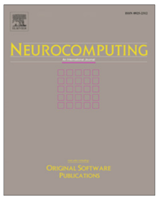
Neurocomputing 526 (2023) 1–8

Contents lists available at ScienceDirect

Neurocomputing

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



Data reduction via multi-label prototype generation

Stefanos Ougiaroglou [a,*], Panagiotis Filippakis [a], Georgia Fotiadou [a], Georgios Evangelidis [b]

[a] Dept. of Information and Electronic Engineering, School of Engineering International Hellenic University, 57400 Sindos, Greece
[b] Dept. of Applied Informatics, School of Information Sciences University of Macedonia, 54636 Thessaloniki, Greece

A R T I C L E   I N F O

Article history:
Received 5 May 2022
Revised 11 December 2022
Accepted 8 January 2023
Available online 12 January 2023

2010 MSC:
00–01
99–00

Keywords:
Multi-label classification
Data reduction techniques
Prototype generation
k-NN classification
Binary relevance
RHC
RSP3
BR*k*NN

A B S T R A C T

A very common practice to speed up instance based classifiers is to reduce the size of their training set, that is, replace it by a condensing set, hoping that their accuracy will not worsen. This can be achieved by applying a Prototype Selection or Generation algorithm, also referred to as a Data Reduction Technique. Most of these techniques cannot be applied on multi-label problems, where an instance may belong to more than one classes. Reduction through Homogeneous Clustering (RHC) and Reduction by Space Partitioning (RSP3) are parameter-free single-label Prototype Generation algorithms. Both are based on recursive data partitioning procedures that identify homogeneous clusters of training data, which they replace by their representatives. This paper proposes variations of these algorithms for multi-label training datasets. The proposed methods generate multi-label prototypes and inherit all the desirable properties of their single-label versions. They consider clusters that contain instances that share at least one common label as homogeneous clusters. It is shown via an experimental study based on nine multi-label datasets that the proposed algorithms achieve good reduction rates without negatively affecting classification accuracy.

1. Introduction

Multi-label classification [1] can be seen as a generalization of the single-label problem of categorizing instances into exactly one class. In the multi-label problem, an instance can belong to many of the classes. Formally, it is a model that maps inputs x to binary vectors y, whose size is equal to the number of available class labels. Classification of images, text, proteins, music, video/movies, etc., are typical examples of multi-label classification problems. For instance, a movie can simultaneously be “crime” and “drama”; an artist or music track may belong to more than one genres or moods; an image may depict “mountain”, “sea” and “beach”.

A typical instance-based, also called lazy, classifier is *k*-Nearest Neighbours (*k*-NN) [2,3]. The classifier retrieves the k nearest neighbours of each unclassified instance and classifies it by applying a majority voting. The *k*-NN classifier is simple, easy to implement and has good performance. It can also be easily adapted for multi-label datasets. Still, its memory and CPU requirements can be lowered and, thus, its performance can be significantly improved by reducing the size of its training set. In single-label classification, *k*-NN is usually applied in conjunction with a Data Reduction Technique (DRT), which can be either a Prototype Selection (PS)[1] [4] or a Prototype Generation (PG) [5] algorithm.

Dimensionality reduction [6] and attribute (feature) selection [7] techniques can be also characterized as DRTs. This paper does not consider dimensionality reduction and attribute selection, but instead focuses on instance reduction via PS and PG algorithms. One can find many such algorithms for single-label classification problems in the literature.

DRTs replace the initial training dataset by a much smaller dataset. This set is called the condensing set and is used by *k*-NN to achieve comparable accuracy to the one achieved when using the initial training dataset, but at a much lower computational cost. Prototype Selection algorithms select instances (or prototypes) from the initial training set, whereas, Prototype Generation

* Corresponding author.
E-mail addresses: stoug@ihu.gr (S. Ougiaroglou), gevan@uom.edu.gr (G. Evangelidis).
URLs: https://people.iee.ihu.gr/stoug (S. Ougiaroglou), http://users.uom.gr/gevan (G. Evangelidis).

[1] Prototype Selection algorithms have two points of view: condensing and editing. We consider them from the first point of view.

https://doi.org/10.1016/j.neucom.2023.01.004
0925-2312/© 2023 Elsevier B.V. All rights reserved.

algorithms generate prototypes by summarizing similar training instances of the same class. Most of the DRTs are based on the following simple idea: only the close to the class decision boundaries training instances are necessary for classification tasks. The training instances belonging to the "internal" area of a class (far from decision boundaries) can be safely removed without loss of classification accuracy. Therefore, DRTs try to select or to generate a sufficient number of prototypes that are close to the class decision boundary data areas. DRTs have been studied in the context of single-label classification problems.

The Label Powerset (LP) transformation technique [1] could be a straightforward approach for using a DRT in a multi-label problem. LP transforms a multi-label dataset into a singe-label dataset by considering each label combination (labelset) as a distinct class. However, LP can be applied only if the number of labels and the possible labelsets are small and there is a sufficient number of instances for each labelset. Otherwise, the total number of different combinations may increase exponentially, the reduction rate may be low and some combinations may be poorly represented.

Binary Relevance (BR) is a widely used transformation technique for multi-label problems. It transforms the multi-label problem into multiple single-label binary problems. Hence, one needs as many classifiers as the available labels in order to predict the labels an unknown instance belongs to. The k-NN classifier in conjunction with BR is called BRkNN [8] and seems to be an ideal combination since the k-NN classifier is a lazy classifier and does not build any classification model. When an instance x needs to be classified, BRkNN searches for the k nearest to x neighbours once, like the single-label k-NN does. Then, the nearest neighbours voting procedure is repeated once for each label. Since each voting procedure concerns a binary classification problem, k should be odd to prevent ties.

The k-NN classifier stops being lazy when it is used in conjunction with a DRT. In effect, the condensing set is the classification model. In multi-label classification, if BR is used, one needs to construct a condensing set for each label. Therefore, the goal of data reduction is not achieved and the k-NN classifier must search for nearest neighbours in each condensing set in order to make each individual label prediction. Thus, the BR transformation method cannot be combined with a Data Reduction algorithm because of the multiple binary condensing sets. This observation makes clear that DRTs must be adapted so that they can be used for multi-label datasets and this is the motive behind the present work.

This paper extends the work presented in [9]. More specifically, it proposes two new prototype generation algorithms for multi-label datasets. Both constitute variations of single-label DRTs. The first proposed algorithm is a variation of Reduction through Homogeneous Clustering (RHC) [10], which is a fast and parameter-free prototype generation algorithm for single-label classification problems. The second proposed algorithm is a variation of RSP3, also a single-label and parameter-free prototype generation algorithm, that belongs to the family of Reduction by Space Partitioning algorithms [11,12]. Both RHC and RSP3 attempt to identify homogeneous clusters of data, i.e., clusters of training instances that belong to the same class label.

The proposed algorithms are called Multilabel RHC (MLRHC) [9] and Multilabel RSP3 (MLRSP3) and inherit the properties of their single-label versions. However, MLRHC and MLRSP3 build multi-label condensing sets to be used by BRkNN. Both MLRHC and MLRSP3 consider a cluster to be homogeneous when all its instances share at least a common label. The experimental study conducted shows that MLRHC and MLRSP3 achieve good reduction rates while classification accuracy is not affected.

The rest of this paper is organized as follows: Section 2 briefly presents the related work, while Section 3 reviews the RHC and RSP3 algorithms. Section 4 presents the proposed MLRHC and MLRSP3 algorithms. Section 5 discusses the experimental study, and, Section 6 concludes the paper and gives directions for future work.

2. Related work

The majority of papers on multi-label classification focus on proposing accurate classification algorithms, without addressing issues related to the high computational cost required when dealing with large multi-label training sets. Only few papers focus on speeding-up lazy classifiers on large multi-label training sets and even fewer are concerned with data reduction or condensing algorithms for that type of datasets. As far as we know, there are no prototype generation algorithms in the literature for multi-label training sets. In this section, we review the limited related literature for fast multi-label classification.

[13] propose the first prototype selection algorithm for multi-label datasets. However, the algorithm focuses on editing imbalanced datasets. Editing removes noise and smooths the class decision boundaries; it does not deal with data reduction. The authors present an under-sampling method for imbalanced training sets inspired by the Edited Nearest Neighbour rule (ENN-rule) [14]. [15] proposes a prototype selection editing algorithm, also based on ENN-rule. The algorithm uses Hamming loss to determine the noisy instances. The idea is simple: instances with high Hamming loss probably lie in close decision boundaries and for this reason they should be removed, like in the case of ENN-rule. We will not further consider the aforementioned approaches since they both deal with editing.

[16] is the first attempt that adapts existing prototype selection algorithms to edit and condense multi-label data. Previously proposed prototype selection algorithms LSBo, for data reduction, and LSSm, for data editing [17] are adapted. The proposed algorithms as well as their predecessors are based on the concept of local sets [18] and on the LP transformation technique. In single-label problems, the local set of an instance x is the largest set of instances centered on x, so that all instances belong to the same class. In multi-label datasets, the authors claim that there is no need for a local set to contain the exact same labelset. The labelset of the instances in a local set may slightly differ. The authors use the Hamming loss calculated over labelsets to measure the differences between labelsets. If the Hamming loss between the labelsets of two instances is greater than a pre-specifed threshold, the instances are considered to be of different "classes". The proposed algorithms are not parameter-free and their performance depends on the pre-specified threshold. Therefore, they are not considered further in this paper.

[19] uses BR, LP and other transformation methods in conjunction with single-label prototype selection algorithms. In the case of BR and its variants, the proposed strategy creates as many single-label training sets as the number of distinct labels. Then, a prototype selection algorithm is applied on each training set to build a condensing set for each label. Every time an instance is selected, it receives a vote that is accumulated in a vector with the votes for all instances. The strategy builds a complete condensing set by selecting all instances with a number of votes that exceeds a pre-specified threshold. Hence, this is also a parameter based approach and will not be considered further. Furthermore, this paper points out the LP drawbacks, mentioned in Section 1, when it is used in conjunction with a prototype selection algorithm.

[20] proposes a scalable lazy classifier for large multi-label datasets. The authors propose an implementation of MLkNN [21] on GPUs. The proposed method implements the execution stages of MLkNN in parallel, offering computational speedup without loss of accuracy.

3. Background knowledge

3.1. The reduction through homogeneous clustering (RHC) algorithm

Reduction through Homogeneous Clustering (RHC) [10] is a prototype generation algorithm based on *k*-means clustering. RHC is parameter-free, hence the size of the condensing set is determined automatically. Also, it is a fast algorithm since it avoids costly and time-consuming pre-processing tasks on the training set, which may be prohibitive for large datasets.

RHC computes the initial means for *k*-means clustering by averaging the instances of each class label. Then, *k*-means forms as many clusters as the number of distinct class labels in the training set. Hence, both the number of clusters *k* and their initial means are determined automatically for *k*-means. If a discovered cluster is homogeneous, i.e., it consists of instances of only one class label, the cluster centroid is placed in the condensing set as a prototype. Otherwise, the aforementioned procedure is recursively applied on that cluster. RHC terminates when all discovered clusters are homogeneous.

RHC generates more prototypes for the close to the decision boundary areas and fewer prototypes for the "internal" class areas. By using the class representatives as initial means for k-means clustering, RHC quickly discovers large homogeneous clusters and achieves a high reduction rate without costly *k*-means iterations. RHC can become even faster by using *k*-means clustering without full cluster consolidation. Contrary to many DRTs, RHC always builds the same condensing set regardless the order of the data in the training set. Moreover, RHC is simple and quite easy to implement. The experimental study presented in [10] shows that RHC is faster and achieves higher reduction rates than state-of-the-art DRTs without harming classification accuracy.

3.2. The reduction by space partitioning v3 (RSP3) algorithm

Reduction by Space Partitioning (RSP) is a group of three PG algorithms that were proposed by Santchez [11]. The three algorithms are based on the Chen and Jozwik algorithm (CJA) [22]. RSP3 is the only parameter-free RSP algorithm (CJA included). Therefore, RSP3 automatically determines the size of the condensing set. Like RHC, RSP3 is based on the concept of cluster homogeneity. Initially, RSP3 determines the two farthest instances in the training set and forms two clusters by assigning each training instance to its closest farthest instance. The algorithm is recursively applied on each non–homogeneous created cluster. In the end, each homogeneous cluster is replaced by its appropriately labeled representative to form the condensing set.

Like RHC, the condensing set created by RSP3 does not depend on the instance order in the training set. The experimental study in [11] and the findings in other studies [12] have shown that RSP3 generates a small and accurate set of training prototypes. When the *k*-NN classifier utilizes the RSP3 output instead of the original training data, it achieves comparable accuracy but at a much lower computational cost.

4. The proposed variations

4.1. The multi-label RHC (MLRHC) algorithm

The Multi-label Reduction through Homogeneous Clustering (MLRHC) algorithm is a variant of RHC for multi-label training data. It works quite similar to RHC, but builds a multi-label condensing set. The key characteristic of MLRHC is the definition of homogeneity for clusters that contain multi-label data. For MLRHC, a cluster is considered homogeneous when it contains instances that share at least one common label.

This is how the MLRHC algorithm works. Assuming that the initial training set is a non–homogeneous cluster, RHC builds a mean (representative) for each label l by averaging the instances that their labelset contains l. Then, it runs *k*-means clustering and discovers as many clusters as the number of distinct labels in the training set. For each cluster C with instances that do not share a common label (i.e., C is non–homogeneous), the aforementioned procedure is repeated considering only the instances that belong to C. For each homogeneous cluster, MLRHC stores the cluster centroid in the condensing set as a prototype. The generated prototype is labeled by the common label(s) in C along with the majority labels, i.e, labels that appear in the labelset of more than half the instances of C. Like the case of RHC, MLRHC terminates when all clusters become homogeneous.

Suppose a training set has three attributes (a, b, c) and three labels (x, y, z). Moreover, suppose that MLRHC discovered a cluster C which contains instances $inst_1, inst_2$ and $inst_3$ with the following BR representations:

- $inst_1 : \{a, b, c, x, y, z\} = \{2, 1, 1, 0, 1, 1\}$
- $inst_2 : \{a, b, c, x, y, z\} = \{1, 2, 2, 0, 1, 0\}$
- $inst_3 : \{a, b, c, x, y, z\} = \{3, 2, 1, 1, 1, 1\}$

Since all instances contain label y, C is homogeneous. The labelset of the generated prototype p will be $\{y, z\}$ because y is the common label that renders C homogeneous and z is a majority label in C. Consequently, in a BR representation, p is $\{a, b, c, x, y, z\} = \{2, 1.67, 1.33, 0, 1, 1\}$.

Fig. 1 demonstrates a two dimensional example. Suppose that a dataset contains sixteen instances (Fig. 1(a)). MLRHC computes representatives for the squares, circles, and stars (Fig. 1(b)). Then, k-means clustering uses the three label representatives as initial means and discovers three clusters (Fig. 1(c)). Two of them are homogeneous because their instances have a common class. Thus, the cluster centroids constitute prototypes (Fig. 1(d)). One prototype is labeled only by "square" because there is no majority label in the cluster. The other prototype is labeled by "square" and "star", because "star" is the common label and "square" is a majority label in a that cluster. MLRHC proceeds by dividing the non homogeneous cluster into three homogeneous clusters (Fig. 1(e)). Consequently, three more prototypes are stored in the condensing set. The final condensing set contains five prototypes instead of the sixteen instances of the initial training set (Fig. 1(f)).

Algorithm 1 shows a non-recursive MLRHC implementation. It uses a queue data structure, Q, to hold clusters. Initially, the whole training set is an unprocessed cluster and is placed in Q (line 2). At each repeat-until iteration, MLRHC dequeues cluster C from Q (line 5) and checks whether C is homogeneous, i.e., it has at least one common label. If it is (line 6), its centroid is placed in the condensing set (CS) as a prototype (line 15) labeled by the common and majority labels in C (lines 9–14). Otherwise, MLRHC computes a list of label-representatives (M), one for each of the labels that exist in C (lines 17–21). Then, MLRHC applies *k*-means clustering, with parameters the non–homogeneous cluster C and the list of the initial label-representatives M to be used as initial means. The result is a new set of unprocessed clusters (*NewClusters*) (line 22) all of which are put into Q (lines 23–25). The repeat-until loop continues until Q becomes empty (line 27), i.e., there are no more clusters to process.

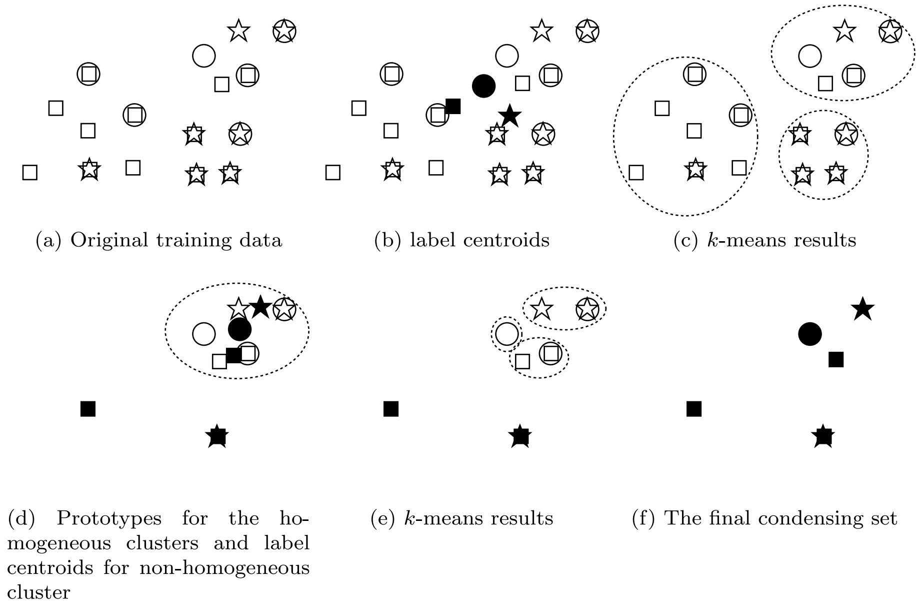

(a) Original training data
(b) label centroids
(c) k-means results
(d) Prototypes for the homogeneous clusters and label centroids for non-homogeneous cluster
(e) k-means results
(f) The final condensing set

Fig. 1. MLRHC execution example.

Algorithm 1: MLRHC

```
Input: $TS$ {original training set}
Output: $CS$ {condensing set}
1: $Q \leftarrow \emptyset$ {queue of unprocessed clusters}
2: Enqueue($Q, TS$) {TS is the first unprocessed cluster}
3: $CS \leftarrow \emptyset$ {initialize the condensing set}
4: repeat
5:   $CS \leftarrow$ Dequeue($Q$) {get a cluster to process}
6:   if all the instances in $C$ have at least one common label then
7:     $r \leftarrow$ centroid of $C$
8:     $r_{labelset} \leftarrow \emptyset$ {initilize the labelset of r}
9:     foreach label $l$ in $C$ do
10:      $n \leftarrow$ count the instances $\in C$ with $l$ in their labelset
11:      if $n > |C|/2$ then
12:        $r_{labelset} \leftarrow r_{labelset} \cup l$
13:      end if
14:    end for
15:    $CS \leftarrow CS \cup \{r\}$ {add r to the condensing set}
16:  else
17:    $M \leftarrow \emptyset$ {initialize the set of label-centroids}
18:    foreach label $l \in C$ do
19:      $m_l \leftarrow$ centroid of instances whose labelset contains $l$
20:      $M \leftarrow M \cup \{m_l\}$
21:    end for
22:    $NewClusters \leftarrow k-means(C, M)$ {determine $|M|$ new clusters}
23:    foreach cluster $cl \in NewClusters$ do
24:      Enqueue($Q, cl$) {put all of them in the processing queue}
25:    end for
26:  end if
27: until IsEmpty($Q$)
28: return $CS$
```

MLRHC inherits all the properties of RHC. Therefore, it is a fast and parameter-free prototype generation algorithm. Also, MLRHC builds the same condensing set regardless the order of the instances in the training set. MLRHC can be ideally combined with BRkNN. For each unclassified instance x, the BRkNN classifier runs over the condensing set (instead of the initial large training set) once and retrieves the k nearest to x prototypes. Then, the labelset of x is predicted by as many voting procedures as the number of labels. For each label l, the same k retrieved nearest prototypes vote in order to predict if x is labeled by l or not.

4.2. The Multi-label RSP3 (MLRSP3) algorithm

The Multi-label Reduction by Space Partitioning (MLRSP3) algorithm is a variant of RSP3 that builds multi-label condensing sets that can be used by the BRkNN classifier. MLRSP3 also uses the idea that a cluster is considered homogeneous when its instances share at least a common label.

Assuming that the initial training set is a non–homogeneous cluster, MLRSP3 retrieves the two farthest instances, $p1$ and $p2$, in the training set. Then, the training set is divided into two clusters, one containing the instances closer to $p1$ and one the instances closer to $p2$. MLRSP3 continues by dividing each non–homogeneous cluster C, i.e., each cluster that contains instances that do not share at least one common label. This procedure continues until all the clusters become homogeneous. A prototype is generated by averaging the instances that belong to each homogeneous cluster C. Like MLRHC, the generated prototype is labeled by the common label(s) in C along with the majority labels in C, i.e., the labels that appear in the labelset of more than half the instances of C.

Fig. 2 shows a two dimensional example. A dataset with sixteen instances is used (Fig. 2(a)). Initially, MLRSP3 finds the farthest training instances A and B (Fig. 2(b)). The training instances are divided into two clusters according to their distances from A and B (Fig. 2(c)). One of the clusters is homogeneous because its instances have a common class. Thus, the cluster centroid constitutes a prototype labeled only by “square” ((Fig. 2(d))). There is no majority label in the cluster. MLRSP3 continues by retrieving the furthest instances C and D in the non–homogeneous cluster ((Fig. 2(d))). The training instances are divided into two clusters according to their distances from C and D. The result is a non–homogeneous cluster and a homogeneous cluster (Figs. 2(e)). MLRSP3 generates a multi-label prototype for the homogeneous cluster. It is labeled by "star" and "square". The star is the common label and the square is the majority label. Then, the same procedure is repeated for each non–homogeneous cluster formed (Figs. 2)) three more prototypes are stored in the condensing set. The final condensing is presented in Fig. 1(i).

A non-recursive MLRSP3 implementation is presented by Algorithm 2. It uses a queue data structure, Q, to hold unprocessed clusters. Initially, the whole training set is an unprocessed cluster and placed in Q (line 2). At each repeat-until iteration, MLRSP3 selects a cluster C from Q (line 5) and checks whether C has at least one common label or not. If it has a common label (line 6), the mean r of the cluster is placed in the condensing set (CS) as a prototype (line 15). r is labeled by the common and majority labels in C (lines 9–14). If C does not contain common labels (i.e., C is non–homogeneous), MLRSP3 finds the farthest instances in C and divides C into $C1$ and $C2$. Both clusters are placed in Q for further processing. The repeat-until loop stops when Q becomes empty (line 23), i.e., all the clusters contain a common label.

MLRSP3 inherits the properties of RSP3. Therefore, it is parameter-free prototype generation algorithm that builds the same condensing set regardless the order of the instances in the training set. MLRSP3 can be combined with BRkNN in a similar manner as MLRHC.

Algorithm 2: MLRSP3

```
Input: TS {the original training set}
Output: CS {the condensing set}
1: $Q \leftarrow \emptyset$ {queue of unprocessed clusters}
2: Enqueue($Q, TS$)
3: $CS \leftarrow \emptyset$ {initialize the condensing set}
4: repeat
5: $C \leftarrow$ Dequeue($Q$) {get a cluster to process}
6: if all the instances in $C$ have at least one common
   label then
7: $r \leftarrow$ centroid of $C$
8: $r_{labelset} \leftarrow \emptyset$ {initilize the labelset of r}
9: for each label $l$ in $C$ do
10: $n \leftarrow$ count the instances $\in C$ with $l$ in their labelset
11: if $n > |C|/2$ then
12: $r_{labelset} \leftarrow r_{labelset} \cup l$
13: end if
14: end for
15: $CS \leftarrow CS \cup \{r\}$ {add r to the condensing set}
16: else
17: $p1$ and $p2$ are the farthest instances in $C$
18: $C1 \leftarrow$ set of instances of $C$ closer to $p1$
19: $C2 \leftarrow$ set of instances of $C$ closer to $p2$
20: Enqueue($Q, C1$)
21: Enqueue($Q, C2$)
22: end if
23: until IsEmpty($Q$)
24: return $CS$
```

5. Performance evaluation

5.1. Datasets

The performance of MLRHC and MLRSP3 was evaluated by conducted experiments on nine multilabel datasets distributed by Mulan datasets repository [23][2]. Table 1 summarizes the key characteristics of the datasets used. The last two columns present the cardinality and the density of the datasets. Cardinality is the mean of the number of labels of the instances. Density is the mean of the number of labels of the instances divided by the number of labels. The second column of Table 1 lists the domain of each dataset.

5.2. Experimental setup

MLRHC and MLRSP3 were coded in C++ and were run only as a pre-processing step to build the multilabel condensing sets. We used the Python implementation of BRkNN[3] available in scikin-multilearn [24]. Scikit-multilearn is a Python library for multi-label classification that is built on top of the popular scikit-learn library.

We compared the performance of BRkNN running over the condensing set built by MLRHC and MLRSP3 against the performance of BRkNN running over the original training set. The Euclidean distance was used as the distance metric. We measured two metrics: (i) Hamming loss, and, (ii) Reduction Rate, that were derived via a fivefold cross-validation schema. Initially, we normalized the datasets in the $[0-1]$ range, and then, we split them into random subsets appropriate for fivefold cross-validation. Since the computational cost of the BRkNN classifier depends on the size of the training set used, the CPU time needed for the classification is not reported. The Hamming Loss (HL) is the fraction of the wrongly predicted labels to the total number of labels. It is computed as follows:

$$HL = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \Delta Z_i|}{|L|}$$

where m is the number of instances in the testing dataset, $|L|$ is the number of labels, Y_i is the set of real labels of instance i, and, Z_i is the set of predicted labels for instance i. Δ is the symmetric difference of two sets and corresponds to the XOR operation. For example, if the labels of an instance are $\{1,1,0,0,1\}$ and the predicted labels are $\{1,1,0,1,0\}$, the Hamming loss is $\frac{2}{5} = 0.4$.

5.3. Experimental results

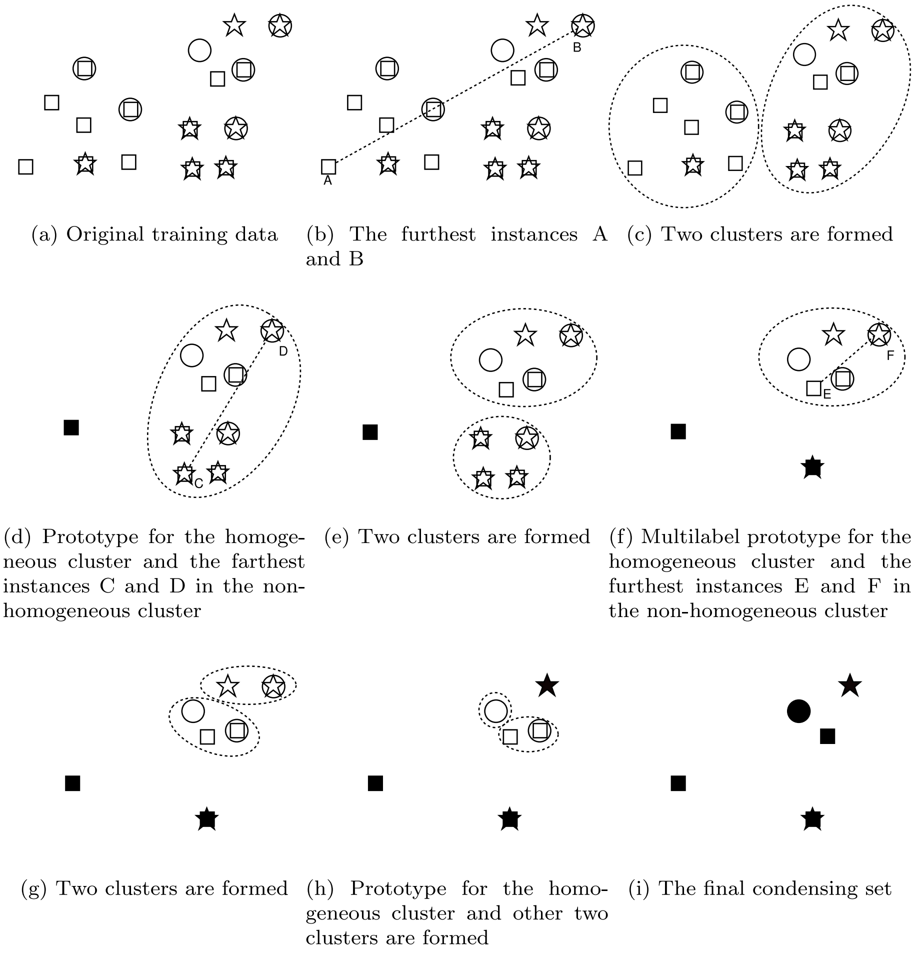
Tables 2–4 present the results obtained by the experimental study. For each dataset, we report Hamming loss and reduction rate achieved by MLRHC and MRPS3. The best measurements are shown in bold face. Moreover, we estimated the pre-processing cost needed by MLRHC and MLRSP3 in order to build their condensing sets. The pre-processing cost was estimated by counting the distances computed by MLRHC and MLRSP3. The last table row reports the average measurements.

Table 2 shows that MLRHC achieved reduction rates between 40% and 85%. The reduction rates achieved by MLRSP3 were between 31% and 76%. The average reduction rate is 57.73% and 51.60% for MLRHC and MLRSP3, respectively. This means that the BRkNN classifier that runs over the condensing set constructed by MLRHC (MLRSP3) is 57.73% (51.60%) faster on average than the BRkNN classifier that runs over the original training set. In the cases of the EMT, SC, BRD, CHD and IMG datasets, MLRHC achieves higher reduction rates than MLRSP3. In the rest four datasets, MLRSP3 achieved higher reduction rates than MLRHC.

One could claim that the reduction rates are not very high. Certainly, they are lower than the reduction rates achieved by the single-label versions of RHC and RSP3. We claim that the comparison between the reduction rates achieved on single-label and multi-label datasets does not make sense. Multi-label data is more complex than single-label data. As a result, MLRHC and MLRSP3 cannot identify large homogeneous clusters in multi-label data like their single-label versions do in single-label data.

[2] http://mulan.sourceforge.net/datasets-mlc.html

[3] http://scikit.ml/api/skmultilearn.adapt.brknn.html



A
B
C
D
E
F
(a) Original training data
(b) The furthest instances A and B
(c) Two clusters are formed
(d) Prototype for the homogeneous cluster and the farthest instances C and D in the non-homogeneous cluster
(e) Two clusters are formed
(f) Multilabel prototype for the homogeneous cluster and the furthest instances E and F in the non-homogeneous cluster
(g) Two clusters are formed
(h) Prototype for the homogeneous cluster and other two clusters are formed
(i) The final condensing set

Fig. 2. MLRSP3 execution example.

Table 1
Dataset characteristics.

Datasets	Domain	Size	Attr.	Labels	Cardinality	Density
CAL500 (CAL)	Music	502	68	174	26.044	0.150
Emotions (EMT)	Music	593	72	6	1.869	0.311
Water quality (WQ)	Chemistry	1060	16	14	5.073	0.362
Scene (SC)	Image	2407	294	6	1.074	0.179
Yeast (YS)	Biology	2417	103	14	4.237	0.303
Birds (BRD)	Sounds	645	260	19	1.014	0.053
CHD49 (CHD)	Medicine	555	49	6	2.580	0.430
Image (IMG)	Image	2000	294	5	1.236	0.247
Mediamill (MDM)	Video	43907	120	101	4.376	0.043

Table 2
Comparison in terms of reduction rate.

Dataset	MLRHC	MLRSP3
CAL	40.50	**76.36**
EMT	**65.73**	46.44
WQ	40.64	**59.91**
SC	**85.13**	39.33
YS	51.85	**53.02**
BRD	**42.7**	31.82
CHD	**65.47**	53.30
IMG	**71.71**	35.65
MDM	55.86	**68.57**
AVG	**57.73**	51.60

Table 3
Comparison in terms of pre-processing cost (in distance computations)

Dataset	MLRHC	MLRSP3
CAL	712,828	**340,001**
EMT	**65,750**	298,839
WQ	**369,905**	1,559,594
SC	**480,152**	6,811,157
YS	**1,341,733**	3,885,708
BRD	**79,764**	160,228
CHD	**65,892**	224,105
IMG	**419,476**	6,410,364
MDM	**806,819,021**	2,604,417,792
AVG	**90,039,391**	291,567,532

Table 4
Comparison in terms of Hamming Loss (HL %). The best three values per dataset are shown in bold (including ties).

Alg.	k	CAL	EMT	WQ	SC	YS	BRD	CHD	IMG	MDM
BrkNN	1	0.197	0.244	0.385	0.133	0.294	0.096	0.368	0.298	0.041
	5	0.153	0.207	0.35	**0.12**	**0.203**	**0.084**	0.33	0.278	**0.033**
	9	0.146	0.197	0.336	**0.12**	**0.195**	**0.088**	0.311	0.276	**0.032**
	13	0.144	**0.195**	0.333	0.122	**0.195**	**0.088**	0.309	0.272	**0.032**
	17	0.141	**0.192**	**0.328**	0.122	**0.195**	**0.088**	**0.301**	0.272	**0.032**
MLRHC BRkNN	1	0.17	0.226	0.379	0.126	0.232	0.093	0.356	0.288	0.038
	5	0.145	0.203	0.34	0.122	**0.202**	**0.09**	0.32	0.259	**0.032**
	9	0.141	0.205	0.33	0.127	**0.202**	0.093	0.307	**0.256**	**0.032**
	13	**0.139**	0.215	**0.327**	0.142	**0.202**	0.095	**0.299**	**0.254**	**0.032**
	17	**0.139**	0.225	**0.326**	0.153	**0.203**	0.097	**0.3**	**0.253**	**0.032**
MLRSP3 BRkNN	1	0.15	0.227	0.368	0.126	0.225	0.131	0.356	0.284	**0.035**
	5	0.14	0.203	0.339	**0.117**	0.204	0.098	0.326	0.27	**0.032**
	9	**0.138**	**0.195**	0.331	**0.121**	**0.202**	0.098	0.322	0.266	**0.032**
	13	**0.138**	**0.191**	0.329	0.125	**0.203**	0.098	0.311	0.262	**0.032**
	17	**0.137**	0.2	0.331	0.128	0.205	0.098	0.306	0.26	**0.032**

Table 5
Results of Wilcoxon signed rank test on Hamming loss, Reduction rate and distance computations (DST) measurements.

Methods	**Hamming Loss**		**Reduction Rate**		**DST**	
	w/l	**Wilc.**	**w/l**	**Wilc.**	**w/l**	**Wilc.**
MLRHC vs MLRSP3	6/3	0.678	5/4	0.515	8/1	0.028
MLRHC vs BR*k*NN	3/6	0.374	-	-	-	-
MLRSP3 vs BR*k*NN	2/6	0.208	-	-	-	-

Table 3 shows that in eight out nine datasets, MLRHC is faster than MLRSP3. In effect, MLRHC is on average three times faster than MLRSP3. In the SC and IMG datasets, the pre-processing cost required by MLRSP3 is 14 and 15 times higher than the pre-processing cost required by MLRHC. In the case of CAL dataset, MLRSP3 computes fewer distances than MLRHC.

We ran experiments using different k values (1, 5, 9, 13 and 17), which is a common practice in the literature. Table 4 shows that the differences in Hamming loss are insignificant. In effect, there is no difference in Hamming loss between BRkNN that uses the multi-label condensing sets constructed by MLRHC and MLRSP3 and the BRkNN classifier that uses the original training set. In most cases, BRkNN achieves similar Hamming loss measurements regardless of whether a condensing set or the original training set is used. In many datasets (i.e., CAL, EMT, WQ, SC, CHD, IMG), the BRkNN classifier that uses condensing sets instead of the original training sets is more accurate. We can safely conclude that MLRHC and MLRSP3 achieve adequate gains in reduction rates while accuracy is not negatively affected.

5.4. Wilcoxon signed rank test

We report the results of a Wilcoxon signed rank test [25] on the experimental measurements, which is a non-parametric statistical test that compares DRTs in pairs. In particular, it examines the experimental measurements achieved by DRTs in each dataset and statistically confirms the validity of the measurements in Tables 2–4.

The results of the statistical test are presented in Table 5. It is worth mentioning that the column with header "Wilc." presents the Wilcoxon value that quantifies the significance of the difference between the two algorithms. If the Wilcoxon value is lower than 0.05, the difference is considered statistically significant.

We ran the test three times, one for each comparison criterion (Hamming loss, Reduction rate and Preprocessing cost in terms of Distance computations). For each dataset and algorithm, we computed the average value of hamming loss achieved by the different k values. Concerning hamming loss and reduction rate measurements, the Wilcoxon test showed that the difference between the MLRHC and MLRSP3 is not significant. In contrast, the difference is statistically significant when the preprocessing cost measurements in terms of distance computations are examined. Consequently, we can conclude that MLRHC is faster than MLRSP3. It is worth mentioning that there is no statistically difference when MLRHC and MLRSP3 are compared to the conventional BRkNN ran over the original training data (without data reduction). Therefore, we conclude that we can use MLRHC and MLRSP3 without loss of classification accuracy.

6. Conclusions and future work

Data Reduction is an essential pre-processing step in order to avoid the drawbacks of high computational cost and storage requirements in instance based classification. However, the vast majority of existing DRTs are not applicable to multi-label classification problems, and also, they can not be effectively used in conjunction with a problem transformation method like Binary Relevance or Label Powerset.

This paper first presented the recent research efforts for speeding-up the k-NN classifier in the context of multi-label classification. Then, it proposed two variations of known Prototype Generation algorithms that are appropriate for multi-label classification problems. The proposed MLRHC and MLRSP3 algorithms can be considered to be the first Prototype Generation algorithms for multi-label training data.

MLRHC and MRPS3 are adaptations of their single-label versions RHC and RSP3. They are parameter-free and are based on clustering procedures that discover homogeneous clusters. In the context of multi-label classification, we consider a cluster to be homogeneous when it contains instances with at least one common label. The centroid of each homogeneous cluster constitutes a prototype labeled by the common labels along with each label that appears in the majority of the cluster instances. Thus, MLRHC and MLRSP3 build a multi-label condensing set that BR*k*NN can use to search for nearest neighbours and make multi-label predictions.

The experimental study used nine multi-label datasets and demonstrated that there is no difference on the accuracy achieved by BR*k*NN when using the condensing sets built by MLRHC and MLRSP3 instead of the original training set. On the other hand, the CPU time needed for the classification process when using the condensing sets is much lower. In many cases, the new variations save more than 70% of CPU time. MLRHC achieved higher reduction rates and lower pre-processing cost than MLRSP3 with no difference in accuracy. Therefore, we can conclude that MLRHC has higher overall classification performance than MLRSP3.

This paper showed that Data Reduction on multi-label problems is an open research field in the data mining and machine learning context. We plan to adapt well-known single-label DRTs on multi-label problems. Next, we plan to developed new parameter-free DRTs as well as scalable classification methods for multi-label training sets.

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.

References

[1] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, Int. J. Data Warehousing Min. 2007 (2007) 1–13.

[2] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27, https://doi.org/10.1109/TIT.1967.1053964.

[3] S. Ougiaroglou, G. Evangelidis, D.A. Dervos, Fhc: an adaptive fast hybrid method for k-nn classification, Logic Journal of IGPL arXiv:http://jigpal.oxfordjournals.org/content/early/2015/03/29/jigpal.jzv015.full.pdf html, doi:10.1093/jigpal/jzv015. http://jigpal.oxfordjournals.org/content/early/2015/03/29/jigpal.jzv015.abstract.

[4] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435, https://doi.org/10.1109/TPAMI.2011.142.

[5] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, Trans. Sys. Man Cyber Part C 42 (1) (2012) 86–100, https://doi.org/10.1109/TSMCC.2010.2103939.

[6] L. Van Der Maaten, E. Postma, J. Van den Herik, Dimensionality reduction: a comparative review, J. Mach. Learn. Res. 10 (2009) 66–71.

[7] J. Tang, S. Alelyani, H. Liu, Feature selection for classification: A review, CRC Press, 2014, pp. 37–64, publisher Copyright: 2015 by Taylor & Francis Group, LLC. doi:10.1201/b17320.

[8] I.V. Eleftherios Spyromitros, Grigorios Tsoumakas, An Empirical Study of Lazy Multilabel Classification Algorithms, in: Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008), 2008.

[9] S. Ougiaroglou, P. Filippakis, G. Evangelidis, Prototype generation for multi-label nearest neighbours classification, in: H. Sanjurjo Gonzalez, I. Pastor Lopez, P. Garcia Bringas, H. Quintian, E. Corchado (Eds.), Hybrid Artificial Intelligent Systems, Springer International Publishing, Cham, 2021, pp. 172–183.

[10] S. Ougiaroglou, G. Evangelidis, RHC: non-parametric cluster-based data reduction for efficient k-NN classification, Pattern Anal. Appl. 19 (1) (2014) 93–109, https://doi.org/10.1007/s10044-014-0393-7.

[11] J.S. Sánchez, High training set size reduction by space partitioning and prototype abstraction, Pattern Recogn. 37 (7) (2004) 1561–1564.

[12] T. Giorginis, S. Ougiaroglou, G. Evangelidis, D.A. Dervos, Fast data reduction by space partitioning via convex hull and mbr computation, Pattern Recogn. 126 (2022), https://doi.org/10.1016/j.patcog.2022.108553, https://www.sciencedirect.com/science/article/pii/S0031320322000346.

[13] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, MLeNN: a First Approach to Heuristic Multilabel Undersampling, in: E. Corchado, J.A. Lozano, H. Quintián, H. Yin (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2014, Springer International Publishing, Cham, 2014, pp. 1–9.

[14] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst., Man, Cybernet. 2 (3) (1972) 408–421.

[15] S. Kanj, F. Abdallah, T. Denœux, K. Tout, Editing training data for multi-label classification with the k-nearest neighbor rule, Pattern Anal. Appl. 19 (1) (2015) 145–161, https://doi.org/10.1007/s10044-015-0452-8.

[16] J.-F. Álvar Arnaiz-González, J.J. Díez-Pastor, C. García-Osorio Rodríguez, Local sets for multi-label instance selection, Appl. Soft Comput. 68 (2018) 651–666, https://doi.org/10.1016/j.asoc.2018.04.016.

[17] E. Leyva, A. González, R. Pérez, Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective, Pattern Recogn. 48 (4) (2015) 1523–1537, https://doi.org/10.1016/j.patcog.2014.10.001.

[18] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Min. Knowl. Discov. 6 (2) (2002) 153–172, https://doi.org/10.1023/A:1014043630878.

[19] J.-F. Álvar Arnaiz-González, J.J. Díez-Pastor, C. García-Osorio Rodríguez, Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning, Exp. Syst. Appl., 109 (2018) 114–130, https://doi.org/10.1016/j.eswa.2018.05.017.

[20] P. Skryjomski, B. Krawczyk, A. Cano, Speeding up k-Nearest Neighbors classifier for large-scale multi-label learning on gpus, Neurocomputing 354 (2019) 10–19, recent Advancements in Hybrid Artificial Intelligence Systems. doi: 10.1016/j.neucom.2018.06.095.

[21] M.-L. Zhang, Z.-H. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recogn. 40 (7) (2007) 2038–2048, https://doi.org/10.1016/j.patcog.2006.12.019.

[22] C.H. Chen, A. Jóźwik, A sample set condensation algorithm for the class sensitive artificial neural network, Pattern Recogn. Lett. 17 (8) (1996) 819–823, https://doi.org/10.1016/0167-8655(96)00041-4.

[23] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining Multi-label Data, Springer, US, Boston, MA (2010) 667–685, https://doi.org/10.1007/978-0-387-09823-4_34.

[24] P. Szymański, T. Kajdanowicz, A scikit-based Python environment for performing multi-label classification, ArXiv e-prints arXiv:1702.01460.

[25] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, A Chapman & Hall book, Chapman & Hall/CRC (2011).