Beyond global and local multi-target learning

Márcio Basgalupp[1]

Institute of Science and Technology, Federal University of São Paulo, São José dos Campos, SP, Brazil, E-mail: basgalupp@unifesp.br

Ricardo Cerri

Department of Computer Science, Federal University of São Carlos, São Carlos, SP, Brazil, E-mail: cerri@ufscar.br

Leander Schietgat

Department of Computer Science, KU Leuven, Leuven, Belgium
Artificial Intelligence Lab, Vrije Universiteit Brussel, Brussels, Belgium, E-mail: leander.schietgat@vub.be

Isaac Triguero

Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, Nottingham, UK, E-mail: isaac.triguero@nottingham.ac.uk

Celine Vens

Department of Public Health and Primary Care, KU Leuven Kulak, Kortrijk Belgium, E-mail: celine.vens@kuleuven-kulak.be

---

Abstract

In multi-target prediction, an instance has to be classified along multiple target variables at the same time, where each target represents a category or numerical value. There are several strategies to tackle multi-target prediction problems: the local strategy learns a separate model for each target variable independently, while the global strategy learns a single model for all target variables together. Previous studies suggested that the global strategy should be preferred because (1) learning is more efficient, (2) the learned models are more compact, and (3) it overfits much less than the local strategy, as it is harder to overfit on several targets at the same time than on one target. However, it is unclear

[1]Authors' list follows alphabetical order. All authors contributed equally.

whether the global strategy exploits correlations between the targets optimally. In this paper, we investigate whether better results can be obtained by learning multiple multi-target models on several partitions of the targets. To answer this, we determined alternative partitions using an exhaustive search and a genetic algorithm strategy.

We used decision trees and random forests as base models. The results show that it is possible to outperform global and local approaches, but finding a good partition without incurring in overfitting remains a challenging task.
Keywords: multi-target regression, multi-label classification, predictive clustering trees, random forests, genetic algorithms

1. Introduction

Traditional prediction problems deal with a set of instances that have a single target value associated with them. This target attribute can be nominal (classification problem) or numeric (regression problem). Several real-life problems, however, have a *set* of target attributes: instead of a single property, one is interested in predicting multiple properties. This setting is known as multi-target prediction. Applications include the prediction of river water quality parameters from bioindicator data [1], olfaction prediction in molecules [2], and estimation of energy performance in residential buildings [3]. A very related setting is multi-label learning, where multiple class labels are to be predicted for the instances. A multi-label prediction problem can be viewed as a binary multi-target prediction problem [4, 5], where each possible label becomes a target. Applications include protein function prediction [6, 7], subcellular localization [8], document annotation [9], and audio classification [10].

Generally, there are two approaches to tackle multi-target problems [11, 4]. In the first approach, called *local*, one converts the problem into a set of single-target problems, and applies a standard prediction model. In the *global* approach, one keeps the multi-target structure and applies a multi-target prediction model to deal with all targets simultaneously, possibly exploiting relation-

ships between the targets. Several machine learning techniques have been extended to multi-target problems, such as decision trees [11], neural networks [7], support vector machines [12], and novel deep learning approaches [13] for image classification problems.

Typically, global-based methods are preferred in terms of efficiency and model size, while local-based methods benefit from their simplicity. In terms of predictive performance, the global approach has typically reported equal or better results than the local approach [14, 4]. However, it is unclear how this strategy is exploiting the relationships between targets.

Most studies in multi-target learning apply a global or local learning approach. However, it is not clear whether these strategies are optimal. As some targets may be more similar than others, it may well be that models learned on a different partitioning than the global or local approach have a better performance. These correlations can be useful for exploring different patterns across different partitions of the targets. This is exactly what we investigate in this study. The few recent studies that have addressed the same topic [15, 16, 17] have investigated it in a randomized ensemble setting. Here, we want to answer the following question:

> *Is there a partitioning of targets that, when each subset in this partition is treated as a separate prediction problem, outperforms the two extremes of global and local learning?*

To answer this question, we perform an empirical study over 16 multi-target prediction datasets. For this, we have to rely on methods that have global and local-based versions. That is why we have chosen the Clus framework [11], which includes decision tree-based local and global variations implemented as Predictive Clustering Trees (PCTs) [18]. Recall that any other predictor that has local and global versions could be used, however our focus is not on the predictors themselves but on how to maximize a given predictor's performance by finding a suitable "in-between" local and global partitioning for it. An in-between local and global partitioning can be solved by a combination of local and

global-based predictors. The predictors are used only to evaluate the partitions found. Note that, since different models explore the target space differently, an in-between local and global partitioning for a model X can be different from an in-between local and global partitioning for a model Y. However, rather than trying to find the best overall partition, considering different models from the literature, the goal of this paper is to show that it is possible to maximize a given predictor's performance by finding a suitable in-between local and global partition for it.

The main idea is that the learning technique and experimental methodology (e.g., parameter optimization) stays exactly the same, while only the partitioning strategy is different. In order to find the optimal partition, when possible, we perform an exhaustive search (ES) over all partitions of targets. However, the number of possible partitions rapidly grows with the number of targets. As an example, a dataset with 20 targets already leads to a search space of 5,832,742,205,057 partitions. If the dataset has too many targets, making an ES unfeasible, we employ a genetic algorithm (GA) that we created in order to efficiently search in a population of partitions for the optimal one. As GAs have been successfully applied to many non-convex combinatorial optimization problems [19], are relatively easy to code and have built-in mechanisms to avoid arriving in local optima, they are our preferred learning method here.

Our results show that, in most cases, models based on different partitions of targets that the ones used in global and local learning obtain better results. However, it turns out that, in the same way as the local strategy may overfit on the target variables, trying to tune the optimal partitioning may lead to overfitting.

The main contributions of this paper are the following: (1) an experimental analysis over 16 multi-target datasets, showing that in some cases, the in-between strategy obtains superior results, (2) the introduction of a genetic algorithm that can search for the optimal partition, which is less prone to overfitting than the exhaustive search, and (3) it sheds new light on approaches between global and local learning, showing that this is an area worth investi-

gating further. Our focus is thus, rather than introducing a novel multi-target predictor, to show that it is possible to use the existing predictors with different partitions of the target space. While trying to find relationships between different targets is not new in the multi-target learning literature, the exploration of new ways of partitioning the target space to better represent the target relationships has never been done before in the literature. As the number of partitions is exponential in the number of target variables, this is an extremely challenging task.

The remainder of this paper is organized as follows. Section 2 describes recent multi-target methods from the literature. Section 3 formally defines the multi-target task, and describes how to solve it using global and local-based decision tree methods. Section 4 describes all methods and the proposed methodology. Section 5 presents the experimental settings and Section 6 provides all experiments performed and a detailed discussion about the results. Finally, Section 8 summarizes the main conclusions and future research directions.

2. Related Work

Multi-target prediction methods can be divided into classification and regression problems. In multi-target classification, an instance is classified along many targets simultaneously, and each target can have multiple categorical values. If the targets are binary, then the problem is called multi-label classification, which is the most widely studied setting in the related work (Section 2.1). In the case the targets have continuous values, the problem is called multi-target regression (Section 2.2). Section 2.3 addresses multi-task learning.

2.1. Multi-label classification

As previously stated, multi-label classification can be considered as an instantiation of multi-target prediction, where each target is binary. According to Tsoumakas et al. [20], multi-label classification methods can be divided in problem transformation methods and algorithm adaptation methods, depending on how the methods treat the classification problem.

A well-known and often used problem transformation method is binary relevance (BR), where a separate classifier is trained for every label independently. BR corresponds exactly to what we call a local model in this paper, dividing the original problem into multiple binary ones. The final prediction is given by the combination of the binary classifiers associated to each label. On the other hand, algorithm adaptation methods solve multi-label classification problems directly by predicting all labels together, possibly exploiting the fact that certain labels are correlated. This corresponds exactly to what we call a global model in this paper. While local methods allow the use of any conventional classification algorithm as base classifier, global methods require to use classifiers specifically designed to predict multiple classes simultaneously.

In the multi-label classification literature, quite some attention has been given towards exploiting label relationships, given that this is a key factor to good multi-label prediction models. Some studies exploit label relationships that are given by the problem domain, such as in hierarchical multi-label classification. In this setting, a hierarchical taxonomy of labels is given, typically representing an 'is-a' relationship. Barutcuoglu et al. [21] use a local approach to make predictions for each label separately, and then use a Bayesian network modeling the hierarchical relations to make the predictions consistent with the constraints implied by the hierarchy. Vens et al. [11] propose a global approach based on predictive clustering trees. The leaf nodes return a numerical vector of predictions, where a higher value indicates more confidence to predict a particular label. The vector automatically fulfills the hierarchical constraint as it is obtained by averaging the label vectors of the training instances falling into the given leaf. Cerri et al. [7] proposed a local approach using neural networks, where a multi-layer perceptron was assigned to each hierarchical level, and trained to predict classes from its associate level. Predictions made in one level were used to augment the feature vectors of the instances from the next level, trying to incorporate label relationships during training. Masera et al. [22] use a global approach based on neural networks. Their method Adjacency Wrapping matriX (AWX) employs a single model where the underlying hierarchy is mapped onto

the loss function.

In the absence of a given set of label relations, several studies use data mining techniques to extract a set of relations from the training label vectors, and then exploit these relations in the learning phase. In this direction, Madjarov et al. [23] construct a hierarchical clustering over the label space, and use this hierarchy to model the learning task as a hierarchical multi-label classification problem, solved by global or local models. Papagiannopoulou et al. [24] use an Apriori-like algorithm to extract label dependencies and then improve the local model by encoding the label correlations into a Bayesian network. Abreu et al. [25] try to extract relations from label vectors considering the distances between instances in the feature space. The label vectors of similar instances are then used to obtain a prototype vector, which in turn augments the instances' feature space used for training. Prati et al. [26] used a biclustering algorithm, where each bicluster was considered a new binary feature. These were then used to augment the instances' feature vectors.

Still trying to extract relations from the label vectors, Cherman et al. [27], Read et al. [28] and Dembczynski et al. [29] used the instances' classes to complement feature vectors, aiming at incorporating label dependencies in the learning process. Huang and Zhou [30] clustered the instances and calculated similarities within each cluster. These were used to augment the original feature vectors. Yu et al. [31] used neighborhood rough sets to find the possibly related labels for an instance, excluding all unrelated ones. Label pairwise correlation was used by Spolaôr et al. [32] to construct new binary labels to augment the original feature vectors. Huang and Zhou [33] proposed an algorithm to try to explore local correlations. By local, the authors mean that label correlation may be shared by only a subset of instances. They encoded this information into a vector, used to augment the original feature space for each instance.

Joly [15] and Breskvar [16] learn an ensemble of multi-target decision trees, where each tree uses only a random projection, resp. random subspace, of the original output space. Both studies show a decreased learning time in combination with predictive performance gains, when compared to a global method.

Breskvar et al. also extended their work towards multi-target regression [17]. While these studies share with us the idea of working in between the global and local approaches, they do so in an ensemble way. In contrast, in this article, we are interested in learning a single partition of the labels, which is more interpretable and can yield more insights in the problem domain. Szymanski et al. [34] create a single partition of labels by constructing a label co-occurrence graph and applying community detection methods on them. The label subsets are subsequently used in a label powerset approach, and hence this approach can be seen as a data driven strategy to select label subsets in the random k-labelsets (RAkEL) method [35]. Since the method builds on the label co-occurrence graph, it is not applicable to multi-target regression problems.

2.2. Multi-target regression

Considering multi-target regression tasks, Spiroumitros-Xioufis et al. [36] adapted two multi-label classification methods proposed by Read et al. [28], and Godbole and Sarawagi [37], to be applied to the context of multi-target regression. The first proposal is called Multi-Target Regressor Stacking (MTRS). The method is divided into two stages: the first one trains m single-target models, one for each target, and the second one trains a set of m meta-models. Each meta-model is trained on a transformed training set, consisting of the original feature vectors augmented by predictions. These predictions are obtained from the m models on the first stage. A second proposal is called Regressor Chains (RC), based on the popular Classifier Chains (CC) method [28]. Similar to CC, RC trains a single model for the first target obtained from an ordered set of targets. The subsequent models are trained into a transformed dataset, where the instance vectors contain the original features, augmented by the previous targets in the chain.

Spiroumitros-Xioufis et al. [36] also pointed out a problem in CC and RC, violating the independent and identically distributed assumption. Particularly in RC, the distribution of the actual target values used in training can diverge radically from the distribution of the predicted values used in the testing phase.

Thus, the authors proposed Regression Chains Corrected (RCC) which uses the predictions of the previous regression models in the chain instead of the true values of the target variables.

A number of studies have also tried to include target correlations in the local approach. Tsoumakas et al. [5] replaced the target space by a (typically increased) space that is obtained by taking random linear combinations of the original target attributes. Afterwards, they constructed a local classifier on the newly constructed targets. Spiroumitros-Xioufis et al. [38] increased the feature space by applying a stacking or chaining strategy. In the stacking strategy, a two-level approach is used. First, a set of single-target models is constructed. Then, instead of using them directly to make predictions, their predictions are added to the training set as extra features, and a second set of single-target models is constructed on the augmented feature set. In the chaining strategy, first a random chain (permutation) of the targets is fixed. Then the targets are treated in this order and for each target a single-target classifier is learned, using the original features augmented with the predictions for the previously seen targets.

Piccart et al. [39] learn which is the optimal set of targets to use in a multi-target model in order to obtain an optimal prediction of a particular target. They learn this optimal set by using a greedy search method estimated from data and show improvement on a number of multi-target regression methods.

2.3. Multi-task learning

Jacob et al. [40] present an optimization procedure for linear models that partitions tasks together in the context of multi-task learning. They show that this procedure outperforms a global model on two synthetic and one real-life datasets.

Inspired by the methods proposed by Spiroumitros-Xioufis et al. [36], Melki et al. [41] proposed two methods to exploit correlations among target variables. The authors proposed to build single-target soft-margin non-linear support vector regressors (NL-SVR) for each target variable. Then, similarly to

Spiroumitros-Xioufis et al., they proposed to construct a series of random chains of base-line SVR, creating an ensemble model named SVR Random Chains (SVRRC). Because the number of possible chains can factorially increase, a second proposal creates a single chain, maximizing the correlations among the target variables. These correlations were imposed on the order of the chain, ensuring that each appended target provides additional knowledge when training on the next one.

3. Multi-target Learning

In this section, we provide a formal definition of the multi-target learning task, and also present how to build local and global models using the predictive clustering trees (PCTs) framework [11].

3.1. Formal Problem Description

Formally, a multi-target prediction task can be defined as follows [4]:

Given:

- a feature space X consisting of tuples of discrete and/or continuous-valued features, i.e., $\forall X_i \in X, X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, with D the size of a tuple (number of features);
- a target space Y, consisting of tuples of discrete or continuous-valued target variables, i.e., $\forall Y_i \in Y, Y_i = (y_{i1}, y_{i2}, \ldots, y_{iT})$, with T the size of a tuple (number of targets);
- a set of instances E, each instance being a pair of tuples from the feature and target spaces, i.e., $E = (X_i, Y_i) | X_i \in X, Y_i \in Y, 1 \leq i \leq N$, with N the number of instances;
- a quality criterion q, usually rewarding models with high prediction performance and low complexity;

Find: a function $f : X \rightarrow Y$ such that f maximizes q.

When the targets Y are discrete, the task is called multi-target classification, and when they are continuous, it is called multi-target regression. Multi-label classification can be seen as an instance of multi-target prediction, where the target variables are binary. In this article, the function f is based on decision trees from the predictive clustering trees (PCT) framework. The use of PCTs makes it easy to evaluate partitions since it has local and global versions. Given that an in-between local and global partition can be solved by a combination of local and global-based predictors, it is important to have a predictor with both local and global-based versions. If a given predictor does not have, for example, a global version, we cannot evaluate a partition having a subgroup composed of more than two targets. As alternatives to PCTs, we could use any method having local and global versions, such as KNN [42]. In this case, we would probably find an in-between local and global partition different from the one found when using Predictive Clustering Trees. This does not affect our main conclusion, which is stating that an in-between local and global partition can lead to better results than the conventional local and global ones.

3.2. Learning a Global Decision Tree Model with the PCT Framework

Multi-target prediction tasks can be handled by the predictive clustering trees (PCTs) framework [11, 14]. This framework views a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the Clus system, which is freely available for download at `https://dtai.cs.kuleuven.be/clus`.

PCTs can be induced with a standard top-down induction of decision trees (TDIDT) algorithm. It takes as input a set of instances and outputs a decision tree. The heuristic that is used for selecting the tests to be put in a tree node is the reduction in variance caused by partitioning the instances. By maximizing the variance reduction, the cluster homogeneity is maximized and the predictive

performance is improved.

The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task, such as multi-label classification or multi-target regression.

In order to instantiate the PCT algorithm for predicting multiple targets, first, each instance is instantiated with a *vector* of target values. The vector contains nominal, numeric or binary values for multi-target classification, multi-target regression, or multi-label tasks, respectively. The ith component of the vector contains the instance's value for the ith target. The variance of a set of instances E with numeric or binary targets is defined as the averaged squared distance between each instance's class vector and the set's mean class vector:

$$Var(E) = \frac{1}{|E|} \sum_{E_i \in E} d(L_i, \overline{L})^2$$

with d the Euclidean distance. In case of nominal targets, the variance is computed as the sum of the Gini indices or entropies of the target variables.

The prototype function computes the mean (or majority class for nominal targets) of the target vectors of the instances in the leaf.

3.3. Learning a Local Decision Tree Model with the PCT Framework

The local approach we investigate here divides the multi-target problem into T single-target problems, with T being the number of targets in the original problem. Individual decision trees are generated for each target, and predictions are obtained combining all individual outputs.

4. Beyond Learning Global and Local models

The local learning strategy is simple and can be done with any standard predictive machine learning method. It allows each model to focus on one specific target. In contrast, global strategies learn a single model that needs to address

all targets at once, which seems a more difficult learning task. However, the literature suggests that global models have a predictive performance advantage over local models [11]. In addition, they are computationally more efficient to learn and make predictions, and the size of the global model is usually much smaller than the sum of the sizes of the local models [11]. In this article, we hypothesize that in practice, approaches in-between the global and local ones may be preferable in terms of predictive performance and model size. Consider the example in Fig. 1. It shows eight instances with two features and four targets. Labels A and B perfectly correlate (coincide) with each other and with feature X_1, the same holds for targets C, D and feature X_2. There is no correlation between the two groups of targets. A local learning strategy will fit four models, it will not consider the existing correlation between the targets. Any noise in the target space will be hard to detect and may easily lead to overfitting. The global approach constructs a single model, with three internal nodes in total. However, it also has some weaknesses: it contains a repetition of the subtree rooted in X_2. As a consequence, each of these subtrees is constructed on a smaller training set than necessary, leading to a smaller number of samples per leaf to compute the prototype, and hence to a reduced predictive performance. The last strategy shown constructs two trees, each making predictions for two targets. It detects the correlation in the target space and exploits this to build a model with only two internal nodes. This approach can be situated in between global and local models.

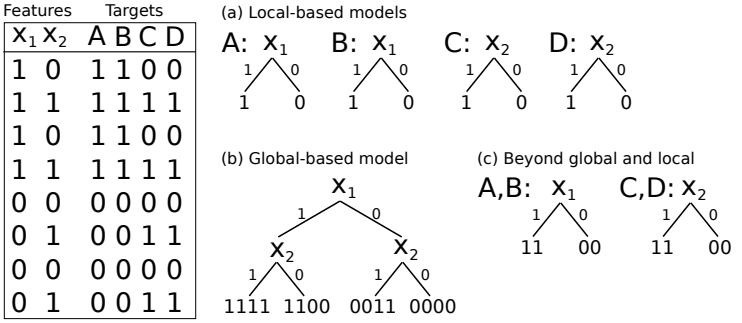


Figure 1: The global, local, and partition-based model on a toy dataset.

In this paper we want to empirically study whether the partition approach, i.e. partitioning the targets and learning a (global) model for each subset, can improve the predictive performance over the purely global or local models. This allows us to explore alternative/feasible partitions of the output space which may be more appropriate for a given problem. In order to find the best partitioning, we use an exhaustive approach where possible. When the number of targets is too high, we need to use a heuristic approach. One possibility would be to cluster the target space if one knows the number of clusters in advance. Here, we want to learn this number as part of the search process and, therefore, we propose to use a genetic algorithm.

4.1. Exhaustive Search

A difficulty in finding the optimal partition from the target set is that the number of possible partitions of a set (also called the Bell number) grows very quickly with the number of elements in the set. For instance, a set of 10 elements has 115,975 possible partitions, while a set of 16 elements has 10,480,142,147 partitions. However, noting that a set of n elements has 2^n possible subsets, this means that the 10,480,142,147 partitions are composed of 65,536 subsets only, which makes it possible to conduct an exhaustive search over all partitions.

More precisely, we propose to train a multi-target classifier on each of the 2^n possible subsets of targets and store the predictions made for these targets on a hold-out validation set. Then, in our implementation, we perform an efficient generation of all set partitions using the procedure outlined in [43], and for each partition combine the corresponding predictions. The partitioning that gives the best predictive performance over the validation set is returned. Using this partitioning, we learn a model using both training and validation sets together, and obtain predictions for the test set. Nevertheless, when the number of targets n is too large, learning 2^n models might not be feasible in terms of runtime. In the experiments of Section 5, we perform an exhaustive search for $n \leq 16$ targets.

4.2. A Genetic Algorithm to find the optimal partition

Searching for the best partitioning of target outputs amongst a great number of potential combinations can be seen as a combinatorial optimization problem. GAs have successfully been applied in these problems for many years [19]. Moreover, GAs are easy to code and more suitable than other meta-heuristics for discrete optimization problems. Another advantage of using GAs here is that they are very suitable for problems with many local optima. This is our case since we can have different in-between partitions resulting in better results than the traditional local and global-based models.

In this section we propose a GA to solve the problem of finding an optimal partitioning of target outputs. In what follows, we present the components of the GA, such as individual representation, population initialization, simulated annealing, genetic operators and fitness function employed.

4.2.1. Individual Representation

Figure 2 illustrates how we code the individuals in the GA. Each individual is represented as a vector where each position (gene) corresponds to a target, and receives a number corresponding to a partition subset (gene value). In the example, we have three subsets (target groups): 1, 2 and 3. Targets at positions 1, 3, and 4 are assigned to groups 2, 1 and 3, respectively. For each group of targets, a Decision Tree model learning those targets simultaneously is then induced.

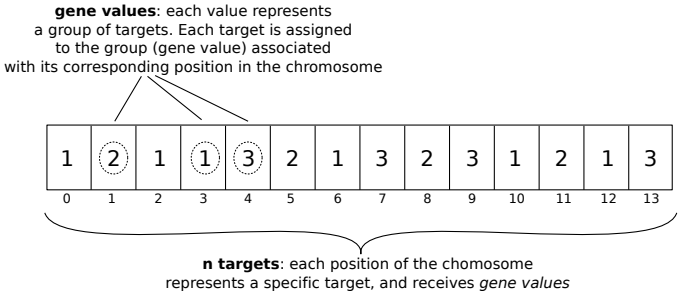


Figure 2: Genetic Algorithm: individual representation for 14 target variables and three target subsets.

4.2.2. Population Initialization

Although a totally random individual generation based on a Normal Distribution is the most common way for generating the initial population in GA applications, we believe that by incorporating task-specific knowledge for multi-target learning we can derive better solutions, or at least equally good solutions in less generations. Our strategy for incorporating task-specific knowledge into the GA is described as follows.

For each individual, a number of groups g is randomly generated according to a Poisson distribution [44]. This integer number g ranges from 2 to $n-1$, where n is the number of targets. Then, the numbers from 1 to g are randomly distributed to the individual's genes, ensuring that the individual represents a solution with g groups. It is important to mention that a Poisson distribution is essential to increase the diversity of the population since there are many more ways to combine groups of $n/2$ targets than 2 or n targets. By using a Poisson distribution, most of the random numbers are generated from $\frac{n}{3}$ to $\frac{2n}{3}$, which is not necessarily true when using Normal distribution. Finally, also the global ($g = 1$) and local ($g = n$) solutions are included in the initial population.

4.2.3. Selection of Individuals and Genetic Operators

The proposed GA uses tournament selection, a popular and effective selection method. In tournament selection, individuals are first ranked according to their fitness. Then, they are selected based on the value of their rank positions. This method overcomes the scaling problems of fitness-proportional assignment, e.g., premature convergence when few individuals with very high fitness values dominate the rest of the population. It also implements the elitism technique, which means it preserves a number of individuals based on their fitness values.

To obtain off-spring the method performs a parameterized uniform crossover, where every gene is crossed over independently with a certain probability. It is important to note that this probability is the per-gene crossover probability, not the probability that the entire individual will be crossed over.

After the crossover, a uniform mutation (uniform gene randomization) is ap-

plied, simply setting a gene to a random value between its minimal and maximal allowed values. Uniform operators will create groups that will be very different from their parents if their parents are not similar, contributing to the diversity. If its parents are similar, the offspring will be similar to its parents, encouraging the convergence.

4.2.4. Fitness Function

We use two different fitness functions, depending on whether regression or classification is being performed. These fitness functions are computed on a separate validation set. For regression we use the mean absolute error (MAE), and for classification we use the area under the ROC curve (AUROC). The MAE is presented in Equation 1, where y_{ij} and $\hat{y}_{ij}$ are, respectively, the desired and predicted outputs for target j given instance i, and N and T denote the number of instances and number of targets.

$$MAE = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{T}|\frac{y_{ij} - \hat{y}_{ij}}{T}| \tag{1}$$

A ROC curve is produced applying threshold values in the interval $[0, 1]$ to the (real valued) outputs of the classifiers. This results in different true positive rates (TPR) and false positive rates (FPR), one pair of values for each threshold used. The combination of these points forms a ROC curve, and the area under the curve (AUROC) is calculated. The FPR and TPR are macro-averaged over the targets. This allows to store and efficiently combine the AUROC values for each target.

4.2.5. Simulated Annealing Procedure

In order to improve the quality of our best solution generated so far, we apply a Simulated Annealing (SA) strategy [45] on the best individual of the population.

The SA strategy works with a single solution and performs a global search by evaluating neighboring solutions. If a neighbor is better than the current solution, it replaces the latter. On the other hand, a worse solution can be accepted

according to an acceptance function. Such behavior allows the SA to escape from non-promising regions and explore the search space by moving to worse neighborhoods. The acceptance probability function considers a parameter T, named temperature, which decreases through the SA iterations. At high temperatures, the probability tends also to be high, while low temperatures usually result in a low acceptance probability, turning the SA into a hill-climbing local search approach. Thus, this cooling factor must be tuned, so that, the SA has enough iterations to explore the search space.

Our SA strategy (Algorithm 1 line 11) is applied at each $SAgen$ generations. It makes a copy of the best individual of the population and applies random mutations to it, trying to avoid local optima. We perform $SAmax$ iterations of the SA, decreasing the number of mutated genes as a temperature parameter t is decreased. After $SAmax$ iterations of the SA procedure, the generated individual replaces the worst individual in the current population.

4.2.6. Evolutionary Process

Algorithm 1 presents the main procedure implemented by the GA in order to find the best target partition. The procedure implements a standard evolutionary strategy, with the previously described elitism, tournament selection, mutation and crossover operators.

The algorithm initially obtains the number of targets from the training dataset. It then randomly generates a population of size p, where each individual represents a partition of the targets. For each subset of the partition, a multi-target decision tree model is induced, or the corresponding result is taken from stored results, if a model has already been induced for that subset earlier. The fitness of an individual is then calculated based on the performance of the combined models for the complete partition on a hold-out validation set. In order to generate a new population, the genetic operators are applied.

Algorithm 4.1: Genetic algorithm to search for the best target partition.

input : X^{train}: train dataset
X^{valid}: validation dataset
p: population size
$gMax$: maximum number of generations
$gBest$: max number of generations without changing the best individual
t: tournament size
e: elitism size
pc: crossover rate
pu: gene crossover rate
pm: mutation rate
pg: gene mutation rate
$SAgen$: each $SAgen$ generations, apply Simulated Annealing
$SAmax$: max number of iterations in SA algorithm

output: $best$: best partition

Function: Evolution(X^{train}, X^{valid}, p, $gMax$, $gBest$, t, e, pc, pu, pm, pg, $SAgen$, $SAmax$)

$numTargets \leftarrow getNumTargets(X^{train})$
$currentPopulation \leftarrow generatePopulation(numTargets, p)$
$calculateFitness(currentPopulation, X^{valid})$
$best \leftarrow getBestPartition(currentPopulation)$
$lastBest \leftarrow best$
$g \leftarrow 1$
$gLast \leftarrow 0$
while $g \leq gMax$ *and* $gLast \leq gBest$ **do**
    if p *mod* $SAgen = 0$ **then**
        $simulatedAnnealing(best, SAmax)$
    $parental \leftarrow tournamentSelection(currentPopulation, t)$
    $offspring \leftarrow uniformCrossover(parental, pc, pu)$
    $offspring \leftarrow mutation(offspring, pm, pg)$
    $currentPopulation \leftarrow$
      $replacementWithElitism(currentPopulation \cup offspring, p, e)$
    $calculateFitness(currentPopulation, X^{valid})$
    $best \leftarrow getBestPartition(currentPopulation)$
    if *best is better than lastBest* **then**
        $lastBest \leftarrow best$
        $gLast = 0$
    else
        $gLast \leftarrow gLast + 1$
    $g \leftarrow g + 1$
return { best }

5. Experimental framework

This article presents a thorough experimental analysis considering both classification and regression settings. In particular, we aim to answer the following research questions:

- Q1 Is it possible to outperform the global or local models by considering other partitions of target variables?
- Q2 How close does the GA come to the optimal solution (exhaustive approach)?

The first question has been addressed by Breskvar et al. [16, 17] for the ensemble setting, where a forest of trees is constructed, each tree focusing on a random subset of targets. Here, we rather focus on a single partition. As base classifiers we consider decision trees and random forests, however, in contrast to Breskvar et al., the forests were constructed with a single target partition.

First, the datasets used in the experiments are explained. The measures employed to evaluate the performance of the algorithms, and the statistical tests conducted to contrast the results are also presented, together with the configurations used in the GA and decision trees. All the code and datasets used in this study are available at `https://github.com/rcerri/Clus-Hyper-Code`.

5.1. Datasets

To evaluate the performance of the proposed approach, we are interested in multi-target datasets with a relatively high number of outputs, in which correlations between target outputs can make a difference in comparison with local and global approaches. Thus, we have selected a total of 16 datasets (seven for classification, and nine for regression) with at least six output labels, that come from different domains such as audio, images, music, bioinformatics and human perception. The number of examples in these datasets ranges from a few hundreds ($\simeq$200) up to more than 40 thousands examples. In terms of features, the used datasets contain a number in the [16-4884] interval. All

these datasets, with the exception of the human olfactory perception one [2], are publicly available and can be downloaded from the Mulan website[2]. Tables 1 and 2 show the main properties of the datasets used. The exact training, validation and test partitions are available on our github.

All these datasets, except for olfaction, have been split into training (8 folds), validation (1 fold) and test (1 fold) sets, following a 10-fold cross validation strategy. The partitions were created using the iterative stratification strategy proposed by Sechidis et. al. [46]. After calculating the desired number of instances in each subset, this strategy iteratively examines each instance in order to select an appropriate subset for distribution. The strategy is implemented within the "utiml" R Package [47]. For the olfaction dataset, we used the original train/test split as used in the corresponding data challenge [2].

Table 1: Classification datasets properties: number of instances $|N|$, number of features $|D|$, number of targets $|T|$.

Dataset	domain	$\|N\|$	$\|D\|$	$\|T\|$
birds	audio	645	260	19
corel5k	images	500	499	374
emotions	music	593	72	6
flags	images	194	19	7
genbase	biology	662	1186	27
mediamill	video	43907	120	101
yeast	biology	2417	103	14

5.2. Evaluation Measures

Following the same evaluation measure used in the fitness function of the proposed GA, we evaluate the results on the multi-label classification datasets with the area under the ROC curve (AUROC). This measure allows us to evaluate the methods independently from a particular prediction threshold, since the choice of an "optimal" value for the threshold is a difficult task. Low values lead to many predictions for each instance, while large values lead to very

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

Table 2: Regression dataset properties: number of instances $|N|$, number of features $|D|$, number of targets $|T|$.

Dataset	domain	$\|N\|$	$\|D\|$	$\|T\|$
water-quality	ecology	1060	16	14
oes10	employment survey	403	298	16
oes97	employment survey	334	263	16
rf1	ecology	9125	64	8
rf2	ecology	9125	576	8
scm1d	supply chain mgmt.	9803	280	16
scm20d	supply chain mgmt	8966	61	16
osales	sales	639	413	12
olfaction	human olfactory perception	476	4884	1029

few predictions. To evaluate the regression datasets, we use the mean absolute error (MAE). We use the macro-averages of these measures, which means that we average them over every label.

5.3. Parameters and algorithms

Decision trees and random forests were used with their default parameter values. We used 50 trees in the random forest. To find the best partition to be used by the prediction algorithms, we used the GA as well as the proposed exhaustive search (when possible).

The GA was implemented within the Java-based Evolutionary Comptuation Research System (ECJ) framework[3], which supports multi-threads. The experiments were executed using 64 threads (the maximal number of cores available in our machines). This means that 64 individuals were evaluated in parallel.

Table 3 presents the parameter values used in the GA. An optimization method could be used to tune the parameters, but it would be very time-consuming for this application, since we would need a search/optimization method (greedy search, for example) to optimize another search/optimisation method (GA). Then, the parameters were set based on the authors' experience with GA and Clus, ensuring that it would converge to good solutions in a fea-

[3] `https://cs.gmu.edu/~eclab/projects/ecj/`

sible time. By choosing $p = 100$ and $gMax = 100$, the user previously knows that the GA will run Clus up to $10,000$ times. By using $gBest = 30$ as a stop criterion we allow the GA to stop the evolution earlier, avoiding unnecessary iterations after it has converged. The other parameters, namely $t = 2$, $e = 1$, $pc = 0.70$, $pu = 0.25$, $pm = 0.40$, $pg = 0.20$, and $SAgen = 10$, do not have a lot of influence in the final results besides increasing/decreasing the time to converge.

After termination of the GA or ES, the classifier is trained again with the best partitioning considering the entire dataset (training + validation). The induced model is then applied to the test set.

Table 3: Parameter values used in the Genetic Algorithm.

p	100
gMax	100
gBest	30
t	2
e	1
pc	0.70
pu	0.25
pm	0.40
pg	0.20
SAgen	10
SAmax	50

The exhaustive search is also used to validate the performance of the GA, comparing the performance of the partitions found by the GA against the optimal partitions.

In addition, the exhaustive approach is used as an "Oracle" to determine if there exists a better target partition than the best partition returned by the compared methods. To do this, training and validation sets are combined to build a model, and the partition that provides the best results on the *test set* is output by the Oracle. The aim of including this Oracle is two-fold: (1) this will help us to answer the first research question, is there any better partition beyond global and local?; (2) this will also allow us to analyze whether the proposed framework is capable of achieving such partitioning using only training and

validation sets.

To provide statistical support for our experimental results, we applied the Friedman and Nemenyi non-parametric statistical tests, as suggested in [48, 49]. The Friedman test ranks the algorithms in terms of their performances, such that, the lower the rank is for an algorithm, the better it is. If the Friedman test detects statistically significant differences in the performances of the algorithms, we apply the Nemenyi post hoc test. In this test, the performances of two classifiers are statistically significantly different only if their average ranks differ by a certain critical difference (CD). The critical difference depends on the number of algorithms, the number of datasets, and the critical value for a significance level provided by a Studentized range statistic. The result from the Nemenyi post hoc test is interpreted with a critical difference diagram. In the diagram, algorithms connected by a line do not present statistically significant differences, since their average rank values differ less than the critical difference.

6. Results

6.1. Regression results

Table 4 shows the results obtained with single decision trees. This table includes the results for global and local approaches, the proposed exhaustive search (ES) when possible, the GA-based search and the Oracle. The best result for each dataset is highlighted in bold (discarding the Oracle that always obtains the highest performance).

Looking at this table, the experiments suggest that in-between local and global partitions are preferred, since both the GA and the exhaustive search were able to find partitions that led the decision tree to obtain better results in the majority of the datasets than the global and local approaches. The results of the Oracle also support that there are indeed partitions different from local and global that could greatly reduce the prediction error. It is noticeable that the GA outperformed the exhaustive search in most datasets. This suggests that the exhaustive search overfits the validation data and does not general-

Table 4: Regression datasets: Test MAE results with Decision Tree as base classifier

Dataset	global	local	ES	GA	*Oracle*
oes10	335.52	335.54	294.20	**293.05**	*218.81*
oes97	507.86	480.96	504.25	**477.02**	*356.12*
rf1	5.04	**0.60**	0.61	**0.60**	*0.56*
rf2	5.04	0.62	**0.61**	**0.61**	*0.57*
scm1d	79.87	68.37	–	**68.19**	–
scm20d	93.22	85.27	84.92	**84.20**	*79.74*
water-quality	0.90	**0.86**	0.87	0.87	*0.83*
osales	2987.28	3270.13	3086.70	**2955.83**	*2363.77*
olfaction	10.44	10.87	–	**10.28**	–
Ranking	2.67	2.17	–	**1.17**	–

ize better than the GA, providing worse results on the test partitions. This highlights the difficulty of determining the best partitioning in practice, as the best output partitioning for training/validation partitions may not always be the most suitable solution in test data.

Table 5: Average number of partitions found by the methods for regression datasets, using single decision trees. The number of partitions of the global approach is always 1, while for local it is $|T|$.

Dataset	$\|T\|$ (local)	ES	GA	Oracle
oes10	16	6.2	8.68	6.2
oes97	16	7.1	8.92	7.3
rf1	8	4.6	4.90	4.7
rf2	8	4.8	4.50	4.6
scm1d	16	–	10.24	–
scm20d	16	7.3	9.12	7.7
water-quality	14	7.8	8.74	7.7
osales	12	4.0	5.20	3.9
olfaction	1029	–	336.00	–

To further illustrate the differences between global, local and alternative partitionings, Table 5 shows the average number of partitions found by the ES, the GA and the Oracle. We also include the original number of target outputs $|T|$, which corresponds to the number of targets used by the local approach. The global approach always uses only one partition. As before, the results confirm

that there are better partitionings than local and global, as witnessed by the Oracle column. We can observe how the number of partitions found by the ES seems to be quite similar to the Oracle. The GA usually provided a slightly higher number of partitions, although still substantially lower than the local method. This may also imply that, apart from the predictive performance gain over the local method, the GA and ES methods end up creating less models, which could help interpret the results more easily.

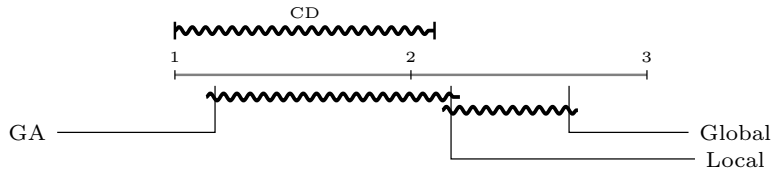
Table 6: Regression datasets: Test MAE results with Random Forest as base classifier

Dataset	global	local	ES	GA	*Oracle*
oes10	240.69	**227.76**	234.89	229.99	*210.04*
oes97	395.10	**379.34**	389.45	385.13	*349.56*
rf1	0.43	**0.41**	0.42	0.42	*0.40*
rf2	0.87	0.72	**0.71**	0.77	*0.69*
scm1d	53.59	**49.56**	–	49.74	–
scm20d	66.07	**64.81**	–	64.86	–
water-quality	0.83	0.83	0.83	**0.82**	*0.80*
osales	2959.88	**2938.13**	2963.47	2945.42	*2684.33*
olfaction	**9.74**	10.05	–	9.83	–
Ranking	2.72	**1.39**	–	1.89	–

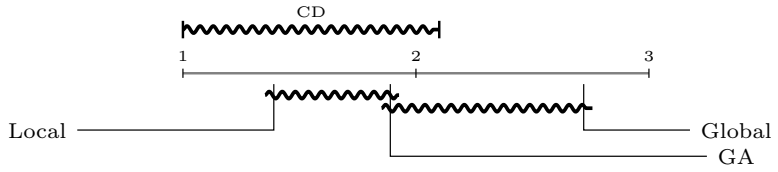
Table 6 summarizes the results in terms of MAE obtained when a Random Forest is considered as base classifier. This table suggests that global methods are rarely preferred, since they obtained better results in only one dataset. Furthermore, the results suggest that, differently from decision trees, random forests generalize better when a model was built for each target separately (local approach). This may be explained by the robustness of the ensemble, which dealt better with many individual single-target problems, not considering target relationships. Conversely with the previous results, both search strategies (GA and ES) seem to overfit when Random Forest is used as base classifier.

Figure 3 presents the critical difference diagrams comparing the results from Local, Global, and GA approaches, after a Nemenyi's statistical test. Approaches connected by a line do not present statistically significant differences, since their average rank values differ less than the critical difference. The results

confirm that using smaller target partitions is preferred over using one singleton global partition. No statistically significant differences were found comparing local and GA approaches. However, using in-between local and global partitions led to statistically superior results in comparison with the global approach when using decision trees.



(a) MAE: decision trees



(b) MAE: random forests

Figure 3: Critical difference diagrams showing average MAE ranks and Nemenyi's critical difference (CD) for the three methods.

6.2. Classification results

Table 7 shows the results obtained with single decision trees. Again, this table includes all the comparison algorithms, while the best result for each dataset is highlighted in bold. From this table we can see that the best results are more spread among local and global partitions. However, it is worth mentioning that the GA or the exhaustive search was able to find some in-between local and global partitions that led to the best results for some datasets.

As before, we also analyse the average number of partitions found by each method. Table 8 presents these results. As for the regression case, the Oracle results confirm that there are better partitions than global and local. For

Table 7: Classification datasets: Test AUROC results with Decision Tree as base classifier

Dataset	global	local	ES	GA	*Oracle*
birds	**0.7601**	0.6586	–	0.6900	–
corel5k	0.5093	**0.5469**	–	0.4973	–
emotions	**0.7720**	0.7173	0.7469	0.7469	*0.8035*
yeast	0.6035	0.5754	**0.6066**	0.5954	*0.6807*
flags	0.6703	0.6589	**0.6843**	0.6701	*0.7658*
genbase	0.8470	**0.8482**	–	0.8476	–
mediamill	**0.7502**	0.6185	–	**0.7502**	–
Ranking	**1.57**	2.50	–	1.93	–

the three datasets where the Oracle provides results, the number of partitions returned by ES and GA is very close to that of the Oracle.

Table 8: Average number of partitions found by the methods for classification datasets, using single decision trees. The number of partitions of the global approach is always 1, while for local it is $|T|$.

Dataset	$\|T\|$ (local)	ES	GA	Oracle
birds	19	–	6.80	–
corel5k	374	–	187.40	–
emotions	6	2.5	2.50	2.1
yeast	14	4.2	5.44	3.5
flags	7	3.5	3.56	3.7
genbase	27	–	10.50	–
mediamill	101	–	1.00	–

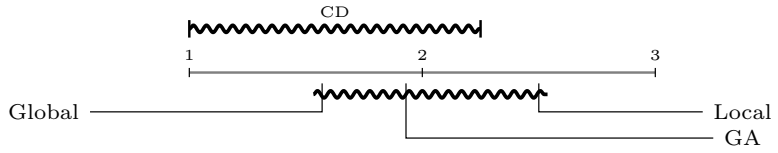
Table 9 presents the results when Random Forest is used as base classifier. The results suggest that, given the existence of relationships among labels, global-based models seem to be more suitable to take these relationships into consideration. Thus, using a singleton partition to induce models seems to be preferred. However, it is worth mentioning that the GA or exhaustive approach could also find some in-between partitions where the results obtained were better than or very similar to those obtained by the global approach. In particular, in 5 of the 7 datasets, the difference between the GA and the best method's AUROC is less than 1%. For the other datasets, it is less than or equal to 2%.

With both base classifiers, on those datasets in which we were able to run the

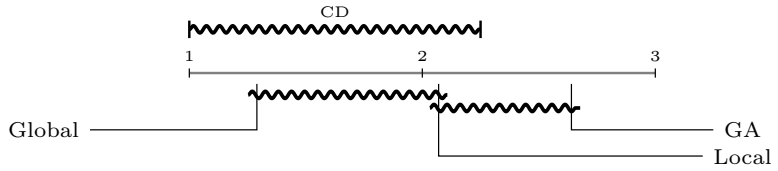
Table 9: Classification datasets: Test AUROC results with Random Forest as base classifier

Dataset	global	local	ES	GA	*Oracle*
birds	0.8830	**0.8853**	–	0.8652	–
corel5k	**0.6766**	0.6707	–	0.6711	–
emotions	**0.8562**	0.8547	0.8524	0.8514	0.8755
yeast	0.6959	**0.7013**	0.6955	0.6871	0.7624
flags	0.7783	0.7736	0.7742	**0.7838**	0.8319
genbase	**0.8421**	0.8380	–	0.8326	–
mediamill	**0.8289**	0.8191	–	0.8257	–
Ranking	**1.29**	2.07	–	2.64	–

Oracle, we can see that none of the approaches was able to provide a solution with AUROC close to that optimal partitioning, suggesting that overfitting may be occurring in these cases.



(a) AUROC: decision trees



(b) AUROC: random forests

Figure 4: Critical difference diagrams showing average AUROC ranks and Nemenyi's critical difference (CD) for the three methods.

Although using a singleton partition led to better results, Figure 4 shows that no statistically significant differences were found among the results of the local, global and GA approaches when using single decision trees as base classifiers. Thus, if the objective is to obtain a more interpretable model, the use of the global approach can be recommended. This is reasonable given that the use of

a singleton partition results in the induction of only one decision tree, which results in less rules than a set of decision trees. Considering random forests, the use of a singleton global partition led to statistically better results in comparison with the use of the partitions found by the GA.

7. Discussion

In both classification and regression problems, we have observed that the Oracle was consistently able to find partitions different from the local and global that improved the performance. To further illustrate the existence of partitions beyond global and local, we have applied the Oracle to rank all potential partitions in terms of MAE and AUROC, respectively. Figure 5 plots the position (as a percentage) of global and local approaches against the potential partitions in 2 classification and 2 regression problems, using Decision Tree as base classifier. Note that the higher the percentage the worse is the solution found by global or local. Thus, this figure allows us to see how many solutions are between the global or local and the best result found by the Oracle, giving a rough idea of how difficult it is to find a partitioning better than global or local.

In these datasets, we can see how the rankings of global and local vary considerably depending on the the dataset. On rf1, the local is a top solution very close to the one found by the Oracle. However, in datasets such as flags, more than 40% of the partitions outperform both global and local.

This analysis shows that there is a good number of partitions that perform better than the global and local models. However, the search algorithms we used (both ES and GA) were not always capable of providing a solution better than the global or the local (especially in classification problems). This suggests that searching for the best partitioning using training and validation sets does not generalize sufficiently well in all cases, and as mentioned before they suffer from overfitting. This is illustrated in Table 10 for the classification results. The table reports the difference in AUROC between validation and test set results, for the global approach and the GA approach. This difference is statistically

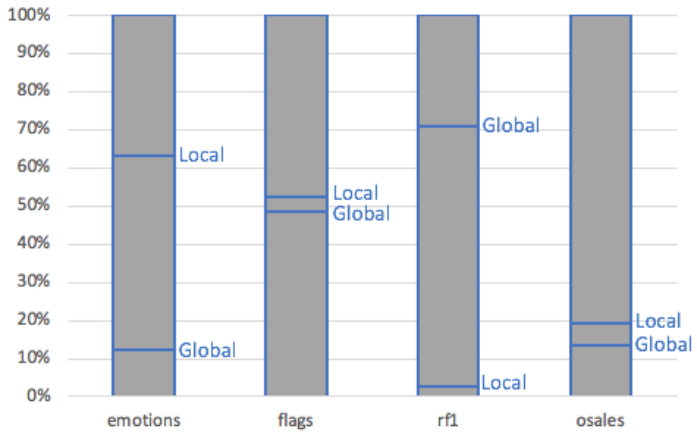


Figure 5: Position (as a percentage) of the Global and Local solutions in 2 classification and 2 regression datasets (Decision Tree as base classifier). Note that higher percentages indicate worse solutions.

significantly larger for the GA than for the global approach (p=0.036 according to a Wilcoxon signed rank test). For the global approach, all differences between validation and test set AUROC are less than 0.2%, sometimes even slightly negative. For the GA, apart from the mediamill dataset, all differences are positive, and can be as high as 10% for the birds dataset. To further illustrate the overfitting phenomenon, we have taken datasets flags and emotions. For these datasets, the GA (using Decision Trees) runs in each fold evolve to the same optimal partition as the exhaustive search for the corresponding fold. Figure 6 plots the position (normalized as a percentage) of these 10 optimal partitions in the corresponding test fold. Ideally, the partition optimized on the validation set should appear as a top ranked partition in the test set, but as displayed in the figure, this is not the case. We see that for the dataset flags, in one fold the partition found by the exhaustive search, or equivalently, by the GA, is also the top partition in the corresponding test set. However, the median position is at 34% and for one fold the partition is only at the 80% position. Similar observations hold for emotion, with a median position at 29%.

Table 10: Classification datasets: difference between validation and test AUROC for the global model and for the GA (decision trees).

Dataset	global			GA		
	validation	test	difference	validation	test	difference
birds	0.7621	0.7601	0.0020	0.7870	0.6900	0.0990
corel5k	0.5084	0.5093	-0.0009	0.5077	0.4973	0.0104
emotions	0.7713	0.7720	-0.0007	0.8044	0.7469	0.0575
flags	0.6718	0.6703	0.0015	0.7524	0.6701	0.0823
genbase	0.8467	0.8470	-0.0003	0.8523	0.8476	0.0047
mediamill	0.7468	0.7502	-0.0034	0.7468	0.7502	-0.0034
yeast	0.6043	0.6035	0.0008	0.6532	0.5954	0.0578
Average diff.			-0.00014			0.04404

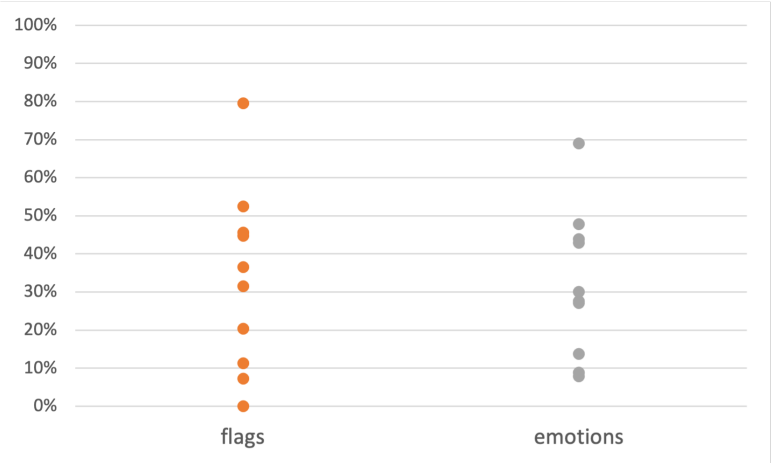


Figure 6: Position (as a percentage) for each fold of the exhaustive search or GA solutions (optimized on validation set) in their corresponding test set (Decision Tree as base classifier). Note that higher percentages indicate worse solutions.

8. Conclusions

In this study we have investigated the task of finding a partition in the target space of multi-target classification and regression problems, which could lead to better results than the traditional global an local approaches, when each subset is treated as a separate prediction problem.

We designed a genetic algorithm to search for the best partition in the tar-

get space, and used Predictive Clustering Trees and random forests as base classifiers to evaluate our partitions.

Given our experiments, complemented with an Oracle method, we can conclude that it is possible to obtain superior performances with partitions in-between global and local. For some datasets, there is even a high percentage of partitions that outperform both global and local approaches. However, using a hold-out validation set to select the best partition - even with an exhaustive search - seems to lead to overfitting. The best approach to find such partition is still an open question at this point.

We believe that this study can encourage future research to propose multi-target models in-between the traditional global and local approaches, which have received the most attention in the literature. As future work, we are considering using various strategies to reduce the overfitting problem. In particular, we will be exploring the use of a windowing approach, such as [50], which has proven to reduce overfitting when search algorithms are used in machine learning .

A limitation of our study, but that can be certainly overcome in future research, is that we did not explore many possibilities on how to use target correlations in order to build the in-between local and global partitions. We could investigate strategies to use similarity measures such as Jaccard index or label conditional probabilities in order to find groups of correlated targets.

We also plan to include a detailed analysis concerning the disturbances/uncertainties of data that intrinsically have noise. This could be performed by using a multi-objective GA considering information about the noise as an additional objective to improve the quality of the partitions.

Acknowledgements

The authors would like to thank CNPq, CAPES and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant 2016/02870-0, for funding this research. They are also grateful for the support from KU Leuven through its Latin America Fund and from the Flemish Government through its AI Research

Program.

References

[1] Džeroski, S., Demšar, D., Grbović, J.: Predicting chemical parameters of river water quality from bioindicator data. Applied Intelligence **13**(1) (Jul 2000) 7–17

[2] Keller, A., Gerkin, R.C., Guan, Y., Dhurandhar, A., Turu, G., Szalai, B., Mainland, J.D., Ihara, Y., Yu, C.W., Wolfinger, R., Vens, C., Schietgat, L., De Grave, K., Norel, R., Consortium, D.O.P., Stolovitzky, G., Cecchi, G.A., Vosshall, L.B., Meyer, P.: Predicting human olfactory perception from chemical features of odor molecules. Science **355**(6327) (2017) 820–826

[3] Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy and Buildings **49** (2012) 560 – 567

[4] Kocev, D., Vens, C., Struyf, J., Deroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition **46**(3) (2013) 817–833

[5] Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I.: Multi-target Regression via Random Linear Target Combinations. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part III. Springer Berlin Heidelberg, Berlin, Heidelberg (2014) 225–240

[6] Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., Džeroski, S.: Predicting gene function using hierarchical multi-label decision tree ensembles. BMC Bioinformatics **11**(2) (2010)

[7] Cerri, R., Barros, R.C., P. L. F. de Carvalho, A.C., Jin, Y.: Reduction strategies for hierarchical multi-label classification in protein function prediction. BMC Bioinformatics **17**(1) (Sep 2016) 373

[8] Wan, S., Mak, M.W., Kung, S.Y.: mgoasvm: Multi-label protein subcellular localization based on gene ontology and support vector machines. BMC Bioinformatics **13**(1) (Nov 2012) 290

[9] Li, X., Ouyang, J., Zhou, X.: Labelset topic model for multi-label document classification. Journal of Intelligent Information Systems **46**(1) (February 2016) 83–97

[10] Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Hadley, S.F., Hadley, A., Betts, M., Fern, X.Z., Irvine, J., Neal, L., Thomas, A., Fodor, G., Tsoumakas, G., Ng, H.W., Nguyen, T.N.T., Huttunen, H., Ruusuvuori, P., Manninen, T., Diment, A., Virtanen, T., Marzat, J., Defretin, J., Callender, D., Hurlburt, C., Larrey, K., Milakov, M.: The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In: 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). (2013) 1–8

[11] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73**(2) (2008) 185–214

[12] Hasan, M.A.M., Ahmad, S., Molla, M.K.I.: Protein subcellular localization prediction using multiple kernel learning based support vector machine. Molecular BioSystems **13** (2017) 785–795

[13] Zeng, Z., Liang, N., Yang, X., Hoi, S.: Multi-target deep neural networks: Theoretical analysis and implementation. Neurocomputing **273** (2018) 634 – 642

[14] Madjarov, G., Kocev, D., Gjorgjevikj, D., Deroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition **45**(9) (2012) 3084–3104

[15] Joly, A., Geurts, P., Wehenkel, L.: Random forests with random projections of the output space for high dimensional multi-label classification. In Calders, T., Esposito, F., Hüllermeier, E., Meo, R., eds.: Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2014. Lecture Notes in Computer Science. Volume 8724., Berlin, Heidelberg, Springer (2014)

[16] Breskvar, M., Kocev, D., Deroski, S.: Multi-label classification using random label subset selections. In: Discovery Science: 20th International Conference, DS 2017. (09 2017) 108–115

[17] Breskvar, M., Kocev, D., Džeroski, S.: Ensembles for multi-target regression with random output selections. Machine Learning **107**(11) (Nov 2018) 1673–1709

[18] Blockeel, H., Raedt, L.D., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the Fifteenth International Conference on Machine Learning. ICML '98, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1998) 55–63

[19] Kramer, O. In: Genetic Algorithms. Springer International Publishing, Cham (2017) 11–19

[20] Tsoumakas, G., Katakis, I., Vlahavas, I. In: Mining Multi-label Data. Springer US, Boston, MA (2010) 667–685

[21] Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics **22**(7) (2006) 830–836

[22] Masera, L., Blanzieri, E.: Awx: An integrated approach to hierarchical-multilabel classification. In Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G., eds.: Proceedings of ECML PKDD 2018: Machine Learning and Knowledge Discovery in Databases, Cham, Springer International Publishing (2019) 322–336

[23] Madjarov, G., Gjorgjevikj, D., Dimitrovski, I., Džeroski, S.: The use of data-derived label hierarchies in multi-label classification. Journal of Intelligent Information Systems **47**(1) (Aug 2016) 57–90

[24] Papagiannopoulou, C., Tsoumakas, G., Tsamardinos, I.: Discovering and exploiting deterministic label relationships in multi-label learning. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '15, New York, NY, USA, ACM (2015) 915–924

[25] de Abreu, I.B.M., Mantovani, R.G., Cerri, R.: Incorporating instance correlations in multi-label classification via label-space. In: 2017 International Joint Conference on Neural Networks (IJCNN). (2017) 581–588

[26] Prati, R.C., de França, F.O.: Extending features for multilabel classification with swarm biclustering. In: 2013 IEEE Congress on Evolutionary Computation. (2013) 2964–2971

[27] Cherman, E.A., Metz, J., Monard, M.C.: Incorporating label dependency into the binary relevance framework for multi-label classification. Expert Systems with Applications **39**(2) (February 2012) 1647–1655

[28] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II. ECML PKDD '09, Berlin, Heidelberg, Springer-Verlag (2009) 254–269

[29] Dembczynski, K., Cheng, W., Húllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In Fúrnkranz, J., Joachims, T., eds.: Proceedings of the 27th International Conference on Machine Learning, Omnipress (2010) 279–286

[30] Huang, S.J., Zhou, Z.H.: Multi-label learning by exploiting label correlations locally. In: AAAI Conference on Artificial Intelligence. (2012) 949–955

[31] Yu, Y., Pedrycz, W., Miao, D.: Multi-label classification by exploiting label correlations. Expert Systems wit Applications **41**(6) (2014) 2989–3004

[32] Spolaôr, N., Monard, M.C., Tsoumakas, G., Lee, H.D.: A systematic review of multi-label feature selection and a new method based on label construction. Neurocomputing **180**(C) (2016) 3–15

[33] Huang, S.J., Zhou, Z.H.: Multi-label learning by exploiting label correlations locally. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. AAAI'12, AAAI Press (2012) 949–955

[34] Szymaski, P., Kajdanowicz, T., Kersting, K.: How is a data-driven approach better than random choice in label space division for multi-label classification? Entropy **18** (06 2016)

[35] Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering **23** (07 2011) 1079–1089

[36] Xioufis, E.S., Groves, W., Tsoumakas, G., Vlahavas, I.P.: Multi-label classification methods for multi-target regression. CoRR **abs/1211.6581** (2012)

[37] Godbole, S., Sarawagi, S. In: Discriminative Methods for Multi-labeled Classification. Springer Berlin Heidelberg, Berlin, Heidelberg (2004) 22–30

[38] Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: treating targets as inputs. Machine Learning (2016) 1–44

[39] Piccart, B., Struyf, J., Blockeel, H. In: Empirical Asymmetric Selective Transfer in Multi-objective Decision Trees. Springer Berlin Heidelberg, Berlin, Heidelberg (2008) 64–75

[40] Jacob, L., Vert, J.P., Bach, F.R.: Clustered multi-task learning: A convex formulation. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., eds.:

Advances in Neural Information Processing Systems 21. Curran Associates, Inc. (2009) 745–752

[41] Melki, G., Cano, A., Kecman, V., Ventura, S.: Multi-target support vector regression via correlation regressor chains. Information Sciences **415** (2017) 53 – 69

[42] Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7) (2007) 2038 – 2048

[43] Orlov, M.: Efficient generation of set partitions. Technical report, Department of Software Engineering, Shamoon College of Engineering, Israel. (2002)

[44] Haight, F.: Handbook of the Poisson distribution. Publications in operations research. Wiley (1967)

[45] Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. In: Simulated annealing: Theory and applications. Springer (1987) 7–15

[46] Sechidis, K., Tsoumakas, G., Vlahavas, I.: On the stratification of multi-label data. In: Machine Learning and Knowledge Discovery in Databases, Berlin, Heidelberg, Springer Berlin Heidelberg (2011) 145–158

[47] Rivolli, A.: utiml: Utilities for Multi-Label Learning. (2016) R package version 0.1.0.

[48] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (December 2006) 1–30

[49] Garcia, S., Herrera, F.: An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. Journal of Machine Learning Research **9** (2009) 2677–2694

[50] Bacardit, J., Goldberg, D.E., Butz, M.V., Llorà, X., Garrell, J.M.: Speeding-up pittsburgh learning classifier systems: Modeling time and ac-

curacy. In: Parallel Problem Solving from Nature - PPSN VIII. Volume
890 3242 of LNCS. (2004) 1021–1031