Ant-Miner Specializations to tackle Imbalanced Data Sets

Murilo Zangari de Souza[1], Aurora Trinidad Ramirez Pozo[1], Wesley Romão[2]
Universidade Federal do Paraná[1]
Universidade Estadual de Maringá[2]
{murilo.zangari, auroratrinidad, wesley.uem }@gmail.com

***Abstract* — A data set is named imbalanced when the classes have not approximately equally representations. Classification algorithms are sensitive of this imbalance and tend to valorize the majority classes and ignore the minority classes, which is a problem when the minority classes are the classes of interest. In this paper we propose two specializations to an efficient and robust classification algorithm inspired by ACO metaheuristic called Ant-Miner. These specializations modify how rules are constructed and evaluated. We compare the results with standard Ant-Miner and C4.5 algorithm. The results show that the proposed algorithms are competitive, finding rules for the minority classes and improve the simplicity of the discovered rule list.**

Keywords — class imbalance, classification algorithms, ACO, Ant-Miner.

I. INTRODUCTION

The class imbalance problem emerged when machine learning matured from an embryonic science to an applied technology, broadly used in the worlds of business, industry and scientific research [1]. This increase in interest gave rise to two workshops held in 2000 at AAAI (Association of the Advancement of Artificial Intelligence) conference [2], in 2003 at ICML (International Conference on Machine Learning) conference [3] and a special edition of ACM SIGKDD Explorations in 2004 [4] [5] [6].

In practical applications, the ratio of the small to the large classes can be drastic such as 1 to 100, 1 to 1.000, 1 to 10.000 and sometimes even greater. In the classification task, a data set is deemed imbalanced when there are more cases of some class than the others. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones, because the cases of minority class have low representation on the training set. In many real data sets, it is the minority class that is of primary interest. Classifiers and metrics that do not take this into account generally do not perform well in these situations [1] [4] [22].

Several techniques to solve the class imbalance problem have been proposed both at the data and algorithmic levels. At data levels (preprocessing), these solutions include many forms of sampling [6] [7] and feature selection [23]. At the algorithmic level, solutions include: hybrid classifiers, adjust cost of learning a class, bias adjusting, learning from one class, appropriate evaluation metric [4] [25].

This paper proposes two specializations to a competitive and robust classification algorithm inspirited by the Ant Colony Optimization (ACO), developed by Parpinelli in 2002 [8], called Ant-Miner classification algorithm. The specializations tackle the imbalance issue, finding rules to minority classes (interest class) a priori improving its predictive accuracy and simplicity.

The reminder of this paper is organized as follows. Section II the class imbalance problem are shortly explained. Section III describes standard Ant-Miner. Section IV explains the Ant-Miner specializations. Section V the setup and results of experiments are discussed. Section VI concludes this paper.

II. THE CLASS IMBALANCE PROBLEM

Classifiers algorithms aim to obtain a model with high prediction accuracy and a good generalization capability. Thus, the inductive bias benefits the covering of the majority examples and the minority examples can be completely ignored by the model. Therefore, the models or results classifiers show low predictive accuracy with respect to the minority class (called, positive class) [4] [5].

Other characteristic of the imbalance problem is the presence of rare cases. Rare cases can lead the occurrence of small disjunts. Small disjunts are rules that cover few cases, this rules can have more errors than large disjunts [5].

A number of solutions to the class imbalance problem were proposed. One of the most common techniques for dealing with rarity is sampling. The basic idea is to minimize the imbalanced by altering the distribution of training examples. Another method to cope with imbalanced data sets is to use more appropriate evaluation metrics. Depending on the problem a method can be more efficient than others. It is possible to use more than one method simultaneously. Three methods are described below.

A. Using a More Appropriate Inductive Bias

Most classifier algorithms use bias induction favoring generalization instead of specialization because the specialization leads to overfitting problems. This bias can adversely impact the ability to learn rare examples. The maximum-generality bias works well for large disjuncts but not for small disjuncts [4]. Weiss [4] cite some studies with this approach, one of them use CN2 algorithm with the maximum specificity bias. It was shown that the approach improves the performance on small disjuncts set but degrades the

performance on large disjuncts set yielding poorer overall performance.

Our work uses bias adjusting to improve the performance of the small disjuncts without affecting the performance of the large disjuncts.

B. Sampling Methods

Sampling methods aim to change the data distributions on the training set thus increasing the predictive accuracy of the models. A sampling can be obtained with deleting cases of the majority class (undersampling) or adding cases of the minority class (oversampling) [1] [5] [13]. There are both simple and advanced methods.

C. More Appropiate Evaluation Metrics

The ROC analysis (Receiver Operating Characteristics) is a graph method for visualizing, organizing and selecting classifiers based on their performance. ROC analysis is able to make more accurate machine learning evaluations, mainly when applied in data set with class imbalance [11] [12].

ROC graphs are two-dimensional graphs in which true positive (*TP*) rate is plotted on the Y axis and false positive (*FP*) rate is plotted on the X axis. A ROC graph depicts relative trade-offs between benefits (true positives) and cost (false positives). Given the confusion matrix, we can find the measures, as follow:

$$TP \text{ rate} = \frac{Positives\ correctly\ classified}{Total\ positives} = \frac{TP}{Pos}$$

$$FP \text{ rate} = \frac{Negatives\ incorrectly\ classified}{Total\ negatives} = \frac{FP}{Neg}$$

Several points in ROC space are important to note. The lower left point (0, 0) represents the strategy of never issuing a positive classification; such a classifier commits no false positive errors but also gains no true positive. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1, 1). The point (0, 1) represents perfect classification.

Some classifiers, such as a Naive Bayes or a neural network, naturally yield an instance *probability* or *score*. Such a *ranking* or *scoring* classifier can be used with a threshold to produce a discrete classifier. Each threshold produces a different point in ROC space, generating a ROC curve. To compare classifiers, the curve ROC is reduced to a single scalar value. A common method is calculating the area under the ROC curve, abbreviated AUC [11] [12] [25].

III. ANT-MINER OVERVIEW

The Ant-Miner, developed by Parpinelli et al. [8] is a classification algorithm based on the ACO (Ant Colony Optimization) metaheuristic [14], in which each ant incrementally builds/modifies a solution to a certain problem. Each detail on the original algorithm can be found in [8].

The rules generated are expressed in the form of *IF-THEN* rules, as follows:

IF (term$_1$ AND term$_2$ AND … term$_n$) THEN (class).

The antecedent part (*IF* part) contains a set of terms "conditions", usually connected by a logical conjunction operator (*AND*). Each term is a triple *(attribute, operator, value)*, such as (*sex=male*). Each term is labeled with a heuristic value (η) and pheromone value (τ). The rule consequent (*THEN* part) specifies the class predicted for cases whose predictor attributes satisfy all the terms specified in the rule antecedent. The Ant-Miner follows a sequential covering approach to discover a list of classification rules covering all the training cases.

The Algorithm 1 describes the pseudo code of Ant-Miner algorithm.

```
TrainingSet = all training cases;
DiscoveryRuleList = empty;
Calcule the heuristic value for each term attribute=value;
while TrainingSet > Max_uncovered_cases do
    i = 1; //ant index;
    j = 1; //convergence test index;
    Initialize all terms with the same amount of
    pheromone;
    repeat
        Ant_i starts with an empty rule and
        incrementally constructs a classification rule R_i
        by adding one term at time to the current rule;
        Prune rule R_i;
        Update the pheromone of all terms by
        increasing pheromone in the terms used by Ant
        proportional to the quality of rule R_i and
        decreasing in the others terms;
        if R_i is equal to R_{i-1} then update convergence
        test
            j = j + 1 ;
        else
            j = 1;
        end
        i = i+1;
    until (i ≥ no_of_Ants) or (j ≥ no_rules_converg);
    Choose the best rule R_best among all the rules R_i;
    Add the rule R_best to DiscoveredRuleList;
    Training = TrainingSet - {set of cases correctly
    covered by R_best};
end
Add the pattern rule at the end of DiscoveredRuleList;
```

Algorithm 1: Ant-Miner pseudo-code

Initially the *DiscoveryRuleList* is empty; the *TrainingSet* has all the training cases. Each iteration of the **while** loop of Algorithm 1, corresponding to a number of executions of the **repeat-until** loop (number of the ants) that discovers a set of candidate classification rules. The best rule is added to the *DiscoveryRuleList* and the training cases that are covered correctly by this rule (i.e., cases satisfying the rule antecedent and having the class predicted by the rule consequent) are removed from the training set. This process is performed iteratively while the number of uncovered training cases is greater than a user-specified threshold (*max_uncovered_cases*)

A. Pheromone Initialization

Let $term_{ij}$ be a rule condition of the form $A_i = V_{ij}$, where A_i is the i^{th} attribute and V_{ij} is the j^{th} value of the domain of A_i. In the beginning all the terms have the same amount of pheromone; this value is inversely proportional to the total number of terms, defined as follow:

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} bi} \quad (1)$$

where: a is the total number of attributes and b_i is the number of possible values that can be taken on by attribute A_i.

B. Heuristic Value

Each $term_{ij}$ has a value η_{ij} of a heuristic function that is an estimate of the quality of this term, with respect to its ability to improve the predictive accuracy of the rule. Parpinelli et al. [8] calculates this value based on Information Theory that involved the entropy associate for each $term_{ij}$. Liu et al. [15] shows an easier form to calculate the value η_{ij} since the pheromone τ_{ij} compensates the smaller errors of η_{ij}.

$$\eta_{ij} = \frac{|majority\_class T_{ij}|}{|T_{ij}|} \quad (2)$$

where: T_{ij} is the total number that the $term_{ij}$ occurs in the training set; $majority\_class T_{ij}$ is the total number that occurs the majority class with the $term_{ij}$ in the training set.

C. Rule Construction

The probability that $term_{ij}$ is chosen to be added to the current partial rule is:

$$P_{ij} = \frac{\tau_{ij}(t) \times \eta_{ij}}{\sum_{i}^{a} \sum_{j}^{bi} \tau_{ij}(t) \times \eta_{ij}, \forall\, i \in I} \quad (3)$$

where: η_{ij} is a problem-dependent heuristic value for $term_{ij}$; τ_{ij} is the amount of pheromone currently available (at time t) on the $term_{ij}$; a is the total number of attributes; b_i is the total number of values in the domain of attribute i; I is the set of attributes that are not yet used by the ant;

D. Quality of Rule

The quality of a rule is measured using the following equation:

$$Q = \left(\frac{TP}{TP + FN}\right) \times \left(\frac{TN}{FP + TN}\right) \quad (4)$$

where: TP (true positives) the number of cases covered by the rule that have the class predicted by the rule; FP (false positives) the number of cases covered by the rule that have a class different from the class predicted by the rule; FN (false negatives) the number of cases that are not covered by the rule but that have the class predicted by the rule; TN (true negatives) the number of cases that are not covered by the rule and that do not have the class predicted by the rule;

E. Pheromone Update

After each ant completes the construction of its rule and measured its quality, the terms that occur in the rule have its pheromone updating:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \times Q\,, \forall\, termij \in \text{current rule} \quad (5)$$

The amount of pheromone associate with each $term_{ij}$ that does not occur in the current rule has to be decreased, to simulate the pheromone evaporation in real ant colony systems. The reduction of pheromone of an unused term is performed by dividing the value of each τ_{ij} by the summation of all τ_{ij}.

Parpinelli et al. [8] evaluated the Ant-Miner with six data sets from UCI (University of California at Irvine). The algorithm was compared with others classification algorithms and achieves good results. The results showed that the Ant-Miner is a promising technique for classification rule discovery. Several researches and modifications have been proposed to improve its efficiency, including: Ant-Miner2 [15], Ant-Miner3 [16], Unordered Rule Set Ant-Miner [24], Ant-Miner+ [17], *c*Ant-Miner [18], Multiple pheromone types Ant-Miner [19].

IV. The Specializations

Most of Ant-Miner extensions aim to improve the predictive accuracy. We change de bias of the algorithm that helps to find rules to minority classes because classification algorithms are sensitive to imbalanced data sets. The specializations are called Ant-MinerCI (Class Imbalanced) and Ant-MinerCIP (Class Imbalanced Precision), the only difference between the two algorithms proposed is how the rule quality is calculated.

A. Heuristic Value

Standard Ant-Miner and other specializations calculate the heuristic value for each $term_{ij}$ according with how often the term occurs with the majority class, so that the ants tend to find the paths (terms) that lead to the majority class more interesting.

In our specializations, the heuristic value of each $term_{ij}$ is not calculated with the majority class, it is calculated with the interest class. The interest class can either be the minority class or not, defined by Equation (6).

$$\eta_{ij} = \frac{|CI \,\&\, T_{ij}|}{|T_{ij}|} \quad (6)$$

where: T_{ij} is how often that the $term_{ij}$ occurs in the training set; $CI \,\&\, T_{ij}$ is how often the interest class occurs with each $term_{ij}$.

The idea is that the ants have a preference for terms that are most relevant to find rules for the interesting class.

B. Rules Construction

Standard Ant-Miner, the rule consequent (class) is chosen after the antecedent part construction. In the Ant-Miner+ [17], the consequent is probabilistically chosen first. The consequent is selected according to the pheromone value

associated, and this pheromone indicates which class contributed the most to the rule construction.

In Ant-MinerCI, during the first iteration to find the best rule, the algorithm selects the consequent first, however not in a probabilistic way, instead it selects the interest class, thus the ants obligatorily select terms to add to the partial rule that maximize the rule quality that has a consequent the interest class. In other words, the ants converge to shorter paths (best rules) with the interest class. This interest class must be set by the user before the algorithm starts the rules construction.

The process to add terms to the current rule stops when one of the following conditions is met:

1. Any term to be added to the rule would make the rule cover a number of cases that is smaller than a user-specified threshold, called *min_cases_per_rule*;
2. All attributes have been already used by the current ant;

In Ant-MinerCI there is one more condition to stop the terms addition, which is *max_no_antec*. Such parameter limits the number of antecedent that a rule can have, since rules that have a large number of terms in the antecedent part can difficult the comprehensibility of the rules.

After the first iteration of the while loop, the best rule R_{best} among all the R_i (that have the interest class) is selected and added at the discovered rules list. The next iterations of the while loop the algorithm can find rules for every class. The formula to calculate the rule quality in Ant-MinerCI is the same that standard Ant-Miner, *Q=sensitivity×specificity*, defined before as Equation (4).

The discovered ordered rule list by the algorithm will have at least one discovered rule for the interesting class, even if the interest class is the minority class, which is the most difficult discovery.

It is important that first discovered rule should be the interest class, because the first discovered rule is related to the entire training set. Hence, the discovered rule R_n is conditioned by the R_{n-1} previous discovered rules.

C. Ant-MinerCI Parameters

The conventional parameters of Ant-Miner are: *no_of_ants* (the maximum number of candidate rules), *min_cases_per_rule* (each rule must cover a minimum number of cases from the training set and this parameter helps to avoid overfitting), *max_uncovered_cases* (the maximum number of cases uncovered from the training set), *no_rules_converg* (the number of consecutives equals rules, then the algorithm concludes that the ants have converged).

Beyond the conventional parameters, the Ant-MinerCI has two others: *interesting_class* (the user specifies which is the interest class and that will be used to discover the first rule), *max_no_antec* (the maximum number of terms that a rule can have in the antecedent part).

D. The Ant-MinerCIP Algorithm

It has the Ant-MinerCI modifications but the evaluation rules are different. The discovered rules from the minority classes have few covered cases considering all the training cases. When the function *Q=sensitivity×specificity* is used, the tendency is that the rules have a reasonable number of covered cases considering all the training cases.

For example: A data set that has class "*Positive*" and the class "*Negative*". The distribution of this data set is 96% to class *Negatives*" and 4% to class "*Positive*". Most rule induction algorithms probably will find rules only to "*Negative*" class, but the most interesting is at 4% of class "*Positive*". Even the rules discovered by Ant-MinerCI for the minority class, this rules can have a low predictive accuracy using *Q=sensitivity×specificity*. Suppose that this data set has 20.000 cases. An ant finds a rule R_i (for class "*Positive*") that cover 200 cases, and among these cases, *TP*=180, *FP*=20, *FN*=609 and *TN*=19191. Using *Q=sensitivity× specificity*, has the following value:

$$Q = \left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\right) \times \left(\frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}\right) = \left(\frac{180}{180+609}\right)\left(\frac{19191}{20+19191}\right) = 0{,}2278 \quad (7)$$

The value from Equation (8) is the quality value to R_i, the probability for this rule to be chosen as the best rule by algorithm is low, even if this rule has a predictive accuracy of 90%. This happens because the algorithm selects the best rule when the *Q* value is high, in other words, the algorithm select the rule that has high coverage (recall). In this case, that there is a rare class, the discovered rules for this class can have a low predictive accuracy, which harms all the predictive accuracy.

The Ant-MinerCIP algorithm uses the precision formula directly, Equation (8), which evaluates rule quality, since the precision aims to analyze only the covered cases by the rule. The lower the *FP* number covered by the rule, better the precision.

$$\text{Precision} = \left(\frac{TP}{TP+FP}\right) \quad (8)$$

Thus, the precision metric does not harm the discovery of rules to the minority class, as well as for other classes. A situation that can be happen is that the rule may be too adjusted, which characterizes overffiting. To avoid this problem, the *min_cas_per_rule* parameter must be used, which matches the minimum value of cases that a rule can cover. Thus adjusting this parameter may avoid overffiting.

V. Results and Analysis

A. Specializations Algorithms Evaluation

Two algorithms are used to compare with the specializations: C4.5 and GUI-Ant-Miner. The C4.5 [21] is a rule induction algorithm that uses de decision tree representation. The GUI-Ant-Miner [20] is the standard

version of Ant-Miner with a better interface for user interaction.

The algorithms were evaluated using public-domain data sets from University of California at Irvine repository. Table I describes the data sets, number of cases, the number of attributes, number of classes and the proportion of the minority class (positive class). From the seven data sets, only one has continuous attributes, but its values were categorized (preprocessing). The data sets Letter-a and Letter-vowel were constructed based on the original data set Letter [6].

For a fair comparison, no adjustments were made in the algorithms parameters to optimize its performance, because each data set can have its optimum parameters set. Table II shows the parameters values, in consonance with [8].

Table I. Data sets used to evaluate the algorithms

Data sets	No. of cases	No. of attributes	No. of classes	Positive (%)
Ljubljana breast cancer	286	9	2	29.7
Wisconsin breast cancer	683	9	2	34.4
tic-tac-toe	958	9	2	34.6
dermatology	366	34	6	5.5
Votes	435	16	2	38.6
Letter-a	20.000	17	2	4
Letter-vowel	20.000	17	2	19.4

Table II. Parameters value

Parameters	Values
no_of_ants	1000
min_cases_per_rule	10
max_uncovered_cases	15
no_rules_converg	10
max_no_antec	4
class_of_interest	minority class

We used two test methods, hold out and cross-validation. Table III shows the results using ROC analysis with AUC and hold out method, because [25] concludes that hold out are sufficient to achieve good model selection for AUC.

Table III. AUC Ant-MinerCI x Ant-MinerCIP x GUI-Ant-Miner x C4.5

	Area Under Curve (AUC)			
Data Sets	Ant-MinerCI	Ant-MinerCIP	GUI-Ant-Miner	C4.5
Ljubljana breast cancer	0.658	0.639	0.612	0.639
Wisconsin breast cancer	0.951	0.99	0.951	0.983
tic-tac-toe	0.914	0.892	0.914	0.984
Dermatology	0.921	0.956	0.956	0.998
Vote	0.986	0.986	0.992	0.986
Letter-a	0.95	0.965	0.96	0.965
Letter-vowel	0.872	0.944	0.902	0,96
Average ± SD	**0.893 ± 0.102**	**0.91 ± 0.124**	**0.898 ± 0.13**	**0.931 ± 1.129**

We use a paired t-test to determine if the performances are significantly different from each other. Each algorithm was compared with the top performance with one-tailed. The means are not significantly different at 90% confidence level. Nevertheless, the great difference was between C4.5 and Ant-MinerCI.

Table IV shows the mean of number of rules that each set (model) obtained with cross-validation with 10 partitions.

Table IV. No. of rules Ant-MinerCI x Ant-MinerCIP x GUI-Ant-Miner x C4.5

	Number of Rules			
Data Sets	Ant-MinerCI (μ)	Ant-MinerCIP (μ)	GUI-Ant-Miner (μ)	C4.5 (μ)
Ljubljana breast cancer	3.1	9	4.4	4
Wisconsin breast cancer	5.8	7	7.2	19
tic-tac-toe	6	38.6	6.5	95
dermatology	5	18	6.5	30
Vote	5.1	10.5	4.8	6
Letter-a	4.2	12.8	7.2	37
Letter-vowel	5	67	14	185

As shown in Table III, the average predictive AUC values of the four algorithms are very similar (no statistical difference) and they are all lower with significant difference than C4.5. AUC replace accuracy in measuring and comparing classifiers as AUC is better measure in general, mainly with imbalance.

The number of discovery rules (Table IV) by the C4.5 in these data sets was higher than the other algorithms. It is important to note that the Ant-MinerCI and GUI-Ant-Miner sacrifice the predictive accuracy to build a model with fewer rules, thus contributing with comprehensibility.

The specializations have a parameter that limits the number of terms antecedent, finding antecedent with at most four terms, thus improve de comprehensibility of the rules. Greater the number of attributes of a data set, the higher might be the number of terms.

Lastly, analyzing the Ant-MinerCI, Ant-MinerCIP and GUI-Ant-Miner results, the fact of learning the minority class first did not decrease the performance (AUC) of the algorithms.

B. Ant-MinerCI x Ant-MinerCIP

The parameters in this section were the same as the ones used in previous section, except for the *min_cases_per_rule* which now is 50. This parameter is used to avoid overfitting.

The goal of Ant-MinerCIP is to discover rules to rare classes with a better predictive accuracy. Thus, the following results of this section are focused on the first rule discovered by each algorithm. For this study were used the data sets that are more imbalanced from the previous section (Letter-a and Letter-vowel). Table V shows the data sets distribution.

Table V. Data sets distribution

Data set	Classes	Distribution (%)
Letter-a	(a, others)	(4, 96)
Letter-vowel	(vowel, others)	(19.4, 80.6)

Table VI and Table VII show the results of predictive accuracy (precision), the coverage (recall) and the number of terms antecedent of the first rule discovered by the algorithms with the interesting class (minority class).

Table VI. Precision (%), Recall (%) e No. of antecedent of the first rule for class "A" from the Letter-a data set (Ant-MinerCI x Ant-MinerCIP)

Letter-a	Class	Precision (μ% ± sd%)	Recall (μ% ± sd%)	No. of terms antecedent
Ant-MinerCI	a	66.07 ± 17.56	39.58 ± 15.31	1.5 ± 0.5
Ant-MinerCIP	a	98.74 ± 0.02	10.64 ± 2.78	3.2 ± 1

Table VII. Precision (%), Recall (%) e No. of antecedent of the first rule for class "vowel" from the Letter-vowel data set (Ant-MinerCI x Ant-MinerCIP)

Letter-vowel	Class	Precision (μ% ± sd%)	Recall (μ% ± sd%)	No. of terms antecedent
Ant-MinerCI	vowel	28.43 ± 1.61	31.17 ± 2.2	1.28 ± 0.45
Ant-MinerCIP	vowel	100 ± 0	3.58 ± 0.8	3 ± 1.2

Ant-MinerCI uses the function $Q=sensitivity \times specificity$ to evaluate the rules, thus the algorithm finds more general rules, this means that the rule covers a large number of cases, consequently the rule has a lower predictive accuracy because it has a many erroneous cases, which are the *FP* (false positive) to the rule. However, the Ant-MinerCIP, which uses the precision as a function to evaluate the rules, finds more specific rules, but with a better predictive accuracy. Even if the rules are more specific, they have at least 50 cases covered, because this is the threshold chosen.

VI. CONCLUSIONS

The results showed that all algorithms have similar predictive AUC. Comparing with the standard Ant-Miner the specializations have the advantage to discovery a rule to minority class first with better predictive accuracy. Other advantage is the simplicity of the rules, because the algorithm limits the number of terms antecedent, which improve the comprehensibility. The way that the Ant-MinerCIP calculates the rule quality improves the predictive accuracy to the minority class. The AUC analysis helped to evaluate the results more fairly, which showed that the algorithm specializations are competitive with those in the literature, mainly when applied imbalanced data sets.

REFERENCES

[1] Chawla, N.V., Japkowicz, N., Kotcz, A.: (Editors) Editorial: Special Issue on Learning from Imbalanced Data Sets. In: ACM SIGKDD Explorations vol. 6, pp. 1-6, 2004.

[2] Japkowicz, N.: Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech Report WS-00-05. AAAI, 2000.

[3] Chawla, N.V., Japkowicz, N., Kotcz, A.: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets, 2003.

[4] Weiss, G.: Mining with Rarity: A Unifying Framework. In: ACM SIGKDD Explorations, vol. 6, pp. 7-19, 2004.

[5] JAPKOAWICZ, N., JO, T.: Class Imbalances versus Small Disjuncts. In: ACM SIGKDD Explorations, vol. 6, pp. 40-49, 2004.

[6] Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. In: ACM SIGKDD Explorations, vol. 6, pp. 20-29, 2004.

[7] Liu, X., Wu, J., Zhou, Z.: Exploratory Under-Sampling for Class-Imbalance Learning. In: IEEE Transactions on Systems, Man and Cybernetics, Part B, pp. 1-14, 2008.

[8] Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data Mining With an Ant Colony Optimization Algorithm. In: IEEE Transactions on Evolutionary Computation, vol. 6, pp.321–332, 2002.

[9] Kubat, M., Matwin, S.: "Addressing the curse of imbalanced training sets: One-sided selection." In Proceedings of the 14th International Conference on Machine Learning, pp. 179-186, 1997.

[10] Chawla, N. V., Hall, L. O., Bowyer, K. W., Kegelmeyer, W. P.: "SMOTE: Synthetic Minority Oversampling Technique". In: Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[11] Prati, R. C., Batista, G. E. A. P. A., Monard, M. C.: "Curvas ROC para avaliação de classificadores" In: IEEE Latin America Transactions, vol.6, no. 2, pp. 215-222, 2008

[12] Fawcett, T.: "ROC Graphs – Notes and Practical Considerations." Kluwer Academic Publishers, Netherlands, 2004.

[13] He, H., Garcia, E.: Learning from Imbalanced Data. In: IEEE Transactions on Knowledge and Data Engineering, vol. 21, pp. 1263-1284, 2009.

[14] Dorigo, M., Di Caro, G.: The Ant Colony Optimization meta-heuristic. In: New Ideas in Optimization, pp.11-32, 1999.

[15] Liu, B., Abbass, H.A., Mckay, B.: Density based Heuristic for Rule Discovery with Ant-Miner. In: The Sixth Australia-Japan Joint Workshop on Intelligent and Evolutionary System, pp. 180-184, 2002.

[16] Liu, B., Abbass, H.A., Mckay, B: Classification Rule Discovery with Ant Colony Optimization. In: In Proc. IEEE/WIC International Conference on Intelligence Agent Technology, pp. 83–88, 2003.

[17] Martens, D., Backer, M. D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. In: IEEE Transactions on Evolutionary Computation. vol. 11, pp. 651–665, 2007.

[18] Otero, F., Freitas, A., Johnson, C.G.: Cant-Miner: an Ant Colony Classification algorithm to cope with continuous attributes. In: Ant Colony Optimization and Swarm Intelligence. Lecture Notes in Computer Science, vol. 5217, pp. 48-59, 2008.

[19] Salama, K.M., Abdelbar, A.M., Freitas, A.A.: Multiple Pheromone Types and other Extensions to the Ant-Miner Classification Rule Discovery algorithm. In: Swarm Intelligence, vol. 6234, pp 1-34, Lecture Notes in Computer Science, 2010.

[20] Meyer, F.; Parpinelli, R.S.: Gui Ant-Miner: uma versão atualizada do minerador de dados baseado em colônias de formigas. In: I Congresso Sul Catarinense de Computação: UNESC – Criciúma, 2005.

[21] Quinlan, J.R.: C4.5: Programs for Machine Learning. San Francisco: Morgan Kaufmann, 1993.

[22] Chawla, N.V: Data Mining for Imbalanced Datasets: An Overview. In: Data Mining and Knowledge Discovery Handbook, Sringer US, 2nd ed., pp 875-886, 2010.

[23] Pappa, G. L.; Freitas, A. A.; Kaestner, C. A. A.: Attribute Selection with a Multiobjective Genetic Algorithm, In: Proc. of 16th Brazilian Symposium on Artificial Intelligence, pp 280-290, 2002.

[24] Smaldon, J; Freitas, A.A: A New Version of the Ant-Miner Algorithm Discovering Unordered Rules Sets, In: GECCO, pp. 43-50, 2006.

[25] Caruana, R., J; Mizil, A. N.: An Empirical Evaluation of Supervised Learning for ROC Area, In: Proceedings of the 1st Workshop on ROC Analysis in AI (ROCAI), pp. 01-08, 2004.