



Article

ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

Manuel González ^{1,*} , José-Ramón Cano ² and Salvador García ¹ 

¹ Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain; salvagl@decsai.ugr.es

² Department of Computer Science, University of Jaén, EPS of Linares, Avenida de la Universidad S/N, 23700 Linares, Jaén, Spain; jrcano@ujaen.es

* Correspondence: manuel.gonzalez.es@gmail.com

Received: 5 April 2020; Accepted: 26 April 2020; Published: 29 April 2020



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. In LDL problems, instance-based algorithms and particularly the adapted version of the k -nearest neighbors method for LDL (AA- k NN) has proven to be very competitive, achieving acceptable results and allowing an explainable model. However, it suffers from several handicaps: it needs large storage requirements, it is not efficient predicting and presents a low tolerance to noise. The purpose of this paper is to mitigate these effects by adding a data reduction stage. The technique devised, called Prototype selection and Label-Specific Feature Evolutionary Optimization for LDL (ProLSFEO-LDL), is a novel method to simultaneously address the prototype selection and the label-specific feature selection pre-processing techniques. Both techniques pose a complex optimization problem with a huge search space. Therefore, we have proposed a search method based on evolutionary algorithms that allows us to obtain a solution to both problems in a reasonable time. The effectiveness of the proposed ProLSFEO-LDL method is verified on several real-world LDL datasets, showing significant improvements in comparison with using raw datasets.

Keywords: label distribution learning; evolutionary optimization; prototype selection; label-specific feature; machine learning

1. Introduction

A supervised learning process is the machine learning task of training a function that maps an input to an output based on data points with known outputs. Classification is the process of predicting to which of a set of categories a new observation belongs. Thus, the purpose of classification is to achieve a model that will be able to classify the right class to an unknown pattern. However, there are an increasing number of problems where a pattern can have several labels simultaneously associated. Examples can be found in genetics [1], image classification [2], etc. The generalization of the classic classification is Multi-Label Learning (MLL) [3–6], where multiple labels can be assigned to each instance.

Nevertheless, in many real-world problems we may face cases where MLL is still not enough, as the degree of description of each label is different. To appoint only one dataset used in this paper, the biological experiments on yeast genes [7] during a period of time yield different levels of gene expression in a time serie. The exact level of expression at any point in time is of less importance.

What becomes really decisive is the overall expression distribution over the whole period of time. If the learning goal is to predict that distribution for a particular gene, it cannot easily fit into the MLL framework because the role of each output in the distribution is critical, and there is no division of relevant and irrelevant labels at all.

The proposal of Label Distribution Learning (LDL) appeared for first time in 2013 [8] and was formally introduced in 2016 [9] in order to handle ambiguity on the label side of the mapping when one instance is not mandatorily matched to one single label. The main objective of this paradigm is to respond to the question “how much does each label describe the instance?” instead of “which label can describe the instance?”.

Numerous studies have been conducted by applying the LDL framework to diverse real-life problem solving scenarios, e.g., facial age estimation [8], pre-release prediction of crowd opinion on movies [10], crowd counting in public video surveillance [11], sense beauty recognition [12], image emotion classification [13], personality recognition on social media [14], head pose estimation [15], etc. Further research has focused more on the development of new learners or on the adaptation of existing ones 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 Label Distribution [18]), or ensembles strategies (e.g., Logistic boosting regression for LDL [19], Structured random forest for LDL [20]).

AA-*k*NN [9] has proven to be a very competitive algorithm in previous experimental studies, achieving acceptable results and allowing an explainable model [21]. However, like any other instance-based algorithm, it suffers from several handicaps: it needs large memory requirements to store the training set, it is not efficient predicting due to the multiple computations of similarities between the test and training samples and it presents a low tolerance to noise because it uses all the data as relevant.

Nowadays, one of the main challenges of supervised learning algorithms is still how to deal with raw datasets. Data collection is usually an unsupervised process, leading to datasets with redundant information, noisy data or irrelevant features. Hence, data pre-processing is an important step in the data mining process, that can mitigate this type of problem and generate improved datasets. The purpose of this paper is to address the LDL problem from the data pre-processing stage, by applying data reduction techniques which will allow us to have a reduced representation of the original data while maintaining the essential structure and integrity [22]. These pre-processing techniques often result in better performance of subsequent learners.

Two of the most widely used data reductions techniques are the instance selection [23], also known as prototype selection in case of instance-based algorithms [24], and the feature selection or reduction of the data dimensionality [25]. The first technique focuses on finding an optimal subset of samples to optimize the performance of the learner while the main idea of the second technique is to replace the original set of features by a new subset that extract the main information and provide an accurate classification.

A few methods are available to approach the instance selection in the MLL domain [26–28] but, to the best of our knowledge, there are no studies concerning instance or prototype selection reported in LDL. With regard to the feature selection, most of LDL algorithms are built on a simple feature space where all features are used to predict all the labels. However, in real-world applications, an instance is characterized by all labels, but some labels can only be determined by some specific features of their own. Although label-specific feature selection has been widely studied in MLL [29,30], the studies we can find on this subject for LDL are still scarce [31].

Prototype selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning, or from now ProLSFEO-LDL, is our proposal for a pre-processing algorithm adapted to LDL problems and specifically designed to mitigate the handicaps of the AA-*k*NN method. We provide a novel method to simultaneously address the prototype selection and the label-specific feature selection. Both pre-processing techniques can be considered as a search problem where the search space is huge.

Therefore, we have devised a search method based on evolutionary algorithms [32] that allows us to obtain a solution to both problems in a reasonable time. To this end, we have adapted elements of classical evolutionary algorithms to manage LDL restrictions, we have designed a representation of the solution and a way to measure its quality, which allows us to address both problems together.

In order to evaluate the proposal proficiency, we will compare an LDL-learner applying our ProLSFEO-LDL algorithm to the raw training set and measuring six aspects of their performance. We will repeat the experiment over 13 real-world datasets and validate the results of the empirical comparisons using Wilcoxon and Bayesian Sign tests [33–35].

The rest of the paper is organized as follows. First, a brief review and discussion of the foundations of LDL, a description of data reduction techniques and an introduction to the evolutionary optimization process are given in Section 2. The proposed ProLSFEO-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 and the most relevant studies carried out on LDL (Section 2.1), are presented. Furthermore, some basic concepts on instance selection and label-specific features selection for classification are introduced (Section 2.2), as well as the notions needed to optimize solutions using evolutionary algorithms (Section 2.3), providing the necessary background required to properly present the study carried out in this paper.

2.1. Foundations of Label Distribution Learning

We can formalize a 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 $y_j \in Y | j \in \{1, \dots, c\}$, such that $Y = \{y_1, \dots, y_c\}$ denotes the complete set of labels. The constant c is the number of labels and $d_{x_i}^{y_j}$ is the description degree of the particular j th label y_j for a particular i th instance x_i . According to the definition, for each x_i the description degree should meet the constraints $d_{x_i}^{y_j} \in [0, 1]$ and $\sum_{j=1}^c d_{x_i}^{y_j} = 1$.

The solution to an LDL problem can be addressed from several perspectives. According to the selected approach, the algorithm to be developed may differ significantly, either a brand-new algorithm designed especially to deal with LDL constraints, or an adaptation of already available classification algorithms, reformulated to work with those constraints. The LDL study presented in [9] suggested six algorithms grouped in three categories. The first one is Problem Transformation (PT), a simple way to convert an LDL problem into a Single-Label Learning or SLL [36] problem. PT transforms the training samples into weighted single-label examples. Thus, any SLL algorithm may be applied. Two representative algorithms are PT-Bayes and PT-SVM. The second one is Algorithm Adaptation (AA), in which the algorithms are tailored to existing learning algorithms to handle directly with the label distribution. Two suitable algorithms were presented: AA- k NN, an adaptation of the well-known k -nearest neighbors method [37], and AA-BP, a three-layer backpropagation neural network. Finally, Specialized Algorithms (SAs), in contrast to the indirect strategy of PT and AA, directly match the LDL problem. SA-IIS and SA-BFGS are two specialized algorithms that learn by optimizing an energy function that is based on the maximum entropy model.

Subsequent works have successfully improved the results achieved by these original algorithms through different approaches. The methods LDLogitBoost and AOSO-LDLogitBoost proposed in [19], are a combination of the boosting method and the logistic regression applied to LDL framework. Deep Label Distribution Learning (DLDL) [18] and LDL based on Ensemble Neural Networks (ENN-LDL) [38] are two examples of success in the application of neural networks in LDL. Modelled on differentiable decision trees [39], an end-to-end strategy LDL forests proposed in [17] was used as the basis for Structured Random Forest (StructRF) [20]. BC-LDL [40] and DBC-LDL [41] use the binary coding strategies to address with the large-scale LDL problem. Classification with LDL (LDL4C) [42] is

also an interesting approach when the learned label distribution model is considered as a classification model. Feature selection on LDL [43] shows encouraging results by applying feature selection on LDL problems.

2.2. Prototype Selection and Label-Specific Feature Learning

Nowadays, one of the main challenges for supervised learning algorithms is still how to deal with raw datasets. Data pre-processing is an often unattended but important step in the data mining process [22]. Data gathering is often a poorly monitored process, resulting in low-quality datasets. If there is a lot of irrelevant and redundant information or noisy and unreliable data, then knowledge discovery is more difficult to carry out.

Data reduction techniques [22] allow us to obtain a reduced representation of the original data but maintaining the essential structure and integrity. Such pre-processing techniques usually lead to improved performance of subsequent learners.

A frequent problem with real datasets is their big volume, as well as the presence of noise and anomalies that complicate the learning process. Instance selection is to choose a subset of data to achieve the original purpose of a data mining application as if the whole data is used [23]. The optimal outcome of instance selection is a stand-alone model, with a minimal data sample that can perform tasks with little or no performance degradation.

Instance selection has been successfully applied to various problem types like imbalanced learning [44], data streams [45], regression [46], subgroup discover in large size datasets [47]. The research conducted in [48] is also of interest to investigate the impact of instance selection on the underlying structure of a dataset by analyzing the distribution of sample types. However, as of today, instance selection has not been extensively researched in the domain of MLL and to date only few methods have been made available [26–28]. As for LDL, we have not been able to find any studies to date.

Prototype Selection [22] methods are Instance Selection methods that expect to find training sets that offer the best ranking accuracy and reduction rates by using instances-based classifiers which consider a certain similarity or distance measure. A widely used categorization of prototype selection methods consists of three types of techniques [24]: Condensation, where the aim is to retain border points, preserving the accuracy of the training system; Edition, where the objective is to eliminate boundary points that are considered noise or do not match their neighbors but without removing the internal points of each dataset; or Hybrid methods that try to find a small set of training data while maintaining the performance of the classifier. The best approach considering the trade-off between reduction and accuracy is usually the hybrid technique.

To name just a few successful cases when applying prototype selection, [49] proposes a prototype selection algorithm called MONIPS that has proved to be competitive with classical prototype selection solutions adapted to monotonic classification. The experimental study presented in [50] shows a significant improvement when prototype selection is applied to dynamic classifier and ensemble selection.

Another widely used pre-processing technique is the reduction of the data dimensionality by means of feature selection [25]. The main idea is to replace the original set of features by a new subset that extract the main information and provide an accurate classification. However, in MLL and LDL, the strategy of selecting a set of characteristics shared by all labels may not be optimal since each label may be described by a specific subset of characteristics of their own. Label-specific feature learning has been widely studied in MLL. For instance, LIFT [29] firstly builds specific features of each label by performing cluster analysis on its positive and negative instances, and then conducts training and testing by querying the cluster results. LLSF [30] proposes learning label-specific features for each class label by considering pairwise (i.e., second-order) label correlations. MLFC [51] is also a multi-label learning method, which attempts to learn the specific characteristics of each label by exploiting the correlations between them. However, the studies we can find on this subject for LDL are still scarce.

LDLSF [31] is a method inspired in LLSF adapted to deal with LDL problems by jointly selecting label-specific features, selecting common features and exploiting label correlations.

2.3. Evolutionary Optimization

Evolutionary algorithms (EAs) [32] are stochastic search mechanisms based on natural selection notions. EAs have been applied to a broad range of problems, including search problems [52], optimization problems [53], and in many areas as in economics, engineering, biology, etc. The primary idea is to maintain a population of chromosomes, that represent valid solutions to the problem and which evolve over time through a process of competition and targeted variation. CHC [54], is a well-known evolutionary model that introduces different techniques to achieve a trade-off between exploration and exploitation; such as incest prevention, reinitialization of the search process when the population converges or the search stops making progress and the competition among parents and offsprings into the replacement process.

Prototype Selection can be considered as a search problem where EAs can be applied. The search space consists of all the subsets of the training set. This can be represented by using a binary chromosome with two possible states for each gene. If the gene value is 1 then the associated instance is included in the subset, if the value is 0, this does not occur. The chromosome will have as many genes as the number of instances in the training set. EAs have been used to solve the prototype selection problem with promising results [55–59].

Label-specific feature selection is a very complex task. If the original set contains q features and c labels, the objective is to find a $q \times c$ binary matrix where each ij position indicates if the individual feature x_i will be taken into consideration to predict the label d_j . Many search strategies can be used to find a solution to this problem but finding the optimal subset can be a huge time-consuming task. Therefore, it is justifiable to use an evolutionary algorithm that gives us an approximate solution in an acceptable time. Several studies have successfully applied feature reduction to multi-label problems using evolutionary algorithms [60–63], but as of today no such technique has been used to solve the label-specific feature learning problem.

3. ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

ProLSFEO-LDL is an evolutionary algorithm proposal adapted to LDL specificities that combines prototype selection and label-specific feature learning.

It uses the framework of CHC where the search space will be represented by a chromosome (or individual) in which we will code the prototype selection and label-specific feature as detailed in the next Section 3.1. Through the evolutionary algorithm described in Algorithm 1, we will optimize this initial solution until we reach a solution that meets our expectations. We start from a parent population P of size N , randomly initialized, to generate a new population P' obtained by crossing the individuals of the parent population. The recombination method to obtain the offsprings is detailed in the Section 3.2. Then, a survival competition is held where the best N individuals from the parent population P and the offspring population P' are selected to compose the next generation. To determine if one individual is better than another we need to evaluate the quality of each chromosome, for this we use the fitness method described in the Section 3.3. When the population converges or the search no longer progresses, the population is reinitiated to introduce a new diversity into the search, for this purpose we use a threshold t to control when the reinitialization will take place, this step is explained in the Section 3.4. The process of evolution iterates over G generations and finally returns the best solution B found in the search.

In the following sections, we explain in depth each part of the proposed method.

Algorithm 1: ProLSFEO-LDL: Prototype selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

```

Function ProLSFEO-LDL( $S, N, G, t$ ):
1  input :  $S \leftarrow$  Training Dataset ;
    $N \leftarrow$  Population size ;
    $G \leftarrow$  Number of generations ;
    $t \leftarrow$  Treshold ;
2  output:  $B \leftarrow$  Best chromosome ;
3   $P =$  Initialize population of size  $N$  ;
4   $L = t$  ;
5   $g = 0$  ;
6  while  $g < G$  do
7    Evaluate fitness of chromosomes in  $P$  ;
8     $P' =$  HUX crossover ( $P$ ) ;
9    Evaluate fitness of chromosomes in  $P'$  ;
10    $New\_P =$  best  $N$  chromosomes from  $P \cup P'$  ;
11   if  $New\_P = P$  then
12     |  $L = L - 1$  ;
   end
13   if  $L = 1$  then
14     |  $B =$  best chromosome in  $P$  ;
15     |  $P =$  Initialize population of size  $N - 1$  ;
16     |  $P = P \cup B$  ;
17     |  $L = t$  ;
   else
18     |  $P = New\_P$  ;
   end
19    $g = g + 1$  ;
   end
20   $B =$  best chromosome from  $P$ ;
End Function

```

3.1. Representation of Individuals

Each individual should represent the two parts of the algorithm. On the one hand, we have the first m genes of the chromosome that represents the selection of instances, if the gene value is 1 then the associated instance is included in the subset, if the value is 0, this does not occur. And, the second part of the chromosome, consists of $q \times c$ genes representing the matrix of label-specific features. A graphical representation of genes, chromosome and population is exposed in Figure 1, where genes named as ss_i represent if the instance i will be included in the final set or not; and genes fs_{jk} represent if feature k will be taken into account to predicte the specific label j . The initialization of the chromosomes that make up the population is randomly performed from the “discrete uniform” distribution in the closed interval $[0, 1]$.

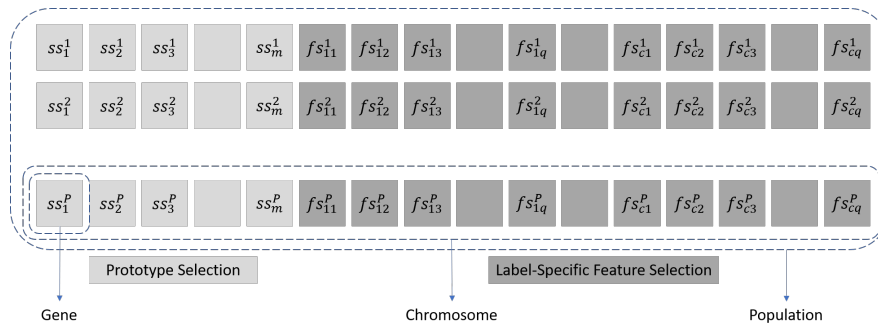


Figure 1. Gene, chromosome and population in evolutionary optimization.

3.2. Crossover

The crossover operator, HUX, performs a uniform cross that randomly exchanges exactly half of the bits that differ between the two parenting chains. No mutation is applied during the recombination phase of the CHC algorithm.

3.3. Fitness Function

In order to measure the quality of a particular chromosome we need to evaluate it using a LDL learner as wrapper. As the prototype selection is focused on the optimization of the AA-*k*NN method, we have chosen this same learner, but with some adaptations to support the selection of features. By default AA-*k*NN uses all features to calculate the distance between each instance. In our adaptation, each output label *j* should be individually predicted using only the selected features from $fs = \{fs_{j1}, \dots, fs_{jq}\}$ coded in the chromosome. The prediction obtained is then normalized so that the result is compatible with the LDL constraints.

The step sequence for evaluating a chromosome is as follows:

- Create a subset of instances: the selected prototypes are coded in the first *m* genes of the chromosome. We create a subset *T* by keeping only the elements of the original training set *S* which have an associated gene value = 1.
- Prediction phase: we use the subset *T* as the training set for AA-*k*NN. The prediction is then computed over a set merged from the test predictions of a 10-fold cross validation set (10-fcv) created from the original training set *S*.
- Evaluation phase: in order to measure the fitness of the chromosome, we calculate the distance between the predicted label distribution \hat{D} and the real label distribution *D* using the Kullback–Leibler divergence formula $KL(D, \hat{D}) = \sum_{j=1}^c d_j \ln \frac{d_j}{\hat{d}_j}$, where *d_j* and \hat{d}_j are the description degree of the particular *j*th label. In our case the fitness of the chromosome directly matches with the KL divergence.

The objective is to minimize the fitness function.

3.4. Reinitialization

When the population converges or the search no longer progresses, the population is reinitiated to introduce a new diversity into the search. The population is considered not sufficiently diverse when the threshold *L* reaches 1 (line 13 in Algorithm 1). This threshold is initialized with the parameterized value *t* (line 4 in Algorithm 1) and decreased by 1 each time that the population is not evolving (line 12 in Algorithm 1). In such a case, only the chromosome that represents the best solution found during the search is kept in the new population, and the remaining individuals are randomly generated, filling in the rest of the population (lines 14–17 in Algorithm 1). In CHC, the typical value used for threshold *t* is equal to the total length of the chromosome divided by 4. In our case, due to the length of the chromosome, we will choose a lower value that allows a faster convergence.

4. Experimental Framework

This section is devoted to introducing the experimental framework used in the empirical study conducted in this paper. In our experiments, we have included 13 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 to evaluate the different aspects of experiments performed in Section 4.3.

4.1. Datasets

There are a total of 13 real-world datasets employed in the experiments. The summary of their characteristics is shown in Table 1.

Table 1. Datasets used in experiments.

No.	Datasets	Examples (m)	Features (q)	Labels (c)
1	Yeast_alpha	2465	24	18
2	Yeast_cdc	2465	24	15
3	Yeast_cold	2465	24	4
4	Yeast_diau	2465	24	7
5	Yeast_dtt	2465	24	4
6	Yeast_elu	2465	24	14
7	Yeast_heat	2465	24	6
8	Yeast_spo	2465	24	6
9	Yeast_spo5	2465	24	3
10	Yeast_spoem	2465	24	2
11	SJAFFE	213	243	6
12	SBU_3DFE	2500	243	6
13	Natural_Scene	2000	294	9

The first to the tenth datasets were gathered from biological experiments on the budding yeast *Saccharomyces cerevisiae* [7]. Each dataset consists of 2465 yeast genes, and an associated phylogenetic profile vector with a length of 24 is used to characterize each gene. In a biological experiment, the level of gene expression is usually uneven at each discrete time point, so the labels correspond to the time point.

Datasets JAFFE [64] and BU_3DFE [65] are two widely used facial expression image datasets. A total of 213 gray-scale expression images in the JAFFE dataset while BU_3DFE contains 2500 facial expression images. The images in JAFFE have been rated by 60 people based on 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 ranked by 23 people using the same scale as used in JAFFE. Each dataset is composed by a 243-dimensional feature vector extracted by the Local Binary Patterns method (LBP) [66]. The score for each emotion is considered the degree of description, and the description degrees (normalized gene expression level) of the six emotions make a label distribution for a particular facial expression image.

The Natural Scene dataset is compiled from 2000 natural scene images that have been inconsistently classified by ten human rankers. A 294-dimensional feature vector extracted in [2] represents each image and is linked with a multi-label selected from nine possible labels, i.e., sky, mountain, water, sun, cloud, snow, building, desert and plant. Then rankers are then required to sort the labels in descending order of relevance. Predictably, the ratings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process [67] is applied to achieve the label distribution.

4.2. Evaluation Measure Selection

Several measures of distance/similarity between probability distributions can be applied to measure the distance/similarity between label distributions. We propose to use a set of six measures when comparing different LDL algorithms:

- Chebyshev Distance: is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension.
- Clark Distance: the Clark distance also called coefficient of divergence is the squared root of half of the divergence distance.
- Canberra Metric: is a numerical measure of the distance between pairs of points in a vector space. It is a weighted version of Manhattan distance. The Canberra distance is a metric function often used for data scattered around an origin.
- Kullback–Leibler Divergence: which is closely related to relative entropy, information divergence, and information for discrimination, is a non-symmetric measure of the difference between two probability distributions $p(x)$ and $q(x)$. Specifically, the Kullback–Leibler divergence of $q(x)$ from $p(x)$ is a measure of the information lost when $q(x)$ is used to approximate $p(x)$.
- Cosine coefficient: is a metric used to measure how similar two non-zero vectors are irrespective of their size. It measures the cosine of the angle between two vectors projected in a multidimensional space. The smaller the angle, higher the cosine similarity.
- Intersection similarity: has its largest value, 1, when all the terms of the first probability distribution are identical to the corresponding terms of the second probability distribution. Otherwise, the similarity is less than 1. In the extreme case, when both distributions are very different, then the similarity will be close to 0.

Each of these measures come from a different syntactic family: Minkowski family, the X^2 family, the L_1 family, the Shannon’s entropy family, the inner product family, and the intersection family, respectively [68]. Taking into account that they come from different families and are significantly different in both syntax and semantics, they have a good chance to reflect different aspects of an LDL algorithm. Supposing that the real label distribution is $D = \{d_1, d_2, \dots, d_c\}$, and the predicted label distribution is $\hat{D} = \{\hat{d}_1, \hat{d}_2, \dots, \hat{d}_c\}$, then the formulation of the six measures is summarized in Table 2.

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 d_j - \hat{d}_j $
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)↓	$Dis(D, \hat{D}) = \sum_{j=1}^c \frac{ d_j - \hat{d}_j }{d_j + \hat{d}_j}$
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)↑	$Sim(D, \hat{D}) = \sum_{j=1}^c \min(d_j, \hat{d}_j)$

4.3. Experimental Setting

The proposed ProLSFEO-LDL algorithm is firstly applied, in a pre-preprocessing step, to the raw datasets obtaining a subset of training instances and a matrix of label-specific features. The subset of selected instances will make up the new pre-processed training set that the learner will train with. As we mentioned before, the selected learner is AA- k NN. Regarding the label-specific features,

we have adapted the AA- k NN algorithm to support the selection of features, for this, each label is predicted separately using only the characteristics marked as selected. The prediction obtained is normalized to make it compatible with the LDL constraints.

The results will be compared with those obtained by the same learner without applying data reduction as a pre-processing step. All measures were computed over a merged set from the test predictions using a wrapped 10-fcv. Note that this procedure is over the entire process and differs from the cross-validation used to estimate the fitness function within the optimization process.

The parameters used in experiments are summarized in Table 3. The AA- k NN algorithm used in the fitness function of ProLSFEO-LDL method and in the subsequent learner requires a value for k , set to four neighbors. We have selected a small value for k in order to provide the most flexible fit with a low bias.

Table 3. Summary of the parameters.

Algorithm	Parameter	Description	Value
ProLSFEO-LDL	N	Population size	100
	G	Number of generations	500
	t	Threshold	10
	k	Number of selected neighbors in AA- k NN used for fitness function	4
AA- k NN	k	Number of selected neighbors	4

5. Results and Analysis

This section presents the results of the empirical studies and their analyses. We will compare the results obtained by AA- k NN with the results obtained by applying the pre-processing step ProLSFEO-LDL. In Table 4 the best outcome for each dataset and measure is highlighted in bold. The last row is the average aggregation result of each column. 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”.

The Wilcoxon test and Bayesian Sign test [33,34] 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 (AA- k NN) and R (ProLSFEO-LDL) is computed into a graphical space divided into three 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 [69] has been used to compute the Wilcoxon test and the R package rNPBST [70] 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.

The outcome of both statistical tests applied to our method is represented in Table 5 (Wilcoxon test) and Figure 2 (Bayesian Sign test).

Comparing ProLSFEO-LDL with the standard LDL learner AA- k NN, we reach the following conclusions:

- The results of the different measures shown in Table 4 highlights the best ranking of ProLSFEO-LDL in the large majority of the datasets and measures.
- The Wilcoxon Signed Ranks test corroborates the significance of the differences between our approach and AA- k NN. As we can see in Table 5, all the hypotheses of equivalence are rejected with small p-values.
- With regard to the Bayesian Sign test, Figure 2 graphically represent the statistical significance in terms of precision between ProLSFEO-LDL and AA- k NN. The following heat-maps clearly indicate the significant superiority of ProLSFEO-LDL, as the computed distributions are always located in the right region.

Table 4. Experimental Results (mean \pm std).

Dataset	Cheby \downarrow		Clark \downarrow		Can \downarrow	
	AA-kNN	ProLSFEO-LDL	AA-kNN	ProLSFEO-LDL	AA-kNN	ProLSFEO-LDL
Yeast_alpha	0.0148 \pm 0.0007	0.0145 \pm 0.0007	0.2321 \pm 0.0112	0.2280 \pm 0.0111	0.7577 \pm 0.0372	0.7451 \pm 0.0376
Yeast_cdc	0.0177 \pm 0.0010	0.0174 \pm 0.0009	0.2375 \pm 0.0134	0.2316 \pm 0.0129	0.7179 \pm 0.0395	0.6972 \pm 0.0395
Yeast_cold	0.0554 \pm 0.0021	0.0545 \pm 0.0015	0.1509 \pm 0.0070	0.1485 \pm 0.0050	0.2611 \pm 0.0129	0.2563 \pm 0.0090
Yeast_diau	0.0393 \pm 0.0009	0.0397 \pm 0.0013	0.2122 \pm 0.0042	0.2125 \pm 0.0055	0.4560 \pm 0.0109	0.4528 \pm 0.0126
Yeast_dtt	0.0393 \pm 0.0016	0.0381 \pm 0.0013	0.1068 \pm 0.0045	0.1035 \pm 0.0036	0.1836 \pm 0.0075	0.1778 \pm 0.0050
Yeast_elu	0.0177 \pm 0.0005	0.0174 \pm 0.0004	0.2182 \pm 0.0048	0.2137 \pm 0.0048	0.6444 \pm 0.0155	0.6285 \pm 0.0137
Yeast_heat	0.0451 \pm 0.0012	0.0445 \pm 0.0007	0.1955 \pm 0.0044	0.1928 \pm 0.0035	0.3924 \pm 0.0096	0.3867 \pm 0.0078
Yeast_spo	0.0643 \pm 0.0024	0.0637 \pm 0.0030	0.2715 \pm 0.0112	0.2690 \pm 0.0124	0.5594 \pm 0.0232	0.5499 \pm 0.0248
Yeast_spo5	0.0962 \pm 0.0043	0.0951 \pm 0.0052	0.1933 \pm 0.0099	0.1914 \pm 0.0110	0.2972 \pm 0.0145	0.2938 \pm 0.0169
Yeast_spoem	0.0924 \pm 0.0036	0.0887 \pm 0.0036	0.1374 \pm 0.0053	0.1320 \pm 0.0055	0.1913 \pm 0.0074	0.1837 \pm 0.0076
SJAFFE	0.1155 \pm 0.0180	0.1088 \pm 0.0090	0.4137 \pm 0.0489	0.4070 \pm 0.0285	0.8527 \pm 0.0962	0.8359 \pm 0.0624
SBU_3DFE	0.1350 \pm 0.0048	0.1272 \pm 0.0056	0.4255 \pm 0.0134	0.4068 \pm 0.0146	0.8746 \pm 0.0290	0.8332 \pm 0.0313
Natural_Scene	0.3168 \pm 0.0081	0.3046 \pm 0.0118	1.8253 \pm 0.0343	1.9161 \pm 0.0347	4.2609 \pm 0.0866	4.5848 \pm 0.1206
Average	0.0807 \pm 0.0038	0.0780 \pm 0.0035	0.3554 \pm 0.0133	0.3579 \pm 0.0118	0.8038 \pm 0.0300	0.8174 \pm 0.0299
Dataset	KL \downarrow		Cos \uparrow		Inter \uparrow	
	AA-kNN	ProLSFEO-LDL	AA-kNN	ProLSFEO-LDL	AA-kNN	ProLSFEO-LDL
Yeast_alpha	0.0066 \pm 0.0006	0.0064 \pm 0.0006	0.9935 \pm 0.0006	0.9937 \pm 0.0006	0.9581 \pm 0.0020	0.9588 \pm 0.0021
Yeast_cdc	0.0083 \pm 0.0009	0.0080 \pm 0.0009	0.9920 \pm 0.0008	0.9924 \pm 0.0008	0.9527 \pm 0.0025	0.9541 \pm 0.0026
Yeast_cold	0.0142 \pm 0.0014	0.0137 \pm 0.0012	0.9866 \pm 0.0012	0.9871 \pm 0.0010	0.9356 \pm 0.0031	0.9368 \pm 0.0022
Yeast_diau	0.0150 \pm 0.0008	0.0146 \pm 0.0009	0.9862 \pm 0.0008	0.9866 \pm 0.0009	0.9367 \pm 0.0017	0.9372 \pm 0.0018
Yeast_dtt	0.0073 \pm 0.0007	0.0069 \pm 0.0006	0.9931 \pm 0.0005	0.9934 \pm 0.0004	0.9547 \pm 0.0017	0.9561 \pm 0.0011
Yeast_elu	0.0074 \pm 0.0003	0.0071 \pm 0.0003	0.9928 \pm 0.0003	0.9932 \pm 0.0003	0.9545 \pm 0.0011	0.9556 \pm 0.0003
Yeast_heat	0.0146 \pm 0.0007	0.0142 \pm 0.0005	0.9861 \pm 0.0007	0.9865 \pm 0.0005	0.9356 \pm 0.0017	0.9365 \pm 0.0013
Yeast_spo	0.0302 \pm 0.0024	0.0287 \pm 0.0026	0.9716 \pm 0.0020	0.9730 \pm 0.0023	0.9076 \pm 0.0036	0.9093 \pm 0.0040
Yeast_spo5	0.0333 \pm 0.0033	0.0315 \pm 0.0030	0.9705 \pm 0.0025	0.9720 \pm 0.0024	0.9038 \pm 0.0043	0.9049 \pm 0.0052
Yeast_spoem	0.0285 \pm 0.0023	0.0256 \pm 0.0021	0.9754 \pm 0.0019	0.9777 \pm 0.0019	0.9076 \pm 0.0036	0.9113 \pm 0.0036
SJAFFE	0.0730 \pm 0.0162	0.0672 \pm 0.0091	0.9308 \pm 0.0163	0.9366 \pm 0.0085	0.8529 \pm 0.0176	0.8572 \pm 0.0106
SBU_3DFE	0.0907 \pm 0.0048	0.0816 \pm 0.0061	0.9118 \pm 0.0047	0.9197 \pm 0.0058	0.8394 \pm 0.0053	0.8474 \pm 0.0060
Natural_Scene	1.1924 \pm 0.0765	0.9908 \pm 0.0654	0.7043 \pm 0.0137	0.7295 \pm 0.0126	0.5634 \pm 0.0098	0.5705 \pm 0.0125
Average	0.1170 \pm 0.0085	0.0997 \pm 0.0072	0.9534 \pm 0.0035	0.9570 \pm 0.0029	0.8925 \pm 0.0045	0.8951 \pm 0.0041

Another interesting information to analyse is the reduction ratio obtained by the proposed method. In Figure 3 we show the percentage of selected prototypes and features with respect to the initial training set. In the case of features we represent the average percentage since each of the output labels can be represented by a different number of features.

The average percentage is around 53% for both, prototypes and features, varying more significantly for the SJAFFE and SBU_3DFE datasets where the reduction ratio decreases significantly. This percentage of data reduction will have a huge impact in the performance of the learner. In instance-based methods like AA-kNN, the fact of handling about half of the prototypes and features, will lead to a reduction of computation time in the prediction phase.

Table 5. Wilcoxon Signed Ranks test comparing ProLSFEO-LDL vs. AA-kNN.

Measure	R^+	R^-	p -Value
Cheby	87	4	0.0017
Clark	77	14	0.0266
Can	78	13	0.0215
KL	91	0	0.0002
Cos	91	0	0.0002
Inter	91	0	0.0002

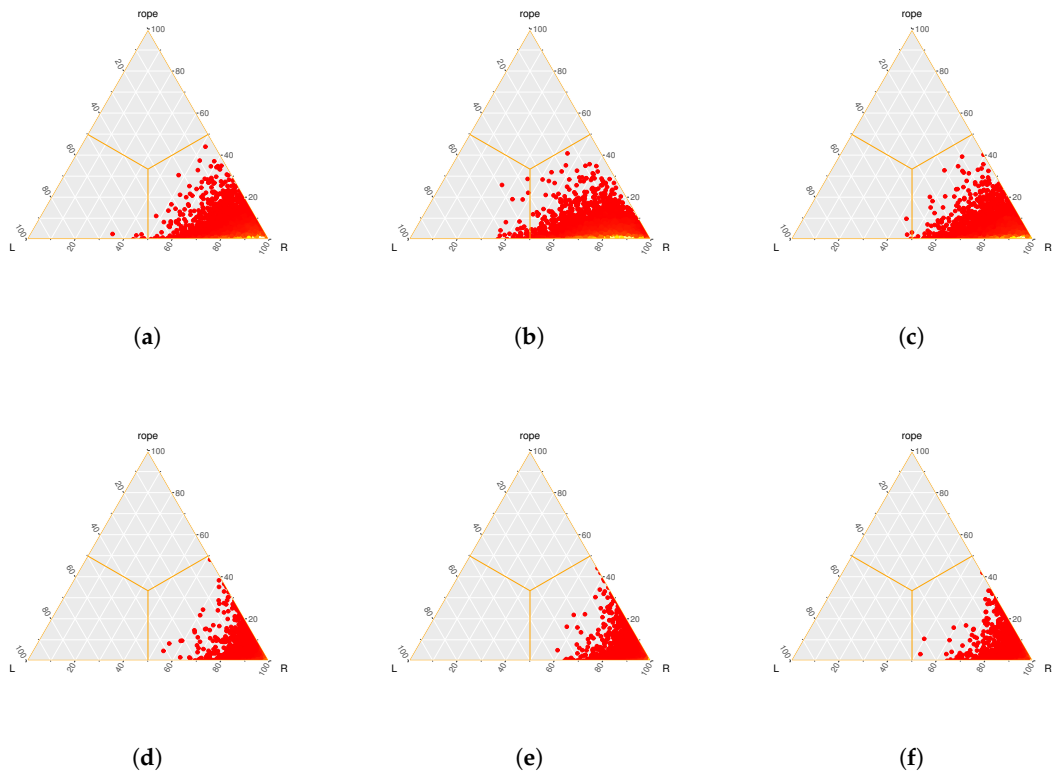


Figure 2. Bayesian Sign test comparing AA-*k*NN(L) vs. ProLSFEO-LDL(R). (a) Chebyshev Distance. (b) Clark Distance. (c) Canberra Metric. (d) Kullback–Leibler Divergence. (e) Cosine Coefficient. (f) Intersection Similarity.

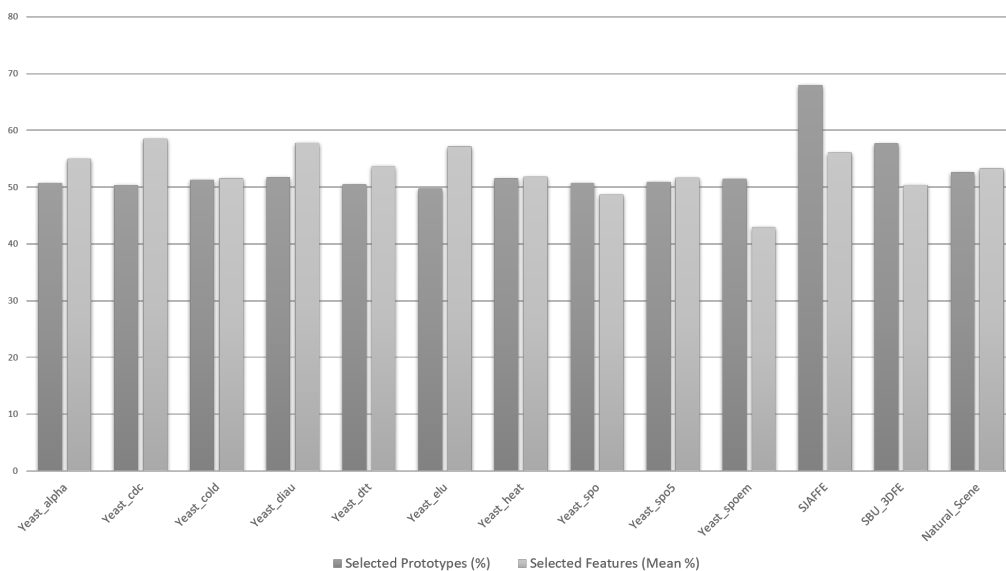


Figure 3. Number of selected prototypes and features (%).

6. Conclusions

In this paper, we proposed a novel data reduction algorithm that adapts to LDL constraints. It simultaneously address the prototype selection and the label-specific feature selection with two objectives: finding an optimal subset of samples to improve the performance of the AA-*k*NN learner

and selecting a subset of characteristics specific for each one of the output label. Both tasks have been addressed as search problems using an evolutionary algorithm, based on CHC, to optimize the solution.

In order to verify the effectiveness of the solution designed, ProLSFEO-LDL has been applied on several real-world LDL datasets, showing significant improvements compared to the use of the raw training set. The results of the different measures highlights the best ranking of ProLSFEO-LDL, outcomes subsequently corroborated by statistical tests. In addition, the percentage of data reduction reached leads to a significant improvement of prediction time.

In future studies we may introduce some further comparisons between already existing approaches and also make improvements to the presented proposal such as:

- It might be interesting to compare the results obtained by this study with the following techniques: LDLSF [31] and Binary Coding bases LDL [40].
- Complement the experimental analysis by dealing with larger datasets (Big Data). To this end, we will complement the current proposal with some of the big data reduction techniques presented in [71].
- The experimental settings use the AA-KNN learner to measure the quality of the solution applied over the pre-processed dataset. Other more powerful LDL learners could be considered for this task but they must previously be adapted to support the label-specific feature selection. With this we will be able to carry out more adequate comparisons with state-of-the-art LDL methods like LDLFs proposed in [17] or StructRF [20].
- Another interesting study that we will undertake is how the presence of noise at the label side can affect the performance. In real scenarios the data gathering is often an automatic procedure that can lead to incorrect sample labelling. It would be interesting to inject some artificial noise to the training labels in order to check the robustness of the implemented approach.

Author Contributions: M.G. and S.G. designed the study; M.G. performed the investigation, carried out tool implementation and all of the analyses and wrote the manuscript; J.-R.C. revised and validated the manuscript; S.G. was in charge of funding acquisition, supervision and project administration. All authors have read and agreed to the published version of the manuscript.

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

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

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

10. Geng, X.; Hou, P. Pre-release prediction of crowd opinion on movies by label distribution learning. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015; pp. 3511–3517.
11. Zhang, Z.; Wang, M.; Geng, X. Crowd counting in public video surveillance by label distribution learning. *Neurocomputing* **2015**, *166*, 151–163. [[CrossRef](#)]
12. Ren, Y.; Geng, X. Sense Beauty by Label Distribution Learning. In Proceedings of the International Joint Conferences on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 2648–2654.
13. Yang, J.; She, D.; Sun, M. Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network. In Proceedings of the International Joint Conferences on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 3266–3272.
14. Xue, D.; Hong, Z.; Guo, S.; Gao, L.; Wu, L.; Zheng, J.; Zhao, N. Personality recognition on social media with label distribution learning. *IEEE Access* **2017**, *5*, 13478–13488. [[CrossRef](#)]
15. Xu, L.; Chen, J.; Gan, Y. Head pose estimation using improved label distribution learning with fewer annotations. *Multimed. Tools Appl.* **2019**, *78*, 19141–19162. [[CrossRef](#)]
16. Zheng, X.; Jia, X.; Li, W. Label distribution learning by exploiting sample correlations locally. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LO, USA, 2–7 February 2018; pp. 4556–4563.
17. Shen, W.; Zhao, K.; Guo, Y.; Yuille, A.L. Label distribution learning forests. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, US, 4–9 December 2017; pp. 834–843.
18. Gao, B.B.; Xing, C.; Xie, C.W.; Wu, J.; Geng, X. Deep label distribution learning with label ambiguity. *IEEE Trans. Image Process.* **2017**, *26*, 2825–2838. [[CrossRef](#)]
19. Xing, C.; Geng, X.; Xue, H. Logistic boosting regression for label distribution learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4489–4497.
20. Chen, M.; Wang, X.; Feng, B.; Liu, W. Structured random forest for label distribution learning. *Neurocomputing* **2018**, *320*, 171–182. [[CrossRef](#)]
21. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. [[CrossRef](#)]
22. García, S.; Luengo, J.; Herrera, F. *Data Preprocessing in Data Mining*; Springer: Berlin/Heidelberg, Germany, 2015.
23. Liu, H.; Motoda, H. On issues of instance selection. *Data Min. Knowl. Discov.* **2002**, *6*, 115–130. [[CrossRef](#)]
24. Garcia, S.; Derrac, J.; Cano, J.; Herrera, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [[CrossRef](#)]
25. Tang, J.; Alelyani, S.; Liu, H. Feature selection for classification: A review. In *Data Classification: Algorithms and Applications*; CRC Press: Boca Raton, FL, USA, **2014**, pp. 37–64.
26. Kanj, S.; Abdallah, F.; Denoeux, T.; Tout, K. Editing training data for multi-label classification with the k-nearest neighbor rule. *Pattern Anal. Appl.* **2016**, *19*, 145–161. [[CrossRef](#)]
27. Arnaiz-González, Á.; Díez-Pastor, J.F.; Rodríguez, J.J.; García-Osorio, C. Local sets for multi-label instance selection. *Appl. Soft Comput.* **2018**, *68*, 651–666. [[CrossRef](#)]
28. Charte, F.; Rivera, A.J.; del Jesus, M.J.; Herrera, F. REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization. *Neurocomputing* **2019**, *326*, 110–122. [[CrossRef](#)]
29. Zhang, M.L.; Wu, L. Lift: Multi-label learning with label-specific features. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 107–120. [[CrossRef](#)]
30. Huang, J.; Li, G.; Huang, Q.; Wu, X. Learning label-specific features and class-dependent labels for multi-label classification. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 3309–3323. [[CrossRef](#)]
31. Ren, T.; Jia, X.; Li, W.; Chen, L.; Li, Z. Label distribution learning with label-specific features. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 3318–3324.
32. Zhou, Z.H.; Yu, Y.; Qian, C. *Evolutionary Learning: Advances in Theories and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2019.

33. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [[CrossRef](#)]
34. Benavoli, A.; Corani, G.; Demšar, J.; Zaffalon, M. Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *J. Mach. Learn. Res.* **2017**, *18*, 2653–2688.
35. Carrasco, J.; García, S.; Rueda, M.; Das, S.; Herrera, F. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm Evol. Comput.* **2020**, *54*, 100665. [[CrossRef](#)]
36. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133–3181.
37. Aha, D.W.; Kibler, D.; Albert, M.K. Instance-based learning algorithms. *Mach. Learn.* **1991**, *6*, 37–66. [[CrossRef](#)]
38. Zhai, Y.; Dai, J.; Shi, H. Label Distribution Learning Based on Ensemble Neural Networks. In Proceedings of the International Conference on Neural Information Processing, Siem Reap, Cambodia, 13–16 December 2018; pp. 593–602.
39. Kotschieder, P.; Fiterau, M.; Criminisi, A.; Rota Bulo, S. Deep neural decision forests. In Proceedings of the IEEE international conference on computer vision, Santiago, Chile, 7–13 December 2015; pp. 1467–1475.
40. Wang, K.; Geng, X. Binary Coding based Label Distribution Learning. In Proceedings of the International Joint Conferences on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 2783–2789.
41. Wang, K.; Geng, X. Discrete binary coding based label distribution learning. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 3733–3739.
42. Wang, J.; Geng, X. Classification with label distribution learning. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 3712–3718.
43. Wang, Y.; Dai, J. Label Distribution Feature Selection Based on Mutual Information in Fuzzy Rough Set Theory. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–2.
44. Millán-Giraldo, M.; García, V.; Sánchez, J. Instance Selection Methods and Resampling Techniques for Dissimilarity Representation with Imbalanced Data Sets. In *Pattern Recognition-Applications and Methods*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 149–160.
45. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [[CrossRef](#)]
46. Song, Y.; Liang, J.; Lu, J.; Zhao, X. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing* **2017**, *251*, 26–34. [[CrossRef](#)]
47. Cano, J.R.; García, S.; Herrera, F. Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes. *Pattern Recognit. Lett.* **2008**, *29*, 2156–2164. [[CrossRef](#)]
48. García, V.; Sánchez, J.S.; Ochoa-Ortiz, A.; López-Najera, A. Instance Selection for the Nearest Neighbor Classifier: Connecting the Performance to the Underlying Data Structure. In Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis, Madrid, Spain, 1–4 July 2019; pp. 249–256.
49. Cano, J.R.; Aljohani, N.R.; Abbasi, R.A.; Alowidbi, J.S.; Garcia, S. Prototype selection to improve monotonic nearest neighbor. *Eng. Appl. Artif. Intell.* **2017**, *60*, 128–135. [[CrossRef](#)]
50. Cruz, R.M.; Sabourin, R.; Cavalcanti, G.D. Analyzing different prototype selection techniques for dynamic classifier and ensemble selection. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3959–3966.
51. Zhang, J.; Li, C.; Cao, D.; Lin, Y.; Su, S.; Dai, L.; Li, S. Multi-label learning with label-specific features by resolving label correlations. *Knowl. Based Syst.* **2018**, *159*, 148–157. [[CrossRef](#)]
52. Khan, S.S.; Quadri, S.; Peer, M. Genetic Algorithm for Biomarker Search Problem and Class Prediction. *Int. J. Intell. Syst. Appl.* **2016**, *8*, 47. [[CrossRef](#)]
53. Ali, A.F.; Tawhid, M.A. A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. *Ain Shams Eng. J.* **2017**, *8*, 191–206. [[CrossRef](#)]
54. Eshelman, L.J. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*; Elsevier: Amsterdam, The Netherlands, 1991; Volume 1, pp. 265–283.

55. García, S.; Cano, J.R.; Herrera, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognit.* **2008**, *41*, 2693–2709. [[CrossRef](#)]
56. Garcia, S.; Cano, J.R.; Bernado-Mansilla, E.; Herrera, F. Diagnose effective evolutionary prototype selection using an overlapping measure. *Int. J. Pattern Recognit. Artif. Intell.* **2009**, *23*, 1527–1548. [[CrossRef](#)]
57. García, S.; Herrera, F. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evol. Comput.* **2009**, *17*, 275–306. [[CrossRef](#)]
58. Vluymans, S.; Triguero, I.; Cornelis, C.; Saeys, Y. EPRENNID: An evolutionary prototype reduction based ensemble for nearest neighbor classification of imbalanced data. *Neurocomputing* **2016**, *216*, 596–610. [[CrossRef](#)]
59. Kordos, M.; Arnaiz-González, Á.; García-Osorio, C. Evolutionary prototype selection for multi-output regression. *Neurocomputing* **2019**, *358*, 309–320. [[CrossRef](#)]
60. Yin, J.; Tao, T.; Xu, J. A multi-label feature selection algorithm based on multi-objective optimization. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–16 July 2015; pp. 1–7.
61. Lee, J.; Kim, D.W. Memetic feature selection algorithm for multi-label classification. *Inf. Sci.* **2015**, *293*, 80–96. [[CrossRef](#)]
62. Zhang, Y.; Gong, D.W.; Rong, M. Multi-objective differential evolution algorithm for multi-label feature selection in classification. In Proceedings of the International Conference in Swarm Intelligence, Beijing, China, 25–28 June 2015; pp. 339–345.
63. Khan, M.; Ekbal, A.; Mencía, E.; Fürnkranz, J. Multi-objective Optimisation-Based Feature Selection for Multi-label Classification. In Proceedings of the International Conference on Applications of Natural Language to Information Systems, Paris, France, 13–15 June 2017; pp. 38–41. [[CrossRef](#)]
64. Lyons, M.; Akamatsu, S.; Kamachi, M.; Gyoba, J. Coding facial expressions with gabor wavelets. In Proceedings of the Third IEEE international conference on automatic face and gesture recognition, Nara, Japan, 14–16 April 1998; pp. 200–205.
65. Yin, L.; Wei, X.; Sun, Y.; Wang, J.; Rosato, M.J. A 3D facial expression database for facial behavior research. In Proceedings of the 7th international conference on automatic face and gesture recognition (FGR06), Los Alamitos, CA, USA, 10–12 April 2006; pp. 211–216.
66. Ahonen, T.; Hadid, A.; Pietikainen, M. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 2037–2041. [[CrossRef](#)]
67. Geng, X.; Luo, L. Multilabel ranking with inconsistent rankers. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3742–3747.
68. Cha, S.H. Comprehensive survey on distance/similarity measures between probability density functions. *Int. J. Math. Model. Methods Appl. Sci.* **2007**, *1*, 300–307.
69. Triguero, I.; González, S.; Moyano, J.M.; García, S.; Alcalá-Fdez, J.; Luengo, J.; Fernández, A.; del Jesús, M.J.; Sánchez, L.; Herrera, F. KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining. *Int. J. Comput. Intell. Syst.* **2017**, *10*, 1238–1249. [[CrossRef](#)]
70. Carrasco, J.; García, S.; del Mar Rueda, M.; Herrera, F. rnpbst: An R package covering non-parametric and bayesian statistical tests. In Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, La Rioja, Spain, 21–23 June 2017; pp. 281–292.
71. Luengo, J.; García-Gil, D.; Ramírez-Gallego, S.; García, S.; Herrera, F. *Big Data Preprocessing: Enabling Smart Data*; Springer: Berlin/Heidelberg, Germany, 2020.

