(5.5)	20.56	(7)	18.47	(3)	19.03	(4)	19.17	(5.5)	17.78	(1)	18.33	(2)
mSLocRain.	22.88	(1.5)	23.08	(7)	22.91	(3)	23.03	(5)	23.05	(6)	23.03	(4)	22.88	(1.5)
Avg. rank		(3.24)		(5.33)		(4.52)		(3.54)		(4.11)		(3.78)		(3.48)

On the other hand, $DBT2_d$ and $DBT1_{sv}$ are the best and the worst tree-based methods, respectively. It should be noted that the DBTs using the metric derived from Eq. (4), $DBT1_d$ and $DBT2_d$, obtain better results than their counterpart approaches, respectively $DBT1_{sv}$ and $DBT2_{sv}$, based on Eq. (3). Similarly, when the DBTs are constructed with the second algorithm from Section 3.5, $DBT2_d$ and $DBT2_{sv}$, the results are better than when the first algorithm is used, $DBT1_d$ and $DBT1_{sv}$. All these facts support the finding that the best performance is obtained using the metric of Eq. (4) with the second greedy algorithm proposed.

Although all the approaches in this experiment use the same pairwise models, sometimes the differences are quite big. This happens again more clearly when the number of classes is larger. For instance, in the case of the librasmovement data set, the difference can be of up to two points, in a problem with an error rate of around 20%. Of course, the differences are much smaller in other cases. In fact, all the methods perform equally in three data sets (soybeansmall, dermatology and wine), which means that the combination strategy sometimes does not make any difference and the errors are only due to the pairwise classifiers.

Analyzing the results shown in Fig. 2 from a statistical perspective, the Friedman test suggests that there are significant differences between the methods and the Nemenyi test indicates that DDAG is significantly worse than SVM_{ovo}, (at a confidence level of 95%) and than $DBT2_d$ and $DBT1_d$(90%). It should be stressed here that there are no significant differences between SVM_{ovo} and all the DBT approaches, a fact that also occurs with ADAG.

It is well known that *C* is a crucial parameter when learning SVM classifiers. Therefore, in order to better analyze the studied combination strategies, in the second experiment each multi-class classifier adjusts the value of *C* as a whole. In this case, the selection is made considering the accuracy of the whole multi-class classifier, including the combination scheme. Notice that now, pairwise models for a multi-class classifier will have the same value of *C*, but it can be different for the pairwise models of other multi-class classifier method. Table 3 shows these scores.

The first conclusion is that the differences between the approaches are now greater. Only in the soybeansmall data set do all the methods obtain the same result. The error reduction between the worst and the best algorithm is 8.3% on average, while in the previous experiment it was just 4.7%. In the first experiment, this kind of reduction was greater than 10% in only three data sets; whereas in the second experiment this occurs eight times. For instance, this reduction is 19.98% in the librasmovement data set, from 22.22 for DDAG to 17.78 for $DBT2_{sv}$. This means that the differences between these methods can sometimes be greater in practice that one might expect.

In the second experiment, the best algorithm is not SVM_{ovo}, but $DBT2_d$. Moreover, two other DBT versions, $DBT1_d$ and $DBT2_{sv}$ perform almost the same as SVM_{ovo}. This suggests that DBT approaches benefit more than SVM_{ovo} from adjusting the value of *C*, although it should be noted that the differences between them are quite small. However, these differences increase if we consider only data sets with ten or more classes. The average rank in these problems is 3.69 for SVM_{ovo} and 2.77 for $DBT2_d$. If one goes up, then the other goes down, but still the difference is not statistically significant.

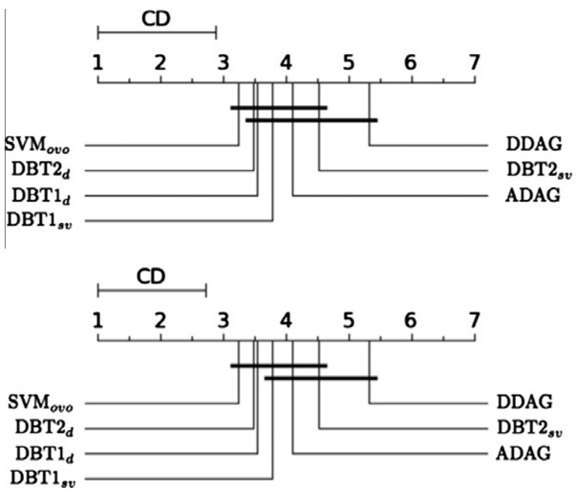


Fig. 2. Comparison of the error rate in the first experiment of all the methods using the Nemenyi test. All the approaches use the same pairwise models (the value of parameter C is adjusted for each pairwise model isolated from the rest). The scale from 1 to 7 shows the average rank of each method. Those classifiers that are not significantly different at $p = 0.05$ (above, $CD = 1.8786$) and at $p = 0.10$ (below, $CD = 1.7155$) are linked by a bold line.

Table 3
Error percentage of the compared methods using different pairwise classifiers (parameter C was established to minimize the error rate of each method). The number in parentheses behind the error rate is the rank (in decreasing order) of the method on the corresponding data set. The average rank is the average of the ranks across all data sets.

Data set	SVM$_{ovo}$		DDAG		DBT1$_{sv}$		DBT1$_d$		ADAG		DBT2$_{sv}$		DBT2$_d$	
	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank	0/1	Rank
Soybeansmall	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)	0.00	(4)
Vehicle	19.44	(1)	19.56	(2)	20.16	(6)	19.68	(3)	20.04	(5)	20.27	(7)	19.98	(4)
Car	14.44	(4)	14.76	(6)	15.39	(7)	14.24	(1)	14.32	(2)	14.64	(5)	14.38	(3)
Pageblocks	3.27	(1)	3.64	(6)	3.76	(7)	3.32	(2)	3.49	(4)	3.57	(5)	3.46	(3)
Glass	36.91	(6)	37.14	(7)	33.20	(2.5)	35.51	(5)	33.43	(4)	33.20	(2.5)	32.74	(1)
Dermatology	3.56	(6)	3.28	(1)	3.55	(3.5)	3.69	(7)	3.55	(3.5)	3.55	(3.5)	3.55	(3.5)
Landsat	13.12	(4.5)	13.17	(7)	13.12	(4.5)	13.14	(6)	13.09	(3)	13.05	(1)	13.09	(2)
Zoo	4.95	(4)	4.48	(3)	4.98	(6)	3.98	(1.5)	4.98	(6)	4.98	(6)	3.98	(1.5)
Image	3.70	(1)	3.79	(2.5)	4.16	(6.5)	3.79	(2.5)	4.13	(5)	4.09	(4)	4.16	(6.5)
Ecoli	10.99	(1.5)	11.14	(3)	11.90	(6.5)	10.99	(1.5)	11.75	(4.5)	11.75	(4.5)	11.90	(6.5)
Wine	3.67	(4)	3.94	(5)	3.38	(1.5)	3.66	(3)	3.952	(6.5)	3.38	(1.5)	3.95	(6.5)
Led1000	28.10	(2)	28.50	(5)	28.60	(6)	28.40	(4)	28.00	(1)	28.80	(7)	28.20	(3)
Yeast	41.95	(6)	42.12	(7)	41.58	(4)	41.81	(5)	41.54	(3)	41.48	(2)	41.37	(1)
Semeion	6.81	(5)	7.19	(7)	6.53	(1)	6.84	(6)	6.75	(4)	6.69	(3)	6.62	(2)
Optdigits	1.91	(2)	2.03	(7)	1.96	(4)	1.89	(1)	2.00	(5.5)	2.00	(5.5)	1.92	(3)
Usps	4.34	(6)	4.36	(7)	4.15	(2)	4.33	(5)	4.17	(4)	4.16	(3)	4.12	(1)
Pendigits	1.87	(4)	2.12	(7)	1.84	(3)	1.97	(6)	1.89	(5)	1.83	(2)	1.79	(1)
Vowel	19.09	(1)	20.71	(7)	20.00	(5)	19.50	(2)	20.01	(6)	19.95	(4)	19.55	(3)
mSLocRST	61.17	(1)	61.46	(3)	61.62	(5)	61.17	(2)	62.06	(7)	61.91	(6)	61.61	(4)
mSLocSun.	29.98	(7)	29.86	(6)	29.86	(5)	29.74	(4)	29.15	(1)	29.74	(3)	29.51	(2)
mSTemp.	23.02	(3)	23.69	(7)	23.62	(6)	22.88	(2)	23.02	(4.5)	22.73	(1)	23.02	(4.5)
Librasmove.	21.11	(6)	22.22	(7)	18.47	(3)	18.89	(4)	19.58	(5)	17.78	(1)	18.33	(2)
mSLocRain.	22.81	(1)	23.36	(7)	22.93	(4)	23.02	(6)	22.97	(5)	22.87	(2)	22.90	(3)
Avg. rank		(3.52)		(5.37)		(4.48)		(3.63)		(4.28)		(3.63)		(3.09)

As regards DBT methods, the second greedy algorithm and the metric of Eq. (4) once more seem to be the best options. In this experiment, DBT1$_d$ and DBT2$_{sv}$ perform almost equally. On the other hand, DDAG once again continues to be the worst method, more clearly so in data sets with a larger number of classes. Even when the method can search for a proper value for the learning parameters, the results are still worse. This clearly implies that the shortcomings of this method are due to the imposed constraints of the tree structure.

From a statistical point of view (see Fig. 3) the conclusions of this experiment are similar to those of the first experiment: there are no differences between the DBT approaches, SVM$_{ovo}$ and ADAG, while the Nemenyi test indicates that DDAG is significantly worse than DBT2$_d$ (at a confidence level of 95%) and SVM$_{ovo}$, DBT1$_d$ and DBT2$_{sv}$ (90%).

The main conclusions to be drawn from both experiments are that DBT algorithms improve the classification performance of decomposition approaches based on trees and are very competitive with respect to the one-vs-one approach. This study

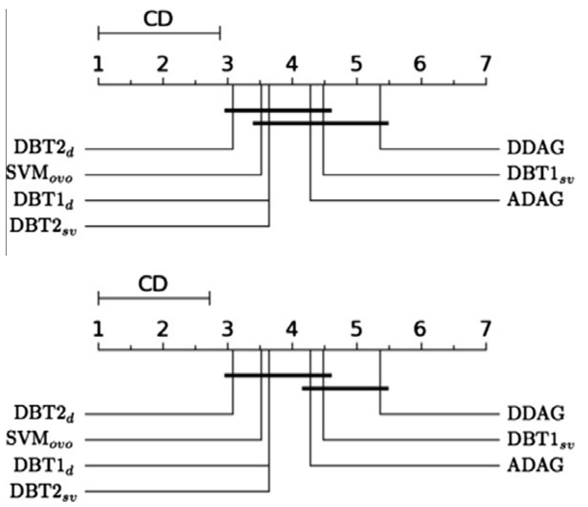


Fig. 3. Comparison of the error rate in the second experiment of all the methods using the Nemenyi test. The compared approaches may use different pairwise models (the value of parameter C is adjusted for the whole multi-class classifier). The scale from 1 to 7 shows the average rank of each method. Those classifiers that are not significantly different at $p = 0.05$ (above, $CD = 1.8786$) and at $p = 0.10$ (below, $CD = 1.7155$) are linked by a bold line.

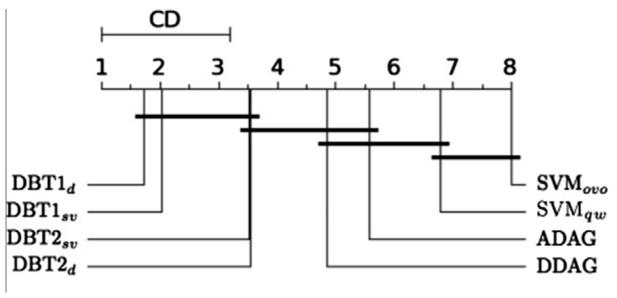
Table 4
Number of kernel evaluations in the testing stage for all the methods. The number in parentheses is the rank (see explanation in Table 2).

Data set	SVM$_{ovo}$		SVM$_{qw}$		DDAG		DBT1$_{sv}$		DBT1$_d$		ADAG		DBT2$_{sv}$		DBT2$_d$	
	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank	# Eval.	Rank
Soybeansmall	343.6	(8)	222.9	(7)	186.4	(6)	171.2	(1)	175.2	(3)	184.4	(5)	177.6	(4)	172.9	(2)
Vehicle	61487.0	(8)	34174.1	(5)	30683.8	(4)	29417.8	(1)	29441.2	(2)	29975.3	(3)	38657.8	(7)	36746.7	(6)
Car	212510.2	(8)	191567.0	(7)	151484.8	(1)	163720.6	(4)	152839.7	(2)	174270.3	(5.5)	153297.2	(3)	174270.3	(5.5)
Pageblocks	366253.6	(8)	281890.6	(7)	156013.2	(2)	279635.4	(6)	142297.1	(1)	176083.0	(4)	194908.3	(5)	175029.6	(3)
Glass	9215.8	(8)	5767.5	(7)	4261.5	(6)	3896.4	(2)	3849.0	(1)	4234.9	(5)	3968.5	(4)	3959.6	(3)
Dermatology	13123.5	(8)	5763.3	(7)	4493.8	(4)	4220.6	(2)	4171.0	(1)	4598.0	(6)	4592.2	(5)	4319.5	(3)
Landsat	2560229.1	(8)	1018871.6	(5)	1443915.1	(7)	601655.8	(1)	602655.1	(2)	1393710.5	(6)	700436.2	(3.5)	700436.2	(3.5)
Zoo	2190.8	(8)	778.7	(7)	639.9	(4)	574.8	(2)	625.9	(3)	692.5	(6)	669.6	(5)	559.6	(1)
Image	138184.2	(8)	58519.0	(7)	45930.5	(5)	32868.9	(1)	34177.3	(2)	47989.8	(6)	40265.8	(4)	39389.7	(3)
Ecoli	13042.2	(8)	5987.3	(6)	5667.6	(5)	4517.0	(3)	4077.7	(1)	7102.1	(7)	4430.2	(2)	4523.0	(4)
Wine	1328.2	(8)	1026.3	(7)	966.6	(5.5)	813.3	(2.5)	813.3	(2.5)	966.6	(5.5)	813.3	(2.5)	813.3	(2.5)
Led1000	482300.0	(8)	135085.9	(7)	100067.7	(5)	84479.7	(1)	85114.2	(2)	102325.7	(6)	90089.4	(3)	94014.3	(4)
Yeast	647579.6	(8)	448982.6	(7)	227711.1	(4)	224859.9	(3)	203964.1	(1)	241541.5	(6)	222161.2	(2)	236926.0	(5)
Semeion	748680.7	(8)	204339.6	(7)	156151.1	(5)	133248.6	(1)	137288.9	(2)	162305.6	(6)	140415.5	(3)	143350.9	(4)
Optdigits	1368807.2	(8)	370229.3	(7)	294566.4	(5)	235926.9	(1)	241347.3	(2)	302281.5	(6)	253413.8	(3)	264930.8	(4)
Usps	8900232.4	(8)	2325554.1	(7)	1906896.6	(6)	1461284.7	(2)	1438043.6	(1)	1811028.4	(5)	1608009.9	(4)	1607457.1	(3)
Pendigits	2304147.0	(8)	719593.8	(7)	442605.0	(6)	350197.5	(1)	356407.6	(2)	428144.3	(5)	420305.3	(4)	404866.3	(3)
Vowel	176616.0	(8)	57287.3	(7)	34366.1	(5)	23282.4	(1)	24646.2	(2)	35996.8	(6)	25887.5	(3)	26533.3	(4)
mSLocRST	70460.7	(8)	20479.8	(7)	17600.7	(5)	12136.4	(2)	10966.5	(1)	17951.6	(6)	13464.6	(4)	12995.1	(3)
mSLocSun.	70945.4	(8)	15935.2	(7)	13394.9	(6)	9355.2	(2)	8552.0	(1)	12977.4	(5)	9615.0	(3)	9788.8	(4)
mSTemp.	109723.6	(8)	42598.0	(7)	18108.0	(5)	15144.1	(4)	13401.7	(1)	18675.8	(6)	14242.1	(2)	14943.6	(3)
Librasmove.	63964.8	(8)	19626.7	(7)	8629.5	(5)	6898.6	(1)	7803.8	(3)	8783.0	(6)	7516.8	(2)	8404.5	(4)
mSLocRain.	4824359.9	(8)	1233845.0	(7)	927612.4	(5)	503068.4	(2)	456543.2	(1)	941665.3	(6)	623840.5	(3)	710393.5	(4)
Avg. rank		(8)		(6.78)		(4.85)		(2.02)		(1.72)		(5.57)		(3.52)		(3.54)

also reveals that this does not occur with previous methods based on trees, particularly in the case of DDAG. Moreover, the algorithms proposed in Section 3.5 do not imply any computational overhead, so DBT approaches can be considered by practitioners in real-world applications, mainly in multi-class tasks in which the number of classes is large.

4.3. Evaluation time

An advantage of multi-class classifiers based on tree-combination schemes is that they speed up testing predictions. In order to compare the evaluation time employed for each method, we used the number of kernel evaluations. This is a good indicator of evaluation time when the base learner is a kernel method, like SVM, because the lower the number of kernel evaluations, the lower the time required for the testing stage. This approach was previously used in the same context by Platt et al. [37].



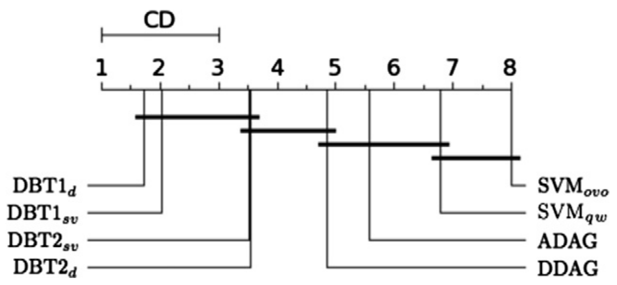


Fig. 4. Comparison of the number of kernel evaluations of all the methods using the Nemenyi test. Notice that eight algorithms are compared in this case. Thus, the critical differences are now $CD = 2.1893, p = 0.05$ (above), and $CD = 2.0080$ at $p = 0.10$ (below).

Table 4 shows the number of kernel evaluations of the different methods in the testing stage. The scores correspond to the first experiment reported in the previous section, in which all the methods use the same pairwise models. Consequently, the differences are due only to the combination strategy. We included an additional method, called QWeighted [35] and labeled as SVM$_{qw}$, which intelligently evaluates only the pairwise classifiers that are actually necessary to predict the same class as SVM$_{ovo}$. The main idea underlying this method is that, during the voting procedure of SVM$_{ovo}$, there exist many situations in which some classes may be excluded from the set of possible most-voted classes, even if they would win their remaining evaluations. This reduced the evaluations needed from $m(m-1)/2$ to only $m \log m$ in practice. To the best of our knowledge, QWeighted is the fastest algorithm to implement the testing procedure of SVM$_{ovo}$.

The behavior of SVM$_{ovo}$ was clearly expected, since all pairwise models need to be evaluated. Similarly, it was also presumed that DBT1$_{sv}$ would perform well, seeing as this method accurately attaches classifiers with less support vectors to nodes belonging to the upper levels of the DBT. These nodes are the ones that evaluate the most number of examples. Therefore, it is worth noting that DBT1$_d$ had the fastest evaluation, better than DBT1$_{sv}$. The difference between them is quite small, but it means that Eq. (4) is not only better in terms of accuracy, but also as good as Eq. (3) as regards evaluation time. Furthermore, the differences between all the tree-based algorithms are small, the biggest difference being between DBT1$_d$ and ADAG, which is a factor of 1.5 times slower.

The differences with respect to other approaches are greater: DBT1$_d$ is a factor of 5.8 times faster to evaluate than SVM$_{ovo}$, and 1.8 faster than SVM$_{qw}$. These factors respectively increase to 6.62 and 1.87 if we consider only data sets with ten or more classes. Notice that in contrast to SVM$_{qw}$, DBT1$_d$ provides a deterministic and fixed testing strategy, while SVM$_{qw}$ needs to run the QWeighted algorithm to test every example. We did not consider this additional computational time here.

The Friedman–Nemenyi test (see Fig. 4) indicates that DBT1$_d$ and DBT1$_{sv}$ are significantly better in terms of kernel evaluations than DDAG, ADAG, SVM$_{qw}$ and SVM$_{ovo}$. There are no differences between DBT approaches, while DDAG and ADAG are only significantly better than SVM$_{ovo}$, but not with respect to SVM$_{qw}$.

4.4. Real application

We also tested our proposal in a real application. Data consist of a collection of spectra obtained using near infrared reflectance microscopy (NIRM) [40] from animal feed samples taken at the Department of Animal Nutrition, Grasslands and Forages of the Regional Institute for Research and Agro-Food Development in Spain. These spectra were collected using a Fourier transform near infrared reflectance (FT-NIR) instrument attached to a microscope with an optical system designed to increase the efficiency of radiation transmission. Each spectrum is a set of absorbances (capacity of a substance to absorb light) in an interval of wavelengths. The interest of dealing with these data lies in detecting the ingredient of animal feed samples in an attempt on the part of the European Union of stopping the spread of bovine spongiform encephalopathy. In fact, NIRM has been proposed as an alternative of classical microscopy for identifying ingredients in animal feeds.

The data consists of 196 samples of 13 ingredients (classes), whose spectra were averaged after removing the trend of each spectrum and where each ingredient had at least five samples. The scores reported in Table 5 are the error rate and the average number of kernel evaluations using the same setting of the second experiment.

Table 5
Results of the compared methods for animal feed data set. The first row shows the error rate (%) and standard error for each method, and the second row the average number of kernel evaluations.

	SVM_{ovo}	DDAG	$DBT1_{sv}$	$DBT1_d$	ADAG	$DBT2_{sv}$	$DBT2_d$
Error rate	16.10 ± 1.93	16.60 ± 1.78	16.35 ± 1.86	15.85 ± 2.00	16.35 ± 1.86	16.60 ± 1.78	15.59 ± 1.98
Kernel eval.	21,590 ± 814	3803 ± 40	2941 ± 42	3341 ± 57	4064 ± 59	3086 ± 37	3502 ± 57

As it can be seen, $DBT2_d$ shows better performance than the rest of methods, also ameliorating the results reported in [18]. With regard to SVM_{ovo}, $DBT2_d$ slightly improves the error rate and, at the same time, it considerably reduces the number of kernel evaluations (in more than 80%). Indeed, these results confirm some conclusions derived from the experiments performed over benchmark data sets. For instance, the ranking orders in terms of error rate and kernel evaluations keep similar. Moreover, these results also corroborate that $DBT2_d$ performs particularly well when the number of classes is large.

5. Conclusions and future work

This paper analyzes the influence of the arrangement of the pairwise classifiers in tree structures. Placing classifiers of classes that are hard to separate in the lower levels of the tree leads to an improvement in the generalization ability of the resulting multi-class classifier. Among several structures, Directed Binary Trees (DBTs) are preferable, since they impose less constraints while at the same time they increase their flexibility.

The experiments carried out clearly show that the original random choice for the location of the models is not a good practice either in terms of error rate or prediction time, particularly for problems with a large number of classes. Hence, a guided arrangement leads to an overall improvement. In terms of the error rate, the methods proposed in this paper for locating the pairwise models are comparable and sometimes better than the one-vs-one decomposition approach combined with max-wins strategy. They also outperform previous tree-based methods such as DDAG and ADAG. As regards the computational time employed in the test stage, DBTs are faster than the rest of the compared approaches. They may therefore be very useful in the highly time-consuming experimental processes commonly performed by practitioners to estimate the true error of several algorithms for a given task.

In summary, DBT multi-class classifiers improve the classification performance and evaluation time of previous decomposition approaches based on trees and are very competitive with respect to the one-vs-one approach. This is even truer for multi-class data sets in which the number of classes is large. Our assumption is that DBTs are very well-suited for problems of this kind, because instead of taking into account the predictions of all pairwise models, they can mainly consider the decision of the binary classifier involving the two principal candidate classes, for instance, whether said model is placed near or just before the leaves.

The study has been particularized to the special case of using Support Vector Machines (SVMs) as binary classifiers. In this sense, it would be interesting as future work to extend the proposal to other binary classifiers in order to check if the performance of the DBTs and the strategy proposed remain.

Acknowledgments

The research reported in this paper has been partially supported by Spanish Ministerio de Economía y Competitividad (Grant TIN2011-23558). Besides, we would like to thank Begoña de la Roza and Ana Soldado from the Department of Animal Nutrition, Grasslands and Forages of the Regional Institute for Research and Agro-Food Development (SERIDA) for providing us with their animal feed data set.

References

[1] E. Allwein, R. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, Journal of Machine Learning Research (2001) 113–141.
[2] A. Asuncion, D. Newman, UCI machine learning repository, 2007.
[3] C. Chang, C. Lin, LIBSVM: a library for support vector machines, 2001.
[4] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (2002) 131–159.
[5] J. Chen, C. Wang, R. Wang, Adaptive binary tree for fast svm multiclass classification, Neurocomputing 72 (2009) 3370–3375.
[6] P. Chen, S. Liu, An improved dag-svm for multi-class classification, in: International Conference on Natural Computation (ICNC '09), pp. 460–462.
[7] K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines, Journal of Machine Learning Research 2 (2001) 265–292.
[8] K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems, Machine Learning 47 (2002) 201–233.
[9] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
[10] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 30.
[11] T. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, Journal of Artificial Intelligence Research 2 (1995) 263–286.
[12] J. Díez, J. del Coz, A. Bahamonde, O. Luaces, Soft margin trees, in: W. Buntine, M. Grobelnik, D. Mladenic, J. Shawe-Taylor (Eds.), Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, 5781, Springer, Berlin, Heidelberg, 2009, pp. 302–314.

[13] J. Fakcharoenphol, B. Kijsirikul, Constructing multiclass learners from binary learners: a simple black-box analysis of the generalization errors, in: S. Jain, H.U. Simon, E. Tomita (Eds.), Algorithmic Learning Theory, Lecture Notes in Computer Science, 3734, Springer, Berlin, Heidelberg, 2005, pp. 135–147.
[14] J. Fakcharoenphol, B. Kijsirikul, Erratum: Constructing multiclass learners from binary learners: a simple black-box analysis of the generalization errors, in: Proceedings of the 19th International Conference on Algorithmic Learning Theory, ALT '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 464–466.
[15] N. Farajzadeh, G. Pan, Z. Wu, M. Yao, Multiclass classification based on meta probability codes, IJPRAI 25 (2011) 1219–1241.
[16] B. Fei, J. Liu, Binary tree of svm: a new fast multiclass training and classification algorithm, IEEE Transactions on Neural Networks (2006) 696–704.
[17] J. Feng, Y. Yang, J. Fan, Fuzzy multi-class svm classifier based on optimal directed acyclic graph using in similar handwritten chinese characters recognition, in: International Symposium on Neural Networks (ISNN '05), vol. 3496, pp. 875–880.
[18] V. Fernández-Ibáñez, T. Fearn, E. Montañés, J. Quevedo, A. Soldado, B. de la Roza-Delgado, Improving the discriminatory power of a near-infrared microscopy spectral library with a support vector machine classifier, Applied Spectroscopy 64 (2010) 46–52.
[19] E. Frank, S. Kramer, Ensembles of nested dichotomies for multi-class problems, in: International Conference of Machine Learning (ICML '04), pp. 305–312.
[20] J. Friedman, Another Approach to Polychotomous Classification, Technical Report, Department of Statistics, Stanford University, 1996.
[21] J. Fürnkranz, Round robin classification, Journal of Machine Learning Research 2 (2002) 721–747.
[22] N. García-Pedrajas, D. Ortíz-Boyer, Improving multiclass pattern recognition by the combination of two strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1001–1006.
[23] N. García-Pedrajas, D. Ortíz-Boyer, An empirical study of binary classifier fusion methods for multiclass classification, Information Fusion 12 (2011) 111–130.
[24] T. Hastie, R. Tibshirani, Classification by Pairwise Coupling, in: NIPS '97, MIT Press, Cambridge, MA, USA, 1998, pp. 507–513.
[25] C. Hsu, C. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2002) 415–425.
[26] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: C. Nédellec, C. Rouveirol (Eds.), Proceedings of ECML-98, 10th European Conference on Machine Learning, Springer, Heidelberg, 1998, pp. 137–142.
[27] B. Kijsirikul, N. Ussivakul, Multiclass support vector machines using adaptive directed acyclic graph, in: International Joint Conference on Neural Networks (IJCNN '02), pp. 980–985.
[28] A. Klautau, N. Jevtić, A. Orlitsky, On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines, Journal of Machine Learning Research 4 (2003) 1–15.
[29] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data, Journal of the American Statistical Association 99 (2004) 67–82.
[30] A. Lorena, A. Carvalho, J. Gama, A review on the combination of binary classifiers in multiclass problems, Artificial Intelligence Review 30 (2008) 19–37.
[31] W. Maass, On the computational power of winner-take-all, Neural Computation 12 (2000) 2519–2535.
[32] G. Madzarov, D. Gjorgjevikj, I. Chorbev, A multi-class svm classifier utilizing binary decision tree, Informatica (Slovenia) 33 (2009) 225–233.
[33] O. Murphy, R. McCraw, Designing storage efficient decision trees, IEEE Transactions on Computers 40 (1991) 315–320.
[34] T.H. Oong, N.A.M. Isa, One-against-all ensemble for multiclass pattern classification, Applied Soft Computing 12 (2012) 1303–1308.
[35] S.H. Park, J. Fürnkranz, Efficient pairwise classification, in: J.N. Kok, J. Koronacki, R. López de Mántaras, S. Matwin, D. Mladenić, A. Skowron (Eds.), Proceedings of 18th European Conference on Machine Learning (ECML-07), Springer-Verlag, Warsaw, Poland, 2007, pp. 658–665.
[36] A. Passerini, M. Pontil, P. Frasconi, New results on error correcting output codes of kernel machines, IEEE Transactions on Neural Networks 15 (2004) 45–54.
[37] J. Platt, N. Cristianini, J. Shawe-Taylor, Large margin DAGs for multiclass classification, in: NIPS '00, pp. 547–553.
[38] J. Quevedo, A. Bahamonde, M. Pérez-Enciso, O. Luaces, Disease liability prediction from large scale genotyping data using classifiers with a reject option, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9 (2012) 88–97.
[39] R. Rifkin, A. Klautau, In defense of one-vs-all classification, Journal of Machine Learning Research 5 (2004) 101–141.
[40] B. de la Roza-Delgado, A. Soldado, A. Martínez-Fernandez, F. Vicente, A. Garrido-Varo, D. Pérez-Marín, M. de la Haba, J. Guerrero-Ginel, Application of near-infrared microscopy (nirm) for the detection of meat and bone meals in animal feeds: a tool for food and feed safety, Food Chemistry 105 (2007) 1164–1170.
[41] W.R.T. Phetkaew, B. Kijsirikul, Reordering adaptive directed acyclic graphs: an improved algorithm for multiclass support vector machines, in: International Joint Conference on Neural Networks (IJCNN '03), vol. 2, pp. 1605–1610.
[42] F. Takahashi, S. Abe, Optimizing directed acyclic graph support vector machines, in: Artificial Neural Networks in Pattern Recognition (ANNPR '03), pp. 166–170.
[43] R. Tibshirani, T. Hastie, Margin trees for high-dimensional classification, Journal of Machine Learning Research 8 (2007) 637–652.
[44] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag New York, Inc., New York, NY, USA, 1995.
[45] J. Weston, C. Watkins, Support vector machines for multi-class pattern recognition, in: Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANNs).
[46] T. Windeatt, R. Ghaderi, Coding and decoding strategies for multi-class learning problems, Information Fusion 4 (2003) 11–21.
[47] J. Zhou, H. Peng, C.Y. Suen, Data-driven decomposition for multi-class classification, Pattern Recognition 41 (2008) 67–76.