One-Class Ant-Miner: Selection of Majority Class Rules for Binary Rule-based Classification

Naser GHANNAD[1,2][0000−0001−8491−9150], Roland DE GUIO[1,2][0000−0002−8201−9551], and Pierre PARREND[1,3][0000−0002−1680−1182]

[1] ICube (Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie), UMR 7357, Université de Strasbourg, CNRS, 67000 Strasbourg, France
[2] INSA de Strasbourg, Strasbourg, France
[3] EPITA, 5, Rue Gustave Adolphe Hirn, Strasbourg, France
`Naser.Ghannad@insa-strasbourg.fr`

Abstract. In recent years, high-performance models have been introduced based on deep learning; however, these models do not have high interpretability to complement their high efficiency. Rule-based classifiers can be used to obtain explainable artificial intelligence. Rule-based classifiers use a labeled dataset to extract rules that express the relationships between inputs and expected outputs. Although many evolutionary and non-evolutionary algorithms have developed to solve this problem, we hypothesize that rule-based evolutionary algorithms such as the AntMiner family can provide good approximate solutions to problems that cannot be addressed efficiently using other techniques. This study proposes a novel supervised rule-based classifier for binary classification tasks and evaluates the extent to which algorithms in the AntMiner family can address this problem. First, we describe different versions of AntMiner. We then introduce the one-class AntMiner (OCAntMiner) algorithm, which can work with different imbalance ratios. Next, we evaluate these algorithms using specific synthetic datasets based on the AUPRC, AUROC, and MCC evaluation metrics and rank them based on these metrics. The results demonstrate that the OCAntMiner algorithm performs better than other versions of AntMiner in terms of the specified metrics.

Keywords: AntMiner · Evolutionary algorithm · Rule-based classifier · Ant colony classification · Imbalanced dataset · Binary classification · Synthetic datasets.

1 Introduction

In recent years, various highly scalable models have been developed using deep learning [6] and other machine learning models such as XGBoost [3]. Most of these models are black-box models that are not interpretable by users. A few of these models specify the importance of features to help users understand which features are more helpful for predicting classes [10]. However, they do not provide explicit relationships that allow human users to understand the relationships between input and output variables, unlike a white-box model.

Because rule-based classifiers [1] explicitly rely on individual variables in the original data, they are powerful candidates for constructing white-box models. Rule-based classifiers extract rules to explain the effects of individual variables on a given class, as follows:

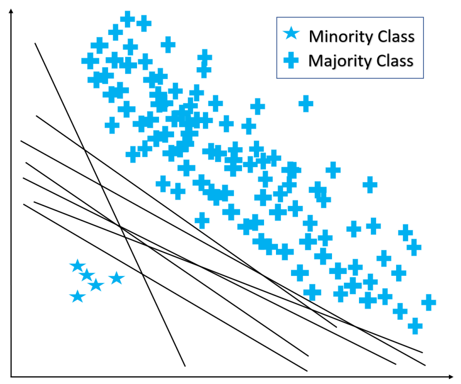
$$IF\ (Conditions)\ THEN\ (Consequent) \tag{1}$$

Conditions are combinations of propositions for different input variables (terms) bound by a logical conjunction (AND). The result of these combinations is the consequent (i.e., classes). In this study, we focused on ordered-rule-based classifiers. There are two methods for extracting rules: direct and indirect. In direct methods, the algorithm directly operates on the data and extracts rules from the data, as seen in the RIPPER [5], CN2 [4], PART [9], and RISE [7] algorithms, or uses evolutionary algorithms such as the ant colony algorithm (AntMiner) [16] to extract rules. In indirect methods, a classifier is first applied to the data and then another method extracts rules from the classifier. Such methods include the C4.5, J48 [18, 19], random tree [17], and REPTree [20] algorithms. These tree-based algorithms first use a decision tree to classify data. Rules are then extracted from the trees provided by the algorithms.

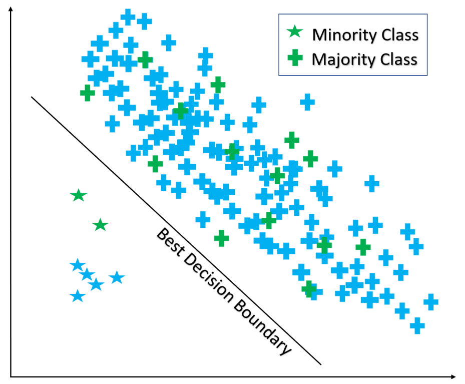
The goal of this study was to develop an algorithm to extract rules from provided datasets that works well with both imbalanced and balanced datasets, and also determine a suitable metric for ranking algorithms based on datasets with various imbalance ratios. Additionally, considering the lack of suitable datasets for evaluating rule-based algorithms, we aimed to generate datasets containing all possible instances for generating output classes so that we could determine which algorithms could achieve the highest values for the defined metrics with the availability of all possible instances and absence of noise. Also, with these data, we can find out which algorithm is over-fitted or under-fitted on the data. The remainder of this article is organized as follows. Section 2 provides an overview of evolutionary approaches. Section 3 defines the developed one-class AntMiner (OCAntMiner) algorithm, and Section 4 provides an evaluation of the considered algorithms. Section 5 presents the evaluation results. Finally, Section 6 concludes this article.

2 Related Work

Evolutionary algorithms (EAs) are used in optimization problems. They represent a subset of evolutionary computations [21]. The EA approach is inspired by biological evolution and uses operations such as mutation, recombination (crossover), and selection. A population evolves with the goal of maximizing a given evaluation function. The main EAs used for learning rules are the learning classifier system (LCS) [2] and AntMiner algorithms [16]. The LCS was introduced by John Holland [11], and genetic algorithms were used to extract rules. In contrast, AntMiner uses a simulated ant colony as a probabilistic approach to solve graph problems and find the best path to optimize an evaluation function. Ants choose their paths based on the amount of pheromones along paths,



(a) Different solutions for defining a boundary



(b) best boundary separator between two classes of data points

Fig. 1: Different data boundaries based on noise or new data entry

which represents how many times a path has been selected successfully before. Releasing pheromones in an environment (along paths) is a method for ants to communicate. The path with the most pheromones should be shorter than the others according to the current state of the search [8].

In this study, we considered both the AUPRC and AUROC to rank different algorithms. By using relevant datasets and metrics, we identified the limitations of the AntMiner algorithms in terms of handling datasets with high imbalance ratios and attempted to overcome these limitations. To this end, we developed a novel algorithm called OCAntMiner, which is presented in section 3.

3 OCAntMiner

This section presents our proposed OCAntMiner (One-Class AntMiner) algorithm. As shown in **Fig1a**, it is possible to define different boundaries for discriminating two classes of data, which becomes particularly important when dealing with imbalanced data (i.e., when the amount of data in one class is significantly less than the amount of data in another class).

The amount of data that can express a minority class is very small and noise in these data can completely distort the output of prediction. As shown in **Fig1a**, only five data points belong to class Star (minority class) and all other data belong to class Plus (majority class). Our goal in this study was to develop an algorithm that works on existing data without the need to add or remove data. As shown in **Fig1b**, if we add green data points to the existing data (either noise or true data) and if these data are added to the minority class, they can significantly change the boundaries of the minority class. However, for the majority class, the addition of new data does not significantly shift the boundary because the rest of the data can pinpoint the location of the boundary.

```
Input  : $TrainingSet : all.training.cases$
Output: Discovered.Rules.list[]
$WCTP = MajorityClass$;
while $TrainingSet$ >$Max$ $uncovered$ $cases$ do
    $t = 1$ ;                                              /*ant index*/
    $j = 1$ ;                                 /*convergence test index*/
    $pheromones = init.phermones()$;
    $Rule = []$ ;                                          /*empty rules*/
    repeat
        Rule[t] = add terms based on heuristic function and pheromones;
        Prune Rule[t] ;                          /*based on quality function*/
        if Consequent of Rule[t] != WCTP then
            Quality[Rule[t]]=0;
        end
        Update the pheromones of all ants;
        if Rule[t] is equal to Rule[t-1] then
            j = j + 1;
        else
            j = 1;
        end
        t = t + 1;
    until $(t >= No.of.ants)$ OR $(j >= No.rules.converged)$;
    R.best =best(Rule) ; /*Rule with highest quality among all Rules*/
    if Consequent of R.best == WCTP then
        Add rule R.best to Discovered.Rules.List; ;
        TrainingSet=TrainingSet-(set of cases correctly covered by R.best);
    end
end
```

Algorithm 1: High-level pseudocode for OCAntMiner

In previous versions of AntMiner, the majority class was considered as the default class and the algorithm searched for rules to explain the minority and majority classes with no restrictions on finding rules for each class. As a result, the algorithm could provide rules to describe the majority class and rules to describe the minority class, or only rules to describe the minority class. Our idea is to limit the algorithm to the majority class and extract rules only for that class. The reason for choosing the majority class to extract rules instead of the minority class is that there are more data for the majority classes in datasets, meaning more precise rules can be derived to express such classes. If we can describe one class very well, then another class will be easily discriminable. One goal of this study was to evaluate the impact of integrating this approach into AntMiner-based algorithms. The OCAntMiner pseudocode is presented in **Algorithm 1**. OCAntMiner extends the original AntMiner by focusing on the extraction of the majority class, which is defined as the class with the highest frequency in the class distribution of training samples. The pheromone update, pheromone initialization, heuristic, and quality functions rely on the AntMiner model. To focus on the majority class, the first step is to detect which class to predict (WCTP), afterward, algorithm was modified to extract rules related to this class (the modifications are highlighted by red in **Algorithm 1**). Most other changes to the original version aim to prevent the algorithm from generating minority class rules. In line 10, the algorithm checks for the consequents of extracted rules. If the consequent does not match the value of the majority class, then the quality of the rule is equal to zero. Similarly, in line 22, when all ants provide their solutions (rules), the best solution (R.best) is selected based on the quality measure. If the consequent of R.best is equal to the value of the majority class, then R.best is added to the list of discovered rules and the cases correctly covered by R.best are removed from the training set.

4 Evaluation

In this section, we first provide a detailed overview of our datasets and then our methodology for evaluating algorithms. Subsequently, relevant evaluation metrics are identified and justified. Finally, the algorithms selected for the evaluation process are presented with their configurations.

4.1 Selected Algorithms

To demonstrate the performance of the proposed OCAntMiner algorithm compared to other algorithms, we selected algorithms from the category of evolutionary algorithms for the sake of fair comparison. However, we also went a step further and compared it to non-evolutionary algorithms to demonstrate the power of the proposed algorithm. We selected different versions of AntMiner, namely, the original version of AntMiner [12], cAntMinerPB [15], UCAntMinerPB [13], and cAntMiner [14]. For direct and non-evolutionary algorithms, we selected RIPPER [5], PART [9], and RISE [7], and for indirect algorithms, we selected J48 [18, 19], and REPTree [20].

4.2 Algorithm parameters

We used the same input parameters for all algorithms, namely, "Size of Ant colony" = 60, "Maximum number of iterations" = 1500, "Minimum covered cases per rule" = 10, "Number of uncovered cases set" = 10, and "Rule quality function" = Sensitivity $\times$ Specificity. In the next section, we present the results of different algorithms based on the described metrics.

5 Results and Discussion

5.1 Ranking of Algorithms

In this section, we rank the algorithms using a statistical approach based on AutoRank tools using both UCI and synthetic datasets. Because the data under analysis do not follow a normal distribution, the Friedman test with the Nemenyi post-hoc test was applied to rank the algorithms and divide the algorithms into different groups based on the critical distance (CD) metric. As shown in **Fig 2**, the algorithms were ranked based on the AUROC metric using UCI datasets. One can see that the most performant algorithm is the proposed OCAntMiner, followed by J48. This figure also shows that OCAntMiner, J48, RIPPER, and IREP are in the same group and there is no significant difference between them, but they are significantly different from the other algorithms. It also shows that the original version of AntMiner is in the group with the worst results along with different versions of AntMiner. With our modifications, it jumps to the first group and first rank. Additionally, we performed another test with 24 synthetic datasets and 50% random sampling. The results are presented in **Fig 3**. As shown in this figure, RIPPER has the highest ranking, followed by OCAntMiner. This figure also shows that OCAntMiner provides significantly better results than the other versions of AntMiner and is clearly in the top group. These results indicate that despite being in the same group as RIPPER, there is still room for improving OCAntMiner.

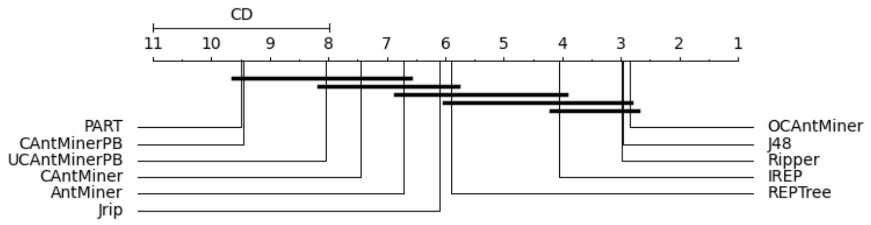


Fig. 2: AUPRC on UCI datasets with 10-fold cross-validation.

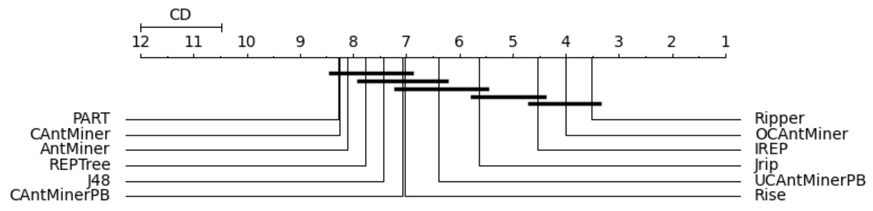


Fig. 3: AUPRC on synthetic datasets with 50% of data with 10-fold.

6 Discussion and Conclusions

In this study, we focused on rule-based classifiers, specifically the AntMiner algorithm family, for extracting classification rules from a given dataset. The most important results for the classification task are validation results because if such results are not calculated properly, it can lead to incorrect directions for improving existing algorithms. For validation, there are two important features, the datasets, and metrics used for validation. Most studies have used the UCI database for evaluating algorithms in different areas with varying complexities. However, most UCI datasets cannot provide all possible instances for data inputs to facilitate the measurement of the intrinsic performance of algorithms when seeking a binary function. Interestingly, no AntMiner algorithm was able to consistently achieve values of 100% for the metrics used when all instances were provided. When evaluating and ranking different algorithms using synthetic datasets with 50% of all data instances, overfitting or underfitting to the given data may have occurred. The same phenomenon should occur with the UCI data, but because we did not have all data instances, we could not verify this phenomenon.

Another limitation of using UCI datasets is that we do not know the extent to which the data used for ranking cover different complexities for the target problem. Our contribution to solving these problems was the introduction of a dataset generator that generates datasets with various imbalance ratios, covers different complexities, and provides all possible instances for a given number of input parameters. With all possible instances, we can check whether the algorithm is overfitted or underfitted to the data. Therefore, we divided our tests into two scenarios. The first scenario used 100% of the data for training and the same data for testing. In the second scenario, we used 50% of all data instances with random sampling for the training phase and 100% of the instances for the testing phase. This allowed us to check how classifier rules were extracted for all data instances. The results of the different algorithms for 100% and 50% of all possible instances are presented in **Table ??**. As shown on the right side of this table (i.e., 100% sampling), the maximum values for AUROC, AUPRC, and MCC are 100% and the Rise algorithm can achieve these values, indicating that this algorithm either overfits the data or fits the data properly. To understand which one of these scenarios occurred, we should consider the results for 50% of the data (i.e., left side of the table). As shown using 50% of data, Rise algorithm is not ranked first anymore and shows a very poor result for the MCC metric (i.e., 45.74%), indicating that the algorithm likely overfitted the data, which is why the solution provided for 50% of the data is not as good as that for 100% of the data. Another interesting point in this table is that the OCAntMiner and RIPPER algorithms retain the same ranks for the two different sampling percentages based on different metrics, indicating that these two algorithms are robust to overfitting. This also indicates that they are likely not underfitted to the data because they achieve the highest values for different metrics.

Another issue that we observed in AntMiner was that this algorithm sets the majority class as the default class and then attempts to find rules to describe

the majority and minority classes. As a result, three outputs may be generated at the end of executing the algorithm: 1) some rules describing the majority class and some rules describing the minority class, 2) all rules describing the minority class, or 3) no rules extracted from the data and only the default class is used for all data instances. As shown in **Fig 1**, the distribution and boundary of majority data points are more reliable than those of minority data samples because there may be too few instances for describing the minority class. As a result, noise in the data or a lack of instances may confuse the extraction of rules. This is why simply describing the majority class seems to be a powerful approach to solving this problem. Our main contribution in this study is highlighting the relevance of this approach, which, to the best of our knowledge, has not yet been applied to any AntMiner-based algorithm. We implemented this approach by modifying the first version of AntMiner and forced the algorithm to extract rules for the majority class alone. Therefore, the algorithm used the minority class as the default class and attempted to find the rules for majority data instances. As shown in **Table ??**, after adding this feature to the first version of AntMiner, the number of extracted rules decreased dramatically (43%) and the classifier with fewer rules could still detect the behavior of the data. This approach also reduced the runtime by 47%. Furthermore, the AUPRC was improved by 15.87%. These results demonstrate that with this added feature, we can reduce the runtime and number of rules while improving data classification performance. Additionally, as shown in **Table ??**, by adding this feature, we achieved a large jump in performance (from the second-worst AntMiner to the best). This jump demonstrates the strength of the components added to the algorithm to handle datasets with various imbalance ratios.
We performed a statistical test using the Demsar method to rank the algorithms considering the uncertainty in ranking caused by the number of datasets used in the experiments and variance of the results. As shown in **Fig 2**, we first ranked the different algorithms based on the UCI datasets with 10-fold cross-validation to demonstrate how the proposed algorithm works on datasets used in previous studies. The results reveal that OCAntMiner ranks first among all evolutionary and non-evolutionary algorithms. Statistically, it is in the same group as the J48, RIPPER, and IREP algorithms. We also applied the Demsar method to the algorithms using 50% of all possible instances, and the results are presented in **Fig 3**. The results show that OCAntMiner ranks second among all algorithms, but ranks first among the different versions of AntMiner. Additionally, this test indicated that OCAntMiner belongs to the same group as RIPPER and IREP. This figure also reveals that the ranking of the J48 algorithm is dramatically reduced compared to that in the previous figure using the UCI data. Several interpretations can explain this phenomenon. For example, synthetic datasets are more general than UCI datasets and cover more complex problems. Under these conditions, J48 may not classify some datasets properly or the five UCI datasets may not be sufficient to rank the algorithm (5% error in ranking). Finally, we demonstrated that modifying the AntMiner process can significantly improve its results without supplementing or modifying the heuristic or quality

functions. In future work, we will consider adding this feature to other AntMiner models and analyzing the resulting algorithm behavior.

Acknowledgements This research was conducted within the framework of the Offensive Sciences project number 13.11 "Virtual Innovative Real Time Factory" (VIRTFac) which benefits from the financial support of the Offensive Sciences programme of the Upper Rhine Trinational Metropolitan Region, the INTERREG V Upper Rhine programme and the European Regional Development Fund (ERDF) of the European Union.

References

1. R. Agrawal, R. Srikant, and others. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Citeseer, 1994.
2. L. Bull and T. Kovacs. Foundations of learning classifier systems: An introduction. In *Foundations of Learning Classifier Systems*, pages 1–17. Springer, 2005.
3. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
4. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, Mar. 1989.
5. W. W. Cohen. Fast Efective Rule Induction. page 10, 1995.
6. S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, Sept. 2020.
7. P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, Aug. 1996.
8. M. Dorigo and L. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
9. E. Frank and I. Witten. Generating Accurate Rule Sets Without Global Optimization. *Machine Learning: Proceedings of the Fifteenth International Conference*, June 1998.
10. N. Ghannad, R. De Guio, and P. Parrend. Feature Selection-Based Approach for Generalized Physical Contradiction Recognition. In *International TRIZ Future Conference*, pages 321–339. Springer, 2020.
11. J. H. Holland. Adaptation**Research reported in this article was supported in part by the National Science Foundation under grant DCR 71-01997. In R. ROSEN and F. M. SNELL, editors, *Progress in Theoretical Biology*, pages 263–293. Academic Press, 1976.
12. B. Liu, H. Abbass, and R. McKay. *Density-based heuristic for rule discovery with ant-miner*. Jan. 2002.
13. F. E. B. Otero and A. A. Freitas. Improving the Interpretability of Classification Rules Discovered by an Ant Colony Algorithm: Extended Results. *Evolutionary Computation*, 24(3):385–409, Sept. 2016.

14. F. E. B. Otero, A. A. Freitas, and C. G. Johnson. Handling continuous attributes in ant colony classification algorithms. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Data Mining (CIDM 2009)*, pages 225–231. IEEE, 2009.
15. F. E. B. Otero, A. A. Freitas, and C. G. Johnson. A New Sequential Covering Strategy for Inducing Classification Rules with Ant Colony Algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):64–74, 2013.
16. R. S. Parpinelli, H. S. Lopes, and A. Freitas. An ant colony based system for data mining: applications to medical data. 2001.
17. B. Pfahringer. Random model trees: an effective and scalable regression method. 2010.
18. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
19. J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
20. D. B. Srinivasan and P. Mekala. Mining social networking data for classification using REPTree. *International Journal of Advance Research in Computer Science and Management Studies*, 2(10), 2014.
21. P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, 2016.