IMPLEMENTATION OF ANT COLONY ALGORITHMS IN MATLAB

R. Seidlová, J. Poživil

Institute of Chemical Technology, Department of Computing and Control Engineering
Technická 5, Prague 6, 166 28, Czech Republic

Abstract

Ant colony optimization (ACO) algorithms have been successfully applied to combinatorial optimization tasks especially to data mining classification problem. The ant miner algorithm is based on the behavior of ants in searching of food. Implementation of ACO algorithm in MATLAB is presented in this study. Our experiments have used data sets from the UCI data set repository. We have selected databases with different properties such as number of instances, character of attributes, percentage of missing values or imbalance ratio. An important parameter of ACO algorithm is a heuristic function. It was determined that the selection of heuristic function has large influence on calculation time of the algorithm. In the contribution the influence of heuristic function on accuracy of the classification algorithm is discussed.

Keywords: Ant Colony Algorithms, Knowledge Discovery, Classification Rules, Multiagent system, Machine Learning

1. INTRODUCTION

Classification is a data mining technique studied by statisticians and machine leaning researchers for a very long period of time. It is a process in which classes are predefined and a classifier is built (learnt) which allocates data samples to classes. Prediction rules are often represented in the IF-THEN form, and it can be described as: IF<conditions> THEN <class>. In each rule, the <condition> part (the antecedent) usually contains a logic combination of predictor attributes in the form: term1 AND term2 AND…AND term¡. And each term is a triple <attribute, operation, value>, such as <Color = Red>. The consequent part <class> shows the predicted class whose cases satisfy the <condition>. Using these rules, people can make prediction of a certain phenomenon or gain insights from large amount of data

Many classification algorithms already exist, such as decision trees [1], (e.g. the C4.5 algorithm [2]) neural networks [3], statistical algorithms [4], and genetic algorithms [5]). In this paper, we study ant colony algorithms [6] inspired by the behavior of ants during searching/finding paths from the nest to food sources.

Social insects like ants, bees and termites work by themselves in their simple tasks, independently of others members of the colony. However, when they act as a community, they are able to solve complex problems emerging in their daily lives, by means of mutual cooperation. This emergent behavior of a group of social insects is known as “swarm intelligence” [7]. Ants are able to find the shortest path between a food source and the nest without the aid of visual information, and also to adapt to a changing environment [8]. It was found that the way ants communicate with each other is based on pheromone trails. While ants move, they drop a certain amount of pheromone on the floor, leaving behind a trail of this substance that can be followed by other ants. The more ants follow a pheromone trail, the more attractive the trail becomes to be followed in the near future. The basic idea is illustrated in figure 1.

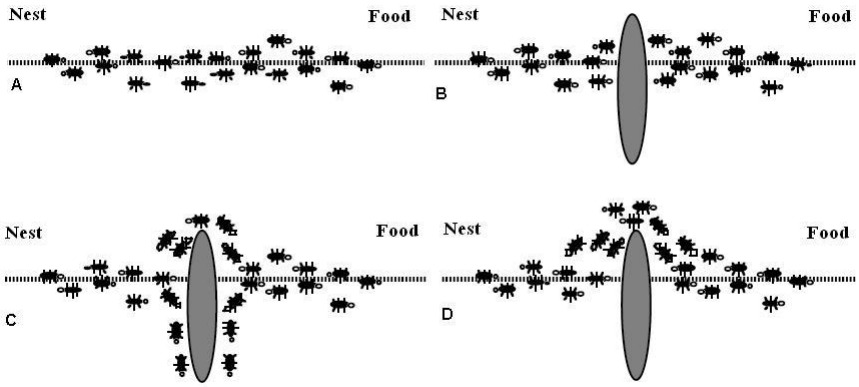

Figure 1 (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

Two ants start from their nest (left) and look for the shortest path to a food source (right). Initially, no pheromone is present on either trails, so there is the same chance of choosing either of the two possible paths. Suppose one ant chooses the upper trail, and the other one the lower trail. The ant that has chosen the upper (shorter) trail will have returned faster to the nest. As a result, there is a greater amount of pheromone on the upper trail as on the lower one. The probability that the next ant will choose the upper (shorter) trail will be higher. More ants will choose this trail, until all (majority) ants will follow the shorter path.

2. ANTMINER ALGORITHM

The core of the algorithm is the incremental construction of classification rules of the form *IF* ($term_1$) *AND* ($term_2$) *AND ... AND* ($term_n$) *THEN* (*class*) from data. Each term is an attribute-value pair related by an operator (relational operator). The *IF* part corresponds to the rule's antecedent and the *THEN* part is the rule's consequent, which represents the class to be predicted by the rule. An example term is "Color = red". The attribute's name is "color" and "red" is one of its possible values. Since we use only "=" any continuous (real-valued) attributes present in the data have to be discretized in a preprocessing step. Pseudo-code of AntMiner is illustrated in figure.2.

```
TrainingSet = {all training cases};
DiscoveredRuleList = []; /* rule list is initialized with an empty list */
WHILE (TrainingSet ≥ Max_Uncovered_Cases)
   i = 1; /* ant index */
  No_Ants_Converg = 1; /* convergence test index */
  Initialize all trails 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 a time to the current rule;
     Prune rule Ri;
     Update the pheromone of all trails, by increasing pheromone in the trail followed by Ant_i
     (in proportion to the quality of R_i) and decreasing pheromone in the other trails
     (simulating pheromone evaporation);
     IF (R_i is equal to R_i – 1) /* update convergence test */
         THEN No_Ants_Converge = No_Ants_Converge + 1;
         ELSE No_Ants_Converge = 1;
     END IF
     i = i + 1;
  UNTIL (i ≥ No_of_Ants) OR (No_Ants_Converg ≥ No_Rules_Converg)
  Choose the best rule R_best among all rules R_i constructed by all the ants;
  Add rule R_best to DiscoveredRuleList;
  TrainingSet = TrainingSet – {set of cases correctly covered by R_best};
END WHILE
```

Figure 2: Pseudo-code of original AntMiner

AntMiner algorithm works as follows: It starts with an empty rule list and iteratively adds one rule at a time to that list while the number of uncovered training examples is greater than a user-specified maximum value. A single ant starts with an empty rule and adds one term at a time to the rule antecedent. The ant keeps adding a term to the partial rule until any term added to the antecedent would make the rule cover less training examples than a user-specified threshold, which would make the rule too specific and unreliable, or all attributes have already been used by the ant. Once this process of rule construction has finished, first the rule constructed by the ant is pruned to remove irrelevant terms from the rule antecedent. Then, the consequent of the rule is chosen to be the class value most frequent among the set of training examples covered by the rule. Finally, pheromone trails are updated and another ant starts to construct a new rule. The process of constructing a rule is repeated until a user-specified number of rules have been reached, or the current ant has constructed a rule that is exactly the same as rules constructed by a predefined number of previous ants, which works as a rule convergence test. The best rule, based on a quality measure *Q*, found along this iterative process is added to the rule list and the correctly classified training examples are removed from the training set.

The heuristic function η is based on the amount of information associated with the attribute *i* with value *j* [9].

$$infoT_{ij} = -\sum_{w=1}^{k}\left(\frac{freqT_{ij}^{w}}{|T_{ij}|}\right) * \log_2\left(\frac{freqT_{ij}^{w}}{|T_{ij}|}\right) \quad (1)$$

where:
k is the number of classes in the dataset
$|T_{ij}|$ is the total number of cases in the data partition T_{ij} (partition that contains the cases where the attribute *i* is equal to the value *j*);
freq T_{ij}^{w} represents the number of cases in T_{ij} that belong to class w.

In Ant-Miner, the heuristic value is taken to be an information theoretic measure for the quality of the term to be added to the rule. The quality here is measured in terms of the entropy for preferring this term to the others; and is given by:

$$\eta_{ij} = \frac{log_2(k) - infoT_{ij}}{\sum_i^a \sum_j^{b_i} log_2(k) - infoT_{ij}} \quad (2)$$

where *a* is the total number of attributes and b_i is the number of values in the domain of attribute *i*.

Authors [10] believe that the AntMiner algorithm does not need accurate information in this heuristic value and propose the following easily computable equation:

$$\eta_{ij} = \frac{majority\_class\ T_{ij}}{|T_{ij}|} \quad (3)$$

where *majority_class*T_{ij} is the majority class in partition T_{ij}. This heuristic has less computational cost and has the same accuracy as original AntMiner.

After each ant completes the construction of its rule, pheromone level updating is carried out in original algorithms as follows [11]:

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

where ρ is the pheromone evaporation rate, large value of ρ indicates a fast evaporation. Typical values for this evaporation factor lie in the range <0.8,0.99>

The quality Q of a rule is measured as:

$$Q = \left(\frac{TruePos}{TruePos + FalseNeg}\right) * \left(\frac{TrueNeg}{FalsePos + TrueNeg}\right) \quad (5)$$

where:

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

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

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

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

3. COMPUTATIONAL

3.1. Examined algorithms

The algorithm AntMiner is coded in MATLAB environment (R2012a). Parameters of algorithm (*Number of Ants*, *Minimum number of cases per rule*, *Maximum number of uncovered cases in the training set*, *Number of rules used to test convergence of the ants, evaporation factor*) were modified as well as the type of heuristic function. The data input files are in ARFF (Attribute-Relation File Format). The original version of AntMiner does not work with continuous attributes directly. It requires continuous attributes to be discretized in a preprocessing step. The discretization method C4.5-Disc [12] was applied prior to AntMiner in a data preprocessing phase. For each continuous attribute, C4.5 is applied to a reduced dataset which only contains the attribute to be discretized and the class attribute. Data input file were discretized with well-known Weka system [13].

3.2 Attribute-Relation File Format (ARFF)

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software.

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.

The **Header** of the ARFF file contains the name of the relation (as the first line), a list of the attributes (the columns in the data), and their types. An example header of dataset looks like this:

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
```

The **Data** of the ARFF file looks like the following:

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
```

```
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Each instance is represented on a single line. Attribute values for each instance are delimited by commas, they must appear in the order that they were declared in the header section. Missing values are represented by a "?" mark.

3.3. Datasets used in the Experiments

AntMiner algorithm has been applied by us to a wide range of public-domain datasets, obtained from the Machine Learning Repository UCI. For all datasets, the continuous attributes were discretized using the C4.5-Disc algorithm. Comparative performance of AntMiner method using ten-fold cross-validation was evaluated. Each database was divided into ten partitions, and each method was run ten times, using a different partition as test set each time, with the other nine as training set. The rule list produced by training set to predict the class of each case in the test set was used, the accuracy rate being calculated according to equation (5). Every rule list includes a default rule, which has no condition and takes the majority class in the set of training cases as its class, so that the default rule could be applied if none of the rules in the list covers the test case.

4. RESULTS AND DISCUSSION

AntMiner algorithm includes several parameters: *Number of Ants*, *Minimum number of cases per rule*, *Maximum number of uncovered cases in the training set*, *Number of rules used to test convergence of the ants, Evaporation factor*. The greatest influence on the accuracy and calculation time have *Number of Ants* and *Evaporation factor*. Increasing evaporation factor will result in a slower convergence process and no significant increase in accuracy. The calculation time increases with the number of ants and with the evaporation factor. The two-dimensional plots (Figs.2 and 3) shows the influence these two parameters on execution time and accuracy and it is obviously that there is flat-maximum effect.

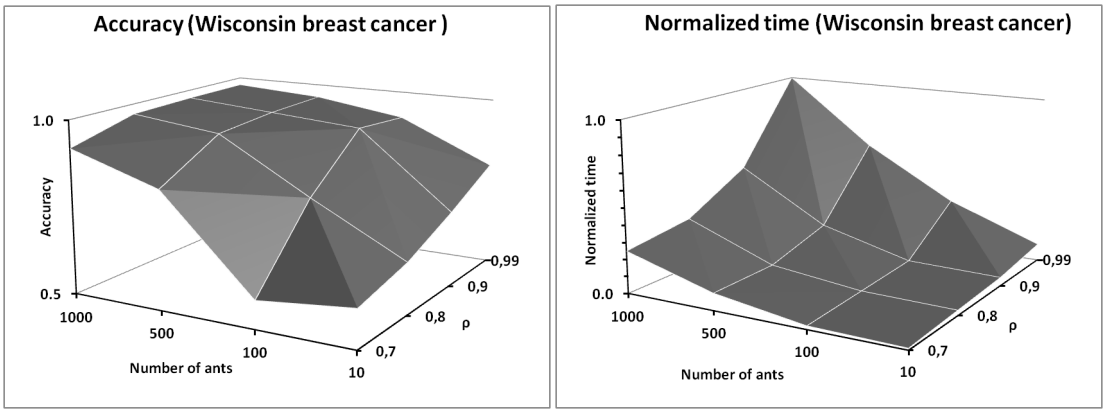

Figure 2 Influence of the number of ants and evaporation factor ρ on accuracy and execution time for Wisconsin breast cancer dataset.

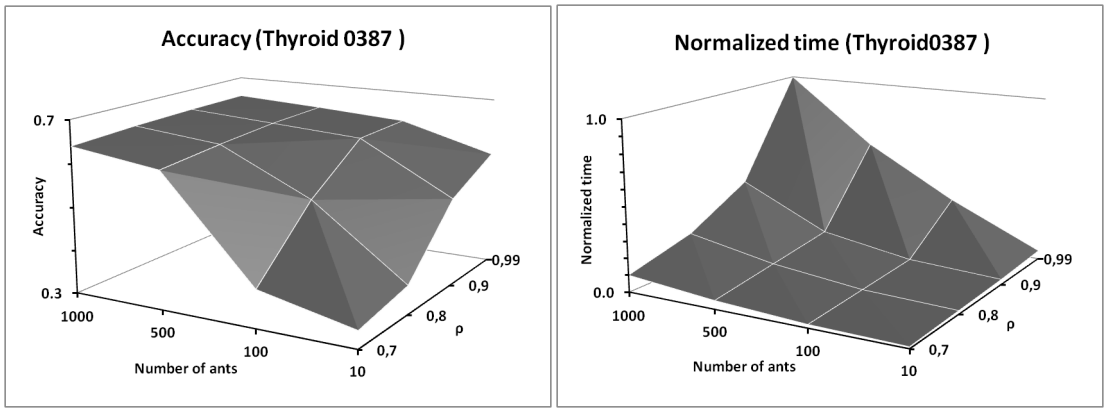

Figure 3 Influence of the number of ants and evaporation factor ρ on accuracy and execution time for Thyroid 0387 dataset.

5. CONCLUSIONS

In this contribution, data mining methods based on ant colony algorithm coded in MATLAB environment were examined. Matlab advantage lies in its powerful graphics and ease of use. AntMiner algorithms generate rules by selecting attribute-value pairs on the basis of ant's entropy and pheromone values. The results of our experiments are summarized in Table 1. For each data set, the number of data instances, attributes and average accuracy for two different heuristic functions (calculated using eq.2 resp.3) are displayed. Both accuracies are approximately the same, but algorithm with the second type of heuristic function is about twenty percent faster.

Dataset	Instances	Attributes	Accuracy 1	Accuracy 2
Cleveland heart disease	303	14	0,8325	0,8319
Horse Colic	368	22	0,8712	0,8701
Hypothyroid	3772	29	0,9891	0,9879
Mammo	1728	7	0,8690	0,8685
Thyroid 0387	9172	28	0,6621	0,6610
Wisconsin breast cancer	699	9	0,9613	0,9602

REFERENCES

[1] Chang, S.-Y., Lin, C.-R., & Chang, C.-T.: A fuzzy diagnosis approach using dynamic fault trees. Chemical Engineering Science, 57(15) (2002), 2971-2985.
[2] Quinlan, J.R.: C4.5: Programs for Machine Learning. San Francisco:Morgan Kaufmann, (1993)
[3] Venkatasubramanian, V., Vaidyanathan, R.,& Yamamoto, Y.: Process fault detection and diagnosis using neural networks-I.Steady-stateeee processes. Computers & Chemical Engineering, 14(7) (1990) 699-712.
[4] Hsiung, J. T., Himmelblau, D.M.: Detection of leaks in a liquid-liguid heat exchanger using passive acoustic noise. Computers & Chemicial Engineering, 20(9) (1996) 1101-1111
[5] Wesley Romão, Alex A. FREITAS, Itana M.de S.Gimenes.: Discovering interesting knowledge from a science and technology database with a genetic algorithm. Applied Soft Computing, 4 (2004) 21-137

[6] Parepinelli, R. S., Lopes, H.S., & Freitas, A.: An Ant Colony Algorithm for Classification Rule Discovery. In H. A. a. R. S. a. C. Newton(Ed.), Data Mining: Heuristic Approach: Idea Group Publishing, (2002).
[7] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press.
[8] Dorigo, M., & Maniezzo, V. (1996). The Ant System: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics , 26(1), pages 1-13.
[9] Weiss, M., & Kulikowski, C. (1991). Computer systems that learn - classification and prediction methods from statistics, neural nets, machine learning, and expert systems. San Francisco: Morgan Kaufmann Publishers Inc.
[10] Liu, B., Abbass, H., & McKay, B. (2002). Density-based Heuristic for Rule Discovery with Ant-Miner. The 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System, pages 180-184.
[11] Liu, B., Abass, H., & McKay, B. (2004). Classification Rule Discovery with Ant Colony Optimization. IEEE Computational Intelligence Bulletin , 3(1)
[12] Kohavi, R., & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. 2nd Int. Conf. Knowledge Discovery and Data Mining, pages 114-119
[13] Available at http://www.cs.waikato.ac.nz/ml/weka/