PREPRINT - In: Khosrow-Pour, M (Ed.), Encyclopedia of Information Science and Technology. Hershey: Idea Group Inc., v. ?, p. ?-?, 2005. ISBN: 1-59140-553-X

Classification–Rule Discovery with an Ant Colony Algorithm

Rafael S. Parpinelli
CEFET-PR, Brazil

Heitor S. Lopes
CEFET-PR, Brazil

Alex A. Freitas
University of Kent, United Kingdom

INTRODUCTION

Ant colony optimization (ACO) is a relatively new computational intelligence paradigm inspired by the behaviour of natural ants (Bonabeau, Dorigo & Theraulaz, 1999). The natural behaviour of ants that we are interested in is the following. Ants often find the shortest path between a food source and the nest of the colony without using visual information. In order to exchange information about which path should be followed, ants communicate with each other by means of a chemical substance called pheromone. As ants move, a certain amount of pheromone is dropped on the ground, creating a pheromone trail. The more ants follow a given trail, the more attractive that trail becomes to be followed by other ants. This process involves a loop of positive feedback, in which the probability that an ant chooses a path is proportional to the number of ants that have already passed by that path.

Hence, individual ants, following very simple rules, interact to produce an intelligent behaviour – a solution to a complex problem – at the higher level of the ant colony. In other words, intelligence is an emergent phenomenon; that is, "the whole is more than the sum of the parts".

In this article we present an overview of Ant-Miner, an ACO algorithm for discovering classification rules in data mining (Parpinelli, Lopes & Freitas, 2002a, 2002b). In essence, in the classification task each case (record) of the data being mined consists of two parts: a goal attribute, whose value is to be predicted, and a set of predictor attributes. The aim is to predict the value of the goal attribute for a case, given the values of the predictor attributes for that case.

To the best of our knowledge, the use of ACO algorithms (Bonabeau, Dorigo & Theraulaz, 1999; Dorigo et al., 2002) for discovering classification rules is a very under-explored research area. There are other ant algorithms developed for the data mining task of clustering – see for example Monmarché (1999) – but that task is very different from the classification task addressed in this article. Note that Ant-Miner was designed specifically for discovering classification rules, rather than for solving other kinds of data mining tasks.

In other research areas ACO algorithms have been shown to produce effective solutions to difficult real-world problems. A detailed review about many other ACO algorithms (designed to solve many other different kinds of problems) and a discussion about their performance can be found in Bonabeau, Dorigo and Theraulaz (1999) and Dorigo et al. (2002).

A typical example of application of ACO is network traffic routing, where artificial ants deposit "virtual pheromone" (information) at the network nodes. In essence, the amount of pheromone deposited at each node is inversely proportional to the congestion of traffic in that node. This reinforces paths through uncongested areas. Both British Telecom and France Telecom have explored this application of ACO in telephone networks.

ANT COLONY OPTIMIZATION

An ACO algorithm is essentially a system based on agents that simulate the natural behavior of ants, including mechanisms of cooperation and adaptation.

ACO algorithms are based on the following ideas:

- Each path followed by an ant is associated with a candidate solution for a given problem;
- When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem;

- When an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone have a greater probability of being chosen by the ant.

As a result, the ants eventually converge to a short path, hopefully the optimum or a near-optimum solution for the target problem.

In essence, the design of an ACO algorithm involves the specification of (Bonabeau, Dorigo & Theraulaz, 1999):

- An appropriate representation of the problem, which allows the ants to incrementally construct/modify solutions through the use of a probabilistic transition rule, based on the amount of pheromone in the trail and on a local, problem-dependent heuristic;
- A method to enforce the construction of valid solutions;
- A problem-dependent heuristic function (h) that measures the quality of items that can be added to the current partial solution;
- A rule for pheromone updating, which specifies how to modify the pheromone trail (t);
- A probabilistic transition rule based on the value of the heuristic function (h) and on the contents of the pheromone trail (t) that is used to iteratively construct a solution.

Artificial ants have several characteristics similar to real ants, namely:

- Artificial ants have a probabilistic preference for paths with a larger amount of pheromone;
- shorter paths tend to have larger rates of growth in their amount of pheromone;
- The ants use an indirect communication system based on the amount of pheromone deposited on each path.

MOTIVATIONS FOR USING ACO

ACO possesses a number of features that are important to computational problem solving (Freitas & Johnson, 2003):

- The algorithms are relatively simple to understand and implement, whilst also offering emergent complexity to deal effectively with challenging problems;
- They can be readily hybridized with other techniques and/or problem-dependent heuristics in a synergistic fashion;
- They are compatible with the current trend towards greater decentralization in computing;
- The algorithms are highly adaptive and robust, enabling them to cope well with noisy data.

Two more features of ACO are particularly useful in data mining applications:

- Many projects in the field of data mining were developed using deterministic decision trees or rule induction algorithms. These algorithms are hill climbing like and are susceptible to finding only locally optimal solutions instead of the global optimum. The utilization of ACO to induce classification rules tries to mitigate this problem of premature convergence to local optima, since ACO algorithms have a stochastic component that favors a global search in the problem's search space;
- Unlike classical methods for rule induction, the ACO heuristic is a population-based one. This characteristic has advantages over other methods because it allows the system to search many different points in the search space concurrently and to use the positive feedback between the ants as a search mechanism.

REPRESENTING A CANDIDATE CLASSIFICATION RULE

In Ant-Miner each artificial ant represents a candidate classification rule of the form:

- IF <*term1* AND *term2* AND … > THEN < *class* >.

Each term is a triple <*attribute*, *operator*, *value*>, where *value* is one of the values belonging to the domain of *attribute*. An example of a term is: <*Sex = female*>. *Class* is the value of the goal attribute predicted by the rule for any case that satisfies all the terms of the rule antecedent. An example of a rule is:

- IF <*Salary = high*> AND <*Mortgage = No*> THEN <*Credit = good*>.

In the current version of Ant-Miner the *operator* is always "=", so that Ant-Miner can cope only with categorical (discrete) attributes. Continuous attributes would have to be discretized in a preprocessing step.

DESCRIPTION OF ANT-MINER

The pseudocode of Ant-Miner is described, at a very high level of abstraction, in Algorithm 1. Ant-Miner starts by

initializing the training set to the set of all training cases, and initializing the discovered rule list to an empty list. Then it performs an outer loop where each iteration discovers a classification rule.

The first step of this outer loop is to initialize all trails with the same amount of pheromone, which means that all terms have the same probability of being chosen by an ant to incrementally construct a rule. This is done by an inner loop, consisting of three steps. First, an ant starts with an empty rule and incrementally constructs a classification rule by adding one term at a time to the current rule. In this step a $term_{ij}$ – representing a triple $<Attribute_i = Value_j>$ – is chosen to be added to the current rule with probability proportional to the product of $h_{ij} \cdot t_{ij}(t)$, where h_{ij} is the value of a problem-dependent heuristic function for $term_{ij}$ and $t_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration (time index) t. More precisely, h_{ij} is essentially the information gain associated with $term_{ij}$ – see Cover and Thomas (1991) for a comprehensive discussion on information gain. The higher the value of h_{ij} the more relevant for classification $term_{ij}$ is and so the higher its probability of being chosen. $t_{ij}(t)$ corresponds to the amount of pheromone currently available in the position i,j of the trail being followed by the current ant. The better the quality of the rule constructed by an ant, the higher the amount of pheromone added to the trail positions ("terms") visited ("used") by the ant. Therefore, as time goes by, the best trail positions to be followed – that is, the best terms to be added to a rule – will have greater and greater amounts of pheromone, increasing their probability of being chosen.

Algorithm 1: High-level pseudocode of Ant-Miner.

```
TrainingSet = {all training cases};
DiscoveredRuleList = [];  /* initialized with an empty list */
REPEAT
 Initialize all trails with the same amount of pheromone;
      REPEAT
  An ant incrementally constructs a classification rule;
  Prune the just-constructed rule;
  Update the pheromone of all trails;
      UNTIL (stopping criteria
   Choose the best rule out of all rules constructed by all the ants;
   Add the chosen rule to DiscoveredRuleList;
  TrainingSet = TrainingSet – {cases correctly covered by the chosen rule};
       UNTIL (stopping criteria)
```

The second step of the inner loop consists of pruning the just-constructed rule, that is, removing irrelevant terms – terms that do not improve the predictive accuracy of the rule. This is done by using a rule-quality measure, the same one used to update the pheromones of the trails, as defined later. In essence, a term is removed from a rule if this operation does not decrease the quality of the rule. This pruning process helps to avoid the overfitting of the discovered rule to the training set.

The third step of the inner loop consists of updating the pheromone of all trails by increasing the pheromone in the trail followed by the ant, proportionally to the quality of the rule. In other words, the higher the quality of the rule, the higher the increase in the pheromone of the terms occurring in the rule antecedent. The quality (Q) of a rule is measured by the equation:

- $Q = Sensitivity \cdot Specificity$,

where $Sensitivity = TP / (TP + FN)$ and $Specificity = TN / (TN + FP)$. The meaning of the acronyms TP, FN, TN and FP is as follows:

- TP = number of true positives, that is, the number of cases covered by the rule that have the class predicted by the rule;
- FN = number of false negatives, that is, the number of cases that are not covered by the rule but that have the class predicted by the rule;
- TN = number of true negatives, that is, the number of cases that are not covered by the rule and that do not have the class predicted by the rule; and
- FP = number of false positives, that is, the number of cases covered by the rule that have a class different from the class predicted by the rule.

See Lopes (1997) for a discussion about these variables and their use to estimate predictive accuracy.

The inner loop is performed until some stopping criterion(a) is(are) satisfied, for example, until a maximum number of candidate rules have been constructed.

Once the inner loop is over, the algorithm chooses the highest-quality rule out of all the rules constructed by all the ants in the inner loop, and then it adds the chosen rule to the discovered rule list. Next, the algorithm removes from the training set all the cases correctly covered by the rule, that is, all cases that satisfy the rule antecedent and have the same class as predicted by the rule consequent. Hence, the next iteration of the outer loop starts with a smaller training set, consisting only of cases that have not been correctly covered by any rule discovered in previous iterations. The outer loop is performed until some stopping criterion(a) is(are) satisfied, for example, until the number of uncovered cases is smaller than a user-specified threshold.

Hence, the output of Ant-Miner is the list of classification rules contained in the discovered rule list.

A SUMMARY OF COMPUTATIONAL RESULTS

We have performed computational experiments comparing Ant-Miner with two well-known rule induction algorithms, namely CN2 (Clark & Niblett, 1989) and C4.5 (Quinlan, 1993) in several public-domain data sets often used as a benchmark in the machine learning literature. More precisely, the data sets used in the experiment are Ljubljana breast cancer, Wisconsin breast cancer, tic-tac-toe, dermatology, hepatitis, and Cleveland heart disease. A detailed description of all these data sets is available online from: *http://www.ics.uci.edu/~mlearn/MLRepository.html*. C4.5 and CN2, as well as many other classification algorithms, are also described in Witten and Frank (2000). C4.5 and CN2 were chosen for comparison because they are, in general, two of the most used algorithms belonging to the rule induction paradigm, where discovered knowledge is expressed by IF-THEN rules. Hence, they are a natural choice to be compared with Ant-Miner, since Ant-Miner also discovers knowledge expressed by IF-THEN rules. The results of the experiments summarized here are described in detail in Parpinelli, Lopes and Freitas (2002a, 2002b).

The results showed that Ant-Miner is competitive with both C4.5 and CN2 concerning predictive accuracy on the test set used to measure the generalization ability of the discovered rules. More precisely, predictive accuracy was measured by the accuracy rate, that is, the ratio of the number of correctly classified test cases divided by the total number of test cases (correctly-classified plus wrongly-classified test cases), as usual in the classification literature (Witten & Frank, 2000). Ant-Miner obtained a considerably better accuracy rate than CN2 in the Ljubljana breast cancer and the dermatology data sets, and a considerably better accuracy rate than C4.5 in the hepatitis data set. However, both CN2 and C4.5 obtained a considerably better accuracy rate than Ant-Miner in the tic-tac-toe data set. In the other data sets the difference in the predictive accuracy of Ant-Miner and the other two algorithms was quite small.

However, concerning rule set simplicity (measured by the number of discovered rules and the number of terms per rule), Ant-Miner discovered rule sets much simpler (i.e., smaller) than the rule sets discovered by both C4.5 and CN2. This is an important advantage in the context of data mining, where discovered knowledge is supposed to be shown to a human user in order to support his/her decision making (Fayyad, Piatetsky-Shapiro & Smyth, 1996).

REFERENCES

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

Clark, P., & Niblett, T. (1989). The CN2 rule induction algorithm. *Machine Learning*, *3*(4), 261-283.

Cover, T.M., & Thomas, J.A. (1991). *Elements of information theory*. New York: Wiley.

Dorigo, M., Gambardella, L.M., Middendorf, M., & Stutzle, T. (2002). Guest editorial: Special section on ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *6*(4), 317-319.

Fayyad, U.M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy (Eds.), *Advances in knowledge discovery & data mining* (pp. 1-34). Cambridge, MA: MIT Press.

Freitas, A.A., & Johnson, C.G. (2003). Research cluster in swarm intelligence. *EPSRC Research Proposal GR/S63274/01 – Case for Support*. Computing Laboratory, University of Kent.

Lopes, H.S., Coutinho, M.S., & Lima, W.C. (1997). An evolutionary approach to simulate cognitive feedback learning in medical domain. In E. Sanchez, T. Shibata & L.A. Zadeh (Eds.), *Genetic algorithms and fuzzy logic systems* (pp. 193-207). Singapore: World Scientific.

Monmarché, N. (1999). On data clustering with artificial ants. In A.A. Freitas (Ed.), *Data mining with evolutionary algorithms, Research directions – Papers from the AAAI Workshop* (pp. 23-26). Menlo Park, CA: AAAI Press.

Parpinelli, R.S., Lopes, H.S., & Freitas, A.A. (2002a). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation, Special Issue on Ant Colony Algorithms, 6*(4), 321-332.

Parpinelli, R.S., Lopes, H.S., & Freitas, A.A. (2002b). An ant colony algorithm for classification rule discovery. In H. Abbass, R. Sarker & C. Newton (Eds.), *Data mining: A heuristic approach* (pp. 191-208). London: Idea Group Publishing.

Quinlan, J.R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Witten, I.H., & Frank, E. (2000). *Data mining: Practical machine learning tools with Java implementations*. San Mateo, CA: Morgan Kaufmann.

KEY TERMS

Data Mining: An interdisciplinary research field, whose core is at the intersection of machine learning, statistics, and databases. We emphasize that the goal – unlike, for example, classical statistics – is to discover knowledge that is not only accurate, but also comprehensible for the user.

Overfitting: Term referring to the situation where the discovered rules fit too much to the training set peculiarities. Overfitting usually leads to a reduction of the predictive accuracy rate on the test cases.

Rule List: An ordered list of IF-THEN rules discovered by the algorithm during training. When the rules are applied to classify cases in the test set, they are applied in order. That is, a case is matched with each of the rules in the list in turn. The first rule whose antecedent (conditions in the IF part) matches the attribute values of the case is then used to classify the case; that is, the case is assigned the same class as the class predicted by the first matching rule found in the discovered rule list.

Testing Case: Each of the cases (records) of the test set.

Test Set: A set of cases unseen during the training phase of the algorithm and used to compute the predictive accuracy of the list of rules discovered during the training phase.

Training Case: Each of the cases (records) of the training set.

Training Set: A set of cases used by the algorithm to discover the classification rules. At each iteration of Ant-Miner only one rule is discovered. The training cases that are covered correctly by the discovered 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.