Information Fusion 66 (2021) 64–75

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Decomposition-Fusion for Label Distribution Learning

Manuel González [a,*], Germán González-Almagro [a], Isaac Triguero [c], José-Ramón Cano [b], Salvador García [a]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain
[b] Department of Computer Science, University of Jaén, EPS of Linares, Avenida de la Universidad S/N, Linares 23700, Jaén, Spain
[c] The Computational Optimization and Learning (COL) Lab, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, United Kingdom

ARTICLE INFO

Keywords:
Label Distribution Learning
Decomposition strategies
One vs. One
Machine learning

ABSTRACT

Label Distribution Learning (LDL) is a general learning framework that assigns an instance to a distribution over a set of labels rather than to a single label or multiple labels. Current LDL methods have proven their effectiveness in many real-life machine learning applications. However, LDL is a generalization of the classification task and as such it is exposed to the same problems as standard classification algorithms, including class-imbalanced, noise, overlapping or irregularities. The purpose of this paper is to mitigate these effects by using decomposition strategies. The technique devised, called Decomposition-Fusion for LDL (DF-LDL), is based on one of the most renowned strategy in decomposition: the One-vs-One scheme, which we adapt to be able to deal with LDL datasets. In addition, we propose a competent fusion method that allows us to discard non-competent classifiers when their output is probably not of interest. The effectiveness of the proposed DF-LDL method is verified on several real-world LDL datasets on which we have carried out two types of experiments. First, comparing our proposal with the base learners and, second, comparing our proposal with the state-of-the-art LDL algorithms. DF-LDL shows significant improvements in both experiments.

1. Introduction

A supervised learning process is the machine learning task of training a predictive model using data points with known outputs. Classification is the problem of identifying to which of a set of categories a new observation belongs. Hence, the aim of classification is to obtain a model that will be able to assign the correct class to an unknown pattern. However, there is a growing number of problems in which a pattern can have several labels simultaneously associated. Examples are found in image classification [1] or genetics [2], etc. Multi-Label Learning (MLL) [3–6] is a generalization of the traditional classification where multiple labels may be assigned to each instance.

Nevertheless, in many real-world problems we can find cases in which MLL is still not sufficient since the level of description of each label is not the same. To name just one example from the datasets used in this paper, the biological experiments on the yeast genes [7] over a period of time result in different levels of gene expression in a time series. The precise degree of expression at each point in time is of minor significance. What is really crucial is the distribution of the overall expression over the entire period of time. If the learning task is to predict that distribution for a given gene, it can hardly fit into the MLL framework because the role of the individual output in the distribution is crucial, and there is no partitioning of relevant and irrelevant labels at all.

The Label Distribution Learning (LDL) concept appeared for first time in 2013 [8] and was formally described in 2016 [9] in order to deal with ambiguity on the label side of the mapping when one instance is not necessarily mapped to one single label. The aim of this paradigm is to answer the question "how much does each label describe the instance?" instead of "which label can describe the instance?".

From the first formulation of the LDL problem, numerous studies have been carried out applying the LDL methodology to various real life problem solving situations, e.g.: sense beauty recognition [10], facial age estimation [8], personality recognition on social media [11], image emotion classification [12], pre-release prediction of crowd opinion on movies [13], crowd counting in public video surveillance [14], head pose estimation [15], etc. Other studies have focused more on developing new learners or on adapting existing learners such as: instance-based algorithms (e.g., AA-*k*NN [9]), optimization algorithms (e.g., SA-IIS, SA-BFGS [9] or LDL-SCL [16]), decision trees (e.g., LDL forests [17]), deep learning algorithms (e.g., Deep

* Corresponding author.
E-mail addresses: manuel.gonzalez.es@gmail.com (M. González), germangalmagro@ugr.es (G. González-Almagro), isaac.triguero@nottingham.ac.uk (I. Triguero), jrcano@ujaen.es (J.-R. Cano), salvagl@decsai.ugr.es (S. García).

https://doi.org/10.1016/j.inffus.2020.08.024
Received 23 February 2020; Received in revised form 10 July 2020; Accepted 29 August 2020
Available online 4 September 2020

Label Distribution [18]), or ensembles strategies (e.g., Logistic boosting regression for LDL [19], Structured random forest for LDL [20]).

LDL is a generalization of the classification task [21] and as a result is vulnerable to the same problems as conventional classification algorithms: imbalanced datasets [22], when there is a disproportion in the number of examples of the different classes; noisy data [23], because of imperfections in data acquisition, transmission or storage; overlapping [24], when the input features are not sufficient to correctly differentiate among instances of different classes; or irregularities [25], situations where the distribution of data points, the sampling of the data space to generate the training set, and the characteristics that describe each data point deviate from what might have been ideal, being biased, skewed, incomplete and/or misleading.

Over the last years, decomposition strategies for addressing classification problems have been widely studied in the literature [26]. The same underlying idea is behind all proposals for decomposition: to solve a multi-class problem using binary classifiers. Decomposition strategies have proven to be efficient in dealing with the difficulties presented above and that is why, in this paper, we propose a decomposition algorithm adapted to LDL restrictions. The devised technique is inspired by one of the most renowned strategies in decomposition: the One-vs-One (OVO) [27] scheme, where the original problem is divided into binary problems that distinguish between the different pairs of classes, every single division will be trained with a base classifier. This method usually requires an additional step to fuse the outputs from single classifiers in order to produce the final result.

We design a decomposition strategy that can handle label distribution, capable of dealing with real values instead of multi-class outputs. While OVO uses a binary classifier as base learner (the learning algorithm used for solving binary problems), in our proposal we will rather rely on a specific LDL learner. In addition to these, we also propose a fusion method, capable of providing an output according to the LDL constraints, and that also will allow us to discard non-competent classifiers when their output is probably not of interest.

Decomposition-Fusion for LDL, from now DF-LDL, is our decomposition proposal for LDL type problems. In order to evaluate the proposal proficiency, we will carry out two types of experiments: on the one hand we will compare the results obtained by the base learners with our DF-LDL algorithm using the same learner as base classifier and, on the other hand, we will compare our proposal with the state-of-the-art LDL algorithms, measuring in all cases six aspects of their performance. We will repeat the experiment over 17 real-world datasets and validate the results of the empirical comparisons using Wilcoxon, Friedman rank and Bayesian Sign tests [28,29].

In summary, the main contributions of this paper are:

- A decomposition strategy to handle LDL problems.
- A fusion method custom-designed to provide an output compliant with LDL restrictions and that also allows to exclude non-competent classifiers.
- From a technical point of view, the proposed solution could be implemented in a very fast way taking advantage of any existing LDL learner.

The rest of the paper is organized as follows. First, a brief review and discussion of the foundations of LDL and decomposition strategies for classification are given in Section 2. The proposed Decomposition-Fusion for LDL method is described in Section 3. Then the details of the experiments are reported in Section 4. Finally, the results and conclusions are drawn in Section 5 and Section 6, respectively.

2. Preliminaries

In this section, the foundations, as well as the most relevant studies carried out on LDL (Section 2.1), are presented. Furthermore, some basic concepts on decomposition strategies for classification are introduced (Section 2.2), providing the necessary background required to properly present the study carried out in this paper. We will conclude this section by explaining the motivations for applying decomposition strategies to LDL-type problems.

2.1. Foundations of label distribution learning

We can formalize an LDL problem as a set of m training samples $S = \{(x_1, D_1), \dots, (x_m, D_m)\}$, where $x_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$ is a q-dimensional vector. For each instance x_i, the label distribution is denoted by $D_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, \dots, d_{x_i}^{y_c}\}$ where $y_i \in Y | i \in \{i, \dots, c\}$, such that $Y = \{y_1, \dots, y_c\}$ denotes the complete set of labels. The constant c is the number of possible labels and $d_{x_i}^{y_j}$ is the description degree of the particular jth label y_j for a particular ith instance x_i. According to the definition, each description degree should meet the constraints $d_x^y \in [0,1]$ and $\sum_{y=1}^{c} d_x^y = 1$.

Solving an LDL problem can be approached from different perspectives. Depending on the approach chosen, the algorithm to be developed will vary considerably, either a completely new algorithm developed specifically to deal with LDL constraints, or an adaptation of existing classification algorithms, reformulated to work with these constraints. The LDL study published in [9] proposed six algorithms classified in three categories. The first one is Problem Transformation (PT), a straightforward way to transform an LDL problem into a Single-Label Learning or SLL [30] problem is to change the training examples into weighted single-label examples. In this way, any SLL algorithm can be applied. Two representative algorithms are PT-Bayes and PT-SVM. The second one is Algorithm Adaptation (AA), where algorithms are adapted to existing learning algorithms to deal directly with the label distribution. Two suitable algorithms were presented: AA-*k*NN, an adaptation of the well-known *k*-nearest neighbors method [31], and AA-BP, a tree-layer backpropagation neural network. Finally, Specialized Algorithms (SAs), unlike the indirect strategy of problem transformation and algorithm adaptation, match directly the LDL problem. SA-IIS and SA-BFGS are two specialized algorithms that learn by optimizing an energy function based on the maximum entropy model.

Further studies have succeeded in improving the results obtained by these original algorithms using different strategies. The methods LDLogitBoost and AOSO-LDLogitBoost proposed in [19], are a combination of the boosting method and the logistic regression applied to LDL model. Deep label distribution learning (DLDL) [18] and Label Distribution Learning Based on ensemble neural networks [32] are two good examples of success applying neural networks on LDL. Inspired by differentiable decision trees [33], an end-to-end strategy LDL forests proposed in [17] which served as the basis for Structured Random Forest (StructRF) [20]. BC-LDL [34] and DBC-LDL [35] use the binary coding techniques to deal with the large-scale LDL problem. Classification with LDL (LDL4C) [21] is another interesting proposition when learned label distribution model is generally treated as a classification model. Feature selection on LDL [36,37] shows promising results by applying selection of characteristics on label distribution problems.

2.2. Decomposition strategies for classification

Decomposition strategies for addressing multi-class problems have been widely studied in the literature [38]. The same underlying idea is behind all proposals for decomposition: to solve a multi-class problem using binary classifiers. Following the “divide and rule” paradigm, the problem of multiple classes is divided into simpler binary classification problems. However, this method needs an additional step because of the simplification of the base classifiers: their outputs must be recombined to obtain the final result. How this aggregation is carried out is crucial to the quality of the final result. An exhaustive comparison of decomposition strategies and aggregation methods can be found in [38].

The most common decomposition strategies are One-vs-All (OVA) [39] and One-vs-One (OVO) [27], which can be included within Error Correcting Output Codes (ECOC) framework [40]. The first learns a binary classifier to discern between each pair of classes, while the second builds a binary classifier to separate each class from all the other classes.

The One-vs-All (OVA) [39] approach consists in dividing the multiple c class problem into binary c classification problems. Each binary classifier is faced up by a binary classifier which is responsible of distinguishing one of the classes from all other classes. The training phase of each classifier is carried out using the complete training set, considering as positive the samples of the single class and as negative all other samples. In the validation phase, the example is classified using each of the binary classifiers. The classifier that obtains a positive output will show the output class. Note that the output may not be unique and in these cases some sort of tiebreaker mechanism must be used. For example, we can calculate the confidence of each classifier to decide the final output by predicting the class from the classifier with the highest confidence.

The One-vs-One (OVO) [27] decomposition scheme divides a problem of c classes into $c(c-1)/2$ binary problems. Each problem is addressed by a binary classifier that distinguishes between the different pairs of classes. The learning phase of each classifier is carried out using a subset of instances containing one of the two output classes. Instances with a different class are ignored.

In the prediction phase, the sample to be validated is predicted by each of the classifiers trained previously, thus obtaining a score matrix R:

$$R = \begin{pmatrix} - & r_{r2} & r_{13} & \dots & r_{1c} \\ r_{21} & - & r_{23} & \dots & r_{2c} \\ . & . & . & . & . \\ . & . & . & . & . \\ r_{c1} & r_{c2} & r_{c3} & \dots & - \end{pmatrix}$$

The output of a classifier given by $r_{ji} \in [0,1]$ is the confidence of the binary classifier discriminating classes i and j in favor of the previous class. The confidence of the classifier for j is calculated as $r_{ji} = 1 - r_{ij}$ if the classifier does not provide it (the class with the larger confidence is the selected output class of a classifier).

The final prediction is calculated based on the score matrix using different aggregation models. The weighted voting strategy is the most used strategy, where confidences are aggregated class by class (by rows) and the one with the highest sum is selected as output. There is a disadvantage, known as the “non-competent classifier problem” [41] in the OVO system. The classifiers in the OVO system are not sufficiently competent to classify all the classes in the problem, as they are only learned through examples of two classes. However, all binary classifiers will be triggered for a given test model, because the competence cannot be known a priori, which can lead to incorrect decisions. To mitigate this problem there exist dynamic selection techniques that are able to distinguish between competent sorters directly in the prediction phase. The studies carried out in [42] and [43] provide a review of the most popular dynamic selection techniques to avoid the non-competent classifiers problem. On their side, [44,45] are some examples of how to successfully apply the OVO technique.

ECOC [40] provides a suitable matrix framework for modeling the decomposition of a multi-class classification problem into simpler sub-problems. How to perform the decomposition to fit better the data using a small number of classifiers has been a key point of the research, as well as the decoding step, which deals with the combination of the sub-problems. The research [46] proposes an evidential unified framework that handles both the coding and decoding steps.

2.3. Motivation for using decomposition strategies on LDL

LDL is a generalization of the classification model and as such is exposed to the same problems as classic classification algorithms: imbalanced datasets, noisy data, overlapping or irregularities. OVO and OVA strategies have proven to be efficient in dealing with these kinds of difficulties. In [22,47] we can find a complete review of how to apply decomposition strategies to imbalanced datasets that solve the disproportion of the number of examples of the different classes. The results obtained in [26,48] show that using the OVO strategy lead to better performances and more robust classifiers when dealing with noisy data, especially with the most disruptive noise schemes. Decomposition performed by OVO in [49] helps to increase the separation between classes, creating more regular decision boundaries where there are overlapping samples. Other irregularities, such as the problem of the “difficult classes”, have also been successfully addressed using the OVO decomposition strategy [50].

Applying a decomposition technique to an LDL-type dataset, comparing output labels and breaking relationships between non-working pairs, could lead to better performance of the base LDL algorithms due to the proven improvement achieved by these techniques when they have been applied on datasets presenting the above problems. However, it should be recalled at this point that the LDL paradigm differs significantly from a multi-class problem, what we have is an output label distribution. Therefore, the OVO strategy would not serve us as it is. On the one hand, we need a new decomposition strategy capable of dealing with real values instead of labels and on the other hand, a new aggregation strategy capable of providing an output according to the LDL constraints. As base learner we can use any of the LDL-specific learners.

3. DF-LDL: Decomposition-Fusion for Label Distribution Learning

DF-LDL is our decomposition proposal for LDL type of problems. The main idea of this method is to decompose the original c-label problem by dividing it into $c(c-1)$ problems, that is, comparing one to one all the labels that make up the output as we will explain in Section 3.1. The next step is to combine the outputs to obtain the final prediction, for this, we have conceived the competent fusion method which will be exposed in Section 3.2. Finally, we will illustrate how it works in Section 3.3.

3.1. Decomposition strategy

As opposed to a multi-class classification problem that would decompose the problem into $c(c-1)/2$ problems (one for each possible class pair), in LDL we need a decomposition mechanism able to carry out a comparison between the real values of each label. The strategy consists in making an inequality comparison ($a \geq b$) between each pair of output labels, creating subsets of samples containing only the samples whose label value is greater than the value of the compared label. The goal is to create subsets that break up relationships between pairs of labels that do not work. In this way, we will obtain the double of classifiers that in the original OVO strategy (because we are comparing real values and not grouping them into pairs of classes), that is $c(c-1)$ subsets on which we will start the training phase obtaining a matrix of learners like:

$$L = \begin{pmatrix} - & l_{12} & l_{13} & \dots & l_{1c} \\ l_{21} & - & l_{23} & \dots & l_{2c} \\ l_{31} & l_{32} & - & \dots & l_{3c} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ l_{c1} & l_{c2} & l_{c3} & \dots & - \end{pmatrix}$$

The detailed process is summarized in Algorithm 1.

Algorithm 1: Training stage of DF-LDL

```
Function Fit(S, m, c, l):
1   input : $S \leftarrow$ Training Dataset ;
            $m \leftarrow$ Number of Samples ;
            $c \leftarrow$ Number of Labels ;
            $l \leftarrow$ Base Learner ;
2   output: $L \leftarrow$ Matrix of learners ;
3   for i=1 to c do
4       for j=i+1 to c do
            #Create the subsets of samples
5           $SS1 = \emptyset$ ;
6           $SS2 = \emptyset$ ;
7           for k=1 to m do
8               if $d_{x_k}^{y_i} \geq d_{x_k}^{y_j}$ then
9                   Append: $(x_k, D_k)$ to $SS1$ ;
                else
10                  Append: $(x_k, D_k)$ to $SS2$ ;
                end
            end
            #Fit the base learners using the subsets
11          $l$.setTrainingSet(SS1);
12          $l$.fit();
13          $L_{ij} = l$;
14          $l$.setTrainingSet(SS2);
15          $l$.fit();
16          $L_{ji} = l$;
        end
    end
End Function
```

Starting from the total set of training samples S (containing m samples in total), we classify each sample (x_k, D_k) into the subsets $SS1$ or $SS2$ depending on whether the value of label i (represented by $d_{x_k}^{y_i}$) is greater or lesser than the value of label j (represented by $d_{x_k}^{y_j}$). Therefore, the subset $SS1$ will contain the samples that maximize the value of the output label i with respect to the output label j, $SS2$ is consequently the complementary set to $SS1$. Each subset is trained separately using the base classifier l. This process is repeated $c(c-1)/2$ times obtaining as result the L learner matrix that will contain a total of $c(c-1)$ learners. Note that the base learner l can be any LDL-compatible learning algorithm.

3.2. Competent fusion method

In the prediction phase, the sample to be validated is usually predicted by each one of the classifiers trained previously but, as we have previously stated, our aim has been to design a dynamic competent aggregation or fusion method that allows us to build the final solution using only the learners that can minimize the error. We have been inspired by the dynamic classifier selection for One-vs-One strategy proposed in [43] that tries to avoid the non-competent classifiers when their output is probably not of interest. We consider an initial prediction of each instance to decide whether a classifier may be competent or not. Obviously, this initial prediction must be very run-time efficient in order to not penalize the total prediction time. For these reasons the technique chosen for this initial prediction is the AA-*k*NN [9] that obtains a prediction with an appropriate quality/time trade-off.

The competent fusion procedure summarized in Algorithm 2 works as follow: (1) Predict the example x_0 using AA-*k*NN and obtaining the initial prediction p_{knn}. Given the instance x_0, its k nearest neighbors are first found in the training set. Then, the mean of the label distributions of all the k nearest neighbors is calculated as the label distribution of x_0, i.e.,

$$p_{knn}(y_j|x_0) = \frac{1}{k} \sum_{i \in N_k(x_0)} d_{x_i}^{y_j}, (j = 1, 2, \ldots, c),$$

where $N_k(x_0)$ is the index set of the k nearest neighbors of x_0 in the training set. (2) Use this initial prediction to compare one by one the output values of the labels. If the value of the label i, $p_{knn}^{y_i}$, is greater or equal than the value of the label j, $p_{knn}^{y_j}$, then we mark as selected the learner L_{ij}, otherwise we select the opposite learner Lji. (3) Predict the label distribution of the example x_0 using the selected learner in previous step and add this prediction to the final output p. (4) Repeat points 2 and 3 for each pair of labels. (5) The final label distribution p is calculated as the average of predictions obtained by all selected learners:

$$p(y_j|x_0) = \frac{1}{c(c-1)/2} \sum_{ij \in L(x_0)} p_{l_{ij}}^{y_j}, (j = 1, 2, \ldots, c),$$

where $L(x_0)$ is the index set of selected learners for x_0. In total $c(c-1)/2$ learners will participate in the prediction.

Algorithm 2: Predict an example in DF-LDL

```
Function Predict(S, c, L, $x_0$, k):
1   input : $S \leftarrow$ Training Dataset ;
            $c \leftarrow$ Number of Labels ;
            $L \leftarrow$ Matrix of learners ;
            $x_0 \leftarrow$ Example to predict ;
2   output: $p \leftarrow$ Prediction ;
    #Initial prediction using AA-kNN
3   AA-$k$NN.setTrainingSet(S);
4   $p_{knn}$ = AA-$k$NN.predict($x_0$);
    #Select learners and predict
5   $p = 0$;
6   for i=1 to c do
7       for j=i+1 to c do
8           if $p_{knn}^{y_i} \geq p_{knn}^{y_j}$ then
9               $p = p + L_{ij}.predict(x_0)$;
            else
10              $p = p + L_{ji}.predict(x_0)$;
            end
        end
    end
11  return $p/(c*(c-1)/2)$ ;
End Function
```

3.3. An illustrative example

In order to show how the proposal works, let us take the toy dataset $S = \{(x_1, D_1), (x_2, D_2), (x_3, D_3)\}$ illustrated below containing $m = 3$ samples with $q = 2$ features and $c = 3$ output labels:

$$S = \begin{Bmatrix} x_1 & D_1 \\ x_2 & D_2 \\ x_3 & D_3 \end{Bmatrix} = \begin{Bmatrix} (0.4, 0.7) & (0.1, 0.85, 0.05) \\ (0.01, 0.9) & (0.75, 0.25, 0) \\ (0.5, 0.5) & (0.05, 0.05, 0.9) \end{Bmatrix}.$$

Following the process described in Section 3.1 we need to obtain a 3×3 matrix of learners. In order to build the first learner l_{12}, we will compare the output label 1 and 2 of each sample. In our case:

$$\begin{pmatrix} d_1^1 & d_1^2 \\ d_2^1 & d_2^2 \\ d_3^1 & d_3^2 \end{pmatrix} = \begin{pmatrix} 0.1 & 0.85 \\ 0.75 & 0.25 \\ 0.05 & 0.05 \end{pmatrix}.$$

As $d_2^1 \geq d_2^2$ and $d_3^1 \geq d_3^2$, the subset of samples used for learner l_{12} will be $(x_2, D_2), (x_3, D_3)$. Therefore the subset of samples for learner l_{21} will be $(x_1, D_1), (x_3, D_3)$.

Repeating this procedure for each pairs of labels we obtain the learner matrix below, for each learner we are showing the subset of samples selected:

$$L = \begin{pmatrix} - & \{(x_2, D_2), (x_3, D_3)\} & \{(x_1, D_1), (x_2, D_2)\} \\ \{(x_1, D_1), (x_3, D_3)\} & - & \{(x_1, D_1), (x_2, D_2)\} \\ \{(x_3, D_3)\} & \{(x_3, D_3)\} & - \end{pmatrix}$$

Once we have trained each of the classifiers using the subsets generated, we will predict the distribution of labels of a test instance, $x_0 = (0.2, 0.8)$, according to the method explained in Section 3.2. Let us remember that the process consisted of five steps: (1) we predicted the output values through a AA-kNN algorithm using the whole training set. The k-NN prediction obtained for x_0 is:

$$p_{knn} = (0.3, 0.38, 0.32)$$

Then, in step (2), we use that prediction to discard incompetent learners by comparing each pair of output labels. In order to check if the first learner l_{12} will be part of the final prediction, we will compare the output label 1 and 2. Value 0.3 is not greater or equal to 0.38, then learner l_{12} will be discarded and, consequently, the opposite one l_{21} will be selected.

In step (3), x_0 will be predicted by the selected learner, adding this prediction to the final output p_{x_0}. Repeating this process for each pair of labels, step (4), the learners that will be considered to obtain the final prediction are:

$$\begin{pmatrix} - & \cancel{l_{12}} & \cancel{l_{13}} \\ \mathbf{l_{21}} & - & \mathbf{l_{23}} \\ \mathbf{l_{31}} & \cancel{l_{32}} & - \end{pmatrix}$$

Finally, as described in step (5), the final label distribution p_{x_0} is divided by 3 (the total of learners that have been involved in building the solution). Assuming that the values obtained by each of the learners are as follows:

$$p_{l_{21}} = (0.2, 0.7, 0.1)$$

$$p_{l_{23}} = (0.4, 0.6, 0)$$

$$p_{l_{31}} = (0.25, 0.5, 0.25)$$

The final label distribution p_{x_0} will be:

$$p_{x_0} = (0.28, 0.6, 0.12)$$

4. Experimental framework

This section is devoted to introducing the experimental framework used in the different empirical studies of the paper. In our experiments, we have included 17 datasets of a wide variety of real-world problems, described in the following Section 4.1.

In order to evaluate the learners proficiency, we will employ six measures (described in Section 4.2) for the different aspects of their performance. For each dataset and learner, these measures were computed over a merged set from the test predictions of a 10-fold cross validation set (10-fcv).

We have run two different kinds of experiments. The first one consists in comparing the results provided by the base learners with the DF-LDL method using the same base learner. The selected base learners have been SA-BFGS [9] and Structured tree (StructTree), this second one has been extracted from the Structured random forest (StructRF) [20] proposal but by training a single tree. The second experiment directly compares the DF-LDL proposal with other LDL state-of-the-art algorithms, such as: SA-BFGS, label distribution learning forests (LDLFs) [17] and StructRF.

The non-parametric statistical Wilcoxon, Friedman rank and Bayesian Sign tests [28,29] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved using methods L (base learner in first experiment) and R (DF-LDL) is computed into a graphical space divided in 3 regions: left, rope and right. The location of most of the distribution in these sectors indicates the final decision: the superiority of algorithm L, statistical equivalence and the superiority of algorithm R, respectively. KEEL package [51] has been used to compute the Wilcoxon and Friedman rank tests and the R package rNPBST [52] was used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies. The Rope limit parameter used to represent the Bayesian Sign test is 0.0001.

Table 1
Datasets used in experiments.

No.	Datasets	Examples (m)	Features (q)	Labels (c)	Type
1	Yeast_alpha	2465	24	18	LDL
2	Yeast_cdc	2465	24	15	LDL
3	Yeast_diau	2465	24	7	LDL
4	Yeast_elu	2465	24	14	LDL
5	Yeast_heat	2465	24	6	LDL
6	Yeast_spo	2465	24	6	LDL
7	SJAFFE	213	243	6	LDL
8	SBU_3DFE	2500	243	6	LDL
9	Movie	7755	1869	5	LDL
10	Natural_Scene	2000	294	9	LDL
11	Human_Gene	30 542	36	68	LDL
12	Optdigits	5620	64	10	Multi-class
13	Semeion	1593	256	10	Multi-class
14	Ecoli	327	7	5	Multi-class
15	LED7digit	500	7	10	Multi-class
16	Wq	1060	16	14	Multi-target
17	Jura	359	15	3	Multi-target

4.1. Datasets

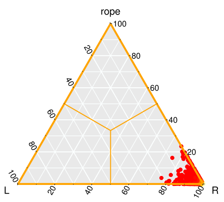
There are a total of 17 real-world datasets employed in the experiments. The summary of their characteristics is shown in Table 1.

The first 11 datasets are originally LDL problems. To complete the experiment, we have also added 4 multi-class classification datasets and 2 multi-target regression datasets, with a double purpose: to see how LDL learners behave on other types of classification problems and because purely LDL datasets are still scarce. Note that non-LDL datasets have been adapted to satisfy LDL restrictions. This transformation is described below along with the description of each set.

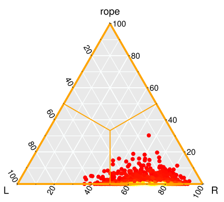
The first six datasets have been collected from biological experiments on the budding yeast Saccharomyces cerevisiae [7]. It includes 2465 yeast genes, and an associated phylogenetic profile vector with a length of 24 is utilized to represent each gene. In a biological experiment, the gene expression level is usually disparate at each discrete time point, so the labels correspond to the time point.

Datasets JAFFE [53] and BU_3DFE [54] are two widely used facial expression image datasets. There are 213 gray-scale expression images in the JAFFE dataset while BU_3DFE contains 2500 facial expression images. The images in JAFFE have been scored by 60 people using the six primary emotion labels with a 5-level scale, i.e., fear, disgust, happiness, anger, sadness, surprise, and the images in BU_3DFE have been scored by 23 people using the same scale as used in JAFFE. Each dataset is represented by a 243-dimensional feature vector extracted using the Local Binary Patterns method (LBP) [55]. The score for each emotion is regarded as the description degree, and the description degrees (normalized gene expression level) of all the six emotions constitute a label distribution for a particular facial expression image.

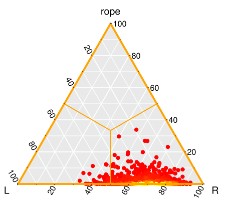
Dataset Movie includes 7755 movies. There is a total of 54,243,292 ratings from 478,656 different users on a scale from 1 to 5 integral stars from ®Netflix. The percentage of each rating level is regarded as the label distribution. There are numeric and categorical attributes in the dataset such as genre, director, country, year, budget and so on. After transforming the categorical attributes into binary vectors, the final feature vector of each movie is 1869-dimensional.



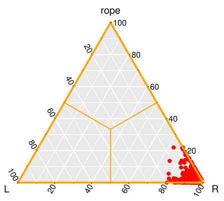
(a) Chebyshev Distance



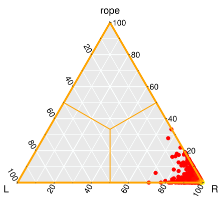
(b) Clark Distance



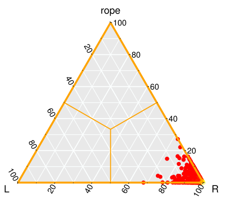
(c) Canberra Metric



(d) Kullback-Leibler Divergence



(e) Cosine Coefficient



(f) Intersection Similarity

Fig. 1. Bayesian sign test comparing StructTree(L) vs. DF-LDL(R).

Table 2
Evaluation measure for LDL learners. ↓ *means that the lowest value is the best and* ↑ *means the opposite.*

Name	Formula
Chebyshev(Cheby)↓	$Dis(D,\hat{D}) = max_j \lvert d_j - \hat{d}_j \rvert$
Clark↓	$Dis(D,\hat{D}) = \sqrt{\sum_{j=1}^{c} \frac{(d_j-\hat{d}_j)^2}{(d_j+\hat{d}_j)^2}}$
Canberra(Can)↓	$Dis(D,\hat{D}) = \sum_{j=1}^{c} \frac{\lvert d_j-\hat{d}_j \rvert}{d_j+\hat{d}_j}$
Kullback–Leibler(KL)↓	$Dis(D,\hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$
Cosine(Cos)↑	$Sim(D,\hat{D}) = \frac{\sum_{j=1}^{c} d_j \hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^2}\sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$
Intersection(Inter)↑	$Sim(D,\hat{D}) = \sum_{j=1}^{c} min(d_j, \hat{d}_j)$

Table 3
Summary of the parameters.

Algorithm	Parameter	Description	Value
AA-*k*NN	k	Number of selected neighbors	4
SA-BFGS	ϵ	Convergence criterion: must be less than ϵ before successful termination	10^{-5}
LDLFs	Trees	Number of trees	5
	Depth	Maximum depth of the tree	7
	out. feat.	Output unit number of the feature learning function	64
	iters. leaf	Iteration times to update leaf node predictions	20
	Batches	Number of mini-batches to update leaf node predictions	100
	Iters	Maximum iterations	2500
StructTree	max. depth	Maximum depth of the tree	20
	min. leaf	Minimum size of the leaf	5
StructRF	Trees	Number of trees	50
	Sampling	Sampling ratio of data	0.8
	max. depth	Maximum depth of the tree	20
	min. leaf	Minimum size of the leaf	5

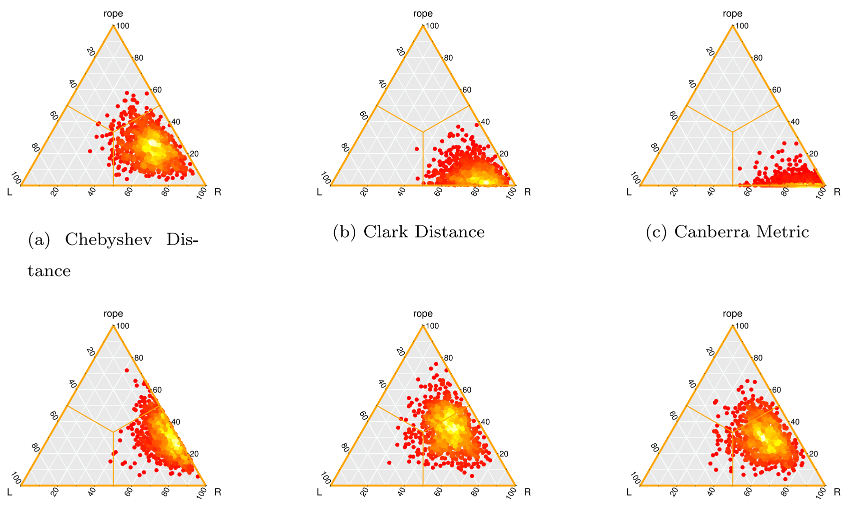
The Natural Scene dataset is collected from 2000 natural scene images that have been ranked inconsistently by ten human rankers. A 294-dimensional feature vector extracted in [1] represents each image and is associated with a multi-label selected from 9 possible labels, i.e., sun, sky, water, cloud, mountain, snow, desert, building, and plant. Then rankers are required to rank the relevant labels in descending order of relevance. As expected, the rankings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process [56] is applied to achieve the label distribution.

The Human Gene dataset is much larger than the other datasets used in this experiment. This dataset is collected from biological research on the relationship between human genes and diseases. Each of the 30,542 human genes is represented by the 36 numerical descriptors for a gene sequence proposed in [57].

The following 4 datasets correspond to standard multi-class classification problems, all of them extracted from UCI Machine Learning Repository [58]. Therefore we need a pre-processing step to transform them into LDL type datasets. Since we only have information from the class, we need to transform the classifier scores into accurate multiclass probability estimates following the process described in [59]. A base classifier is fit on the training set of the cross-validation generator and the test set is used for calibration. The probabilities for each of the folds are then averaged for prediction. We have made use of 4 different base classifiers: SVC [60], *k*-NN [31], a decision tree classifier [61] and Gaussian Naive Bayes classifier [62].

The selected multi-class datasets are: Optidigits (Optical Recognition of Handwritten Digits), Semeion (Semeion Handwritten Digit Dataset, where 1593 handwritten digits from around 80 persons were scanned and documented), Ecoli, an E.Coli bacteria protein classification and LED7digit, a simple domain containing 7 boolean attributes, one for each light-emitting diode of a 7-segment display.

Jura and Wq are two multi-target regression datasets collected from the Mulan website [63]. The Jura dataset consists of measurements of concentrations of seven heavy metals (cadmium, cobalt, chromium, copper, nickel, lead, and zinc), recorded at 359 locations in the topsoil of a region of the Swiss Jura. We are interested in the distribution prediction of the concentration of metals that are more expensive to measure (primary variables) using measurements of metals that are cheaper to sample (secondary variables). The Water Quality dataset has 14 target attributes that refer to the relative representation of plant and animal species in Slovenian rivers and 16 input attributes that refer to physical and chemical water quality parameters. LDL algorithms can deal directly with multi-target datasets, the only prerequisite is to normalize the output vector in such a way that the sum is equal to 1.



(a) Chebyshev Distance (b) Clark Distance (c) Canberra Metric

(d) Kullback-Leibler Divergence (e) Cosine Coefficient (f) Intersection Similarity

Fig. 2. Bayesian Sign test comparing BFGS(L) vs. DF-LDL(R).

4.2. *Evaluation measure selection*

We use a set of six measures when comparing different LDL algorithms: Chebyshev Distance, Clark Distance, Canberra Metric and Kullback–Leibler Divergence for distance calculation; Cosine Coefficient and Intersection for Similarity as shown in Table 2. Each of these measures come from a different syntactic family summarized in [64] and are relatively widely used in the related areas. Thus, they can adequately represent the different aspects of the algorithms.

4.3. *Parameters*

The DF-LDL proposal itself is parameter-free, but the AA-kNN algorithm used in the competent fusion method requires a value for k, set to 4 neighbors (we have selected a small value for k in order to provide the most flexible fit, with a low bias). For the SA-BFGS algorithm, the convergence criterion ϵ has been set up to 10^{-5}. Regarding the parameters used for the StructRF and LDLFs algorithms we have kept the same values as those used in the original proposal papers, for LDLFs the maximum number of iterations has been adapted to meet the time constraints. Since StructTree starts from the same base as StructRF, the parameters used are exactly the same except that we use a single classification tree. An overview of all these parameters can be found in Table 3.

5. Results and analysis

This section presents the results of the empirical studies and their analyses. We have differentiated between two types of experiments: first we will compare the results obtained by the base learners with the results obtained by DF-LDL using the same base learner (Section 5.1), then we will compare our DF-LDL proposal with other state-of-the-art LDL algorithms, showing their different strengths (Section 5.2).

In each of the result tables the best outcome is highlighted in bold. The last column is the mean of each row. The best average is also highlighted in bold. As those algorithms have been tested using 10-fcv, the performance is represented using "mean±standard deviation".

5.1. *Evaluation of DF-LDL vs. base learners*

Comparing DF-LDL with the base classifiers SA-BFGS and StructTree we reach the following conclusions:

- The results of the different measures shown in Tables 4 and 5 highlights the best ranking of DF-LDL in the large majority of the datasets and measures.
- The Wilcoxon Signed Ranks test corroborate the significance of the differences between our approach and the base learners. Table 6 includes the outcome of the Wilcoxon test comparing DF-LDL with the base learners. All the hypotheses of equivalence are rejected with small p-values.
- With regard to the Bayesian Sign test, Figs. 1 and 2 graphically represent the difference between using DF-LDL or directly the base learner and its statistical significance in terms of precision. The following heat-maps clearly indicate the significant superiority of DF-LDL, as the computed distributions are always located in the right region.

Special mention about the excellent results obtained with our proposal for the SJAFFE dataset, for which we obtained an outstanding improvement over the base algorithms.

Nevertheless the outcomes obtained for the Clark and Canberra measures on the StructTree method deserve special attention. We note that the values obtained when using these two measures differ greatly from those obtained for the others. To understand this peculiarity we must look previously at the label distribution of each dataset, when this distribution has many zeros in its composition and we train this set with the StructTree algorithm the probability of obtaining label values equal to zero increases, falsifying the total computation of the distances. Therefore these measures are not representative of the quality of the algorithm for the specific case of the StructTree method and the datasets that present this peculiarity.

5.2. *Evaluation of DF-LDL vs. LDL state-of-the-art algorithms*

We will now see how DF-LDL behaves compared to three other state-of-the-art algorithms as SA-BFGS, LDLFs and StructRF. Note than the selected base learner used by DF-LDL for this comparison is StructTree. We reach the following conclusions:

- The results of the different measures shown in Table 7 highlight the best average score of DF-LDL in all cases.

Table 4
Experimental results DF-LDL (using StructTree as base learner) vs. StructTree (mean ± std).

Measure	Method	Datasets					
		Yeast_alpha	Yeast_cdc	Yeast_diau	Yeast_elu	Yeast_heat	Yeast_spo
Cheby↓	StructTree	0.0168 ± 0.0005	0.0204 ± 0.0009	0.0459 ± 0.0015	0.0201 ± 0.0007	0.0518 ± 0.0008	0.0734 ± 0.0032
	DF-LDL	**0.0133 ± 0.0008**	**0.0162 ± 0.0009**	**0.0367 ± 0.0013**	**0.0161 ± 0.0005**	**0.0418 ± 0.0008**	**0.0609 ± 0.0024**
Clark↓	StructTree	0.2615 ± 0.0082	0.2729 ± 0.0123	0.2474 ± 0.0076	0.2476 ± 0.0082	0.2239 ± 0.0035	0.3078 ± 0.0135
	DF-LDL	**0.2084 ± 0.0125**	**0.2158 ± 0.0134**	**0.1990 ± 0.0062**	**0.1977 ± 0.0053**	**0.1813 ± 0.0028**	**0.2575 ± 0.0124**
Can↓	StructTree	0.8506 ± 0.0298	0.8216 ± 0.0380	0.5321 ± 0.0181	0.7302 ± 0.0221	0.4481 ± 0.0058	0.6369 ± 0.0280
	DF-LDL	**0.6772 ± 0.043**	**0.6472 ± 0.0406**	**0.4279 ± 0.0155**	**0.5804 ± 0.0153**	**0.3629 ± 0.0057**	**0.5295 ± 0.0251**
KL↓	StructTree	0.0086 ± 0.0006	0.0113 ± 0.0009	0.0209 ± 0.0015	0.0087 ± 0.0004	0.0200 ± 0.0004	0.0310 ± 0.0030
	DF-LDL	**0.0054 ± 0.0006**	**0.0070 ± 0.0009**	**0.0132 ± 0.0010**	**0.0061 ± 0.0004**	**0.0126 ± 0.0005**	**0.0270 ± 0.0023**
Cos↑	StructTree	0.9916 ± 0.0006	0.9894 ± 0.0008	0.9811 ± 0.0013	0.9907 ± 0.0006	0.9817 ± 0.0005	0.9627 ± 0.0029
	DF-LDL	**0.9947 ± 0.0006**	**0.9934 ± 0.0008**	**0.9878 ± 0.0009**	**0.9941 ± 0.0003**	**0.9880 ± 0.0005**	**0.9746 ± 0.0019**
Inter↑	StructTree	0.9530 ± 0.0016	0.9460 ± 0.0025	0.9262 ± 0.0026	0.9485 ± 0.0015	0.9265 ± 0.0009	0.8948 ± 0.0043
	DF-LDL	**0.9626 ± 0.0024**	**0.9574 ± 0.0026**	**0.9406 ± 0.0023**	**0.9591 ± 0.0011**	**0.9405 ± 0.0010**	**0.9126 ± 0.0038**
Measure	Method	Datasets					
		SJAFFE	SBU_3DFE	Movie	Natural_Scene	Human_Gene	Optdigits
Cheby↓	StructTree	0.1392 ± 0.0263	0.1520 ± 0.0062	0.1294 ± 0.0053	0.3581 ± 0.0216	0.0828 ± 0.0143	0.1214 ± 0.0073
	DF-LDL	**0.1044 ± 0.0160**	**0.1198 ± 0.0064**	**0.1123 ± 0.0050**	**0.2672 ± 0.0117**	**0.0499 ± 0.0039**	**0.0874 ± 0.0040**
Clark↓	StructTree	0.4759 ± 0.0691	0.4515 ± 0.0160	0.5649 ± 0.0222	**1.6344 ± 0.0350**	2.5795 ± 0.1732	**0.7966 ± 0.0170**
	DF-LDL	**0.3792 ± 0.0527**	**0.3783 ± 0.0188**	**0.5081 ± 0.0267**	2.3719 ± 0.0221	**2.0755 ± 0.0433**	1.0088 ± 0.0199
Can↓	StructTree	0.9775 ± 0.1426	0.9084 ± 0.0314	1.0954 ± 0.0456	**3.7282 ± 0.1147**	17.9475 ± 1.3568	**1.7797 ± 0.0344**
	DF-LDL	**0.7839 ± 0.1088**	**0.7881 ± 0.0378**	**0.9737 ± 0.0515**	6.3743 ± 0.0934	**14.0834 ± 0.7432**	2.5748 ± 0.0573
KL↓	StructTree	0.1084 ± 0.0330	0.1157 ± 0.0080	0.1322 ± 0.0098	2.1534 ± 0.1533	0.3896 ± 0.0646	0.3063 ± 0.0255
	DF-LDL	**0.0589 ± 0.0145**	**0.0705 ± 0.0064**	**0.0961 ± 0.0072**	**0.6300 ± 0.0265**	**0.2289 ± 0.0239**	**0.0827 ± 0.0053**
Cos↑	StructTree	0.8973 ± 0.0311	0.8857 ± 0.0075	0.9154 ± 0.0063	0.6487 ± 0.0222	0.7342 ± 0.0374	0.9166 ± 0.0086
	DF-LDL	**0.9438 ± 0.0139**	**0.9298 ± 0.0065**	**0.9367 ± 0.0043**	**0.7889 ± 0.0098**	**0.8497 ± 0.0167**	**0.9837 ± 0.0021**
Inter↑	StructTree	0.8254 ± 0.0286	0.8268 ± 0.0065	0.8157 ± 0.0075	0.5409 ± 0.0203	0.7143 ± 0.0267	0.8698 ± 0.0073
	DF-LDL	**0.8660 ± 0.0192**	**0.8568 ± 0.0072**	**0.8397 ± 0.0069**	**0.6061 ± 0.0102**	**0.8009 ± 0.0123**	**0.9035 ± 0.0042**
Measure	Method	Datasets					
		Semeion	Ecoli	LED7digit	Wq	Jura	**Average**
Cheby↓	StructTree	0.2355 ± 0.0193	0.0953 ± 0.0378	0.0568 ± 0.0141	0.3465 ± 0.0230	0.0886 ± 0.0162	0.1196 ± 0.0117
	DF-LDL	**0.1799 ± 0.0079**	**0.0749 ± 0.0272**	**0.0503 ± 0.0087**	**0.2840 ± 0.0171**	**0.0791 ± 0.0144**	**0.0938 ± 0.0076**
Clark↓	StructTree	**1.0923 ± 0.0463**	0.4596 ± 0.0879	**0.3028 ± 0.0577**	**2.6063 ± 0.0596**	0.2998 ± 0.0353	**0.7544 ± 0.0396**
	DF-LDL	1.2406 ± 0.0286	**0.4245 ± 0.069**	0.3040 ± 0.0477	3.1031 ± 0.0425	**0.2768 ± 0.0255**	0.7842 ± 0.0264
Can↓	StructTree	**2.5888 ± 0.1278**	0.7740 ± 0.1648	**0.6971 ± 0.1468**	**7.6999 ± 0.2775**	0.4280 ± 0.0529	**2.5085 ± 0.1551**
	DF-LDL	3.3518 ± 0.0929	**0.7230 ± 0.1348**	0.7093 ± 0.1240	10.6209 ± 0.2191	**0.3829 ± 0.0399**	2.6230 ± 0.1087
KL↓	StructTree	0.5342 ± 0.0542	0.1022 ± 0.0692	0.0555 ± 0.0272	1.0220 ± 0.0500	0.0336 ± 0.0101	0.2973 ± 0.0301
	DF-LDL	**0.1756 ± 0.0125**	**0.0502 ± 0.0336**	**0.0305 ± 0.0075**	**0.9255 ± 0.0413**	**0.0266 ± 0.0063**	**0.1439 ± 0.0112**
Cos↑	StructTree	0.8091 ± 0.0177	0.9572 ± 0.0331	0.9760 ± 0.0109	0.5178 ± 0.0230	0.9809 ± 0.0070	0.8904 ± 0.0124
	DF-LDL	**0.9468 ± 0.0077**	**0.9785 ± 0.0181**	**0.9844 ± 0.0052**	**0.6672 ± 0.0091**	**0.9848 ± 0.0053**	**0.9369 ± 0.0061**
Inter↑	StructTree	0.7344 ± 0.0206	0.9013 ± 0.0403	0.9343 ± 0.0157	0.4081 ± 0.0180	0.9114 ± 0.0162	0.8281 ± 0.0130
	DF-LDL	**0.7948 ± 0.0095**	**0.9220 ± 0.0292**	**0.9423 ± 0.0102**	**0.4828 ± 0.0134**	**0.9209 ± 0.0144**	**0.8593 ± 0.0088**

- Table 8 includes the outcome of the Friedman rank and Holm tests in relation to the obtained results over the computed measures. DF-LDL is ranked first compared to SA-BFGS, LDLFs and StructRF. Although DF-LDL has a better ranking than StructRF, both yield very competitive results.
- Note also that the results vary considerably depending on the dataset used. LDLFs proves to be very effective on the Yeast datasets but decreases in strength on the other sets. In this experiment, DF-LDL uses the same base algorithm than StructRF, so when the number of learners used by DF-LDL is much lower than the used by StructRF, this last one provides a better accuracy (remember that the number of learners is related to the number of output labels in DF-LDL while it remains a fix value in the case of StructRF).

Another measure that is of interest to compare is the execution time. At this point we will focus only on the two proposals that have achieved the best results. In our experiment, DF-LDL uses the same base algorithm than StructRF, Struct Tree. Hence the computational complexity of the training phase is similar for both methods: $O(m^2 \times q \times trees)$ in StructRF and $O(m^2 \times q \times c(c-1))$ in DF-LDL. If $c(c-1)$ is less than the number of trees used in StructRF, our proposal DF-LDL will be faster to train and vice versa. As for the prediction, the computational complexity for StructRF is $O(q \times trees)$. For DF-LDL it is also necessary to add the time of the previous prediction AA-KNN, resulting in a complexity of $O(q \times c(c-1)/2 + mq)$. We have two factors that can make one algorithm more efficient predicting than the other. On the one hand, if $c(c-1)/2$ is less than the number of trees used in StructRF, our proposal DF-LDL will use fewer prediction trees and vice versa. But in the case of DF-LDL it is also necessary to add the time of the previous prediction AA-KNN.

We can see an overview of the runtimes on each dataset in Table 9 that corroborate this fact, obtaining prediction times that improve StructRF in the vast majority of cases.

6. Conclusion

In this paper, we proposed a novel decomposition strategy that adapts to LDL constraints. We have based the design of this algorithm on techniques like OVO that have already demonstrated their potential. The DF-LDL algorithm can use any of the already existing LDL learners as base to build a stronger classifier. In addition, the developed fusion

Table 5
Experimental results DF-LDL (using SA-BFGS as base learner) vs. SA-BFGS (mean ± std).

Measure	Method	Datasets					
		Yeast_alpha	Yeast_cdc	Yeast_diau	Yeast_elu	Yeast_heat	Yeast_spo
Cheby↓	SA-BFGS	0.0135 ± 0.0008	0.0163 ± 0.0009	0.0370 ± 0.0014	**0.0163 ± 0.0006**	0.0423 ± 0.0008	**0.0584 ± 0.0037**
	DF-LDL	**0.0134 ± 0.0008**	**0.0162 ± 0.0009**	**0.0369 ± 0.0012**	**0.0163 ± 0.0005**	**0.0421 ± 0.0007**	0.0603 ± 0.0028
Clark↓	SA-BFGS	0.2107 ± 0.0126	**0.2165 ± 0.0129**	0.2008 ± 0.0079	0.1992 ± 0.0055	0.1828 ± 0.0027	**0.2500 ± 0.0164**
	DF-LDL	**0.2103 ± 0.0127**	0.2166 ± 0.0129	**0.2001 ± 0.0067**	**0.1989 ± 0.0053**	**0.1819 ± 0.0025**	0.2552 ± 0.0132
Can↓	SA-BFGS	0.6847 ± 0.0432	**0.6493 ± 0.0394**	0.4310 ± 0.0186	0.5838 ± 0.0159	0.3647 ± 0.0058	**0.5137 ± 0.0335**
	DF-LDL	**0.6832 ± 0.0435**	0.6501 ± 0.0393	**0.4298 ± 0.0166**	**0.5831 ± 0.0157**	**0.3631 ± 0.0057**	0.5246 ± 0.0269
KL↓	SA-BFGS	**0.0055 ± 0.0006**	**0.0070 ± 0.0008**	**0.0131 ± 0.0011**	**0.0062 ± 0.0004**	**0.0126 ± 0.0004**	**0.0246 ± 0.0029**
	DF-LDL	**0.0055 ± 0.0006**	**0.0070 ± 0.0008**	**0.0131 ± 0.0011**	**0.0062 ± 0.0004**	**0.0126 ± 0.0004**	0.0260 ± 0.0025
Cos↑	SA-BFGS	**0.9946 ± 0.0006**	**0.9933 ± 0.0007**	**0.9879 ± 0.0010**	**0.9940 ± 0.0004**	0.9880 ± 0.0004	**0.9769 ± 0.0026**
	DF-LDL	**0.9946 ± 0.0006**	**0.9933 ± 0.0007**	**0.9879 ± 0.0010**	**0.9940 ± 0.0004**	**0.9881 ± 0.0004**	0.9755 ± 0.0021
Inter↑	SA-BFGS	0.9622 ± 0.0024	**0.9573 ± 0.0026**	0.9403 ± 0.0027	0.9588 ± 0.0011	0.9401 ± 0.0010	**0.9154 ± 0.0053**
	DF-LDL	**0.9623 ± 0.0024**	0.9572 ± 0.0025	**0.9404 ± 0.0024**	**0.9589 ± 0.0011**	**0.9404 ± 0.0010**	0.9135 ± 0.0041
Measure	Method	Datasets					
		SJAFFE	SBU_3DFE	Movie	Natural_Scene	Human_Gene	Optdigits
Cheby↓	SA-BFGS	0.1603 ± 0.0160	**0.1100 ± 0.0039**	0.1398 ± 0.0161	0.3549 ± 0.0159	0.0533 ± 0.0036	0.0744 ± 0.0032
	DF-LDL	**0.1007 ± 0.0126**	0.1110 ± 0.0004	**0.1309 ± 0.0101**	**0.3359 ± 0.0159**	**0.0523 ± 0.0041**	**0.0675 ± 0.0028**
Clark↓	SA-BFGS	0.6466 ± 0.0462	0.3784 ± 0.0122	0.6035 ± 0.0560	2.3817 ± 0.0239	2.1111 ± 0.0820	1.6751 ± 0.0167
	DF-LDL	**0.3787 ± 0.0416**	**0.3667 ± 0.0106**	**0.5635 ± 0.0390**	**2.3824 ± 0.0239**	**2.0906 ± 0.0711**	**1.4124 ± 0.0176**
Can↓	SA-BFGS	1.3595 ± 0.1100	0.7831 ± 0.0246	1.1679 ± 0.1163	6.5546 ± 0.0914	14.4531 ± 0.6124	4.4333 ± 0.0553
	DF-LDL	**0.7776 ± 0.0798**	**0.7593 ± 0.0186**	**1.0846 ± 0.0785**	**6.5532 ± 0.0914**	**14.3439 ± 0.7324**	**3.7289 ± 0.0544**
KL↓	SA-BFGS	0.1639 ± 0.0245	0.0634 ± 0.0038	0.1543 ± 0.0405	1.1120 ± 0.0999	0.2365 ± 0.0184	0.0896 ± 0.0048
	DF-LDL	**0.0583 ± 0.0117**	**0.0622 ± 0.0035**	**0.1266 ± 0.0203**	**1.0047 ± 0.0999**	**0.2301 ± 0.0135**	**0.0702 ± 0.0034**
Cos↑	SA-BFGS	0.8739 ± 0.0179	**0.9389 ± 0.0034**	0.9072 ± 0.0188	**0.6724 ± 0.0149**	0.8343 ± 0.0102	0.9831 ± 0.0019
	DF-LDL	**0.9454 ± 0.0111**	0.9385 ± 0.0035	**0.9173 ± 0.0120**	0.6711 ± 0.0149	**0.8399 ± 0.0099**	**0.9873 ± 0.0016**
Inter↑	SA-BFGS	0.7788 ± 0.0162	0.8616 ± 0.0041	0.8040 ± 0.0199	**0.5248 ± 0.0145**	0.7842 ± 0.0092	0.9142 ± 0.0035
	DF-LDL	**0.8682 ± 0.0131**	**0.8637 ± 0.0036**	**0.8163 ± 0.0140**	**0.5248 ± 0.0145**	**0.7898 ± 0.0081**	**0.9223 ± 0.0031**
Measure	Method	Datasets					
		Semeion	Ecoli	LED7digit	Wq	Jura	**Average**
Cheby↓	SA-BFGS	**0.1368 ± 0.0084**	0.0875 ± 0.0188	0.0981 ± 0.0086	0.3026 ± 0.0223	0.0732 ± 0.0046	0.1044 ± 0.0076
	DF-LDL	0.1462 ± 0.0062	**0.0753 ± 0.0156**	**0.0754 ± 0.0084**	**0.3006 ± 0.0216**	**0.0728 ± 0.0057**	**0.0973 ± 0.0067**
Clark↓	SA-BFGS	1.4282 ± 0.0208	0.8979 ± 0.0439	1.2670 ± 0.0419	**3.1255 ± 0.0427**	0.2592 ± 0.0248	0.9432 ± 0.0276
	DF-LDL	**1.2428 ± 0.0185**	**0.7059 ± 0.0372**	**0.8589 ± 0.0280**	3.1257 ± 0.0422	**0.2540 ± 0.0243**	**0.8614 ± 0.0240**
Can↓	SA-BFGS	3.7601 ± 0.0579	1.6400 ± 0.0938	3.2747 ± 0.0991	10.7824 ± 0.2188	0.3674 ± 0.0308	3.0473 ± 0.0980
	DF-LDL	**3.2911 ± 0.0616**	**1.2878 ± 0.0804**	**2.2037 ± 0.0652**	**10.7794 ± 0.2164**	**0.3616 ± 0.0303**	**2.8474 ± 0.0975**
KL↓	SA-BFGS	0.1652 ± 0.0132	0.0730 ± 0.0254	0.0932 ± 0.0145	1.0540 ± 0.0714	0.0231 ± 0.0035	0.1940 ± 0.0192
	DF-LDL	**0.1460 ± 0.0093**	**0.0466 ± 0.0124**	**0.0538 ± 0.0108**	**1.0266 ± 0.0604**	**0.0226 ± 0.0036**	**0.1717 ± 0.0150**
Cos↑	SA-BFGS	0.9512 ± 0.0078	0.9847 ± 0.0078	0.9732 ± 0.0084	0.6248 ± 0.0134	**0.9875 ± 0.0023**	0.9215 ± 0.0066
	DF-LDL	**0.9574 ± 0.0053**	**0.9889 ± 0.0055**	**0.9806 ± 0.0064**	**0.6300 ± 0.0128**	**0.9875 ± 0.0026**	**0.9281 ± 0.0053**
Inter↑	SA-BFGS	**0.8330 ± 0.0095**	0.9073 ± 0.0199	0.8737 ± 0.0117	0.4431 ± 0.0170	**0.9628 ± 0.0046**	0.8448 ± 0.0085
	DF-LDL	0.8270 ± 0.0067	**0.9211 ± 0.0162**	**0.9041 ± 0.0102**	**0.4490 ± 0.0166**	0.9272 ± 0.0057	**0.8521 ± 0.0074**

Table 6
Wilcoxon Signed Ranks test comparing DF-LDL vs. base learners.

Measure	DF-LDL vs. StructTree			DF-LDL vs. SA-BFGS		
	R^+	R^-	p-value	R^+	R^-	p-value
Cheby	153	0	1.5258×10^{-5}	109.5	26.5	0.0313
Clark	93	60	≥ 0.2	135.5	17.5	0.0035
Can	94	59	≥ 0.2	142	11	0.0001
KL	153	0	1.5258×10^{-5}	124	12	0.0021
Cos	153	0	1.5258×10^{-5}	110	26	0.0290
Inter	153	0	1.5258×10^{-5}	102.5	33.5	0.0786

method allows us to combine the outputs in a way that discards the less competent classifiers.

In order to verify the effectiveness of the solution designed, it has been compared, firstly, with the base learners where we have demonstrated a clear superiority of DF-LDL over practically all the datasets and measures used, and secondly, with the state-of-the-art learner in the LDL scope where DF-LDL achieves improvements in many of the cases.

We also want to highlight the performance improvement obtained in prediction times with respect to other multiple learning approaches thanks to the fusion method devised that only makes use of the most competent classifiers for each case.

As future work we want to extend the current proposal with some ideas that have emerged during the course of this study. We can anticipate some of them such as the following:

- DF-LDL can be considered an LDL-oriented framework, compatible with any LDL learning algorithm. In this study, we have experimented with two different LDL base learners but in future work we would like to complete the analysis with further ones. For instance, the LDLogitBoost [19] and the Label Distribution Learning Based on ensemble neural networks [32] are two ensemble proposals from which we can extract the underlying base classifier and use it in our DF-LDL framework.
- DF-LDL needs to train a large number of learners, especially when the number of output labels is high. An interesting approach could be to design a decomposition strategy based on ECOC [40] in order to perform the decomposition to fit better the data using a small number of classifiers.

Table 7
Experimental results DF-LDL vs. LDL state-of-the-art algorithms (mean ± std).

Measure	Method	Datasets					
		Yeast_alpha	Yeast_cdc	Yeast_diau	Yeast_elu	Yeast_heat	Yeast_spo
Cheby↓	SA-BFGS	0.0135 ± 0.0008	0.0163 ± 0.0009	0.0370 ± 0.0014	0.0163 ± 0.0006	0.0423 ± 0.0008	0.0584 ± 0.0037
	LDLFs	**0.0129 ± 0.0003**	**0.0159 ± 0.0007**	**0.0344 ± 0.0017**	**0.0156 ± 0.0005**	**0.0392 ± 0.0011**	**0.0561 ± 0.0020**
	StructRF	0.0134 ± 0.0008	0.0162 ± 0.0009	0.0359 ± 0.0015	0.0161 ± 0.0005	0.0407 ± 0.0009	0.0578 ± 0.0030
	DF-LDL	0.0133 ± 0.0008	0.0162 ± 0.0009	0.0367 ± 0.0013	0.0161 ± 0.0005	0.0418 ± 0.0008	0.0609 ± 0.0024
Clark↓	SA-BFGS	0.2107 ± 0.0126	0.2165 ± 0.0129	0.2008 ± 0.0079	0.1992 ± 0.0055	0.1828 ± 0.0027	0.2500 ± 0.0164
	LDLFs	**0.2021 ± 0.0031**	**0.2126 ± 0.0065**	**0.1872 ± 0.0112**	**0.1908 ± 0.0052**	**0.1706 ± 0.0040**	**0.2427 ± 0.0082**
	StructRF	0.2095 ± 0.0125	0.2158 ± 0.0134	0.1947 ± 0.0076	0.1974 ± 0.0054	0.1770 ± 0.0028	0.2467 ± 0.0139
	DF-LDL	0.2084 ± 0.0125	0.2158 ± 0.0134	0.1990 ± 0.0062	0.1977 ± 0.0053	0.1813 ± 0.0028	0.2575 ± 0.0124
Can↓	SA-BFGS	0.6847 ± 0.0432	0.6493 ± 0.0394	0.4310 ± 0.0186	0.5838 ± 0.0159	0.3647 ± 0.0058	0.5137 ± 0.0335
	LDLFs	**0.6555 ± 0.0099**	**0.6385 ± 0.0146**	**0.4011 ± 0.0221**	**0.5595 ± 0.0148**	**0.3408 ± 0.0076**	**0.4969 ± 0.0173**
	StructRF	0.6812 ± 0.0434	0.6472 ± 0.0410	0.4177 ± 0.0178	0.5780 ± 0.0156	0.3541 ± 0.0063	0.5066 ± 0.0274
	DF-LDL	0.6772 ± 0.043	0.6472 ± 0.0406	0.4279 ± 0.0155	0.5804 ± 0.0153	0.3629 ± 0.0057	0.5295 ± 0.0251
KL↓	SA-BFGS	0.0055 ± 0.0006	0.0070 ± 0.0008	0.0131 ± 0.0011	0.0062 ± 0.0004	0.0126 ± 0.0004	0.0246 ± 0.0029
	LDLFs	**0.0051 ± 0.0002**	**0.0067 ± 0.0004**	**0.0115 ± 0.0013**	**0.0057 ± 0.0003**	**0.0111 ± 0.0005**	**0.0231 ± 0.0016**
	StructRF	0.0055 ± 0.0006	0.0070 ± 0.0009	0.0125 ± 0.0010	0.0061 ± 0.0004	0.0120 ± 0.0004	0.0243 ± 0.0024
	DF-LDL	0.0054 ± 0.0006	0.0070 ± 0.0009	0.0132 ± 0.0010	0.0061 ± 0.0004	0.0126 ± 0.0005	0.0270 ± 0.0023
Cos↑	SA-BFGS	0.9946 ± 0.0006	0.9933 ± 0.0007	0.9879 ± 0.0010	0.9940 ± 0.0004	0.9880 ± 0.0004	0.9769 ± 0.0026
	LDLFs	**0.9950 ± 0.0001**	**0.9935 ± 0.0003**	**0.9895 ± 0.0011**	**0.9945 ± 0.0003**	**0.9894 ± 0.0004**	**0.9786 ± 0.0014**
	StructRF	0.9946 ± 0.0006	0.9933 ± 0.0008	0.9885 ± 0.0010	0.9941 ± 0.0003	0.9886 ± 0.0004	0.9773 ± 0.0021
	DF-LDL	0.9947 ± 0.0006	0.9934 ± 0.0008	0.9878 ± 0.0009	0.9941 ± 0.0003	0.9880 ± 0.0005	0.9746 ± 0.0019
Inter↑	SA-BFGS	0.9622 ± 0.0024	0.9573 ± 0.0026	0.9403 ± 0.0027	0.9588 ± 0.0011	0.9401 ± 0.0010	0.9154 ± 0.0053
	LDLFs	**0.9638 ± 0.0006**	**0.9580 ± 0.0009**	**0.9445 ± 0.0028**	**0.9606 ± 0.0010**	**0.9441 ± 0.0012**	**0.9186 ± 0.0028**
	StructRF	0.9624 ± 0.0024	0.9574 ± 0.0027	0.9421 ± 0.0026	0.9592 ± 0.0011	0.9419 ± 0.0011	0.9167 ± 0.0042
	DF-LDL	0.9626 ± 0.0024	0.9574 ± 0.0026	0.9406 ± 0.0023	0.9591 ± 0.0011	0.9405 ± 0.0010	0.9126 ± 0.0038
Measure	Method	Datasets					
		SJAFFE	SBU_3DFE	Movie	Natural_Scene	Human_Gene	Optdigits
Cheby↓	SA-BFGS	0.1603 ± 0.0160	**0.1100 ± 0.0039**	0.1398 ± 0.0161	0.3549 ± 0.0159	0.0533 ± 0.0036	**0.0744 ± 0.0032**
	LDLFs	0.1158 ± 0.0153	0.1302 ± 0.0046	0.1170 ± 0.0033	0.3604 ± 0.0260	0.0531 ± 0.0159	0.6270 ± 0.0129
	StructRF	0.1094 ± 0.0108	0.1183 ± 0.0058	**0.1104 ± 0.0048**	0.2738 ± 0.0116	0.0553 ± 0.0048	0.1034 ± 0.0048
	DF-LDL	**0.1044 ± 0.0160**	0.1198 ± 0.0064	0.1123 ± 0.0050	**0.2672 ± 0.0117**	**0.0499 ± 0.0039**	0.0874 ± 0.0040
Clark↓	SA-BFGS	0.6466 ± 0.0462	0.3784 ± 0.0122	0.6035 ± 0.0560	2.3817 ± 0.0239	2.1111 ± 0.0820	1.6751 ± 0.0167
	LDLFs	0.4175 ± 0.0361	0.4011 ± 0.0110	0.5269 ± 0.0123	2.4841 ± 0.0300	2.1240 ± 0.0637	2.5616 ± 0.0242
	StructRF	0.3900 ± 0.0350	**0.3680 ± 0.0173**	**0.5039 ± 0.0233**	2.3946 ± 0.0227	2.1776 ± 0.1232	1.0620 ± 0.0225
	DF-LDL	**0.3792 ± 0.0527**	0.3783 ± 0.0188	0.5081 ± 0.0267	**2.3719 ± 0.0221**	**2.0755 ± 0.0433**	**1.0088 ± 0.0199**
Can↓	SA-BFGS	1.3595 ± 0.1100	0.7831 ± 0.0246	1.1679 ± 0.1163	6.5546 ± 0.0914	14.4531 ± 0.6124	4.4333 ± 0.0553
	LDLFs	0.8797 ± 0.0804	0.8610 ± 0.0253	0.9995 ± 0.0233	6.8694 ± 0.1319	14.5737 ± 0.4372	7.8649 ± 0.0948
	StructRF	0.8089 ± 0.0769	**0.7817 ± 0.0354**	**0.9595 ± 0.0441**	6.4559 ± 0.0950	14.8633 ± 0.8857	2.7195 ± 0.0638
	DF-LDL	**0.7839 ± 0.1088**	0.7881 ± 0.0378	0.9737 ± 0.0515	**6.3743 ± 0.0934**	**14.0834 ± 0.7432**	**2.5748 ± 0.0573**
KL↓	SA-BFGS	0.1639 ± 0.0245	0.0634 ± 0.0038	0.1543 ± 0.0405	1.1120 ± 0.0999	0.2365 ± 0.0184	0.0896 ± 0.0048
	LDLFs	0.0707 ± 0.0132	0.0765 ± 0.0045	0.0988 ± 0.0043	0.9453 ± 0.0569	0.2398 ± 0.0719	1.0301 ± 0.0446
	StructRF	0.0603 ± 0.0089	**0.0659 ± 0.0054**	**0.0901 ± 0.0061**	0.6436 ± 0.0211	0.2480 ± 0.0265	0.1056 ± 0.0081
	DF-LDL	**0.0589 ± 0.0145**	0.0705 ± 0.0064	0.0961 ± 0.0072	**0.6300 ± 0.0265**	**0.2289 ± 0.0239**	**0.0827 ± 0.0053**
Cos↑	SA-BFGS	0.8739 ± 0.0179	**0.9389 ± 0.0034**	0.9072 ± 0.0188	0.6724 ± 0.0149	0.8343 ± 0.0102	0.9831 ± 0.0019
	LDLFs	0.9334 ± 0.0125	0.9249 ± 0.0040	0.9347 ± 0.0028	0.6653 ± 0.0185	0.8353 ± 0.0251	0.6879 ± 0.0193
	StructRF	0.9429 ± 0.0083	0.9348 ± 0.0055	**0.9407 ± 0.0039**	**0.7895 ± 0.0091**	0.8202 ± 0.0205	0.9758 ± 0.0035
	DF-LDL	**0.9438 ± 0.0139**	0.9298 ± 0.0065	0.9367 ± 0.0043	0.7889 ± 0.0098	**0.8497 ± 0.0167**	**0.9837 ± 0.0021**
Inter↑	SA-BFGS	0.7788 ± 0.0162	**0.8616 ± 0.0041**	0.8040 ± 0.0199	0.5248 ± 0.0145	0.7842 ± 0.0092	**0.9142 ± 0.0035**
	LDLFs	0.8501 ± 0.0151	0.8450 ± 0.0046	0.8350 ± 0.0040	0.4605 ± 0.0220	0.7833 ± 0.0235	0.3619 ± 0.0131
	StructRF	0.8622 ± 0.0132	0.8592 ± 0.0065	**0.8432 ± 0.0060**	0.5919 ± 0.0097	0.7739 ± 0.0159	0.8871 ± 0.0050
	DF-LDL	**0.8660 ± 0.0192**	0.8568 ± 0.0072	0.8397 ± 0.0069	**0.6061 ± 0.0102**	**0.8009 ± 0.0123**	0.9035 ± 0.0042
Measure	Method	Datasets					
		Semeion	Ecoli	LED7digit	Wq	Jura	**Average**
Cheby↓	SA-BFGS	**0.1368 ± 0.0084**	0.0875 ± 0.0188	0.0981 ± 0.0086	0.3026 ± 0.0223	0.0732 ± 0.0046	0.1044 ± 0.0076
	LDLFs	0.3569 ± 0.0459	0.4888 ± 0.0329	0.4719 ± 0.0461	0.3065 ± 0.0116	0.1115 ± 0.0168	0.1949 ± 0.0140
	StructRF	0.2099 ± 0.0042	**0.0716 ± 0.0240**	0.0532 ± 0.0114	0.2859 ± 0.0185	**0.0689 ± 0.0090**	0.0965 ± 0,0069
	DF-LDL	0.1799 ± 0.0079	0.0749 ± 0.0272	**0.0503 ± 0.0087**	**0.2840 ± 0.0171**	0.0791 ± 0.0144	**0.0938 ± 0.0076**
Clark↓	SA-BFGS	1.4282 ± 0.0208	0.8979 ± 0.0439	1.2670 ± 0.0419	3.1255 ± 0.0427	0.2592 ± 0.0248	0.9432 ± 0.0276
	LDLFs	1.6710 ± 0.0889	1.5454 ± 0.0502	1.9121 ± 0.1527	3.1115 ± 0.0295	0.3429 ± 0.0348	1.0767 ± 0.0336
	StructRF	1.3193 ± 0.0197	0.4650 ± 0.0748	0.3293 ± 0.0591	**3.1006 ± 0.0443**	**0.2416 ± 0.0216**	0.7996 ± 0.0305
	DF-LDL	**1.2406 ± 0.0286**	**0.4245 ± 0.069**	**0.3040 ± 0.0477**	3.1031 ± 0.0425	0.2768 ± 0.0255	**0.7842 ± 0.0264**
Can↓	SA-BFGS	3.7601 ± 0.0579	1.6400 ± 0.0938	3.2747 ± 0.0991	10.7824 ± 0.2188	0.3674 ± 0.0308	3.0473 ± 0.0980
	LDLFs	4.7213 ± 0.3042	3.2622 ± 0.1264	5.5698 ± 0.5812	10.7510 ± 0.1516	0.5034 ± 0.0588	3.5264 ± 0.1236
	StructRF	3.5802 ± 0.0661	0.7990 ± 0.1509	0.7832 ± 0.156	10.6299 ± 0.2312	**0.3406 ± 0.0309**	2.7004 ± 0.1169
	DF-LDL	**3.3518 ± 0.0929**	**0.7230 ± 0.1348**	**0.7093 ± 0.1240**	**10.6209 ± 0.2191**	0.3829 ± 0.0399	**2.6230 ± 0.1087**

(*continued on next page*)

Table 7 (*continued*).

Measure	Method	Yeast_alpha	Yeast_cdc	Yeast_diau	Yeast_elu	Yeast_heat	Yeast_spo
KL↓	SA-BFGS	**0.1652 ± 0.0132**	0.0730 ± 0.0254	0.0932 ± 0.0145	1.0540 ± 0.0714	0.0231 ± 0.0035	0.1940 ± 0.0192
	LDLFs	0.4456 ± 0.0874	0.7172 ± 0.0748	0.7356 ± 0.0945	1.1061 ± 0.0372	0.0435 ± 0.0099	0.3278 ± 0.0296
	StructRF	0.2135 ± 0.0095	**0.0448 ± 0.0267**	0.0326 ± 0.0095	0.9391 ± 0.0406	**0.0206 ± 0.0031**	0.1489 ± 0.0101
	DF-LDL	0.1756 ± 0.0125	0.0502 ± 0.0336	**0.0305 ± 0.0075**	**0.9255 ± 0.0413**	0.0266 ± 0.0063	**0.1439 ± 0.0112**
Cos↑	SA-BFGS	**0.9512 ± 0.0078**	**0.9847 ± 0.0078**	0.9732 ± 0.0084	0.6248 ± 0.0134	0.9875 ± 0.0023	0.9215 ± 0.0066
	LDLFs	0.8451 ± 0.0406	0.6986 ± 0.0341	0.6735 ± 0.0359	0.5887 ± 0.0104	0.9744 ± 0.0076	0.8648 ± 0.0126
	StructRF	0.9316 ± 0.0065	0.9831 ± 0.0141	0.9835 ± 0.0053	0.6634 ± 0.0101	**0.9880 ± 0.0033**	0.9347 ± 0.0056
	DF-LDL	0.9468 ± 0.0077	0.9785 ± 0.0181	**0.9844 ± 0.0052**	**0.6672 ± 0.0091**	0.9848 ± 0.0053	**0.9369 ± 0.0061**
Inter↑	SA-BFGS	**0.8330 ± 0.0095**	0.9073 ± 0.0199	0.8737 ± 0.0117	0.4431 ± 0.0170	**0.9628 ± 0.0046**	0.8448 ± 0.0085
	LDLFs	0.6098 ± 0.0480	0.4876 ± 0.0315	0.4854 ± 0.0451	0.3999 ± 0.0118	0.8885 ± 0.0168	0.7469 ± 0.0144
	StructRF	0.7656 ± 0.0062	**0.9251 ± 0.0026**	0.9389 ± 0.0132	0.4745 ± 0.0149	0.9311 ± 0.0090	0.8548 ± 0.0082
	DF-LDL	0.7948 ± 0.0095	0.9220 ± 0.0292	**0.9423 ± 0.0102**	**0.4828 ± 0.0134**	0.9209 ± 0.0144	**0.8593 ± 0.0088**

Table 8
Friedman rank and Holm test applied to the results among the tested algorithms.

Measure	Control method: DF-LDL (2.1176)				Measure	Control method: DF-LDL (1.9706)			
	i	Algorithm (Rank)	Z	*p*-value		i	Algorithm (Rank)	Z	*p*-value
Cheby	3	SA-BFGS (3)	1.9926	0.0463	Clark	3	SA-BFGS (3.2941)	2.9889	0.0028
	2	LDLFs (2.7059)	1.3284	0.1840		2	LDLFs (2.7059)	1.6605	0.0968
	1	StructRF (2.1765)	0.1328	0.8943		1	StructRF (2.0294)	0.1328	0.8943
Measure	Control method: DF-LDL (1.9706)				Measure	Control method: DF-LDL (2.1765)			
	i	Algorithm (Rank)	Z	*p*-value		i	Algorithm (Rank)	Z	*p*-value
Can	3	SA-BFGS (3.2941)	2.9889	0.0028	KL	3	SA-BFGS (2.8824)	1.5941	0.1109
	2	LDLFs (2.7059)	1.6605	0.0968		2	LDLFs (2.7059)	1.1956	0.2319
	1	StructRF (2.0294)	0.1328	0.8943		1	StructRF (2.2353)	0.1328	0.8943
Measure	Control method: DF-LDL (2.2353)				Measure	Control method: DF-LDL (2.1471)			
	i	Algorithm (Rank)	Z	*p*-value		i	Algorithm (Rank)	Z	*p*-value
Cos	3	SA-BFGS (2.7941)	1.2620	0.2069	Inter	3	SA-BFGS (2.8824)	1.6605	0.0968
	2	LDLFs (2.7059)	1.0627	0.2879		2	LDLFs (2.7647)	1.3948	0.1631
	1	StructRF (2.2647)	0.0664	0.9470		1	StructRF (2.2059)	0.1328	0.8943

Table 9
Execution times (in seconds), measured on a machine with®Intel Core i7-7300HQ processor (4 cores, 6 MB cache, 2.5 GHz–3.5 GHz) and 16 GB DDR4 2400 MHz RAM.

Datasets	Fit time		Prediction time	
	StructRF	DF-LDL	StructRF	DF-LDL
Yeast_alpha	**574.35**	4709.50	**0.10**	0.52
Yeast_cdc	**736.34**	2196.00	**0,12**	0,29
Yeast_diau	717.97	**421.10**	0.11	**0,09**
Yeast_elu	**712.89**	1365.01	**0.11**	0,24
Yeast_heat	710.97	**203.28**	0.10	**0.06**
Yeast_spo	522.11	**201.99**	0.10	**0.06**
SJAFFE	143.77	**86.00**	0.05	**0.01**
SBU_3DFE	2272.50	**1370.02**	0.07	**0.04**
Movie	10413.56	**4160.43**	0.39	**0.20**
Natural_Scene	**2866.39**	4122.04	0.08	**0.06**
Human_Gene	**4317.67**	390,365.00	**0.78**	35.34
Optdigits	**1432.00**	2335.03	0.24	**0.22**
Semeion	**630.05**	1161.71	0.06	**0.05**
Ecoli	42.28	**11.52**	0.008	**0.006**
LED7digit	**2866.38**	5058.16	0.08	**0.07**
Wq	**306.38**	717.15	**0.03**	0.06
Jura	54.53	**4.23**	0.007	**0.001**

CRediT authorship contribution statement

Manuel González: Conceptualization, Methodology, Software, Investigation, Writing - review & editing. **Germán González-Almagro:** Software, Writing - review. **Isaac Triguero:** Writing - review & editing. **José-Ramón Cano:** Supervision, Resources. **Salvador García:** Funding acquisition, Project administration, Supervision.

Declaration of competing interest

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

Acknowledgment

This work is supported by the Spanish National Research Project TIN2017-89517-P.

References

[1] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[2] Z. Barutcuoglu, R.E. Schapire, O.G. Troyanskaya, Hierarchical multi-label prediction of gene function, Bioinformatics 22 (7) (2006) 830–836.
[3] E. Gibaja, S. Ventura, A tutorial on multilabel learning, ACM Comput. Surv. 47 (3) (2015) 1–38.
[4] F. Herrera, F. Charte, A.J. Rivera, M.J. Del Jesus, Multilabel classification, in: Multilabel Classification, Springer, 2016, pp. 17–31.
[5] I. Triguero, C. Vens, Labelling strategies for hierarchical multi-label classification techniques, Pattern Recognit. 56 (2016) 170–183.
[6] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: models, experimental study and prospects, Inf. Fusion 44 (2018) 33–45.
[7] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proc. Natl. Acad. Sci. 95 (25) (1998) 14863–14868.
[8] X. Geng, C. Yin, Z.-H. Zhou, Facial age estimation by learning from label distributions, IEEE Trans. Pattern Anal. Mach. Intell. 35 (10) (2013) 2401–2412.
[9] X. Geng, Label distribution learning, IEEE Trans. Knowl. Data Eng. 28 (7) (2016) 1734–1748.
[10] Y. Ren, X. Geng, Sense beauty by label distribution learning, in: International Joint Conferences on Artificial Intelligence, 2017, pp. 2648–2654.

[11] D. Xue, Z. Hong, S. Guo, L. Gao, L. Wu, J. Zheng, N. Zhao, Personality recognition on social media with label distribution learning, IEEE Access 5 (2017) 13478–13488.

[12] J. Yang, D. She, M. Sun, Joint image emotion classification and distribution learning via deep convolutional neural network., in: International Joint Conferences on Artificial Intelligence, 2017, pp. 3266–3272.

[13] X. Geng, P. Hou, Pre-release prediction of crowd opinion on movies by label distribution learning, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[14] Z. Zhang, M. Wang, X. Geng, Crowd counting in public video surveillance by label distribution learning, Neurocomputing 166 (2015) 151–163.

[15] L. Xu, J. Chen, Y. Gan, Head pose estimation using improved label distribution learning with fewer annotations, Multimedia Tools Appl. (2019) 1–22.

[16] X. Zheng, X. Jia, W. Li, Label distribution learning by exploiting sample correlations locally, in: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, 2018, pp. 4556–4563.

[17] W. Shen, K. Zhao, Y. Guo, A.L. Yuille, Label distribution learning forests, in: Advances in Neural Information Processing Systems, 2017, pp. 834–843.

[18] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, X. Geng, Deep label distribution learning with label ambiguity, IEEE Trans. Image Process. 26 (6) (2017) 2825–2838.

[19] C. Xing, X. Geng, H. Xue, Logistic boosting regression for label distribution learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4489–4497.

[20] M. Chen, X. Wang, B. Feng, W. Liu, Structured random forest for label distribution learning, Neurocomputing 320 (2018) 171–182.

[21] J. Wang, X. Geng, Classification with label distribution learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 3712–3718.

[22] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, Springer, 2018.

[23] R.C. Prati, J. Luengo, F. Herrera, Emerging topics and challenges of learning from noisy data in nonstandard classification: a survey beyond binary class noise, Knowl. Inf. Syst. 60 (1) (2019) 63–97.

[24] R.C. Prati, G.E. Batista, M.C. Monard, Class imbalances versus class overlapping: an analysis of a learning system behavior, in: Mexican International Conference on Artificial Intelligence, Springer, 2004, pp. 312–321.

[25] S. Das, S. Datta, B.B. Chaudhuri, Handling data irregularities in classification: Foundations, trends, and future challenges, Pattern Recognit. 81 (2018) 674–693.

[26] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowl. Inf. Syst. 38 (1) (2014) 179–206.

[27] T. Hastie, R. Tibshirani, Classification by pairwise coupling, in: Advances in Neural Information Processing Systems, 1998, pp. 507–513.

[28] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, J. Mach. Learn. Res. 18 (1) (2017) 2653–2688.

[29] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.

[30] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, J. Mach. Learn. Res. 15 (1) (2014) 3133–3181.

[31] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.

[32] Y. Zhai, J. Dai, H. Shi, Label distribution learning based on ensemble neural networks, in: International Conference on Neural Information Processing, Springer, 2018, pp. 593–602.

[33] P. Kontschieder, M. Fiterau, A. Criminisi, S. Rota Bulo, Deep neural decision forests, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1467–1475.

[34] K. Wang, X. Geng, Binary coding based label distribution learning, in: International Joint Conferences on Artificial Intelligence, 2018, pp. 2783–2789.

[35] K. Wang, X. Geng, Discrete binary coding based label distribution learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 3733–3739.

[36] Y. Wang, J. Dai, Label distribution feature selection based on mutual information in fuzzy rough set theory, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–2.

[37] M. González, J.-R. Cano, S. García, Prolsfeo-ldl: Prototype selection and label-specific feature evolutionary optimization for label distribution learning, Appl. Sci. 10 (9) (2020) 3089.

[38] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.

[39] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (Jan) (2004) 101–141.

[40] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artificial Intelligence Res. 2 (1994) 263–286.

[41] J. Fürnkranz, E. Hüllermeier, S. Vanderlooy, Binary decomposition methods for multipartite ranking, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2009, pp. 359–374.

[42] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, Inf. Fusion 41 (2018) 195–216.

[43] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers, Pattern Recognit. 46 (12) (2013) 3412–3424.

[44] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Nmc: nearest matrix classification–a new combination model for pruning one-vs-one ensembles by transforming the aggregation problem, Inf. Fusion 36 (2017) 26–51.

[45] Z.-L. Zhang, X.-G. Luo, Y. Yu, B.-W. Yuan, J.-F. Tang, Integration of an improved dynamic ensemble selection approach to enhance one-vs-one scheme, Eng. Appl. Artif. Intell. 74 (2018) 43–53.

[46] M. Lachaize, S. Le Hégarat-Mascle, E. Aldea, A. Maitrot, R. Reynaud, Evidential framework for error correcting output code classification, Eng. Appl. Artif. Intell. 73 (2018) 10–21.

[47] J. Bi, C. Zhang, An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme, Knowl.-Based Syst. 158 (2018) 81–93.

[48] L.P. Garcia, J.A. Sáez, J. Luengo, A.C. Lorena, A.C. de Carvalho, F. Herrera, Using the one-vs-one decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems, Knowl.-Based Syst. 90 (2015) 153–164.

[49] J.A. Sáez, M. Galar, B. Krawczyk, Addressing the overlapping data problem in classification using the one-vs-one decomposition strategy, IEEE Access 7 (2019) 83396–83411.

[50] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, Empowering difficult classes with a similarity-based aggregation in multi-class classification problems, Inform. Sci. 264 (2014) 135–157.

[51] I. Triguero, S. González, J.M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera, et al., KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining, Atlantis Press, 2017.

[52] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, Rnpbst: An r package covering non-parametric and bayesian statistical tests, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 281–292.

[53] M. Lyons, S. Akamatsu, M. Kamachi, J. Gyoba, Coding facial expressions with gabor wavelets, in: Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, IEEE, 1998, pp. 200–205.

[54] L. Yin, X. Wei, Y. Sun, J. Wang, M.J. Rosato, A 3d facial expression database for facial behavior research, in: 7th International Conference on Automatic Face and Gesture Recognition (FGR06), IEEE, 2006, pp. 211–216.

[55] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: Application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2037–2041.

[56] X. Geng, L. Luo, Multilabel ranking with inconsistent rankers, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3742–3747.

[57] J.-F. Yu, D.-K. Jiang, K. Xiao, Y. Jin, J.-H. Wang, X. Sun, Discriminate the falsely predicted protein-coding genes in aeropyrum pernix k1 genome based on graphical representation, MATCH Commun. Math. Comput. Chem. 67 (3) (2012) 845.

[58] D. Dua, C. Graff, UCI machine learning repository, 2017, URL http://archive.ics.uci.edu/ml.

[59] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 694–699.

[60] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (3) (2011) 27.

[61] R. Lior, et al., Data Mining with Decision Trees: Theory and Applications, Vol. 81, World scientific, 2014.

[62] H. Zhang, The optimality of Naive Bayes, in: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004, Vol. 2, 2004, pp. 562–567.

[63] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas, Mulan: A java library for multi-label learning, J. Mach. Learn. Res. 12 (Jul) (2011) 2411–2414.

[64] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, Int. J. Math. Models Methods Appl. Sci. 1 (2) (2007) 1.