MOCA-I: Discovering Rules and Guiding Decision Maker in the Context of Partial Classification in Large and Imbalanced Datasets

Julie Jacques, Julien Taillard, David Delerue, Laetitia Jourdan, Clarisse Dhaenens

MOCA-I: Discovering Rules and Guiding Decision Maker in the Context of Partial Classification in Large and Imbalanced Datasets

Julie Jacques[1,2,3], Julien Taillard[1], David Delerue[1], Laetitia Jourdan[2,3], and Clarisse Dhaenens[2,3]

[1] Société ALICANTE, 50 Rue Philippe de Girard, 59113 Seclin, France
[2] INRIA Lille Nord Europe, 40 Av. Halley, 59650 Villeneuve d'Ascq, France
[3] LIFL, Université Lille 1, Bât. M3, 59655 Villeneuve d'Ascq cedex, France

Abstract. This paper focuses on the modeling and the implementation as a multi-objective optimization problem of a Pittsburgh classification rule mining algorithm adapted to large and imbalanced datasets, as encountered in hospital data. We associate to this algorithm an original post-processing method based on ROC curve to help the decision maker to choose the most interesting rules. After an introduction to problems brought by hospital data such as class imbalance, volumetry or inconsistency, we present MOCA-I - a Pittsburgh modelization adapted to this kind of problems. We propose its implementation as a dominance-based local search in opposition to existing multi-objective approaches based on genetic algorithms. Then we introduce the post-processing method to sort and filter the obtained classifiers. Our approach is compared to state-of-the-art classification rule mining algorithms, giving as good or better results, using less parameters. Then it is compared to C4.5 and C4.5-CS on hospital data with a larger set of attributes, giving the best results.

1 Introduction

Data mining on real datasets can lead to handling imbalanced data. It occurs when many attributes are available for each observation, but only a few are actually entered. This is especially the case with medical data: ICD-10 [4] – a medical coding system – allows encoding up to 14,199 diseases and symptoms. However in hospital data, for each patient, only a very small subset of these codes will be used: up to 100 symptoms and diseases. This implies that most frequent symptoms, like *high blood pressure*, are found on at best 10% of the patients. For less common diseases, like *transient ischemic stroke* it can be lower to less than 0.5% of patients. This can also happen with market-basket data: many different items are available in the store but only a few are actually bought by a single customer. Additionally, more and more information is available and collected

[4] International classification of diseases; `http://www.who.int/classifications/icd/en/`

nowadays: algorithms must be able to deal with larger datasets. According to Fernández *et al.* dealing with large datasets is still a challenge that needs to be addressed [1]. This work is a part of the OPCYCLIN project – an industrial project involving Alicante company, hospitals and academics as partners – that aims at providing a tool to optimize screening of patients for clinical trials.

Different tasks are available in datamining, this paper focuses on the *classification* task, useful to predict or explain a given *class* (e.g.: *cardiovascular risk*) on unseen observations. Classification will use known data, composed of a set of N known observations $i_1, i_2, ..., i_N$ to build a model. Each observation can be described by M attributes $a_1, a_2, ..., a_M$ and a class c. Therefore each observation i is associated with a set of values $v_{i1}, v_{i2}, ..., v_{iM}$ where $v_{ij} \in V_j\{val_1, val_2, \dots val_p\}$; V_j being the set of possible values for attribute a_j. In the same manner, each observation i is associated to a class value $c_i \in C$, C being the set of all possible values for the class. A classification algorithm will be able to generate a model that describes how to determine c_v on an unseen observation v, using its values $v_{v1}, v_{v2}, ..., v_{vM}$. This paper focuses on models able to give a good interpretability: they allow medical experts to give a feed back about them. Decision trees and classification rules give easy-to-interpret models, by generating trees or rules — like "$a_j = val_j$ and $a_g = val_g \Rightarrow class$", where $val_j \in V_j$, $val_g \in V_g$ and $class \in C$; using combinations of attributes $a_1, a_2, ..., a_M$ and one of their possible values $val_1, val_2, ..., val_M$ to lead to the decision.
Decision trees – like C4.5 [2] or CART (classification and regression trees) – are popular and efficient solutions for knowledge extraction. However the tree representation is composed of conjunctions, not allowing expressing classes explained by different contexts (e.g.: presence of *overweight* or *high blood pressure* implies an increased *cardiovascular risk*, having both increases the risk more). Separate and conquer strategy, frequently implemented in tree algorithms often contribute to miss rules issued from different contexts: each sub-tree is constructed using a sub-part of data (observations not corresponding to the top of the tree are removed from learning). To avoid this problem we will focus on classification rule mining approaches.
The majority of state-of-the-art classification algorithms will have trouble to deal with imbalanced data because they use *Accuracy* to build their predictive model [1]. *Accuracy* focuses on counting good classifications obtained by a given algorithm: *true positives* and *true negatives*. However, when predicting a class available on only 1% of the observations, an algorithm can get a very good classification *Accuracy* – 99% – while predicting each observation as negative (99% of observations) and missing each positive observation. Some resampling methods exist to pre-process the data and convert it into balanced data, an overview can be found in [3]. Jo and Japkowicz showed that combining data resampling and an algorithm able to deal with class imbalance is more effective than using resampling alone [4]. Moreover, in addition to class imbalance and a huge amount of data, hospital data is subject to uncertainty. When data is missing on one patient, two cases can happen: the patient does not have the disease or the patient

has the disease but was not diagnosed yet (or the diagnosis was not entered in the system). This is difficult to predict the consequences of resampling on such data.

The remaining of this paper is organised as follows: Section 2 will introduce some common rule interestingness measures, and will show how the classification rule mining problem can be seen as a multi-objective problem. Section 3 will propose the modeling as a multi-objective local search optimization problem. Then, the *Dominance-based local search* algorithm will be presented, as well as the associated implementation details such as neighborhood. This section will conclude by the description of an original post-processing method to select rules based on ROC curve. In section 4, we will assess the performance of our approach. At first we will compare our results to those gathered by Fernandez *et al.* with 22 state-of-the-art classifiers in the context of imbalanced data [1], showing our approach can be applied on more general datasets. Secondly, we will compare our approach to C4.5 – a state-of-the-art decision tree algorithm – and C4.5-CS – an adaptation of the C4.5 algorithm to imbalanced data – on real hospital data. Finally, section 5 gives conclusions and perspectives for future works.

2 A Multi-objective Model to Discover Partial Classification Rules in Imbalanced Data

This section will present some rule interestingness measures and their meaning. Then it presents the 3 objectives that will be used to find rules.

2.1 Rule Interestingness Measures

When mining rules, an important question will raise: how can we assess that a rule is better than another? Over 38 common rule interestingness measures are referenced by Geng and Hamilton in their review [5], while Ohsaki *et al.* studied measures used in medical domain [6] and Greco *et al.* studied Bayesian confirmation measures [7].

Table 1. Confusion matrix

	P	$\overline{P}$	
C	TP	FP	
$\overline{C}$	FN	TN	
			N

The majority of rule interestingness measures are based on a confusion matrix, like the one provided in Table 1. For a given rule $C \rightarrow P$, TP (*true positives*) will represent count of observations having both C and P; TN (*true negatives*) count of observations not having C and not having P. FN (*false negatives*) and FP (*false positives*) count observations on which C and P do not

match. When dealing with imbalanced data

$$\overline{P} = FP + TN >> P = TP + FN, \tag{1}$$

therefore problems may rise with some measures like previously seen with the *Accuracy*.
To ease the conception of a rule mining algorithm we must focus on a subset of these measures. Indeed, handling too many measures will add complexity and will increase computational time. An analysis of these measures showed that *Confidence* and *Sensitivity*

$$Confidence = \frac{TP}{TP + FP},\ Sensitivity = \frac{TP}{TP + FN}, \tag{2}$$

are two interesting complementary measures. Increasing *Confidence* decreases the number of *false positives* while increasing *Sensitivity* decreases the number of *false negatives*. However, increasing *Confidence* often decreases *Sensitivity* while increasing *Sensitivity* decreases *Confidence*. To the medical domain point of view, only rules having both good *Confidence* and *Sensitivity* are interesting. Moreover, Bayardo and Agrawal showed that mining rules optimizing both *Confidence* and *Support* leads to obtain rules optimizing several other measures including *Gain*, *Chi-squared value*, *Gini*, *Entropy gain*, *Laplace*, *Lift*, and *Conviction* [8]. Since in classification, *Sensitivity* and *Support* measures are proportional, optimizing *Confidence* and *Support* will bring the same rules than optimizing *Confidence* and *Sensitivity*.
When mining variable-length rules, *bloat* can happen: rules endlessly grow with no predictive enhancement. Because of *bloat*, a rule R_1: $C \Rightarrow P$ can turn into R_2: C OR C $\Rightarrow P$, then R_3: C OR C OR C $\Rightarrow P$, increasing computational time and preventing the algorithm to stop. Most of all, R_3 is needlessly complex and harder to interpret than R_1. Rissanen introduced the *Minimum Description Length (MDL) principle* that can be used to overcome this problem [9]. Given two equivalent rules, the simplest must be preferred. The addition of one objective promoting simpler rules is a common solution, successfully applied in Reynolds and Iglesia work [10]. In addition to this, Barcadit *et al.* used rule deletion operators [11]. In application of this principle, we introduce a third objective to promote simpler rulesets: minimizing the count of terms of each solution. Finally, we choose to find rules optimizing the 3 following objectives:

- maximize *Confidence*
- maximize *Sensitivity*
- minimize number of terms

3 A Multi-objective Model to Discover Partial Classification Rules in Imbalanced Data

In addition to class imbalance, our hospital data raises another problem: more than 10,000 attributes are available for each patient. This implies a huge number

of possible rules to explore. As a rule may be seen as a combination of pairs <attribute, value>, the rule mining problem is a combinatorial one. Moreover, regarding the large number of attributes, it requires methods dedicated to deal with very large search spaces such as combinatorial optimization methods and metaheuristics. Moreover, some datamining tasks contain NP-hard problems; in their review, Corne *et al.* explain how operations research and metaheuristics can help solving these problems that may be seen as combinatorial optimization problems [12].
The three objectives identified in the previous section highlight the need of methods able to deal with several objectives. Multi-objective optimization can handle such problems; Srinivasan and Ramkrishnan made a review of rule mining approaches using multi-objective optimization [13].

As explained later, in our work we will adopt a Dominance-based approach: each objective will be treated separately. Metaheuristics working on a population of solutions are particularly well suited for this type of problems [14] and we will adopt one of them to our classification rules problem. In the following, the solution modeling and the algorithm proposed are detailed.

3.1 Solution Modeling

Solution Representation Two main solution representations exist for rule mining in metaheuristics: *Michigan* and *Pittsburgh*. Michigan is the widely used one, where each solution represents a single rule. However algorithms using this representation can miss some rules: an interesting rule will be removed if a slightly better rule is found, even if that rule targets different observations from the dataset.

This problem does not appear in Pittsburgh representation where each solution is a set of rules. This more complex representation will impact the size of the search space – now larger than the one of Michigan – and introduce new problems such as rule redundancy or conflicts between rules: which prediction must be chosen when different rules in the same ruleset have conflicting predictions? This is only an overview of problems that may happen. Casillas *et al.* identified more possible inconsistencies risen by Pittsburgh modeling [15].
We propose to use a Pittsburgh representation where each solution is a ruleset. Each ruleset is composed only of rules predicting the positive class, called partial rules. A rule is a conjunction of terms; a term is the expression of a test made on an attribute, for a given observation. Attributes can be binary (e.g: *hasHighBloodPressure?*), or associated to an operator (=,<,>) and a value, where the value is taken from a list of values, ordered or not. Any observation that triggers at least one rule from the ruleset will be labeled as positive class. Our representation is designed to handle binary classes, thus observations not triggering any rules are labeled as negative class. Since a ruleset groups together only partial classification rules predicting the positive class, there is no need to store the right part of each rule. Moreover, there are no inconsistencies since rules in a same ruleset cannot predict different classes. Rule redundancy is avoided by

minimizing the size of each ruleset. Thus, a rule will be added only if it improves *Sensitivity* (*true positives rate*) or *Confidence* of the ruleset.

Evaluation Function Previously we saw three objectives can be used to find rulesets: maximizing *Confidence*, maximizing *Sensitivity* and minimizing the number of terms. The third objective corrects one drawback of Pittsburgh representation that brings *bloat*. We use a dominance-based approach to handle the different objectives, in opposition to some classification rule mining algorithms like GAssist using scalar approaches [11]. Dominance-based approaches use a dominance relation to compare solutions over several objectives, avoiding searching the good adjustment of weights to combine the different objectives, needed in scalar approaches. Moreover, with a weighted fitness function two solutions having different objective values can have the same fitness score. Our method is based on Pareto Dominance. This will generate a population of rulesets, in our case rulesets with very high *Confidence* but relatively low *Sensitivity*, rulesets with middle *Confidence* and *Sensitivity*, rulesets with high *Sensitivity* and relatively low *Confidence*, etc. These rule sets are stored in an *archive*, thus we need an algorithm able to handle populations.

3.2 DMLS Algorithm

Dominance-based multi-objective local search (DMLS) is a local search algorithm, based on a dominance relation [16]. It needs the definition of a neighborhood function that associates to each solution a set of solutions – called neighbors – by applying a small modification on it. A neighborhood of a rule can be, for example, all rules having one more or one less term.
Most of multi-objective rule mining contributions presented in the review of Srinivasan and Ramkrishnan are based on the metaheuristic NSGA-II (genetic algorithm dedicated to multi-objective problems) [13]. DMLS has previously proven to give at least as good results as NSGA-II on several problems [16]. Moreover, DMLS is easier to parameter than a genetic algorithm as we only have to define a neighborhood operator. Therefore we used DMLS implemented by Liefooghe *et al.* [16] in ParadisEO framework [17], with an unbounded archive, using the natural stopping criterion. DMLS algorithm is detailed in Algorithm 1. It evolves a population of non dominated rulesets. At first, all rulesets are marked as unvisited. While unvisited rulesets exist, DMLS will chose randomly one of them from the archive, visit its neighborhood and add all the non-dominated neighbors to the archive for future visits.

Initialization DMLS is initialized with a population of 100 rulesets. Each is made of two rules, whose attributes are randomly picked in a same observation. This ensures the chosen attributes appears together at least on one observation. Then, one or two random attributes are replaced to add some diversity.

Algorithm 1 Dominance-based multi-objective local search

```
generates 100 rulesets RS_a, composed of 2 initial rules
RS_a.setVisited(false)
archive.add(RS_a)
while RS_current ← archive.selectRandomUnvisitedSol() do
  RS_n ← generateNeighbors(RS_current)
  for RS_neighbor ∈ RS_n do
    /** add non dominated neighbor to archive, for future visits **/
    if !RS_current ≻ RS_neighbor then
      RS_neighbor.setVisited(false)
      archive.add(RS_neighbor)
    end if
    /** stops when a dominating neighbor is found **/
    if RS_neighbor ≻ RS_current then
      break
    end if
  end for
  if all neighbors in RS_n were visited then
    RS_current.setVisited(true)
  end if
end while
```

Neighborhood The neighborhood function is defined as a generator of all rulesets having a one-term difference: one term removed, one term added or one term modified. They are randomly visited. The neighborhood size is important, since a term addition can happen on each rule of the ruleset, for each available attribute. To reduce the neighborhood size on attributes taking values in ordered lists, we designed a simplified term neighborhood where only boundaries (adjacent values) are visited. Table 2 indicates for each operator (=,<,>) the list of possible neighbors, assuming values are ordered ($v_{i-1} < v_i < v_{i+1}$). Ø means remove the term.

Table 2. Neighborhood of list-valued terms

$a = v_i$	$a < v_i$	$a > v_i$
Ø	Ø	Ø
$a > v_{i-1}$	$a = v_{i-1}$	$a = v_{i+1}$
$a < v_{i+1}$	$a < v_{i-1}$	$a > v_{i+1}$
$a = v_{i-1}$	$a < v_{i+1}$	$a < v_{i+2}$
$a = v_{i+1}$	$a > v_{i-2}$	$a > v_{i-1}$

On a dataset with mixed attributes (ordered and not) (heart dataset, introduced in results section), this simplified neighborhood decreases computational time (in average by 14%), while not degrading too much the classification performance (less than 1%). Another optimization on the neighborhood exploration is

done on rules with *Confidence* = 1: adding one term can only result in decreasing *Sensitivity* because the obtained rule will be more specific and will concern less observations. In this case, we restrict neighbors to modification or removing of one random term.

3.3 Post-processing Using ROC Curve

Multi-objective optimization finds a population of rulesets, corresponding to compromise solutions between the different objectives. Some datasets may obtain up to 400 rulesets after one single run. As it is hard to choose one ruleset, we propose a post-processing method based on ROC curve which combines all obtained rules to take advantage of the diversity issued from the multi-objective algorithm, and a tool to help choosing rules. Then we present how the result can be used by the decision maker to choose the final ruleset. In addition, we propose a method to determine automatically the final ruleset.

ROC Curve is often used in data mining to assess the performance of classification algorithms, especially ranking algorithms. It is plotted using *true positive rate (TPR)* (known as *Sensitivity*) and *false positives rate (FPR)* (also called *1 - Specificity*) as axes and allows comparing algorithms. Fawcett presented different ROC curve usages [18]. In our case, we use ROC curve to select which rules to keep. Since the objective is to use the developed method in a medical context, it can also be used to help our medical users to calibrate classifier or choosing rules using a tool they are familiar with. Algorithm 2 describes how ROC curve can be generated for a given ruleset. Rules are first ordered from the highest *Confidence* score to the lower; rules having the same *Confidence* are ordered by descending order according to *Sensitivity*. Then, *TPR* and *FPR* are computed and drawn for each subruleset $\{R_1\}$,$\{R_1, R_2\}$,...,$\{R_1, R_2 \dots R_i\}$.

Algorithm 2 Draw ROC curve of a ruleset RS

```
order rules of RS by Confidence DESC, Sensitivity DESC
create an empty ruleset RS_roc
for rule R_i ∈ RS {R_1, R_2, ..., R_n} do
   /* get TPR and FPR for sub-rules-set R_1, R_2, ..., R_i */
   RS_roc.add(R_i)
   tpr ← RS_roc.computeTruePositiveRate()
   fpr ← RS_roc.computeFalsePositiveRate()
   plot(fpr,tpr)
end for
```

Rule Selection Using ROC Curve One drawback of dominance-based methods lies in obtaining a set of compromise solutions, that are difficult to handle by the decision maker. The following post-processing method is proposed to cope with this problem. Figure 1 shows on the right, one sample ruleset containing rules $R_1 \dots R_{10}$, ordered from the highest *Confidence* to the lower and in descending order of *Sensitivity* for rules having the same *Confidence* score. On the left the matching ROC curve is drawn. Each point on this curve depicts the performance of a subruleset, e.g.: R_1, R_2, R_3. The higher is the point, the more observations of positive class it detects. Additionally, the more a point is on the right, the more false positive it brings. Consequently, point $(0, 1)$ is the ideal point where all positive observations are found, without bringing any false positive. This figure shows the performance of the ruleset when cut at different places, allowing to choose the subruleset giving the most interesting performance according to decision maker's needs. On this curve we can see that between point a and point b, and after point c there is only a small improvement of *True positives rate*, but it brings much more *false positives*. Performance is more interesting before point a, matching ruleset R_1, R_2, R_3. Subruleset $R_1 \dots R_4$ (cut b) does not seem to be a good choice because it brings much more false positives than positives cases. Point c brings more true positives cases, giving ruleset $R_1, R_2, ..., R_7$. Cutting at point d finds a ruleset able to detect all positive observations: $R_1, R_2, ..., R_9$, keeping rule R_{10} is useless and will only increase false positives. Depending on how many false positive are tolerable, the cut point can be changed. In medical context, cut points bringing less false positives will be preferred (like cut a). To the contrary, an advertising campaign will accept more false positives, to deal with a larger audience. In fraud detection, the cut point can be moved until a given number of positive observations are found.

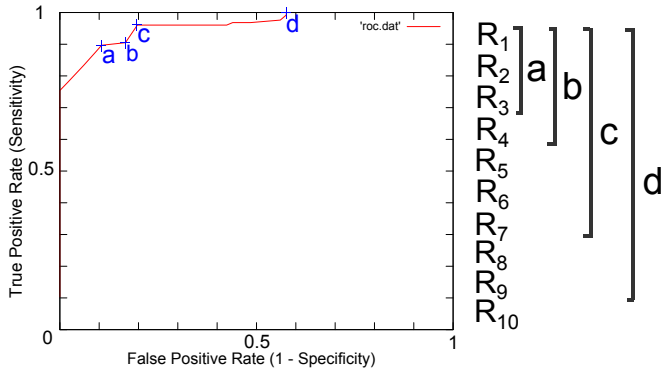

Fig. 1. Example of ROC curve obtained from one ruleset
$R_1 \dots R_{10}$ is one ruleset obtained after the post-processing. a, b, c and d represent different cuts and their associated position on the roc curve.

Ruleset Post-processing Regarding the OPCYCLIN project, the decision maker will have to deal with up to 30 different predictions for each clinical trial, leading to 30 rulesets and 30 ROC curves. It makes the manual rule selection harder. In addition to the previous rule selection method, we propose a solution to determine automatically the best cut point. Thus, we can obtain a classifier with good performance without the intervention of the decision maker. A final ruleset classifier is generated from all obtained rulesets coming from archive, as described in Algorithm 3. After merging all rules into one ruleset, the ROC curve is drawn. The subruleset giving the point closest to the ideal point (0,1) according to the Euclidean distance is chosen. All rules after this point are removed (or disabled if we want to allow the decision maker to change the *Sensitivity* accordingly to his needs). Once this ruleset is obtained, common data mining measures can be computed on the entire ruleset: *Confidence*, *Support*, etc. Diverse cut conditions have been tested but only the above presented one gives classifiers with interesting performance. An improvement of this condition could consist in weighting *true positives rate* and *false positives rate* according to decision maker's needs, since *false positives* can be more or less important than *true positives*, depending on the context.

Algorithm 3 Obtain a ruleset RS_{all} from a set of rulesets RS_i

```
create an empty ruleset RS_all
/* merge all obtained rules into RS_all */
for obtained ruleset RS_i do
   for each rule R_j ∈ RS_i {R_1, R_2, ..., R_n} do
      /* avoid duplicates */
      if R_j ∉ RS_all then
         RS_all.add(R_j)
      end if
   end for
end for
rocCurve ← RS_all.plotROCcurve()
/* best point of ROC curve is (0,1) */
i ← rocCurve.getIndexOfPointClosestToBestPoint()
RS_all.removeRules(i+1,N)
```

4 Experiments and Results

This section first introduces the protocol used in all our experiments. Both benchmarks and real datasets were used for experiments; the first part presents results obtained on benchmarks and compares them to the ones obtained by algorithms of literature. In the last part we compare *C4.5* and *C4.5-CS* decision tree algorithms and our approach on a real dataset having an important imbalance and a large number of attributes.

4.1 Protocol

According to the protocol proposed by Fernandez *et al.*, our algorithm was run 25 times for each dataset. We use 5-fold cross-validation: datasets are split into 5 parts, each containing 20% of observations. Then 4 parts are used for training, 1 for evaluation. For each available partition, as the algorithm contains some stochastic components, it was run 5 times. So we obtain for stochastic algorithms 25 Pareto fronts for each dataset. For each partition, solutions are evaluated on both training and test partitions. In our case, the objective is to maximize the results on test data, because it shows the ability of the algorithm to handle unseen data. A discretization of data was applied with Weka when necessary (*weka.filters.unsupervised.attribute.Discretize ; bins=10, findNumBins=true*) to allow our algorithm to handle datasets containing continuous attributes.
Generally *accuracy* measure is used to assess the performance of classification. Previously we saw that *accuracy* is not effective to handle class imbalance. Therefore Fernandez *et al.* proposed to use *Geometric mean of the true rates* (GM):

$$GM = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{FP+TN}}. \tag{3}$$

In order to have a good score, a classifier has now to classify correctly both classes: positive and negative. GM has one drawback though: when a classifier is not able to predict one class, score is worth 0. Here, when two classifiers failed to predict the negative class, there is no difference between the classifier able to find 50% of positive observations and an other classifier predicting 70% of positive observations: both have a score of 0.

For each dataset we computed the average of GM values obtained in each 25 runs. In order to get a single GM value from the rulesets proposed by our algorithm, we generated a ruleset, its ROC curve and cut it automatically as shown previously. Then we computed GM on the resulting ruleset: if an observation matches a rule from the ruleset it is considered as positive class. Observations not matching any rule are considered as negative class.

Tests were carried out on a computer with a Xeon 3500 quad core and 8 GB of memory, under Ubuntu 12. We used Weka software version 3.6 for discretization of datasets and for running C4.5 tests. Our approach is implemented in C++, using metaheuristics from ParadisEO framework [17]. In our experimentations we set *MOCA-I* max ruleset size = 5, max rule size = 9 for each dataset.

4.2 Experiments on Imbalanced Benchmarks Datasets

Fernandez *et al.* performed a comparison of 22 classification rule mining algorithms on imbalanced datasets and provided material to compare to their results [1]. Since our algorithm is designed to handle discrete attributes, datasets with less continuous attributes were preferred. We selected 6 imbalanced datasets in

those proposed by Fernandez *et al.* Their details are available in Table 3. The degree of class imbalance varies from 0.77 % to 27.42 %: in the *abalone19* dataset the positive class happens on only 0.77% of observations. In addition to these datasets, *tia* dataset - a real dataset - will be used in the next experiments. The *tia* dataset comes from hospital data. It is composed of 10,000 patients taking values in 10,250 available attributes: medical procedures and diagnoses. The objective is to predict the presence of the diagnosis *Transient cerebral ischemic attack*, available on 0.74% of the observations. In order to allow some state-of-the-art algorithms processing this dataset, the number of attributes is reduced. Only attributes available on at least one observation having the class are kept, leading to 699 attributes.

Table 3. Datasets main attributes

#ind.: count of observations; #feat.: count of attributes; % repar.: Percentage of observations having the class to be predicted

name	#ind.	#feat.	% repar.
haberman	306	3	27.42
ecoli1	336	7	22.92
ecoli2	336	7	15.48
yeast3	1484	8	10.38
yeast2vs8	482	8	4.15
abalone19	4174	8	0.77
tia	10,000	699	0.74

Results are available in Table 4. Our approach is denoted *MOCA-I* (Multi-Objective Classifier Algorithm for Imbalanced data). We compared only to algorithms giving the best results regarding the average of GM over the 25 runs. In addition, we will also compared to *C4.5-CS* – a cost-sensitive version of *C4.5* available in KEEL Framework [19]. For each dataset the best average of GM is recorded. Then we computed relative error to the obtained best: a score of 0 indicates the algorithm got the best result on the dataset. As an indication, the last line indicates the average of the relative errors, over the 6 datasets.

We can observe that the majority of algorithms had some difficulties to handle *abalone19* dataset, which has a high imbalance. Our model outperforms other algorithms on 3 datasets. On the 3 remaining datasets, *C4.5-CS* outperforms all algorithms. However, when outperformed the model still gives interesting results.

4.3 Experiments on a Real Dataset

In addition to literature datasets, we tested the scalability of our method on one real large dataset: the previously presented *tia* dataset. We compared to results obtained by J48 – the *C4.5* algorithm implementation of Weka; well-known by some medical users. Since *C4.5* may encounters trouble to deal with imbalanced data, we compared to results obtained by *C4.5-CS* algorithm that obtained good

Table 4. Relative error to the best average of GM: algorithms with 0 obtained the best average of GM

MOCA-I: Multi-Objective Classifier Algorithm for Imbalanced data; SIA: Supervised Inductive Algorithm, O-DT: Oblique Decision Tree, CORE: CO-Evolutionary Rule Extractor, GAssist: Genetic Algorithms based claSSIfier sySTem, OCEC: Organizational Co-Evolutionary algorithm for Classification, DT-GA: Hybrid Decision Tree - Genetic Algorithm, HIDER: HIerarchical DEcision Rules

	MOCA-I	XCS	O-DT	SIA	CORE	GAssist	OCEC	DT-GA	HIDER	C4.5	C4.5-CS
haberman	**0.00**	0.41	0.04	0.16	0.40	0.27	0.27	0.42	0.48	0.42	0.19
ecoli1	0.05	0.04	0.10	0.19	0.03	0.05	0.36	0.07	0.16	0.06	**0.00**
ecoli2	**0.00**	0.66	0.06	0.05	0.15	0.04	0.42	0.15	0.35	0.07	0.03
yeast3	0.01	0.81	0.10	0.10	0.23	0.06	0.01	0.10	0.43	0.07	**0.00**
yeast2vs8	0.11	0.18	0.18	0.88	0.14	0.26	0.16	0.88	0.18	0.88	**0.00**
abalone19	**0.00**	1.00	0.88	1.00	1.00	1.00	0.03	1.00	1.00	1.00	0.52
err. average	0.03	0.52	0.40	0.23	0.32	0.28	0.21	0.44	0.43	0.42	0.12

Table 5. Comparison to *C4.5* and *C4.5-CS* on the real dataset *tia*

	GM on Training	GM on Test
MOCA-I	0.92 ± 0.02	**0.74 ± 0.07**
C4.5	0.58 ± 0.03	0.52 ± 0.11
C4.5-CS	**0.99 ± 0.0009**	0.47 ± 0.11

results on the benchmark datasets. Each algorithm was run 5 times using 5-fold cross-validation to obtain 25 Pareto fronts for *MOCA-I*. *C4.5* and *C4.5-CS* uses default parameters provided by Weka and KEEL.

As observed in Table 5, reporting average and standard deviation of GM, on training and test data, our approach obtains a better GM score than *C4.5* and *C4.5-CS*, on test datasets. *C4.5-CS* is more effective than *C4.5* and *MOCA-I* on training data but is subject to over-fitting: it encounters problems when dealing with unknown observations, like on the test data. In addition to its best performance on test data, *MOCA-I* uses the previously presented post-processing method to output a ruleset with different cut possibilities. In Table 5 the ruleset is cut to improve GM; Table 6 shows results obtained with different cut points over the ROC curve. *Cut 1* is a cut where there is no false positive. *Cut 3* is the cut presented previously in our post-processing method. *Cut 2* is between these two cuts on the ROC curve. The cut point can be adapted depending on the cost of a *false positive*. When no error is tolerable, cut points bringing less false positives will be preferred (like *Cut 1* or *Cut 2* in Table 6).

5 Conclusion and Further Research

We proposed and implemented a Pittsburgh classification rule mining system using partial classification rules, adapted to imbalanced data. Our method based on a multi-objective local search using *Confidence*, *Sensitivity* and rule length

Table 6. Impact of rule selection

Confidence (Cf), Sensitivity (Se) and Geometric mean of the true rates (GM) on one fold, with different cuts after ROC post-processing. Tra=training data, Tst=test data

	Cf tra	Cf tst	Se tra	Se tst	GM tra	GM tst
MOCA-I cut 1	1	1	0.36	0.14	0.60	0.38
MOCA-I cut 2	1	0.75	0.53	0.21	0.72	0.46
MOCA-I cut 3	0.10	0.06	0.86	0.71	0.90	0.81
C4.5	1	0.75	0.39	0.21	0.62	0.46

was shown to be effective in this context, compared to state-of-the-art rule mining classification algorithms. Moreover, it was proven to be more effective than *C4.5* and *C4.5-CS* on real hospital data to predict unknown observations. The use of partial classification rules avoids some common inconsistencies brought by Pittsburgh modeling, simplifying the conception of neighborhood operators. Thanks to DMLS algorithm, parameters are easier to configure than in other approaches based on genetic algorithm, while giving best results. To overcome one of the drawback of dominance-based algorithms, obtaining an archive of 400 and more compromise solutions, we proposed two methods based on ROC curve. The first helps the final user to choose the rules to keep, while the second automatically generates one single solution without the intervention of the user.
Further research may include the development of new neighborhood operators, like a covering operator that introduces rules concerning uncovered individuals. Operators dealing with attribute granularity can be interesting, like the one presented in Plantevit *et al.* work [20]. They will allow generalizing or specializing rules, defining new rule neighbors:

- $R1$: *juvenile diabetes* $\rightarrow$ *increased risk of stroke*
- $R1'$: *diabetes* $\rightarrow$ *increased risk of stroke*

With enhanced computational power another interesting approach would be to use the *area under the ROC curve* (AUC) as an optimization criterion, instead of *Sensitivity* and *Confidence*. Or the left-most portion of the area under the curve (LAUC) as defined by Zhang *et al.* [21], if we want to allow a fixed number of false positives.

References

1. A. Fernández, S. Garciá, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: State of the art, taxonomy, and comparative study," *Evolutionary Computation, IEEE Transactions on*, vol. 14(6), pp. 913–941, dec. 2010.
2. J. R. Quinlan, *C4.5: programs for machine learning*, 1993.
3. N. V. Chawla, "Data mining for imbalanced datasets: An overview," *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, 2010.
4. T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM SIGKDD Explorations Newsletter*, vol. 6(1), pp. 40–49, 2004.

5. L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Computing Surveys (CSUR)*, vol. 38(3), 2006.
6. M. Ohsaki, H. Abe, S. Tsumoto, H. Yokoi, and T. Yamaguchi, "Evaluation of rule interestingness measures in medical knowledge discovery in databases," *Artificial Intelligence in Medicine*, vol. 41, pp. 177–196, 2007.
7. S. Greco, Z. Pawlak, and R. Slowiński, "Can bayesian confirmation measures be useful for rough set decision rules?" *Engineering Applications of Artificial Intelligence*, vol. 17, no. 4, pp. 345 – 361, 2004.
8. J. Bayardo and R. Agrawal, "Mining the most interesting rules," in *Proceedings of the fifth ACM SIGKDD*, ser. KDD '99, 1999, pp. 145–154.
9. J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14(5), pp. 465–471, 1978.
10. A. Reynolds and B. D. L. Iglesia, "Rule induction for classification using multi-objective genetic programming," in *Evolutionary Multi-Criterion Optimization*, ser. LNCS, 2007, vol. 4403, pp. 516–530.
11. J. Bacardit, M. Stout, J. D. Hirst, K. Sastry, X. Llorà, and N. Krasnogor, "Automated alphabet reduction method with evolutionary algorithms for protein structure prediction," in *GECCO*, 2007, pp. 346–353.
12. D. Corne, C. Dhaenens, and L. Jourdan, "Synergies between operations research and data mining: the emerging use of multi-objective approaches," *European Journal of Operational Research*, vol. 221(3), pp. 469–479, 2012.
13. S. Srinivasan and S. Ramakrishnan, "Evolutionary multi objective optimization for rule mining: a review," *Artificial Intelligence Review*, pp. 1–44.
14. C. A. Coello Coello, C. Dhaenens, and L. Jourdan, *Advances in Multi-Objective Nature Inspired Computing*, ser. Studies in Computational Intelligence, 2010.
15. J. Casillas, P. Martínez, and A. Benítez, "Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 451–465, 2009.
16. A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E.-G. Talbi, "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems," *J. Heuristics*, vol. 18, pp. 317–352, 2012.
17. A. Liefooghe, L. Jourdan, and E.-G. Talbi, "A software framework based on a conceptual unified model for evolutionary multiobjective optimization: Paradiseo-moeo," *European Journal of Operational Research*, vol. 209, no. 2, pp. 104–112, 2011.
18. T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27(8), pp. 861–874, 2006.
19. J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, and F. Herrera, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 307–318, 2009.
20. M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, and Y. W. Choong, "Mining multidimensional and multilevel sequential patterns," *ACM TKDD*, vol. 4(1), pp. 1–37, Jan. 2010.
21. J. Zhang, J. W. Bala, A. Hadjarian, and B. Han, "Learning to rank cases with classification rules," *Preference Learning*, pp. 155–177, 2011.