Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 302 (2014) 155–176

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Framework to Generate Synthetic Multi-label Datasets

Jimena Torres Tomás[1,a,2] Newton Spolaôr[a,3]
Everton Alvares Cherman[a,4] Maria Carolina Monard[a,5]

[a] *Laboratory of Computational Intelligence*
Institute of Mathematics and Computer Science
University of São Paulo
13560-970 São Carlos, SP, Brazil

Abstract

A controlled environment based on known properties of the dataset used by a learning algorithm is useful to empirically evaluate machine learning algorithms. Synthetic (artificial) datasets are used for this purpose. Although there are publicly available frameworks to generate synthetic single-label datasets, this is not the case for multi-label datasets, in which each instance is associated with a set of labels usually correlated. This work presents *Mldatagen*, a multi-label dataset generator framework we have implemented, which is publicly available to the community. Currently, two strategies have been implemented in *Mldatagen*: hypersphere and hypercube. For each label in the multi-label dataset, these strategies randomly generate a geometric shape (hypersphere or hypercube), which is populated with points (instances) randomly generated. Afterwards, each instance is labeled according to the shapes it belongs to, which defines its multi-label. Experiments with a multi-label classification algorithm in six synthetic datasets illustrate the use of *Mldatagen*.

Keywords: data generator, artificial datasets, multi-label learning, publicly available framework, Java, PHP

1 Introduction

Classical supervised learning algorithms are single-label, in which only one label from a disjoint set of labels L is associated to each example in the dataset. If $L = 2$, the task is called binary classification, and it is called multi-class classification if

[1] This research was supported by the São Paulo Research Foundation (FAPESP), grants 2011/02393-4, 2010/15992-0 and 2011/12597-6. The authors would like to thank Victor Augusto Moraes Carvalho for his help in additional analysis, as well as the anonymous reviewers for their helpful comments.

[2] Email: jimenat.tomas@gmail.com

[3] Email: newtonspolaor@gmail.com

[4] Email: echerman@icmc.usp.br

[5] Email: mcmonard@icmc.usp.br

http://dx.doi.org/10.1016/j.entcs.2014.01.025

$L > 2$. However, an increasing number of applications in which examples are annotated using more than one label, such as bioinformatics, emotion analysis, semantic annotation of media and text mining [15], requires different algorithms to extract patterns from data. These applications, in which the examples can be associated to several labels simultaneously, characterize a multi-label learning problem.

In practice, the effectiveness of machine learning algorithms depends on the quality of the generated classifiers. This makes fundamental research in machine learning inherently empirical [6]. To this end, the community carries out extensive experimental studies to evaluate the performance of learning algorithms [9]. Synthetic (artificial) datasets are useful in these empirical studies, as they offer a controlled environment based on known properties of the dataset used by the learning algorithm to construct the classifier [2]. A good classifier is generally considered to be one that learns to correctly identify the label(s) of new examples with a high probability. Thus, synthetic datasets can be used instead of real world datasets to derive rigorous results for the average case performance of learning algorithms.

Several frameworks to generate synthetic single-label datasets are publicly available to the community [6] [7] [8]. However, despite some proposals of strategies to generate synthetic multi-label datasets [17,18,3,11], to the best of our knowledge, there is a lack of publicly available frameworks to generate data for multi-label learning. Thus, this work contributes to bridging this gap by proposing the *Mldatagen* framework, which we have implemented and it is hosted at `http://sites.labic.icmc.usp.br/mldatagen`.

The remainder of this work is organized as follows: Section 2 briefly describes multi-label learning concepts and strategies to generate synthetic multi-label datasets. The proposed framework is presented in Section 3 and illustrated in Section 4. Section 5 presents the conclusions and future work.

2 Background

This section presents basic concepts and terminologies related to multi-label learning, including the evaluation measures used in this work, as well as some strategies proposed in the literature to generate synthetic multi-label datasets.

2.1 Basic concepts of multi-label learning

Let D be a dataset composed of N examples $E_i = (\mathbf{x}_i, Y_i)$, $i = 1..N$. Each example (instance) E_i is associated with a feature vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iM})$ described by M features (attributes) X_j, $j = 1..M$, and a subset of labels $Y_i \subseteq L$, where $L = \{y_1, y_2, \ldots y_q\}$ is the set of q labels. Table 1 shows this representation. In this scenario, the multi-label classification task consists in generating a classifier H which, given an unseen instance $E = (\mathbf{x}, ?)$, is capable of accurately predicting its subset of labels (multi-label) Y, *i.e.*, $H(E) \rightarrow Y$.

[6] `http://archive.ics.uci.edu/ml/machine-learning-databases/dgp-2`
[7] `http://www.datasetgenerator.com`
[8] `http://www.burningart.com/meico/inventions/datagen/index.html`

Table 1
Multi-label data.

	X_1	X_2	...	X_M	Y
E_1	x_{11}	x_{12}	...	x_{1M}	Y_1
E_2	x_{21}	x_{22}	...	x_{2M}	Y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	...	x_{NM}	Y_N

Multi-label learning methods can be organized into two main categories: algorithm adaptation and problem transformation [15]. The first one consists of methods which extend specific learning algorithms to handle multi-label data directly, such as the Multi-label Naive Bayes (MLNB) algorithm [18]. The second category is algorithm independent, allowing one to use any state of the art single-label learning method. Methods which transform the multi-label classification problem into several single-label classification problems, such as the *Binary Relevance* (*BR*) approach, fall within this category. Specifically, *BR* transforms a multi-label dataset into q single-label datasets, classifies each single-label problem separately and then combines the outputs.

2.1.1 Evaluation measures

The evaluation of single-label classifiers has only two possible outcomes, correct or incorrect. However, evaluating multi-label classification should also take into account partially correct classification. To this end, several multi-label evaluation measures have been proposed, as described in [15]. In what follows, we briefly describe the example-based and label-based evaluation measures used in this work. All these performance measures range in [0..1].

Example-based evaluation measures Let Δ be the symmetric difference between two sets; Y_i and Z_i be, respectively, the set of true and predicted labels; $I(\text{true}) = 1$ and $I(\text{false}) = 0$. The *Hamming-Loss*, *Subset-Accuracy*, *Precision*, *Recall* and *Accuracy* measures are defined by Equations 1 to 5.

$$Hamming\,Loss(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \Delta Z_i|}{|L|}. \tag{1}$$

$$Subset\,Accuracy(H,D) = \frac{1}{N}\sum_{i=1}^{N} I(Z_i = Y_i). \tag{2}$$

$$Precision(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Z_i|}. \tag{3}$$

$$Recall(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Y_i|}. \tag{4}$$

$$Accuracy(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}. \tag{5}$$

Unlike all other evaluation measures described in this section, for *Hamming-Loss*, the smaller the value, the better the multi-label classifier performance is. Moreover, it is worth observing that *Subset-Accuracy* is a very strict evaluation measure as it requires an exact match of the predicted and the true set of labels.

Label-based evaluation measures In this case, for each single-label $y_i \in L$, the q binary classifiers are initially evaluated using any one of the binary evaluation measures proposed in the literature, such as Accuracy, F-Measure, ROC and others, which are afterwards averaged over all labels. Two averaging operations, macro-averaging and micro-averaging, can be used to average over all labels.

Let $B(T_{P_{y_i}}, F_{P_{y_i}}, T_{N_{y_i}}, F_{N_{y_i}})$ be a binary evaluation measure calculated for a label y_i based on the number of true positive (T_P), false positive (F_P), true negative (T_N) and false negative (F_N). The macro-average version of B is defined by Equation 6 and the micro-average by Equation 7.

$$B_{macro} = \frac{1}{q}\sum_{i=1}^{q} B\left(T_{P_{y_i}}, F_{P_{y_i}}, T_{N_{y_i}}, F_{N_{y_i}}\right). \tag{6}$$

$$B_{micro} = B\left(\sum_{i=1}^{q} T_{P_{y_i}}, \sum_{i=1}^{q} F_{P_{y_i}}, \sum_{i=1}^{q} T_{N_{y_i}}, \sum_{i=1}^{q} F_{N_{y_i}}\right). \tag{7}$$

Thus, the binary evaluation measure used is computed on individual labels first and then averaged for all labels by the macro-averaging operation, while it is computed globally for all instances and all labels by the micro-averaging operation. This means that macro-averaging would be more affected by labels that participate in fewer multi-labels, *i.e.*, fewer examples, which is appropriate in the study of unbalanced datasets [5].

In this work, we use Precision (PC), Recall (RL) and F-Measure (FM), defined by Equations 8 to 10 respectively, as binary evaluation measures.

$$PC(H,D) = \frac{T_P}{T_P + F_P}. \tag{8}$$

$$RL(H,D) = \frac{T_P}{T_P + F_N}. \tag{9}$$

$$FM(H, D) = \frac{2T_P}{2T_P + F_P + F_N}. \quad (10)$$

2.2 Strategies to generate synthetic multi-label datasets

As already mentioned, few strategies to generate synthetic multi-label datasets have been proposed in the literature. In what follows, some of these strategies are briefly described.

In [3] it is proposed to build a synthetic multi-label dataset based on specific functions to label instances which extends an earlier proposal for single-label learning [1]. Many of the $M = 9$ features of the dataset are defined according to uniform distribution.

Synthetic datasets with different properties are generated in [11] to study multi-label decision trees. In this study, the power to identify good features, related to the feature selection task [10], was also verified. The strategy used to generate the datasets considers several functions to define feature values related to the labels.

To illustrate a new multi-label learning algorithm, a synthetic dataset is specified in [17]. Given three labels and a covariance matrix, instances are labeled according to seven Gaussian distributions, such that each distribution is related to one multi-label. The number of instances per multi-label is arbitrarily defined.

Hyperspheres are used to generate twelve synthetic datasets in [18]. First, a hypersphere HS with radius r is generated in $\mathbb{R}^{\frac{M}{2}}$. For each label in a dataset, a smaller hypersphere hs_i, $i = 1..q$, is also randomly generated. Then, the small hyperspheres are populated with points (instances) randomly generated. The dimension of these points increases by adding $\frac{M}{4}$ irrelevant features, with random values, and $\frac{M}{4}$ redundant ones, which copy the original features. Finally, each instance E_i is labeled according to the small hyperspheres it belongs to, which define the multi-label Y_i.

3 Synthetic multi-label dataset generator proposed

This section describes in detail the two strategies we have implemented to generate multi-label synthetic datasets, *HyperSpheres* and *HyperCubes*, which are based on the proposal presented in [18]. These two strategies are integrated in the framework *Mldatagen*, implemented in Java [9] and PHP [10].

The framework outputs a compressed file which contains a noiseless synthetic dataset with the characteristics specified by the user, as well as similar datasets in which noise has been inserted. *Mldatagen* will generate as many datasets with noise as the number of different noise levels requested by the user. Each dataset is in the Mulan format [11], which consists of two files: an ARFF file and a XML file. These

[9] http://www.oracle.com/technetwork/java/index.html
[10] http://php.net
[11] http://mulan.sourceforge.net/format.html

files can be directly submitted to the multi-label learning algorithms available in the Mulan library [16].

3.1 The HyperSpheres strategy

This strategy has 9 input parameters to generate datasets with $M = M_{rel} + M_{irr} + M_{red}$ features.

- M_{rel} - Number of relevant features.
- M_{irr} - Number of irrelevant features.
- M_{red} - Number of redundant features.
- q - Number of labels in the dataset.
- N - Number of instances in the dataset.
- $maxR$ - Maximum radius of the small hyperspheres.
- $minR$ - Minimum radius of the small hyperspheres.
- μ - Noise level(s).
- *name* - Name of the relation in the header of the ARFF file.

As mentioned, a dataset with noise is generated by *Mldatagen* for each noise level. Moreover, the optional parameters $maxR$, $minR$, μ and *name* have default values in *Mldatagen*, as described in Section 3.3.

To prevent the user setting invalid values, the following constraints are assigned to the parameters.

$M_{rel} > 0$	$q > 0$	$minR > 0$	$minR < maxR \leq 0.8$
$M_{irr} \geq 0$	$N > 0$	$\mu \geq 0$	$0 < minR \leq \left\lfloor \frac{\frac{q}{10}+1}{q} \right\rfloor$
$M_{red} \geq 0$	$maxR > 0$	$M_{red} \leq M_{rel}$	

It should be emphasized that *Mldatagen* extends the strategy proposed in [18] by enabling the user to choose the number of different kinds of features, *i.e.*, relevant, irrelevant and redundant, the maximum and minimum radius of the small hyperspheres, as well as the noise level of the datasets generated.

The main steps to generate the synthetic datasets are:

(i) Generating the hypersphere HS in $\mathbb{R}^{M_{rel}}$.

(ii) Generating the q small hyperspheres hs_i, $i = 1..q$ inside HS.

(iii) Generating the N points (instances) in $\mathbb{R}^{M}$.

(iv) Generating the multi-labels based on the q small hyperspheres and inserting the noise level(s) μ.

In what follows, each of these steps is described in detail.

3.1.1 Generating the hypersphere HS

A hypersphere HS in $\mathbb{R}^{M_{rel}}$, centered on the origin of the Cartesian coordinate system and with radius $r_{HS} = 1$, is created.

3.1.2 Generating the q small hyperspheres

All the small hyperspheres are generated in such a way that they are inside the hypersphere HS in $\mathbb{R}^{M_{rel}}$. In addition, each hypersphere $hs_i = (r_i, C_i)$ is defined by a specific radius r_i and center $C_i = (c_{i1}, c_{i2}, c_{i3}, \ldots, c_{iM_{rel}})$. This radius is randomly generated and ranges from $minR$ to $maxR$. On the other hand, the C_i coordinates must fulfill the initial requirement defined by Equation 11 to generate hs_i inside HS.

$$\forall\, i \in [1..q]\ c_{ij} \leq (1 - r_i), j = 1..M_{rel}. \tag{11}$$

Nevertheless, this requirement is insufficient to ensure that hs_i is inside HS. For example, if a value $c_{i1} = (1 - r_i)$ were generated, the other c_{ij} values for $j \neq 1$ would be zero, *i.e.*, the remaining $M_{rel} - 1$ features have to be zero. Thus, all coordinates of each C_i, $i = 1..q$, must range in the restricted domain defined by Equation 12.

$$\sum_{j=1}^{M_{rel}} {c_{ij}}^2 \leq (1 - r_i)^2. \tag{12}$$

The filled area in Figure 1 shows the domain of the C_i coordinates of a hypersphere hs_i for $M_{rel} = 2$, such that hs_i is inside HS.

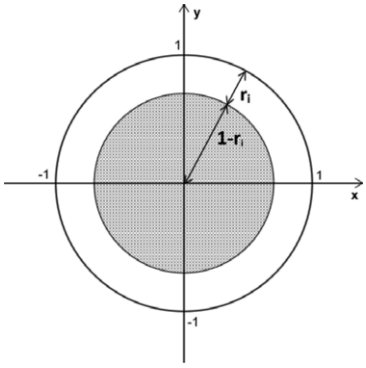


Fig. 1. Domain to define C_i, given r_i, in $\mathbb{R}^2$

As specified in Equation 12, setting the coordinates of C_i, $i = 1..q$, must take into account its radius r_i, as well as the already set coordinate values of the hs_i hypersphere center. This requirement is defined by Equation 13 for each randomly generated coordinate c_{ij}, $j = 1..M_{rel}$, in which only coordinates c_{is}, $s \neq j$, already set are considered.

$$-\sqrt{(1 - r_i)^2 - \sum_{s=1, s\neq j}^{M_{rel}} (c_{is})^2} \leq c_{ij} \leq \sqrt{(1 - r_i)^2 - \sum_{s=1, s\neq j}^{M_{rel}} (c_{is})^2}. \tag{13}$$

As the possible range to define each coordinate of C_i reduces whenever a new coordinate is set, it is necessary to avoid determinism during the generation of the C_i coordinates of the hypersphere hs_i, *i.e.*, to avoid always generating c_{i1} as the first coordinate and $c_{iM_{rel}}$ as the last one. To this end, the index of the coordinates j to be set, $j = 1..M_{rel}$, is randomly defined.

Algorithm 1 summarizes the generation of the hyperspheres $hs_i = (r_i, C_i)$, $i = 1..q$. In this algorithm, the function $random(x, y)$ outputs a value ranging from x to y according to the uniform distribution. The functions $updateminC(x)$ and $updatemaxC(x)$ respectively refresh the lower and upper bounds of the next C_i coordinate to be defined using Equation 13. These procedures are useful to avoid generating instances with empty multi-labels.

Algorithm 1 Generation of the small hyperspheres

for $i = 1 \rightarrow q$ **do**
  $C_i \leftarrow \emptyset$
  $r_i \leftarrow random(minR, maxR)$
  $maxC \leftarrow (1 - r_i)$
  $minC \leftarrow -(1 - r_i)$
  for $j = 1 \rightarrow M_{rel}$ (j *randomly defined*) **do**
    $c_{ij} \leftarrow random(minC, maxC)$
    $C_i \leftarrow C_i \cup \{c_{ij}\}$
    $minC \leftarrow updateminC(minC)$
    $maxC \leftarrow updatemaxC(maxC)$
  end for
  $hs_i \leftarrow (r_i, C_i)$
end for
return $\{hs_1, hs_2, \ldots, hs_q\}$

3.1.3 Generating the points

After defining the q small hyperspheres, the main hypersphere HS is populated with N points (instances). Recall that in each instance $E_i = (\mathbf{x}_i, Y_i)$, $i = 1..N$, $\mathbf{x}_i$ denotes the vector of feature values and $Y_i \subseteq L$ the multi-label of the instance. Thus, to populate HS in $\mathbb{R}^M$, it is required to generate the values $(x_{i1}, x_{i2}, \ldots, x_{iM})$ and the multi-label Y_i for each instance E_i.

To ensure that the multi-labels contain at least one of the q possible labels, the generation of the N instances is oriented, such that, for each instance E_i, the point with coordinates $(x_{i1}, x_{i2}, \ldots, x_{iM_{rel}})$ is at least inside one small hypersphere. By using this procedure, none of the instances will have an empty multi-label. All small hyperspheres hs_i, $i = 1..q$, are populated using this criterion.

Moreover, as the radiuses of the small hyperspheres are different, the distribution (balance) of instances in each hypersphere should be kept, *i.e.*, an hypersphere with greater radius should be proportionally more crowded than an hypersphere with smaller radius. To this end, the framework uses the balance factor defined by Equation 14. Therefore, $N_i = round(f \times r_i)$ points are generated inside each

hypersphere hs_i, $i = 1..q$.

$$f = \frac{N}{\sum_{i=1}^{q} r_i}. \tag{14}$$

As the procedure to generate the small hypersphere centers does, a random point $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kM_{rel}})$ inside $hs_i = (r_i, C_i)$ must be generated obeying the restrictions given by Equation 15.

$$\sum_{j=1}^{M_{rel}} (x_{kj} - c_{ij})^2 \leq r_i^2. \tag{15}$$

However, the domain to generate the $\mathbf{x}_k$ coordinates is different, and is reduced to the region inside hs_i. As the hs_i center is not on the origin, it is required to take into account the coordinates of this center. Figure 2 exemplifies, for $M_{rel} = 2$, that a random point $\mathbf{x}_k$ must be inside the filled area to belong to the hypersphere $hs_i = (r_i, C_i)$.

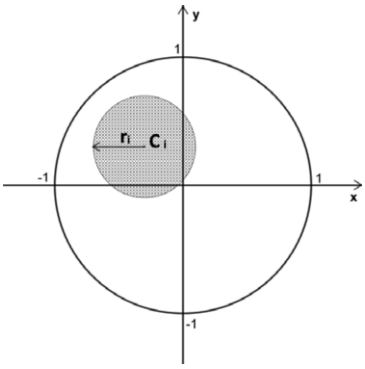


Fig. 2. Domain to define $\mathbf{x}_k$, given r_i and C_i, in $\mathbb{R}^2$

Therefore, to randomly generate each coordinate x_{k_j}, $j = 1..M_{rel}$, of point $\mathbf{x}_k$, it is required to assure that $|x_{kj} - c_{ij}| \leq r_i$. However, in an extreme case, if the first coordinate were $x_{k1} = c_{i1} + r_i$, then the remaining x_{kj} values, $j \neq 1$, would be mandatorily equal to c_{ij} to ensure that point $\mathbf{x}_k$ is inside hs_i. Thus, the x_{kj} coordinate, $\forall j \in [1..M_{rel}]$, should be randomly generated taking into account the already set coordinates. To this end, the range should be constrained as defined by Equation 16 for each randomly generated coordinate x_{k_j}, $j = 1..M_{rel}$, in which only coordinates x_{ks}, $s \neq j$, already set are considered.

$$c_{ij} - \sqrt{r_i^2 - \sum_{s=1, s\neq j}^{M_{rel}} (x_{ks} - c_{is})^2} \leq x_{kj} \leq c_{ij} + \sqrt{r_i^2 - \sum_{s=1, s\neq j}^{M_{rel}} (x_{ks} - c_{is})^2}, \quad \forall i \in [1..q] \, e \, \forall k \in [1..N]. \tag{16}$$

As the procedure to generate the C_i coordinates does, it is required to avoid determinism during the specification of the coordinates of point $\mathbf{x}_k$, *i.e.*, to avoid always generating x_{k1} as the first coordinate and $x_{kM_{rel}}$ as the last one. Thus, the index of coordinates j, $j = 1..M_{rel}$, is also randomly defined.

Algorithm 2 summarizes the generation of the instances $\mathbf{x}_k$, $k = 1..N$, inside the small hyperspheres $hs_i = (r_i, C_i)$, $i = 1..q$. Similar to Algorithm 1, the $updateminX(x)$ and $updatemaxX(x)$ functions refresh respectively the lower and upper bounds of the next $\mathbf{x}_k$ coordinate to be defined inside the corresponding hypersphere hs_i. The already set coordinates are also taken into account to randomly generate the remaining coordinates.

Algorithm 2 Generation of the points inside the hyperspheres

```
for k = 1 → N do
    x_k ← ∅
    maxX ← c_ij − r_i
    minX ← c_ij + r_i
    for j = 1 → M_rel (j randomly defined) do
        x_kj ← random(minX, maxX)
        x_k ← x_k ∪ {x_kj}
        minX ← updateminX(minX)
        maxX ← updatemaxX(maxX)
    end for
end for
return {x_1, x_2, ..., x_N}
```

After generating the N points related to M_{rel}, M_{irr} and M_{red} features are set by adding M_{irr} irrelevant features, with random values, and M_{red} redundant features. The features to be replicated as redundant are chosen randomly. In the end, the N points are in $\mathbb{R}^M$, $M = M_{rel} + M_{irr} + M_{red}$.

3.1.4 Generating the multi-labels

Any instance $\mathbf{x}_k$ $\forall k \in [1..N]$ has the label y_i, $i = 1..q$, in its multi-label Y_k, if $\mathbf{x}_k$ is inside the hypersphere hs_i. The final multi-label Y_k consists of all labels fulfilling this condition, which can be easily verified according to the distance between $\mathbf{x}_k$ and each center C_i, $i = 1..q$. If this distance is smaller than the radius r_i, then $\mathbf{x}_k$ is inside hs_i and $y_i \in Y_k$; otherwise, $y_i \notin Y_k$. The procedure to assign the label y_i to the multi-label Y_k of $\mathbf{x}_k$ $\forall k \in [1..N]$ is implemented as defined by Equation 17. Note that only the M_{rel} features have to be considered.

$$\sqrt{(x_{kj} - c_{ij})^2} \leq r_i, \tag{17}$$

$\forall i \in [1..q]$, $\forall j \in [1..M_{rel}]$, $\forall k \in [1..N]$.

After constructing this dataset, if requested by the user, the noisy dataset(s) are generated. To this end, for each instance E_i, the procedure to insert noise in the constructed dataset flips the label $y_j \in Y_i$, $j = 1..q$, with probability μ. In other words, if the label y_j is in the multi-label Y_i, the flip will remove y_j from the multi-label Y_i with probability μ. Otherwise, the label y_j will be inserted with probability μ.

Based on the *HyperSpheres* strategy, we propose a similar strategy named *HyperCubes* described next, in which hypercubes are used instead of hyperspheres.

3.2 The HyperCubes strategy

The main motivation behind this strategy is that hypercubes could be appropriate to evaluate multi-label learning algorithms, such as decision trees [4], which classify instances by dividing the space using hyperplanes.

The *HyperCubes* strategy also has 9 parameters and the corresponding constraints to generate synthetic datasets without and with noise. However, different from the *HyperSpheres* strategy, $maxR$ and $minR$ denote, respectively, the maximum and minimum half-edges ($\frac{edge}{2}$) of the small hypercubes.

The main steps considered to generate synthetic datasets according to the *HyperCubes* strategy are similar to the ones required by *HyperSpheres*.

(i) Generating the hypercube HC in $\mathbb{R}^{M_{rel}}$.

(ii) Generating the q small hypercubes hc_i, $i = 1..q$ inside HC.

(iii) Generating the N points (instances) in $\mathbb{R}^{M}$.

(iv) Generating the multi-labels based on the q small hypercubes and inserting the noise level(s) μ.

3.2.1 Generating the hypercube HC

A hypercube HC in $\mathbb{R}^{M_{rel}}$, centered on the origin of the Cartesian coordinate system with half-edge $e_{HC} = 1$, is created.

3.2.2 Generating the q small hypercubes

As *HyperSpheres* does, *HyperCubes* generates q small hypercubes $hc_i = (e_i, C_i)$, $i = 1..q$, such that they are inside HC in $\mathbb{R}^{M_{rel}}$. Each small hypercube is defined by a specific half-edge e_i and center $C_i = (c_{i1}, c_{i2}, c_{i3}, \ldots, c_{iM_{rel}})$. This half-edge is randomly generated and ranges from $minR$ to $maxR$. On the other hand, the C_i coordinates must fulfill the initial requirement defined by Equation 18 to generate hc_i inside HC.

$$\forall\, i \in [1..q]\ c_{ij} \leq (1 - e_i), j = 1..M_{rel}. \tag{18}$$

Different from *HyperSpheres*, this requirement is enough to ensure that hc_i is inside HC. Therefore, the domain in which the C_i coordinates range is given by Equation 18. The filled area in Figure 3 shows the domain of the C_i coordinates of a hypercube hc_i for $M_{rel} = 2$, such that hc_i is inside HC.

Different from *HyperSpheres*, the possible ranges to define each coordinate c_{ij} are the same for all coordinates. Thus, Algorithm 3 is simpler than Algorithm 1, as the functions $updateminC(x)$ and $updatemaxC(x)$ are not needed to generate the hypercubes $hc_i = (e_i, C_i)$, $i = 1..q$.

3.2.3 Generating the points

As *HyperSpheres* does, after defining the q small hypercubes, the main hypercube HC is populated with N points (instances). To populate HC in $\mathbb{R}^{M}$, it is required to generate the values $(x_{i1}, x_{i2}, \ldots, x_{iM})$ and the multi-label Y_i for each instance

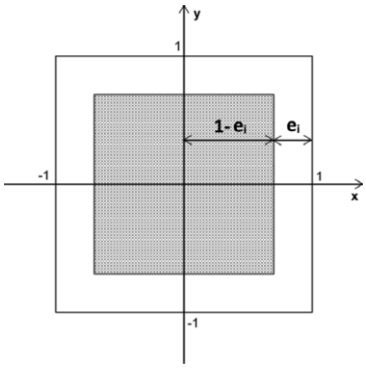


Fig. 3. Domain to define C_i, given e_i, in $\mathbb{R}^2$

Algorithm 3 Generation of the small hypercubes

```
for i = 1 → q do
    C_i ← ∅
    e_i ← random(minR, maxR)
    maxC ← (1 − e_i)
    minC ← −(1 − e_i)
    for j = 1 → M_rel do
        c_ij ← random(minC, maxC)
        C_i ← C_i ∪ {c_ij}
    end for
    hc_i ← (e_i, C_i)
end for
return {hc_1, hc_2, ..., hc_q}
```

E_i.

In addition, HyperCubes also oriented the generation of points, such that each point is at least inside one small hypercube. By using this procedure, no instances have empty multi-labels and all small hypercubes hc_i, $i = 1..q$, are populated.

To ensure that the distribution of instances inside each small hypercube is proportional to the corresponding half-edge, the framework uses a balance factor similar to the one in Equation 14. However, instead of summing radiuses, half-edges are summed. Thus, $N_i = round(f \times e_i)$ points are generated inside each hypercube hc_i, $i = 1..q$.

As the procedure to generate the small hypercube centers does, a random point $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kM_{rel}})$ inside $hc_i = (e_i, C_i)$ must be generated as defined by Equation 19.

$$|(x_{kj} - c_{ij})| \leq e_i,\ \forall i \in [1..q]\, e\, \forall j \in [1..M_{rel}]. \tag{19}$$

Similar to Section 3.1.3, the domain to generate the $\mathbf{x}_k$ coordinates is different, being reduced to the region inside hc_i. As the hc_i center is not on the origin, it is required to take into account the coordinates of this center. Figure 4 exemplifies, for $M_{rel} = 2$, that a random point $\mathbf{x}_k$ must be inside the filled area to belong to the hypercube $hc_i = (e_i, C_i)$.

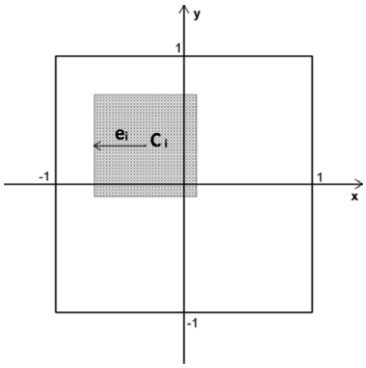


Fig. 4. Domain to define $\mathbf{x}_k$, given e_i, in $\mathbb{R}^2$

Algorithm 4 summarizes the generation of the instances $\mathbf{x}_k$, $k = 1..N$, inside a small hypercube $hc_i = (e_i, C_i)$, $i = 1..q$. This algorithm is simpler than Algorithm 2 by discarding the functions $updateminX(x)$ and $updatemaxX(x)$, as the coordinates range in the same domain.

Algorithm 4 Generation of the points inside the hypercubes

```
for k = 1 → N do
    x_k ← ∅
    maxX ← c_ij − e_i
    minX ← c_ij + e_i
    for j = 1 → M_rel do
        x_kj ← random(minX, maxX)
        x_k ← x_k ∪ {x_kj}
    end for
end for
return {x_1, x_2, ..., x_N}
```

As *HyperSpheres* does, after generating the N points related to M_{rel}, the M_{irr} and M_{red} features are set by adding M_{irr} irrelevant features, with random values, and M_{red} redundant features. The features to be replicated as redundant are chosen randomly. In the end, the N points are in $\mathbb{R}^M$, $M = M_{rel} + M_{irr} + M_{red}$.

3.2.4 Generating the multi-labels

Similar to *HyperSpheres*, any instance $\mathbf{x}_k$ $\forall k \in [1..N]$ has the label y_i, $i = 1..q$, in its multi-label Y_k, if $\mathbf{x}_k$ is inside the hypercube hc_i. The final multi-label Y_k is thus composed of all labels fulfilling this condition, which can be easily verified according to the distance between $\mathbf{x}_k$ and each center C_i, $i = 1..q$. If this distance is smaller than the half-edge e_i, then $\mathbf{x}_k$ is inside hc_i and $y_i \in Y_k$; otherwise, $y_i \notin Y_k$. The procedure to assign the label y_i to the multi-label Y_k of $\mathbf{x}_k$ $\forall k \in [1..N]$ is implemented as defined by Equation 20. Note that only the M_{rel} features have to be considered.

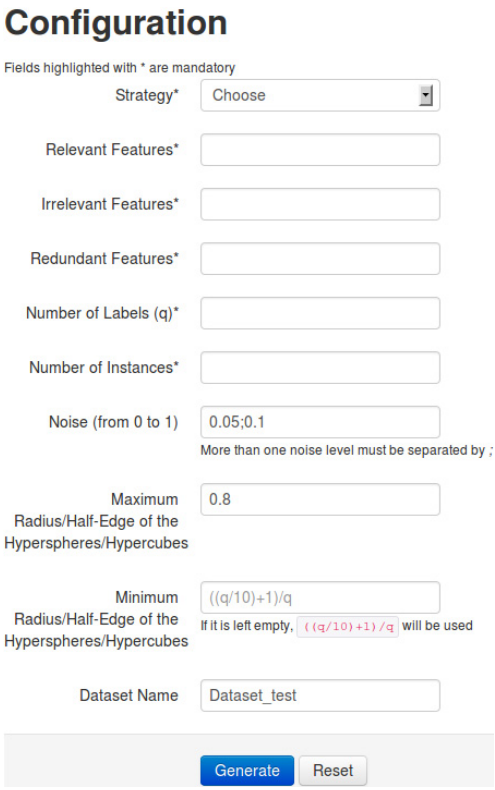
$$|(x_{kj} - c_{ij})| \leq e_i,\ \forall i \in [1..q]\, e\, \forall j \in [1..M_{rel}]. \tag{20}$$

After constructing this dataset, if requested by the user the noisy dataset(s) are generated in the same way as the *HyperSpheres* strategy does.

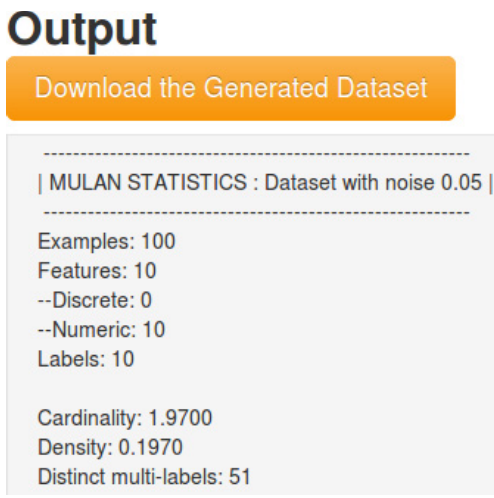
3.3 The Mldatagen framework

Currently, both strategies *HyperSpheres* and *HyperCubes* (Sections 3.1 and 3.2) are implemented in the *Mldatagen* framework, which is publicly available at `http://sites.labic.icmc.usp.br/mldatagen`. In this web site, the user finds a short introduction to *Mldatagen*, as well as the interface to configure the framework parameters and download the output.

Figure 5a shows the parameter setting interface, which considers mandatory and optional parameters. The user can set one or more noise levels in parameter μ by separating them with the ";" character. Furthermore, the optional parameters $maxR$, $minR$, μ and $name$ have default values: 0.8, $\lfloor(\frac{q}{10}+1)/q\rfloor$, $\{0.05;\ 0.1\}$ and "Dataset_test", respectively. After filling in the fields, the user should click on the "Generate" button.



(a) *Mldatagen* parameter setting



(b) Mulan statistics of a dataset generated by *Mldatagen*

Fig. 5. Framework screenshots

To avoid setting invalid values, the constraints described in Section 3.1 are verified. If any constraint is not fulfilled, *Mldatagen* will display an error message indicating the parameter that should be reviewed.

After execution, *Mldatagen* displays information about the generated dataset, such as the distribution of instances inside the geometrical shape used and multi-label statistics, which are calculated by Mulan, as shown in Figure 5b.

To download the *Mldatagen* output, the user should click on the "Download the generated dataset" button. The output consists of a compressed file named according to the pattern

`<x>_<y>_<w>_<z>.tar.gz`

in which `<x>` is the strategy (*HyperSpheres* or *HyperCubes*), `<y>`, `<w>` and `<z>` are numbers denoting, respectively, the number of relevant, irrelevant and redundant features. As already mentioned, this compressed file contains the user specified synthetic dataset without noise, as well as one synthetic dataset per noise level set in the μ parameter. Each dataset can be directly submitted to Mulan for learning purposes.

4 Illustrative example

Mldatagen was used to generate 6 synthetic multi-label datasets, 3 using the *HyperSpheres* strategy and the other 3 the *HyperCubes* strategy. To generate the datasets, different values of the M_{rel}, M_{irr} and M_{red} parameters were used. These values were chosen to analyze how the number of features ($M = M_{rel} + M_{irr} + M_{red}$) and the number of unimportant features (M_{irr} and M_{red}) influence the performance of the multi-label *BRkNN-b* learning algorithm [13], available in Mulan. In what follows, information about the synthetic datasets generated, *BRkNN-b* and the classification results are presented.

4.1 Dataset description

The parameters M_{rel}, M_{irr} and M_{red} were set with different values. Table 2 shows these values and the generation strategy used for each dataset.

Table 2
Number of relevant, irrelevant and redundant features in the six synthetic datasets

Dataset	Strategy	M_{rel}	M_{irr}	M_{red}	M
A	*HyperSpheres*	20	0	0	20
B	*HyperCubes*	20	0	0	20
C	*HyperSpheres*	10	5	5	20
D	*HyperCubes*	10	5	5	20
E	*HyperSpheres*	5	0	0	5
F	*HyperCubes*	5	0	0	5

Aside the features, *Mldatagen* was executed with default value parameters, except for the noise level μ, the number of instances N and the number of labels q,

which were set as shown in Table 3.

Table 3
Setting of other *Mldatagen* parameters

q	N	$maxR$	$minR$	μ
10	1000	0.8	$\lfloor(\frac{q}{10}+1)/q\rfloor$	0

Table 4 shows, for each dataset, the single-label frequencies; the lowest and the highest single-label frequencies, as well as the first, second (median) and third quartiles, as suggested by [14]; the Label Cardinality (LC), which is the average number of single-labels associated with each example defined by Equation 21; and the Label Density (LD), which is the normalized cardinality ($LD(D) = LC(D)/|L|$) defined by Equation 22.

$$LC(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i|. \qquad (21) \qquad\qquad LD(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|}. \qquad (22)$$

Figure 6 depicts dispersion of the single-label frequencies by boxplots, with a dashed line at frequency = 500 (50%). As can be observed, all single-label frequencies are not higher than 50% for datasets A and B. Furthermore, the datasets with less features (E and F) have higher single-label frequencies, with the 3rd quartiles above the dashed line.

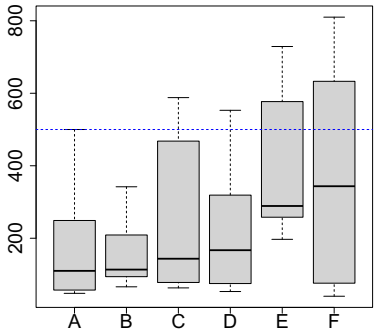


Fig. 6. Boxplots of the single-label frequencies

Table 5 shows, for each synthetic dataset, the percentage of multi-labels in the dataset with number of labels ranging from 1 to 10. For example, in dataset A, 45.90% of the multi-labels have only one single label, 30.90% have two single-labels and this percentage reduces to 0.50% for 6 single-labels. Moreover, there is no multi-label with more than 6 single-labels.

As can be observed, in the four datasets with $M = 20$ features, the higher the number of single-labels, the lower the percentage in the multi-label is. On the other hand, the datasets E and F with $M = 5$ features show a better frequency distribution. These different behaviors could be explained by the curse of dimensionality [8], which can harm analyses of high-dimensional data.

Table 4
Single-label frequencies and multi-label statistics of the synthetic datasets

	Datasets					
	A	*B*	*C*	*D*	*E*	*F*
y_1	65	382	553	109	284	40
y_2	476	130	63	97	274	633
y_3	249	94	88	225	729	361
y_4	69	97	588	67	726	648
y_5	500	155	78	553	446	810
y_6	57	209	171	53	210	187
y_7	151	342	177	75	294	461
y_8	48	77	468	504	258	55
y_9	221	66	116	236	577	76
y_{10}	49	97	69	319	197	326
lowest	48	66	63	53	197	40
1*st quartile*	59	94.8	80.5	80.5	262	103.8
2*nd quartile*	110	113.5	143.5	167	289	343.5
3*rd quartile*	242	195.5	395.3	298.3	544.3	590
highest	500	382	588	553	729	810
LC	1.89	1.65	2.37	2.24	4.00	3.60
LD	0.19	0.17	0.24	0.22	0.40	0.36

4.2 The multi-label BR*k*NN-b *learning algorithm*

Lazy algorithms are useful in the evaluation of datasets with irrelevant features, as the classifiers built by these algorithms are usually susceptible to irrelevant features. The multi-label learning algorithm *BRkNN* is an adaptation of the single-label lazy k Nearest Neighbor (*kNN*) algorithm to classify multi-label examples proposed in [13]. It is based on the well-known *Binary Relevance* approach, which transforms a multi-label dataset into q single-label datasets, one per label. After transforming the data, *kNN* classifies each single-label problem separately and then *BRkNN* aggregates the prediction of each one of the q single-label classifiers in the corresponding multi-label, which is the one predicted. Despite the similarities between the algorithms, *BRkNN* is much faster than *kNN* applied according to the *BR* approach, as *BRkNN* performs only one search for the k nearest neighbors.

Table 5
Percentage of multi-labels with different number of labels

	Datasets					
$\|Y\|$	*A*	*B*	*C*	*D*	*E*	*F*
1	45.90	58.60	33.70	37.90	16.20	14.50
2	30.90	25.80	25.70	29.00	16.60	16.70
3	14.40	9.80	20.00	15.20	12.50	20.30
4	6.90	3.90	13.30	10.40	16.40	16.90
5	1.40	1.70	5.20	4.70	11.10	14.20
6	0.50	0.20	1.90	2.40	10.80	10.40
7	0.00	0.00	0.20	0.70	8.20	6.10
8	0.00	0.00	0.00	0.00	5.10	0.90
9	0.00	0.00	0.00	0.00	2.50	0.00
10	0.00	0.00	0.00	0.00	0.60	0.00

To improve the predictive performance and to tackle directly the multi-label problem, the extensions *BRkNN-a* and *BRkNN-b* were also proposed in [13]. Both extensions are based on a label confidence score, which is estimated for each label from the percentage of the k nearest neighbors having this label. *BRkNN-a* classifies an unseen example E using the labels with a confidence score greater than 0.5, *i.e.*, labels included in at least half of the k nearest neighbors of E. If no label satisfies this condition, it outputs the label with the greatest confidence score. On the other hand, *BRkNN-b* classifies E with the $[s]$ (nearest integer of s) labels which have the greatest confidence score, where s is the average size of the label sets of the k nearest neighbors of E.

In this work, we use the *BRkNN-b* extension proposed in [7] and implemented in Mulan, which was executed with $k = 11$ and the remaining parameters with default values.

4.3 Results and discussion

All the reported results were obtained by Mulan using 5x2-fold cross-validation, which randomly repeats 2-fold cross-validation five times.

The evaluation measures described in Section 2.1.1 were used to evaluate the classifiers constructed by *BRkNN-b*. Table 6 shows the average and the standard deviation (in brackets) of the evaluation measure values. The example-based evaluation measures are denoted in Table 6 as: Hamming-Loss (*HL*); Subset-Accuracy (*SAcc*); Precision (*Pr*); Recall (*Re*); Accuracy (*Acc*). The label-based evaluation measures are denoted as: Micro-averaged Precision (P^{μ}); Micro-averaged Recall

(R^{μ}); Micro-averaged F-Measure ($F1^{\mu}$) and Macro-averaged Recall (R^{M}). Best results among the datasets with $M = 20$, as well as between the two datasets with $M = 5$ are highlighted in gray.

Table 6
Classification performance

	Example-based measures				
	HL	*SAcc*	*Pr*	*Re*	*Acc*
A	0.20 (0.00)	0.06 (0.01)	0.48 (0.01)	0.54 (0.01)	0.37 (0.00)
B	0.16 (0.01)	0.12 (0.01)	0.53 (0.01)	0.75 (0.02)	0.47 (0.01)
C	0.20 (0.01)	0.08 (0.01)	0.55 (0.02)	0.66 (0.01)	0.45 (0.01)
D	0.17 (0.00)	0.10 (0.01)	0.60 (0.01)	0.81 (0.02)	0.52 (0.00)
E	0.19 (0.01)	0.12 (0.01)	0.68 (0.01)	0.80 (0.02)	0.61 (0.01)
F	0.12 (0.01)	0.23 (0.02)	0.76 (0.01)	0.93 (0.01)	0.71 (0.01)

	Label-based measures			
	P^{μ}	R^{μ}	$F1^{\mu}$	R^{M}
A	0.49 (0.01)	0.57 (0.01)	0.52 (0.00)	0.29 (0.01)
B	0.52 (0.02)	0.71 (0.02)	0.60 (0.01)	0.73 (0.02)
C	0.57 (0.02)	0.69 (0.01)	0.62 (0.01)	0.36 (0.01)
D	0.60 (0.01)	0.76 (0.02)	0.67 (0.01)	0.75 (0.02)
E	0.73 (0.02)	0.85 (0.01)	0.78 (0.01)	0.78 (0.01)
F	0.79 (0.01)	0.91 (0.01)	0.85 (0.01)	0.89 (0.01)

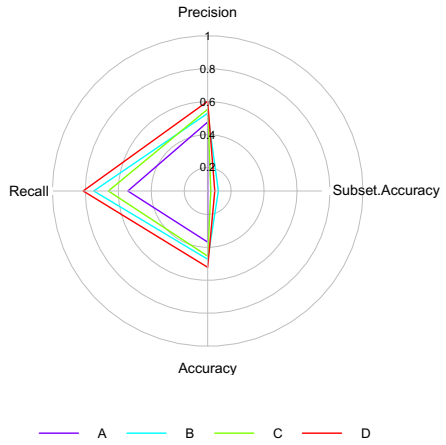
It can be observed that between the two datasets with $M = 5$ features, dataset F, which was created using the *HyperCubes* strategy, shows the best results. Among the four datasets with $M = 20$, dataset B obtained the best example-based measure values for Hamming-Loss (HL) and Subset-Accuracy ($SAcc$), while dataset D obtained the best results for the remainder of the example-based measures and for all the label-based measures considered. Both datasets were also created using the *HyperCubes* strategy.

In what follows, spider graphs summarizing the results of the performance measures in Table 6 (except Hamming-loss) are shown. Thus, greater values indicate better performance. These graphs were generated using the R framework [12]. Figure 7 shows the graphs for datasets A, B, C and D ($M = 20$), while Figure 8 shows the graphs for datasets E and F ($M = 5$). Each colored line plots the performance of the classifier built with an specific dataset, while each axis represents an evaluation measure.

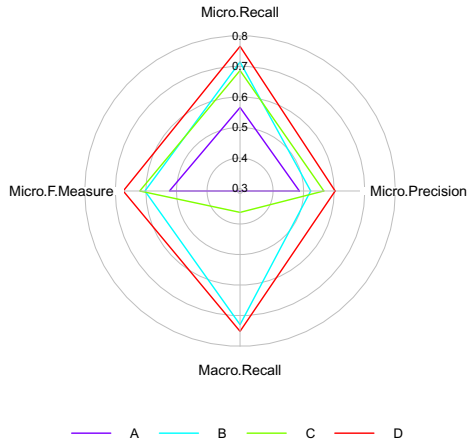
As stated before, for $M = 20$ the classifiers build with datasets B and D show better performance than the ones build with datasets A and C. Moreover, for $M = 5$ the classifier build with dataset F shows better results than the one build with dataset E. Observe that these datasets (B, D and F) were generated using the *HyperCubes* strategy.

One possible cause of this behavior is that in the *HyperSpheres* strategy the volume of the main hypersphere HS increases with the dimensionality up to a maximum value, which depends on the initial volume of the hypersphere. Afterwards, the volume decreases with the dimensionality, tending to zero. As the main hypersphere HS used by the *HyperSpheres* strategy has radius one, its maximum volume

[12] http://www.r-project.org

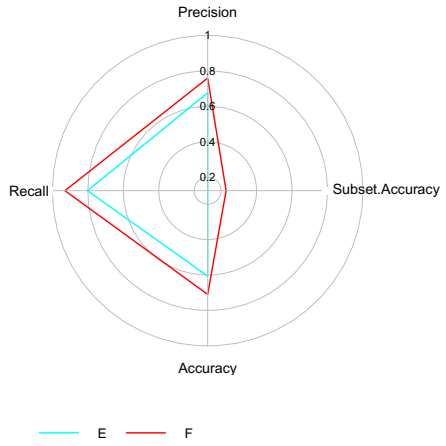


(a) Example-based measures

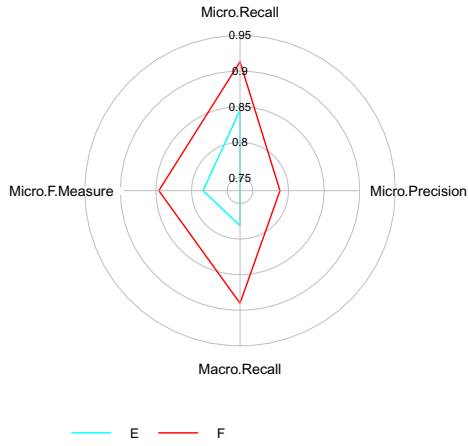


(b) Label-based measures

Fig. 7. Spider graphs for the A, B, C and D datasets



(a) Example-based measures



(b) Label-based measures

Fig. 8. Spider graphs for the E and F datasets

is reached for $M = 5$, decreasing afterwards. This problem is well described in [12]. On the other hand, in the *HyperCubes* strategy the volume of the main hypercube HC always increases with the dimensionality. Both cases are related to the curse of dimensionality.

5 Conclusion

This work proposes two strategies to generate synthetic multi-label datasets. Both strategies were implemented in the framework *Mldatagen*, which is publicly available, improving the lack of publicly available multi-label generators. Although these

strategies are based on one initial proposal described in [18] for a specific problem, we have extended the initial proposal by offering the user of *Mldatagen* the flexibility to choose the number of unimportant features and the noise level for the datasets to be generated, as well as avoiding the occurrence of empty multi-labels. Furthermore, *Mldatagen* outputs the generated datasets in the Mulan format, a well-known framework for multi-label learning, which is frequently used by the community.

To illustrate *Mldatagen*, multi-label classifiers were built from six synthetic datasets. The results suggest that the datasets generated by the *HyperCubes* strategy provide better classification results than the ones based on *HyperSpheres*. In fact, despite both strategies being sensitive to the curse of dimensionality, the volume of a hypercube always increases with the dimensionality, showing better behavior than a hypersphere, whose volume decreases from one dimension upwards. As future work, we plan to implement more strategies into the *Mldatagen* framework and make them available to the community.

References

[1] Agrawal, R., S. Ghosh, T. Imielinski, B. Iyer and A. Swami, *An interval classifier for database mining applications*, in: *International Conference on Very Large Databases*, 1992, pp. 560–573.

[2] Bolón-Canedo, V., N. Sánchez-Maroño and A. Alonso-Betanzos, *A review of feature selection methods on synthetic data*, Knowledge and Information Systems **34** (2013), pp. 483–519.

[3] Chou, S. and C.-L. Hsu, *MMDT: a multi-valued and multi-labeled decision tree classifier for data mining*, Expert Systems with Applications **28** (2005), pp. 799–812.

[4] Clare, A. and R. D. King, *Knowledge discovery in multi-label phenotype data*, in: *European Conference on Principles of Data Mining and Knowledge Discovery*, 2001, pp. 42–53.

[5] Dendamrongvit, S., P. Vateekul and M. Kubat, *Irrelevant attributes and imbalanced classes in multi-label text-categorization domains*, Intelligent Data Analysis **15** (2011), pp. 843–859.

[6] Dietterich, T. G., *Exploratory research in machine learning*, Machine Learning **5** (1990), pp. 5–10.

[7] dos Reis, D. M., E. A. Cherman, N. Spolaôr and M. C. Monard, *Extensions of the multi-label learning algorithm BRkNN (in Portuguese)*, in: *Encontro Nacional de Inteligência Artificial*, 2012, pp. 1–12.

[8] Hastie, T., R. Tibshirani and J. Friedman, "The Elements of Statistical Learning," Springer-Verlag, 2001.

[9] Langley, P., *Crafting papers on machine learning*, in: *International Conference on Machine Learning*, 2000, pp. 1207–1212.

[10] Liu, H. and H. Motoda, "Computational Methods of Feature Selection," Chapman & Hall/CRC, 2008.

[11] Noh, H. G., M. S. Song and S. H. Park, *An unbiased method for constructing multilabel classification trees*, Computational Statistics & Data Analysis **47** (2004), pp. 149–164.

[12] Richeson, D., *Volumes of n-dimensional balls* (2010).
URL http://divisbyzero.com/2010/05/09/volumes-of-n-dimensional-balls

[13] Spyromitros, E., G. Tsoumakas and I. Vlahavas, *An empirical study of lazy multilabel classification algorithms*, in: *Hellenic conference on Artificial Intelligence* (2008), pp. 401–406.

[14] Tsoumakas, G., Personal communication (2013).

[15] Tsoumakas, G., I. Katakis and I. P. Vlahavas, *Mining multi-label data*, in: O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, Springer, 2010 pp. 667–685.

[16] Tsoumakas, G., E. S. Xioufis, J. Vilcek and I. P. Vlahavas, *Mulan: A java library for multi-label learning*, Journal of Machine Learning Research **12** (2011), pp. 2411–2414.

[17] Younes, Z., F. Abdallah, T. Denoeux and H. Snoussi, *A dependent multilabel classification method derived from the k-nearest neighbor rule*, EURASIP Journal on Advances in Signal Processing **2011** (2011), pp. 1–14.

[18] Zhang, M.-L., J. M. Peña and V. Robles, *Feature selection for multi-label naive bayes classification*, Information Sciences **179** (2009), pp. 3218–3229.