Ant Colony Optimization for First-order Rule Discovery

Rafael Ramirez
Department of Information and
Communications Technologies
Pompeu Fabra University
08018 Barcelona, Spain
Email: rafael.ramirez@upf.edu

***Abstract*—In the past, ant colony optimization has been applied to learning sets of *propositional* rules. In this paper, we present an algorithm for learning sets of *first-order* rules with ant colony optimization. First-order rules can sometimes provide a more intuitive and accurate concept description as they are more expressive than traditional propositional rules. As a case study, we apply our algorithm to expressive music performance modeling, one of the most challenging problems in music informatics, and compare our results with the results obtained by state-of-the-art first-order rule learning algorithms.**

I. Introduction

Classification rules are a popular approach to classification learning in which the *antecedent* of an *if-then* rule is generally a conjunction of tests (or conditions) and the *consequent* is the predicted class or classes (possibly with a probability associated to each class). It is often assumed implicitly that the conditions in classification rules involve testing an attribute value against a constant. Such rules are called *propositional* because they have the same expressive power as *propositional logic*. In many cases, propositional rules are sufficiently expressive to describe a concept accurately. However, there are cases where more expressive rules would provide a more intuitive concept description. These are cases where the knowledge to be learned is best expressed by allowing variables in attributes. Algorithms that are capable of learning first-order rules are natural candidates for dealing with data in domains where background knowledge is available, e.g. biological or musical data.

In this paper we describe Relational-AntMiner (RAM), a new algorithm for inducing first-order rules using ant colony optimization. RAM extends previous approaches to learning classification rules using ant colony optimization by inducing sets of first order logic rules, and incorporates ant colony optimization in inductive logic programming. We apply Relational-AntMiner to expressive music performance modeling, one of the most challenging problems in music informatics.

II. Related work

A. Ant colony optimization

Swarm intelligence [20] studies the emergent collective intelligence of groups of simple agents that interact locally through their environment [21]. It is inspired by groups of insects who live in colonies, such as ants and bees. These insects can only do simple tasks individually, while the colony as a whole shows complex intelligent behavior. Thus, the intelligent behavior can be seen as an emergent property of the group. Ants, for instance, initially wander randomly. Upon finding food they return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food.

Over time, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when an ant finds a good (i.e. short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants to follow a single path. The idea of the ant colony algorithm is to mimic this behavior with simulated ants walking around the graph representing the problem to solve.

Ant colony optimization algorithms [4], [5], [6] have been applied to a variety of problems, including vehicle routing (e.g. [19]), scheduling (e.g. [3]), timetabling (e.g. [16]), and the traveling salesman problem (e.g. [18]). They have an advantage over simulated annealing and genetic algorithms approaches when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time.

The application of ant colony optimization to learning classification rules was first introduced by Parpinelli et al.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.163

[11]. Their proposed algorithm, AntMiner, was soon followed by the extensions AntMiner2 and AntMiner3 proposed by Liu et al. [7], [8]. Martens et al. [9] recently proposed AntMiner+, an algorithm improving on the previous AntMiner versions. However, the aim of all these algorithms is to obtain propositional classification rules of the form

IF *antecedent* **THEN** *consequent*

where *antecedent* is a conjunction of terms of the form $variable = value$, e.g. $sex = female$. Here we extend these algorithms by inducing first-order rules as opposed to propositional rules.

B. Inductive logic progrmming

As mentioned before, in many cases, propositional rules are sufficiently expressive to describe a concept accurately. However, there are cases where more expressive rules would provide a more intuitive concept description. These are cases where the knowledge to be learned is best expressed by allowing variables in attributes (e.g. $father(X, Y)$). One important special case involving learning sets of rules containing variables is called *inductive logic programming* [15], [14]. While most existing machine learning approaches tend to look for patterns in a single table of data and require preprocessing to integrate data from multiple tables, inductive logic programming looks for patterns that involve multiple relations (i.e. tables) from a *relational* database. This is, the input data in inductive logic programming typically consists of several tables, as opposed to only one table. Integrating data from several tables may cause loss of meaning or information. In relational data bases, a relation can be defined either *extensionally* (e.g. by tables) or *intensionally* (as logical rules). Relations defined intensionally typically represent relationships that can be inferred from other relationships and correspond to general knowledge about the domain of discourse. Such general knowledge is often referred as *background knowledge*. Inductive logic programming techniques are natural candidates for dealing with data in domains where background knowledge is available. This is the case for musical data where there is often background knowledge available.

As an example of rules containing variables, consider the following two rules jointly describing the concept of ancestor ($parent(X, Y)$ indicates that Y is the mother or the father of X and $ancestor(X, Y)$ indicates that Y is the ancestor of X):

IF parent(X,Y) THEN ancestor(X,Y)
IF parent(X,Z) AND ancestor(Z,Y) THEN ancestor(X,Y)

These two rules describe a recursive predicate that would be very difficult to represent using a propositional representation.

III. Relational-AntMiner

Relational-AntMiner creates a directed construction graph as the ants' environment. However, the construction of the construction graph in Relational-AntMiner is *dynamic*. This is, it is constructed lazily at run-time as the ants walk through the environment. As mentioned before, ant colony optimization algorithms have the advantage (over genetic algorithms and simulated anealing) of being able to run continuously and adapt to dynamic changes in the construction graph in real time. Relational-AntMiner takes advantage of this fact.

```
Initialize TrainingExamples
Construct graph
while (# of positive examples > threshold)
   while (not convergence)
      create ants
      run ants
      prune rule of best ant
      update edge pheromones levels:
          evaporate edge levels
          reinforce best path
      update probabilities of edges
   end
   collect best rule
   remove examples correctly
          classified by best rule
end
return collected rules
```

A. Construction graph

The graph $G = (V, E)$ is defined as follows:

- **Vertices.** First, a vertex with the predicate $q_{target}(v_1, \ldots, v_m)$, corresponding to the target concept, is defined. Second, for each predicate name q_i $i = 1, \ldots, n$ in the training data and in the background knowledge we define a vertex group. The exact number and form of each vertex in the vertex group for q_i is determined dynamically by the position of the current ant. Each vertex corresponds to a candidate specialization of the rule of the current ant. For each predicate symbol q_i in the training data and background knowledge a vertex $q_i(v_1, \ldots, v_r)$ is defined where each v_k $(1 \leq k \leq r)$ is either a new variable or a variable already appearing in a vertex visited by the ant. At least one of the v_k in $q_i(v_1, \ldots, v_r)$ must already exist as a variable in the rule, i.e must appear in a vertex already visited by the ant.
- **Edges.** At any time point, the set of bidirectional edges in the graph is defined as the set containing the pairs of vertices:

  $(q_i(var_1, \ldots, var_k), q_j(v_1, \ldots, v_r))$ for $j = 1, \ldots, n$

  where $q_i(var_1, \ldots, var_k)$ is the current position of the current ant, and $q_j(v_1, \ldots, v_r)$ $(j = 1, \ldots, n)$ are the vertices defined as indicated above. At any time point, the number of vertices in a vertex group is $(|V| + 1)^r$

where $|V|$ is the number of variables already occurring in the clause, and r is the arity of the predicate associated with the vertex group.

B. Edge probabilities

The probability for an ant in vertex $q_i(v_1, \ldots, v_r)$ to choose the edge to vertex $q_j(v_1, \ldots, v_s)$ is a function of the pheromone value (τ_{ij}) of the edge and an heuristic value (η_{ij}) associated to vertex $q_j(v_1, \ldots, v_s)$ and normalized over all possible edge choices.

$$P_{ij}(t) = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{i=1}^{n} \sum_{l=1}^{m_k} x_{kl} (\tau_{kl}(t) \cdot \eta_{kl})}$$

where x_{kl} is a binary variable which is set to one if vertex kl (i.e. the lth vertex with predicate name q_k) is not yet used by the current ant and to zero, otherwise. The choice of the next edge is restricted in order to avoid the same predicate symbol with the same arguments (i.e. variables) to be chosen more than once in a rule.

Relational-AntMiner determines the heuristic value associated to vertices by considering the performance of the rule over the training data. The estimation of the utility of adding a new vertex is based on the number of positive and negative bindings covered before and after adding a new vertex. More precisely, if R_0 is a rule corresponding to the path ending at vertex $q_i(v_1, \ldots, v_r)$, and R_1 is the rule created by adding vertex $q_j(v_1, \ldots, v_s)$ to rule R_0, then the heuristic value η_{ij} (associated to adding $q_j(v_1, \ldots, v_s)$ to the antecedent of R_0) is defined as

$$\eta_{ij} = t(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0})$$

where p_0 is the number of positive bindings of rule R_0, n_0 is the number of negative bindings of R_0, p_1 is the number of positive bindings of R_1, n_1 is the number of negative bindings of R_1, and t is the number of positive bindings of R_0 that are still covered by R_1.

The definition of η_{ij} is based on information theory. According to information theory, $-log_2 \frac{p_0}{p_0+n_0}$ is the minimum number of bits needed to encode the classification of an arbitrary positive binding among the bindings covered by rule R_0. Similarly, $-log2 \frac{p_1}{p_1+n_1}$ is the number of bits required if the binding is one of those covered by rule R_1. Since t is just the number of positive bindings covered by R_0 that remain covered by R_1, η_{ij} can be seen as the reduction due to $q_j(v_1, \ldots, v_s)$ in the total number of bits needed to encode the classification of all positive bindings of R_0.

IV. Expressive performance modeling

In this section, we present a case study on the application of Relational-AntMiner to one of the most challenging problems in music informatics, namely expressive music performance modeling. We show the benefits of our approach by comparing Relational-AntMiner's accuracy with the accuracies of both traditional propositional methods (e.g. decision trees) and ILP systems (i.e. Tilde).

A. Melodic description

We use sound analysis techniques based on spectral models [13] for extracting high-level symbolic features from the recordings. The spectral model analysis techniques are based on decomposing the original signal into sinusoids plus a spectral residual. From the sinusoids of a monophonic signal it is possible to extract high-level information such as note pitch, onset, and duration among other information (the details can be found in [12]).

B. Musical Analysis

It is widely recognized that humans perform music considering a number of abstract musical structures. Once the pitch of each note in the performances is extracted, we implement Narmour's Impliction/Realization theory [10] in order to provide structure for the recordings. Very briefly, the Implication/Realization model proposed by Narmour is a theory of perception and cognition of melodies. The theory states that a melodic musical line continuously causes listeners to generate expectations of how the melody should continue. Melodic patterns or groups can be identified that represent different degrees of expectations in the listener.

C. Training data

The data set is composed of monophonic recordings of four Jazz standards (*Body and Soul*, *Once I Loved*, *Like Someone in Love* and *Up Jumped Spring*) performed by a professional saxophonist at 11 different tempi. The data set consists of 3192 performed notes. It contains musical-context descriptors of notes for which several expressive transformations have been observed. The observed deviations are in terms of note *duration, onset* and *energy*. Each note in the training data is annotated with a class for each expressive transformation.

D. Note descriptors

We characterize each performed note by a set of features representing both properties of the note itself and aspects of the musical context in which the note appears. Information about the note includes note pitch, note duration amd metrical strength, while information about its melodic context includes the relative pitch and duration of the neighboring notes (i.e. previous and following notes) as well as a musicological analysis (Narmour analysis [10]) of the pitch context in which the note appears. In Relational-AntMiner the characterization of each note is specified by predicates *context/6*, *narmour/2*, *succ/2* and *member/3*. Predicate *context/6* specifies the local context of a note, *narmour/2* specifies the note's Narmour analysis, *metric/2* specifies the note's metrical strength. *succ(X,Y)* means note *Y* is the successor of note *X*, and *member(X,L)* means *X* is a member of list *L*. Note that *succ(X,Y)* also mean *X* is the predecessor of *Y*.

Algorithm	C.C.I(%)
Relational-AntMiner	**84.7**
Tilde	80.3
Foil	77.2
Decision trees	74.5

TABLE I
Cross validation results (correctly classified instances percentage) for the duration transformation model. In the case Relational-AntMiner the standard deviation is $\sigma = 1.03$

Algorithm	C.C.I(%)
Relational-AntMiner	**92.3**
Tilde	90.0
Foil	84.6
Decision trees	78.6

TABLE II
Cross validation results (correctly classified instances percentage) for onset transformation. In the case Relational-AntMiner the standard deviation is $\sigma = 1.65$

E. Learning task

For each expressive transformation, we approach the problem as a classification problem. The classes that interest us are *lengthen, shorten* and *same* for duration transformation; *advance*, *delay*, and *same* for onset deviation; and *soft*, *loud* and *medium* for energy variation. We apply Relational-AntMiner with the following target predicates: *duration/3*, *onset/3* and *energy/3* (where */n* at the end of the predicate name refers to the predicate arity).

F. Results

The correctly classified instances percentage (C.C.I:%) for each expressive transformation (i.e. duration, onset and energy) and each considered algorithm is presented in Tables I-III. Results were obtained using 10-fold cross valiadtion. Relational AntMiner is consistently the algorithm with higher C.C.I:% with 84.7% (σ=0.43), 92.3% (σ=0.4) and 84.9% (σ=0.31) for duration, onset and energy, respectively.

V. Conclusions and future research

We have described Relational-AntMiner, a new algorithm for inducing first-order rules using ant colony optimization. The algorithm provides a dynamic environment in which the ants choose a paths and implicitly construct a rule. Each time a best rule is collected, it is added to the disjunctive hypothesis represented by the set of learned rules. Viewed at this level, the search is a specific-to-general search through the space of hypotheses. By walking in the environment, each ant searches a second hypothesis space, consisting of conjunctions of literals, to find a conjunction that will form the preconditions for its own rule. Within this hypothesis space, each ant conducts a general-to-specific, hill-climbing search, beginning with the most general preconditions possible. One advantage of such process is that it produces a comprehensible model, which is important in many application domains. Our algorithm extends previous approaches to learn classification rules using ant colony optimization by inducing sets of first-order rules, and incorporates ant colony optimization to inductive logic programming. We have applied our algorithm to expressive music performance modeling and the showed the benefits of our approach by comparing the obtained accuracy with the accuracies of state-of-the-art algorithms.

Algorithm	C.C.I(%)
Relational-AntMiner	**84.9**
Tilde	79.8
Foil	74.9
Decision Trees	79.5

TABLE III
Cross validation results (correctly classified instances percentage) for energy variation. In the case Relational-AntMiner the standard deviation is $\sigma = 1.08$

Acknowledgment

This work has been partly sponsored by the Spanish TIN project TIMUL (TIN2013-48152-C2-2-R)

References

[1] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In ed. J. Shavlik, editor, Proceedings of the 15th International Conference on Machine Learning, pages 53-63, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.
[2] Srinivasan, A. (2001). The Aleph Manual.
[3] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, Ant system for job-shop scheduling, J. Oper. Res., Stat., Comput. Sci., vol. 34, no. 1, pp. 39-53, 1994.
[4] M. Dorigo, V. Maniezzo, and A. Colorni, Positive feedback as a search strategy Dipartimento di Elettronica e Informatica, Politecnico di Milano, Milano, Italy, TR 91016, 1991.
[5] M. Dorigo, V. Maniezzo, and A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst., Man, Cybern. Part B, vol. 26, no. 1, pp. 29-41, 1996.
[6] M. Dorigo and T. Sttzle. Ant Colony Optimization, MIT Press, 2004
[7] B. Liu, H. A. Abbass, and B. McKay. Density-based heuristic for rule discovery with ant-miner, in Proc. 6th Australasia-Japan Joint Workshop on Intell. Evol. Syst., Canberra, Australia, pp. 180-184, 2002.
[8] B. Liu, H. A. Abbass, and B. McKay. Classification rule discovery with ant colony optimization, in Proc. IEEE/WIC Int. Conf. Intell. Agent Technol., pp. 83-88, 2003.
[9] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, Classification With Ant Colony Optimization, IEEE Transactions on Evolutionary Computation, 11(5), 2007.
[10] E. Narmour. The Analysis and Cognition of Basic Melodic Structures: The Implication Realization Model. University of Chicago Press (1990).
[11] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. An ant colony based system for data mining: Applications to medical data, in Proc. Genetic and Evol. Comput. Conf., pp. 791-797, 2001.
[12] R. Ramirez, A. Hazan. Discovering Expressive Transformation Rules from Saxophone Jazz Performances, Journal of New Music Research, Vol. 34, No. 4, (2005) pp. 319-330
[13] X. Serra, S. Smith. Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition, Computer Music Journal 14(4), (1990).
[14] J.R. Quinlan. (1990) Learning logical definitions from relations. Machine Learning, 5, pp. 239-266, 1990
[15] E. Shapiro. Algorithmic Program Debugging. Cambridge MA, MIT Press, 1983.
[16] K. Socha, J. Knowles, and M. Sampels, A MAX-MIN ant system for the university timetabling problem, in Proc. 3rd Int. Workshop on Ant Algorithms, M. Dorigo, G. Di Caro, and M. Sampels, Eds., vol. 2463, pp. 1-13, Lecture Notes in Computer Science, 2002.

[17] A. Srinivasan, S. Muggleton, R. D. King: Comparing the use of background knowledge by inductive logic programming systems. International Workshop on Inductive Logic Programming, 2005.
[18] L. M. Gambardella and M. Dorigo, Ant-Q: A reinforcement learning approach to the traveling salesman problem, in Proc. 12th Int. Conf. Mach. Learn., A. Prieditis and S. Russell, Eds., Palo Alto, CA, pp. 252-260, 1995.
[19] A. Wade and S. Salhi, An ant system algorithm for the mixed vehicle routing problem with backhauls, in Metaheuristics: Computer Decision-Making. Norwell, MA: Kluwer, pp. 699-719, 2004.
[20] G. Beni, J. Wang. Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, 1989.
[21] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems. New York: Oxford Univ. Press, 1999.