Classification Using Unstructured Rules and Ant Colony Optimization

Negar Zakeri Nejad, Amir H. Bakhtiary, and Morteza Analoui

***Abstract*—In this paper a new method based on the Ant-Miner algorithm is proposed to discover sets of unstructured classification rules. This method, called the Tree-Miner, creates a directed graph made up of nodes representing operators and operands. Each ant in a colony of artificial ants traverses this graph to find routes that represent the best unstructured rule antecedents. These antecedents are used to classify the given data and are also interpreted as knowledge hidden in the training data. The performance of the Tree-Miner algorithm was evaluated against that of the Ant-Miner according to the accuracy and the simplicity of the constructed rules. The results showed that our method has an acceptable predictive accuracy while discovering rules that are simpler and more comprehensive.**

***Index Terms*—Ant Colony Optimization, rule based classifiers, unstructured rules.**

I. INTRODUCTION

After sufficient training, a classifier's task is to try to correctly predict the classes of a number of given instances of data. By examining the way that a particular system has been trained to respond, they can also be used to enhance our knowledge of a particular phenomenon. One such system, that directly fits this bill is the rule based classifier.

During the training of a rule based classifier a set of rules is assigned to it. These rules not only decide the class of each instance of data, but also present the knowledge that is inherent in the data set that was used to train the classifier.

Recently in [1] the Ant-Miner Algorithm was proposed which used an ant colony optimization algorithm to discover a set of ordered rules. From this a number of research activities originated that focused on optimizing the Ant-Miner algorithms. In [2] a modified version of Ant-Miner was proposed where the computation of the heuristic value was based on a simple density estimation of the heuristic. [3] Introduced another ant-based algorithm which uses a different pheromone updating strategy and a state transition rule. A new heuristic function, simple state transition and a self-adaptive parameter are used in [4]. Further works include the discovery of unordered sets [5], pruning using a hybrid rule pruning strategy [6], an extension to address the multi label problem [7] and the use of punishment for reducing the number of rules and conditions [8].

All the previous works have focused on deriving rules that have a fixed structure where each rule chosen by an ant is a set of <attribute, operator, value> triples that are bound together using AND operators to form the antecedent of a single rule.

In this work an algorithm is devised which allows an ant to construct rules that do not have a fixed structure. This is done by having each ant traverse a predefined graph. The path that each ant chooses is in the form of a tree that represents the antecedent of a rule.

This paper is organized as follows: In section II, the Ant Colony Optimization (ACO) is considered. Section III gives an overview of the original Ant-Miner algorithm. Section IV describes the proposed algorithm. In section V we present the experimental results and in section VI we conclude the paper and suggest future work.

II. PROCEDURE ANT COLONY OPTIMIZATION

Swarm Intelligence is a field that deals with the emergent behavior in systems where independent simple agents work together to achieve complex behaviors. The Ant Colony Optimization (ACO) is one such system where the actions of artificial ants resembling the behavior of real ants are used to sub-optimally solve computationally expensive problems. This algorithm was first devised by Dorigo et al. [9] as a way to solve the TSP problem. It was later extended to be applied to other optimization problems.

A. Cooperative ants

Although blind, ants are able to find the shortest path from their nest to sources of food. This is possible through the deposition and sensing of a substance named pheromone. When an ant is randomly searching for a route it leaves a trail of pheromone on the path that it chooses. Other ants that pick up the pheromone decide to follow the path or not. This form of communication allows a very intelligent behavior to be displayed from a group whose individuals are very simple.

III. ANT-MINER

Algorithm I presents the pseudo code for the Ant-Miner Algorithm as is described in [3]. This Algorithm has two nested loops. The inner loop executes in three main stages. The first stage is to construct a rule according to the following template given in (1):

Manuscript received January 15, 2008.

Negar Zakeri Nejad is a graduate student at the Computer Engineering dept. of the Iran University of Science and Technology, Iran (e-mail: n.zakerinejad@gmail.com).

Amir H. Bakhtiary is a graduate student at ECE dept. of the University of Tehran, Iran (e-mail: amirishere@yahoo.com).

Morteza Analoui is a professor at the Computer Engineering dept. of the Iran University of Science and Technology, Iran (e-mail: analoui@iust.ac.ir).

ALGORITHM I
THE ORIGINAL ANT-MINER ALGORITHM.

```
Training set = all training cases;
WHILE (No. of cases in the Training set > Max_uncovered_ cases)
   i = 0;
   REPEAT
      i = i+1;
      $Ant_i$ incrementally constructs a classification rule;
      Prune the constructed rule;
      Update the pheromone of the trail followed by $Ant_i$;
   UNTIL (i ≥ No_of_ants) or ($Ant_i$ constructed the same rule as the previous No_Rules_Converg-1 Ants)
   Select the best rule among all constructed rules;
   Remove the cases correctly covered by the selected rule from the training set;
END WHILE
```

$$If\,(term1\ AND\ term2\ AND\ldots) Then < class > \qquad (1)$$

where the antecedent is made up of a number of terms joint together with the AND operator and the consequent is the class that is predicted by the rule. Each term is a <attribute, operator, value> triple such as <Gender = female> [1]. In the original Ant-Miner uses only the "=" operator and can only manage categorical attributes.

In this stage the choosing of each term is done with a probability proportional to the amount of pheromone associated with the term at the time of the choosing and a heuristic function which is the information gain criteria for that term [10].

The next step is to prune the constructed rule. This is done by repetitively adjusting the antecedent of the rule, evaluating the class that the rule represents best and evaluating the quality of the rule according to the following equation:

$$Q = \frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP} \qquad (2)$$

Where the first part denotes sensitivity and the second part denotes specificity [5] and:

TP (true positives) is the number of cases covered by the rule that have the class predicted by the rule.

FP (false positives) is the number of cases covered by the rule that have a class different from the class predicted by the rule.

FN (false negatives) is the number of cases that are not covered by the rule but that have the class predicted by the rule.

TN (true negatives) is the number of cases that are not covered by the rule and that do not have the class predicted by the rule.

In the final stage, the pheromone of the terms included in the antecedent of the rule is updated according to the quality of the rule.

The loop stops when either it has executed a certain number of times or when the convergence criterion is met.

In the outer loop, the best rule after being found by the inner loop is added to the list of the discovered rules. Then the cases that are covered by the discovered rule are removed from the training set. This loop continues until the number of cases in the training set is less than a maximum number of uncovered cases set by the user.

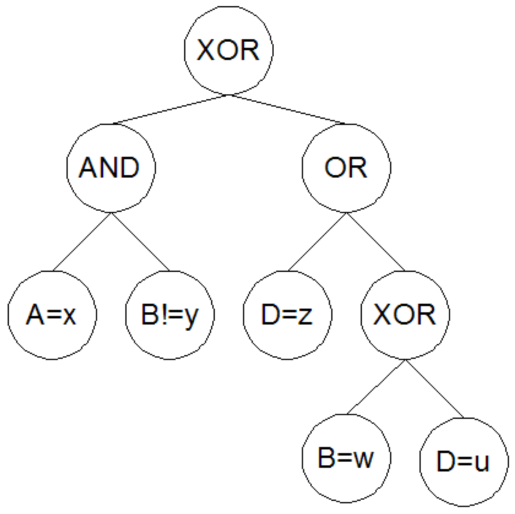

Fig. 1. Each rule antecedent can be represented using a tree.

IV. PROPOSED ALGORITHM

The main difference in this work is the construction of rules that have less structural constraints allowing us to model more complex relationships with rules that are more concise.

To allow more complete rules the rules are allowed to have OR and XOR operators as well as AND operators. Also the concept of operator precedence has been added to allow the interpretation of these rules. The NOT operators have only been introduced on the terms. These rules are represented by trees that are searched for by ants, hence the name Tree-Miner.

Another difference of the Tree-Miner with the Ant-Miner is that no pruning has been used for the rules. Thus the consequent of each rule is determined right after the construction of the antecedent.

A. *Unstructured rules*

The representation of an unstructured rule is easily managed using trees an example of which is shown in Fig. 1. In these trees terms appear in the leaves of the tree while the internal nodes are the binary operators that act on the leaves of the tree. Notice that in the trees that we construct only binary operators are used and that the NOT operator has not been directly included in the tree as a node, because by recursively applying De Morgan's rule to an arbitrary antecedent, we can form an equivalent antecedent in which the NOT operators act directly on the <value, operator, attribute> triples.

In the proposed algorithm each search that is performed by an ant leads to the construction of one tree that represents one unstructured rule antecedent.

B. *Rule Graph*

An example of this graph is shown in Fig. 2. This graph contains a start node, a set of binary nodes and a set of term nodes. The start node is the node that the search for the antecedent begins from; the binary nodes represent binary operators that can be used in the forming of the antecedent. The each term node represents one <attribute, operator, value> triple.

The start node is connected to all of the binary nodes and all pairs of binary nodes are connected together.

We use two edges called left and right, for connecting two binary nodes one for each of its operands. In the same way, each binary node is connected to each term nodes by two edges.

C. Traversal of the tree and rule construction

As we have already mentioned that for the construction of a rule, each ant needs to build a tree by traversing the rule graph. To do this first each ant begins from the start node and chooses one of the outgoing edges according to the roulette wheel schema where the probability of choosing edge *x* is giving as follows:

$$P(x) = \frac{Ph(x,t)}{\sum_{i \in S} Ph(i,t)} \tag{3}$$

In (3) *S* is the set of outgoing edges from start node and *Ph(x,t)* is the amount of existing pheromone on edge *x* at time *t*.

Each time a binary node is reached; the ant will choose two of the outgoing edges, one from the left edges and one from the right edges. From there each of the chosen edges is traversed.

An outgoing edge from a binary node x is chosen with probability *P(x)* as in (4).

$$P(x) = \frac{Ph(x,t) \times W(x,d)}{\sum_{i \in B} Ph(i,t) \times W(i,d)} \tag{4}$$

In (4), *B* is the set of edges from which the ant is choosing from. *d* is the depth of the binary node in the currently constructed tree. *W(x,d)* is a weight that serves to limit the depth of the tree and is defined in (5).

$$W(x,d) = \begin{cases} 1 & type(x) = term \\ 1 & (type(x) = operator) \wedge (d < Max\_depth) \\ 0 & otherwise \end{cases} \tag{5}$$

In (5), *type(x)* denotes the type of the node at the end of the edge *x*. *Max_depth* is a user given parameter that limits the depth of the constructed tree since choosing edges that lead to binary nodes is only possible when the depth of the current node is less than *Max_depth*. Otherwise the ant is force to choose from the edges that lead to the term nodes. This poses a limit on the height of the tree since nodes at depth *Max_depth* are only chosen from the term nodes in the graph and the term nodes can not have children in the tree.

D. Pheromone updating

At each iteration of the while loop of algorithm, the amount of pheromone on all of the edges, is equal to one.

For updating the pheromone for each edge that is included in a certain rule, we use equation (6):

$$Ph(x,t+1) = Ph(x,t) \times (1 + Q \times Conv\_rate) \tag{6}$$

In the above equation *Q* is the quality of the rule and is

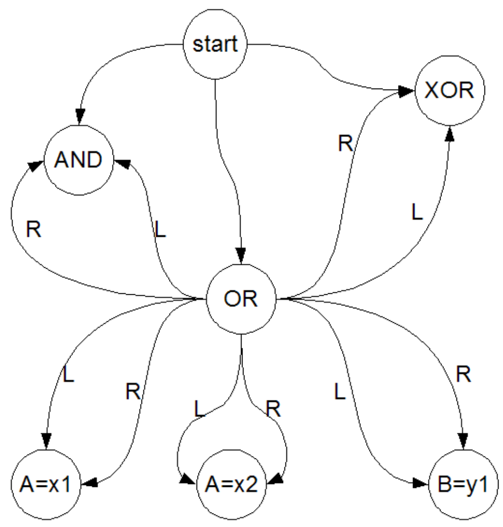

Fig. 2. An ant traverses through a graph similar to this to construct the antecedent of a rule. Only one of the binary nodes has all its connections shown.

calculated as is shown in (2). *Conv_rate* is a parameter that controls the speed at which the algorithm converges. This was set to stop immature convergence to answers that are too far from optimal.

Keeping in mind that in the original Ant-Miner pheromones are associated with terms where as in the Tree-Miner they are associated with links in the graph; equation (6) is very similar to the formula used in the original Ant-Miner to update the pheromones except for the addition of C*onv_rate*.

By setting C*onv_rate* to a small value we are reducing the effect that each ant has on the pheromone trails. This delays the convergence of the constructed rules to a later time thus allowing more exploration to take place before the algorithm stops.

V. EXPERIMENTAL RESULTS

To evaluate the performance of the Tree-Miner algorithm, it was compared to the original Ant-Miner algorithm.

A. Datasets

The performance of the algorithm was evaluated using four public-domain data sets obtained from the UCI repository [11]. The datasets were chosen so as to have a large number of instances each, leading to results that were stable during testing. The main characteristics of these datasets are shown in Table I.

Some of the datasets used contained numerical attributes which the Ant-Miner and the Tree-Miner can not manage. To overcome this the discretization method available in Weka [12] was used in conjunction with its “use equal Frequency” option to change these variables into nominal attributes.

B. Test setup

For testing the Ant-Miner all of its parameters were kept to a set which were tested in the setups used in the original paper [1].

TABLE I. CHARACTERISTICS OF THE DATASETS

Test set	**Characteristics**		
	Number of Instances	***Number of attributes***	***Number of classes***
Abalone	4177	9	25
House Price	506	14	3
Segmentation	2100	19	7
Wisconsin Cancer	699	10	2

TABLE II. PARAMETERS USED FOR CONDUCTING THE TESTS

Parameters	**Classifier Algorithm**	
	Ant-Miner	***Tree-Miner***
No_of_ants	7000	7000
Min_cases_per_rule	5	-
Max_uncovered_cases	10	10
No_rules_converge	10	10
Conv_rate	-	0.01
Max_depth	-	5

For the Tree-Miner all the parameters that it shared in common with the Ant-Miner where kept the same. To keep the test fair, the newly introduced parameters, namely *Conv_rate* and *Max_depth*, were first tuned using a separate dataset and after that the chosen value were used for conducting the experiments. Table II summarizes the parameters used for the tests.

The ten-fold cross-validation process was used to conduct the tests. In this process each data set is partitioned in to ten subsets, and the classification algorithm being tested is run ten times, each time with a different subset used as the test set and the rest used as the training set. Finally the ten results are averaged [13].

C. *Test results*

The results of the experiments are summarized in Tables III, IV and V. From Table III, which presents the average classification rate of both the Ant-Miner and the Tree-Miner, we can see that the Ant-Miner and the Tree-Miner have very similar prediction accuracy.

Table IV and V present the average number of rules found and the average number of terms used in each of the classifiers respectively. We can see that in all of the datasets the Tree-Miner has been able to decrease the number of rules that are needed for classifying the data. This is very observable in the Wisconsin Cancer case where the number of rules has decreased significantly.

Turning our attention to Table V we can see that in the Segmentation test the number of terms used is the same in both algorithms while we have a reduction in the number of terms used in two of the datasets and an increase in one. Note that since the number of rules has decreased in all of the cases, not having a change in the average number of terms suggests an aggregation of the rules, while the reductions in the number of terms suggest the discovery of newer and more effective relationships between the terms.

TABLE III. CLASSIFICATION RATE

Dataset	**Classifier Algorithm**	
	Ant-Miner	***Tree-Miner***
Abalone	22.62%+/-0.63%	22.6%+/-0.7%
House Price	64.49%+/-1.49%	66.21%+/-1.8%
Segmentation	77.44%+/-1.71%	75.08%+/-1.05%
Wisconsin Cancer	92.71%+/-1.14%	93.57%+/-1.17%

TABLE IV. AVERAGE NUMBER OF RULES FOUND

Dataset	**Classifier Algorithm**	
	Ant-Miner	***Tree-Miner***
Abalone	32.6+/-1.14	25.4+/-1.02
House Price	11.5+/-0.45	8.0+/-0.76
Segmentation	26.8+/-0.51	17.3+/-0.82
Wisconsin Cancer	11.4+/-0.16	4.7+/-0.26

TABLE V. AVERAGE NUMBER OF TERMS USED FOR EACH CLASSIFIER

Dataset	**Classifier Algorithm**	
	Ant-Miner	***Tree-Miner***
Abalone	62.9+/-3.19	52.1+/-2.33
House Price	14.5+/-0.56	17.1+/-1.68
Segmentation	35+/-0.83	34.9+/-1.65
Wisconsin Cancer	12.6+/-0.31	9.9+/-0.69

VI. CONCLUSION

We have compared the performance of our method and Ant-Miner according to the predictive accuracy and simplicity of the discovered classification rules.

The results of experiments have shown that the rules discovered by the proposed algorithm have an acceptable predictive accuracy and that they are able to find more complex relationships and give a simple representation for them.

In this work no rule pruning was used before evaluating the quality of each work. Adding this can improve the simplicity of the rules leading to simpler rule sets.

Further research should also be conducted into the effects of discovering a list of unordered set of unstructured rules that may increase the performance of classification.

Additionally, the effect of pheromones on the types and quality of the rules discovered needs to be assessed.

REFERENCES

[1] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, “Data mining with an ant colony optimization algorithm,” IEEE Trans. on Evolutionary Computation, 6(4), Aug 2002, pp. 321-332.
[2] B. Liu, H. A. Abbass, and B. Mckay, “Density-Based heuristic for rule discovery with Ant-Miner,” The 6th Australasia-Japan Joint Workshop on Intelligent and Evolutionary Systems (AJWIS 2002), Canberra, Australia, 2002.
[3] B. Liu, H. A. Abbass, and B. Mckay, “Classification rule discovery with ant colony optimization,” Proceeding of the IEEE/WIC International Conference on Intelligent Agent Technology, Beijing, China (2003), pp. 83-88.
[4] Z. Q. Wang and B. Q. Feng, “Classification rule mining with an improved ant colony algorithm,” G.I. Webb and X. Yu (Eds.): AI 2004, LNAI 3339, 2004, pp. 357-367, Springer.

[5] J. Smaldon and A. A. Freitas, "A new version of the Ant-Miner algorithm discovering unordered rule sets," Proc. Genetic and Evolutionary Computation Conf. (GECCO-2006), pp. 43-50.

[6] A. Chan and A. A. Freitas, "A new classification-rule pruning procedure for an ant colony algorithm," Springer, 2006.

[7] A. Chan and A. A. Freitas, "A new ant colony algorithm for multi-label classification with applications in bioinformatics," Proc. Genetic and Evolutionary Computation Conf. (GECCO-2006), pp. 27-34.

[8] Junzhong Ji, Ning Zhang, Chunnian Liu, and Ning Zhong, "An ant colony optimization algorithm for learning classification rules," Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence.

[9] M. Dorigo, A. Colorni, and V. Maniezzo, "An investigation of some properties of an ant algorithm," Proceedings of the parallel problem solving from nature conference, Elsevier publishing, 1992.

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

[11] UCI Machine Learning Repository. (University of California at Irvine). Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[12] Publisher: Machine Learning Group, University of Waikato, Hamilton, NZ. Available: http://www.cs.waikato.ac.nz/ml/weka

[13] S. M. Weiss and C. A. Kulikowski, Computer Systems that learn. San Francisco, CA: Morgan Kaufmann, 1991.