Journal of Circuits, Systems, and Computers
Vol. 19, No. 1 (2010) 297–306
© World Scientific Publishing Company
DOI: 10.1142/S0218126610006244

COMPATIBILITY AS A HEURISTIC FOR CONSTRUCTION OF RULES BY ARTIFICIAL ANTS

WASEEM SHAHZAD* and ABDUL RAUF BAIG†

Department of Computer Science,
National University of Computer & Emerging Sciences,
Islamabad, Pakistan
**Waseem.Shahzad@nu.edu.pk*
†*Rauf.Baig@nu.edu.pk*

Ant Miner algorithms are used for the classification task of data mining. This paper proposes a new version of Ant Miner algorithm called CAnt Miner that considers compatibility between the attributes, when constructing a rule. In addition to a new heuristic function based on compatibility among attributes, we also introduce a strategy to dynamically stop adding terms in the rule antecedent part instead of using a user specified threshold that specifies when to stop adding terms. We compare the performance of proposed approach with other Ant Miner versions and C4.5 in nine public domain data set. The experiment results show that the proposed approach achieves higher accuracy rate then other approaches.

Keywords: Ant colony optimization; data mining; classification rules.

1. Introduction

The goal of data mining is to extract useful knowledge from data. There are different data mining tasks, such as clustering, classification and association rules mining. In this paper our focus is on the classification model. The goal of the classification process is to find a set of rules from training data and then assign a class label to each test case according to its characteristics. Many classification algorithms are being used such as decision tree, neural networks, fuzzy logic, statistical and rough sets, k-nearest neighbor classifiers and support vector machines (SVMs). The neural networks and SVMs techniques are algorithmically more powerful. They are based on complex mathematical functions. But they are incomprehensible and opaque to humans. In many real time applications both comprehensibility and accuracy are required.

In this paper we propose an ant colony optimization algorithm for classification rule discovery. The proposed Ant Miner algorithm has both properties of accuracy and comprehensibility. The overall approach of the Ant Miner algorithm is sequential covering. It starts with a full training set, creates a best rule that covers a subset of the training data, adds the best rule in the discovered rules list and removes the training samples that are correctly classified by the best rule. This process continues until a few training samples are left, fewer than the maximum uncovered cases specified by the user.

The remainder of this paper is organized as follows. In Sec. 2, we present the basic idea of ant colony optimization systems. Section 3 gives an overview of the existing Ant Miner algorithms. Section 4 describes the proposed approach. In Sec. 5 the experiment results on various publicly available data sets are presented and discussed. Finally Sec. 6 concludes the proposed technique and gives the future direction.

2. Ant Colony Optimization

Ant colony optimization (ACO) is a branch of evolutionary computation inspired by the behavior of natural ants. It includes mechanisms of cooperation and adaptation.[1] Ants communicate with each other by means of chemical substance called pheromone. Each ant perceives pheromone concentrations in its local environment. If many ants follow a given trail, it becomes more attracted and is followed by other ants. In ACO each ant incrementally constructs a candidate solution. The candidate solution is associated with a path in graph representing the search space.

When constructing a solution, an ant typically has to choose which solution component should be added to the current partial solution among several alternative solution components. It chooses a solution component probabilistically. The probability is based on two factors, one is the amount of pheromone associated with the path and the second is the value of a problem dependent heuristic function. The ants that find better solutions are allowed to lay pheromone on their paths (i.e., solution components). These solution components become more attractive for the ants of next iteration and after a while the whole process converges to a solution which is good, if not optimal.

3. Related Work

The first Ant Miner algorithm was proposed by Parpinelli *et al.*, in 2002.[7] In Ant Miner each ant constructs a candidate classification rule. At first an ant starts with an empty rule and incrementally constructs a classification rule by adding one term at a time to current partial rule. The term to be added in the current partial rule is based on the probability proportional to the product of a heuristic function and pheromone value. Heuristic function is calculated by using the information gain. In the second step, after each ant has constructed its antecedent part, the consequent of the rule is assigned by majority vote. Then the constructed rule of each ant is pruned to remove the irrelevant terms to improve the accuracy of rule. The next step is to update the pheromone value of the trail followed by the ant proportional to the quality of rule.

The extensions of basic Ant Miner algorithm were proposed by Liu *et al.* in Ant Miner2[3] and Ant Miner3[4,5]. In Ant Miner2 the authors proposed an easily computable density estimation heuristic function instead of the entropy measure. They proved that a simpler heuristic function does the job as well as the complex one. Ant Miner2 is computationally less expensive than the original Ant Miner. In Ant Miner3 the authors proposed a different pheromone update method. They update and evaporate the pheromone of only those terms that occur in the rule and do not evaporate the pheromones of unused terms.

David Martens *et al.* proposed a Max-Min ant system based Ant Miner that differs from the previously proposed Ant Miner versions in three aspects: after each iteration only the best ant is allowed to update the pheromone, the range of the pheromone trail is limited within an interval, and they fix the class in advance before constructing the rule.[6] They changed the heuristic function calculation method of a $term_{ij}$. They also changed the rule quality measure that is based on confidence and coverage. Other works on Ant-Miner include an algorithm for discovering unordered rule sets has been presented.[9]

4. Proposed Technique

In this section, we discuss in detail our Ant Miner algorithm. The section is divided into 7 sections that describe the proposed approach.

4.1. *Description of the new ant miner*

In our proposed algorithm, given below, each ant incrementally constructs a candidate solution. It is a sequential covering algorithm that learns one rule at a time and then removes the training samples correctly classified by the rule. At the start the discovered rule list is empty and training set contains all the training samples. Each iteration of the outer while loop, executes a number of executions of inner repeat-until loop, which constructs as many rules as the number of ants. The best rule among all the constructed rules is selected and added in the final discovered rule list. This process is continued until the number of uncovered training samples becomes less than or equal to a user specified threshold.

Initially an ant starts with an empty rule. It first probabilistically chooses the class label of the rule from a set of class labels belonging to the remaining cases not covered yet. This class label becomes fixed for all the ants in that iteration of the inner repeat-until loop. The ant then constructs its antecedent part by adding one term at a time in the current partial rule. The choice of adding a term in the current partial rule is based on the pheromone value and heuristic value associated with a term.

In our proposed approach we consider compatibility among the attributes. Therefore in our heuristic function that will be discussed in detail in next sections, the terms which are already chosen by an ant in current partial rule also play an important role. An ant continues adding one term at a time in the current partial rule until by adding a term in the rule, the cases covered by the rule have the same class label or the rule does not cover any case or all the attributes already have been used.

The stopping criteria of adding terms in the rule antecedent of our approach is different then original Ant Miner, in which the user needs to specify a threshold called minimum cases per rule, in which a rule must cover at least a minimum cases specified by the user. We dynamically stop adding terms in the current partial rule. The experiments results show that this strategy improves the accuracy of the classifier.

```
Training Set = all training cases;
Discovered_Rulelist ={ };
While (Training Set > Max_uncovered_cases)
    t = 1;
    j = 1;
    Initialize pheromones of all trails;
    Initialize the matrix of heuristic values;
    Select the class label (Rule Consequent);
    REPEAT
        $Ant_t$ incrementally constructs the classification rule
        for the selected class;
        Evaporate the pheromones values of the terms used in the rule;
        Update the pheromones values of the terms by increasing the
        pheromone in the trail followed by current ant;
        IF($R_t$ is equal to $R_{t-1}$)
            j = j + 1;
        ELSE
            j = 1;
        END IF
            t = t + 1;
    UNTIL (t >= No_of_ants) OR (j >= No_rules_converg)
    Choose the best rule among all the rules constructed by all ants;
    Prune the best rule;
    Add best rule in the Discovered_Rulelist;
    Remove the training samples correctly classified by the best rule;
END WHILE
```

Once an ant constructs its rule the pheromone values of each trail are updated. First we evaporate the pheromone values of all those terms, which occur in the rule and then we increase the pheromone values of the terms used in the rule proportional to the quality of the rule constructed by the ant. Pheromone values of unused terms are updated by normalizing all the terms. By evaporating only the used terms more exploration can be achieved. We prune the best rule before adding it in the final discovered rule list. This helps to improve the accuracy of the classifier. Then the training cases correctly classified by the best rule are removed. This process continues until the number of cases in the training set becomes lesser than or equal to a user specified threshold called maximum uncovered cases. The output of the algorithm is a final ordered rule list used to classify unseen data.

4.2. *Term selection*

The probability of a term to be added in the current partial rule is given by Eq. (1). The term is a value under the domain of an attribute in the data set.

$$P_{ij} = \frac{\tau_{ij}(t) \cdot \eta_{ij}(t)}{\sum_{i=1}^{a} x_i \cdot \sum_{j=1}^{b_i} (\tau_{ij}(t) \cdot \eta_{ij}(t))}\,, \tag{1}$$

where $\tau_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration t, η_{ij}(t) is the value of the heuristic function at iteration t, a is the total number of attributes, x_i is a binary variable that is set to 1 if the attribute a_i was not previously used by the current ant and else set to 0 and b_i is the number of values in the ith attribute's domain.

4.3. *Pheromone initialization*

Each term has a pheromone value associated with every other term. The pheromone matrix is initialized at the beginning of each while loop iteration. The initial value of each cell is initialized according to Eq. (2).

$$\tau_{ij} = \frac{1}{\sum_{i=1}^{a} b_i}\,, \tag{2}$$

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

4.4. *Pheromone updating*

After an ant constructs its rule, the pheromone value of each term that occurs in the current rule is updated by Eqs. (3) and (4), and the pheromone values of unused terms are updated by the normalization.

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t)\,, \tag{3}$$

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \left(1 - \frac{1}{1+Q}\right) \cdot \tau_{ij}(t)\,, \tag{4}$$

where ρ is the pheromone evaporation rate, Q is the quality of the current rule. For large value of ρ, pheromones evaporate rapidly and that tends to favor the exploration, while for small value of ρ, pheromones evaporate slowly and that tends to favor the exploitation. We fix the value of ρ at 0.15 in our experiments. The quality of the rule denoted by Q is computed by using confidence and coverage of the rule of ant and is given in Eq. (5).

$$Q = \frac{TP}{Covered} + \frac{TP}{N}, \tag{5}$$

where *Covered* is the number of cases covered by the rule of ant, TP is the number of cases covered by the rule that have the class predicted by the rule of ant and N is the number of cases in the training set not covered yet by the any rule.

4.5. *Heuristic function*

The heuristic function is an estimate of the quality of the term to be added in current partial solution. The ants know the rule consequent before constructing the rule antecedent hence our heuristic function is dependent on the chosen class. The heuristic function favors the selection of those terms that increase the probability that the rule will predict the current class and improve the predictive accuracy of the rule. When an ant is on vertex v_i and wants to go to vertex v_j, the heuristic function considers compatibility among vertex v_i and vertex v_j and the overall importance of vertex v_j in the current class. In this way more appropriate terms are added in the current rule. If we do not consider compatibility among vertex v_i and v_j then the ant can choose a $term_j$ that may not occur with term_i in the data set. The proposed heuristic function is given in Eqs. (6) and (7).

$$\eta_{ij} = Correct\_Coverage_{ij} \cdot \frac{|\, term_j, k \,|}{|\, total\_term_j \,|}, \tag{6}$$

$$Correct\_Coverage_{ij} = \frac{|\, term_{ij}, k \,|}{|\, term_i, k \,|}, \tag{7}$$

where $|term_j, k|$ is the number of training cases having $term_j$ with the current class k chosen by the ant, $|\, total\_term_j|$ is the number of training cases having $term_j$, $|term_{ij}, k|$ is the number of training cases having $term_j$ occuring with $term_i$ in the current class k and $|term_i, k|$ is the number of times $term_i$ that occurs with the current class k.

Our heuristic function is a more appropriate heuristic function because it considers the overall importance of a term to be added in current partial rule for the class chosen by the ant and relationship of this term with previously chosen terms. This compatibility based heuristic function is very effective in large dimensional data space. In large dimensional cases if we do not use compatibility among attributes then an ant has a large number of possible vertices to go and most of them are not related to the vertices already chosen by the ant. Considering compatibility among attributes, we can achieve effective decision-making and the speed of the algorithm is also increased.

The heuristic value is also calculated for the class attribute. It is the ratio of uncovered data instances that have the class chosen by the ant. For example if 30% of the uncovered data instances of class 1, the heuristic value for the class 1 is 0.3. At the start of outer while loop in the algorithm, class is chosen probabilistically on the basis of the heuristic value of a class and that class is fixed, all the ants of repeat-until loop construct their rules for that class.

The first term of the rule antecedent is chosen by an ant on the basis of following Laplace-corrected confidence of a term given in Eq. (8).

$$L = \frac{|\,term_{ij}, k\,| + 1}{|\,term_{ij}\,| + No\_of\_classes}, \tag{8}$$

where $|term_{ij}, k|$ is the number of training cases having $term_{ij}$ and the current class k, $term_{ij}$ is the number of training cases having $term_{ij}$ and $No\_of\_classes$ is the number of values in the domain of class attribute. It has the advantage of penalizing the terms that has too specific rules, helping to reduce over-fitting. For example if a term occurs in just one training case, and the sample has current class, without the Laplace correction its confidence is 1. In the Laplace correction, supposing that we have 2 classes, its confidence will be 66% instead of 1.

4.6. *Rule pruning*

Rule pruning is a procedure to remove the irrelevant terms from the rule to improve the predictive accuracy of the rule. The idea is to iteratively remove one term at a time from the rule until removing any term will decrease the predictive accuracy of the rule. We change the rule pruning procedure of the original Ant Miner. Instead of pruning each rule constructed by each ant we only prune the best rule before inserting it in the final discovered rule list. This strategy helps to reduce the computational complexity of the algorithm, because rule pruning is a complex procedure.

4.7. *Classifying unseen cases*

A new test case, unseen during training is classified by applying the rules in the order they were discovered. The first rule, that correctly classifies the new case is fired and assigned the class predicted by the rule's consequent. If none of the rule from the discovered rule list is fired, we use a default rule that predicts the majority class in the set of uncovered training cases and assign the unseen case the class of that default rule.

5. Simulation Study

In our experiments we used 9 data sets obtain from the UCI data set repository.[2] Main characteristics of these data sets are summarized in Table 1. The first column of the table contains the name of the data set, the second column is the number of

January 28, 2010 4:23:03pm WSPC/123-JCSC 00624 FA1

Table 1. Data sets used in the experiments.

Dataset name	No. of attributes	No. of instances	No. of classes
Breast cancer − W	9	683	2
Wine	13	178	3
Credit (Australia)	15	690	2
Credit (Germany)	19	1000	2
Car	6	1728	4
Tic-tac-toe	9	958	2
Iris	4	150	3
Balance scale	4	625	3
TAE	6	151	3

attributes in the data set, the third column is the number of samples in the data set and the final column contains the number of classes in the domain of class attribute. Four data sets are binary classification and 5 are multi classification problems. The experiments are performed using a ten-fold cross validation procedure. In a ten-fold cross validation, the data set is split into 10 equally sized mutually exclusive subsets. Each of the 10 subsets is used once for testing while the other 9 are used for training. The results of the 10 runs are then averaged and reported as the final result.

The results are compared with the previously proposed Ant Miner's versions, i.e., Ant Miner, Ant Miner2, Ant Miner3, and a popular decision tree builder called C4.5.[8] In the decision tree each leaf assigns class labels to observation and it also builds comprehensible classifiers. Ant Miner versions discover ordered rule list whereas C4.5 discovers unordered rule list.

The CAnt Miner has four user defined parameters: number of ants, maximum uncovered cases, number of rules converged and evaporation rate. In our experiments we use the following values of these parameters:

- Number of ants = 1000;
- Maximum uncovered cases = 10;
- Number of rules converged = 10;
- Evaporation rate = 0.15.

Same parameter values are used for the algorithms being compared. In addition we use the value of 10 for the parameter "minimum cases covered" for the compared Ant Miners. The results are compared on the basis of predictive accuracy, number of rules generated by each classifier and the number of terms per rule.[10] The results of ten-fold cross validation are shown in Tables 2−4.

The results show that the proposed technique achieves higher accuracy rate on all the data sets. However, the numbers of rules generated and the number of terms per rule for the proposed technique are higher than the other versions of Ant Miner.

The reason for the higher number of rules is that in our proposed technique a rule may even cover only one training sample. The better results of CAnt Miner, in terms

Table 2. Predictive accuracies obtained by 10 fold cross validation.

Data sets	CAnt Miner	Ant Miner	Ant Miner2	Ant Miner3	C4.5
BC − W	**97.54 ± 0.98**	94.64 ± 2.74	92.70 ± 2.82	93.56 ± 3.45	94.84 ± 2.62
Wine	**98.24 ± 2.84**	90.0 ± 9.22	90.49 ± 10.13	94.44 ± 5.24	96.60 ± 3.93
Credit (Aust)	**89.42 ± 4.21**	86.09 ± 4.69	84.20 ± 4.55	86.67 ± 5.46	81.99 ± 7.78
Credit (Ger)	**73.64 ± 2.67**	71.62 ± 2.71	73.16 ± 5.21	72.07 ± 4.32	70.73 ± 6.71
Car	**98.02 ± 0.96**	82.38 ± 2.42	81.89 ± 2.63	78.82 ± 3.76	96.0 ± 2.13
Tic-tac-toe	**100 ± 0.0**	74.95 ± 4.26	72.54 ± 5.98	72.02 ± 4.50	94.03 ± 2.44
Iris	**97.33 ± 4.66**	95.33 ± 4.50	94.67 ± 6.89	96.0 ± 4.66	94.0 ± 6.63
Balance Scale	**86.61 ± 6.18**	75.32 ± 8.86	72.78 ± 9.23	75.06 ± 6.91	83.02 ± 3.24
TAE	**77.33 ± 10.5**	50.67 ± 6.11	53.58 ± 7.33	53.04 ± 10.67	51.33 ± 9.45

Table 3. Average number of rules obtained by 10 fold cross validation.

Data sets	CAnt Miner	Ant Miner	Ant Miner2	Ant Miner3	C4.5
BC − W	18.60	11.0	11.40	11.10	**10.50**
Wine	**4.10**	5.50	5.30	4.20	5.30
Credit (Aust)	7.80	3.90	3.70	**3.0**	74.80
Credit (Ger)	10.0	8.50	8.10	**6.40**	73.60
Car	57.60	**11.40**	11.80	13.40	80.26
Tic-tac-toe	14.80	**6.60**	6.80	7.10	38.60
Iris	10.90	9.20	10.0	8.10	**5.50**
Balance Scale	98.30	17.70	**16.40**	17.50	40.10
TAE	44.50	20.90	**9.30**	12.30	18.30

Table 4. Average number of terms/rule obtained by 10 fold cross validation.

Data sets	CAnt Miner	Ant Miner	Ant Miner2	Ant Miner3	C4.5
BC − W	1.45	**1.02**	1.03	1.03	2.32
Wine	1.65	**1.04**	1.45	1.53	1.41
Credit (Aust)	1.58	**1.0**	1.16	**1.0**	3.22
Credit (Ger)	1.71	**1.13**	1.22	1.30	3.21
Car	2.49	**1.03**	1.13	1.18	2.59
Tic-tac-toe	2.50	**1.09**	1.18	1.20	2.64
Iris	1.05	**1.0**	**1.0**	**1.0**	1.22
Balance Scale	2.51	**1.0**	**1.0**	**1.0**	2.85
TAE	1.48	**1.0**	1.02	1.05	2.69

of predictive accuracy, is due to compatibility based heuristic function, dynamic stopping of terms addition in the rule antecedent part of the ant, and the fact that all the ants search for the same class label in an iteration of the outer "while" loop to extract a single best rule to guide the search to a specific direction.

6. Conclusion and Future Work

This paper proposed a new version of Ant Miner algorithm for classification problem called CAnt Miner that uses compatibility based heuristic function to select the terms for addition in the rule, a method for dynamically stopping/adding terms in the rule antecedent part, and the choice of class label before rule construction. For checking the robustness of the proposed approach, we experimented with both binary and multi class data sets. We have also compared the performance of CAnt Miner with other versions of Ant Miner and C4.5 on 9 public domain data sets. The experiments results show that proposed approach achieves a higher accuracy rate.

In future we will try to optimize the parameters used in the algorithm. Another future direction is to develop the algorithm for multi labels classification problems, where a single rule can predict multiple classes, it involves the prediction of two or more class attributes rather then just one class attribute.

References

1. M. Dorigo and T. Stutzle, *Ant Colony Optimization* (MIT Press, MA, 2004).
2. S. Hettich and S. D. Bay, The UCI KDD Archive (1996), http://kdd.ics.uci.edu.
3. B. Liu, H. A. Abbass and B. McKay, Density based heuristic for rule discovery with Ant Miner, *Proc. 6th Australiasia-Japan Joint Workshop on Intelligent Evolutionary Systems*, Australia, Canberra (2002), pp. 180−184.
4. B. Liu, H. A. Abbass and B. McKay, Classification rule discovery with ant colony optimization, *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.* (2003), pp. 83−88.
5. B. Liu, H. A. Abbass and B. McKay, Classification rule discovery with ant colony optimization, *IEEE Comput. Intell. Bull.* **3** (2004) 31−35.
6. D. Martens, M. de Backer, R. Haesen, J. Vanthienen, M. Snoeck and B. Baesens, Classification with ant colony optimization, *IEEE Trans. Evolutionary Computations*, Vol. 11 (2007).
7. R. S. Parpeinelli, H. S. Lopes and A. A. Freitas, Data mining with an ant colony optimization algorithm, *IEEE Trans. Evolutionary Computations* **6** (2002) 321−332.
8. T. R. Quinlan, *C4.5*: *Programs for Machine Learning* (Morgan Kaufmann, CA, 1993).
9. J. Smaldon and A. Freitas, A new version of the Ant-Miner algorithm discovering unordered rule sets, *Proc. GECCO* (2006), pp. 43−50.
10. I. H. Witten and E. Frank, *Data Mining*: *Practical Machine Learning Tools and Techniques*, 2nd edn. (Morgan Kaufmann, San Francisco, 2005).