Data classification by Fuzzy Ant-Miner

Mohamed Hamlich[1] and Mohammed Ramdani[1]

[1] Computer science lab, UH2, FSTM
Mohammadia, BP 146 20650, Morocco

Abstract

In this paper we propose an extension of classification algorithm based on ant colony algorithms to handle continuous valued attributes using the concepts of fuzzy logic. The ant colony algorithms transform continuous attributes into nominal attributes by creating clenched discrete intervals. This may lead to false predictions of the target attribute, especially if the attribute value history is close to the borders of discretization.

Continuous attributes are discretized on the fly into fuzzy partitions that will be used to develop an algorithm called Fuzzy Ant-Miner. Fuzzy rules are generated by using the concept of fuzzy entropy and fuzzy fitness of a rule.

Keywords: *Fuzzy Ant Miner, fuzzy entropy, fuzzy fitness,* discretization on the fly, *classification*.

1. Introduction

Ant miner is a classification algorithm that learns rules from training sets by using the simulation of real ants [9]. Each artificial ant is a mere agent in the system and finds a solution to the problem and communicates with other ants indirectly through the amount of pheromone deposited.

Ant miner is an efficient algorithm with discrete attributes. But with the continuous attributes, the ant colony algorithm uses intervals with strict boundaries. This can lead to false predictions of the target attribute, especially if the attribute's value is close to the borders of discretization.

To solve this problem, we propose an extension of the classification algorithm based on ant colony algorithms to treat continuous attributes using the concepts of fuzzy logic [12]: Continuous attributes are discretized into fuzzy partitions that will be used in a Fuzzy Ant-Miner algorithm.

The originality of this approach is that it generates fuzzy rules by using the concept of fuzzy entropy and fuzzy fitness of a rule. The basic idea of this algorithm is that the probability with which ant chooses a value of a fuzzy continuous attribute among all those possible depends on two things: the amount of pheromone deposited by the ant, and the fuzzy entropy of the value. After obtaining the complete rule, the method proceeds to pruning based on the quality of a fuzzy rule. This provides simplified and easy to interpret rules.

The simplified fuzzy rules obtained will be exploited by using fuzzy inference systems for classification and prediction.

In this paper we first describe the base Ant-Miner algorithm in section 2. In the third section, we present the principle of the cAnt-Miner algorithm which is an extension of Ant Miner. In section (4) we present our contribution: Fuzzy Ant Miner. Section (5) discusses the results obtained and their comparison with those of the original algorithm and its extension cAnt Miner. The conclusion comes in the sixth section.

2. Ant-Miner

Ant-Miner's [9] goal is to extract, from the data set, the classification rules of type: if-then in the form: IF (term 1) AND (term 2) ... AND .. AND (term n) THEN class. Each term in the rule is a triple (attribute, operator, value).

The pseudo code of Ant-Miner [2] shown in Figure 1 begins with an empty list of rules and adds iteratively a rule to this list as long as the number of training examples not covered remains larger than the maximum number specified by the user.

In order to construct rules, the ant starts with a blank rule (no background terms). The terms are chosen with a probability to be added to the current partial rule according to the amount of pheromone and heuristic information (η) associated with these terms:

The probability that the term $A_i = v_j$ is chosen to be added to the current partial rule is given by Eq. (1):

$$P_{ij}(t) = \frac{\tau_{ij}(t) * \eta_{ij}}{\sum_{i}^{n} \sum_{j}^{b_i} \tau_{ij}(t) * \eta_{ij}} \qquad \forall i \in I \qquad (1)$$

A_i : The i^{th} attribute;

v_j : The j^{th} value of the field A_i .

η_{ij} : The heuristic function of the term *ij*

$\tau_{ij}(t)$: Available pheromone quantity (at time t) in the track that will be explored by the ant.

n: Total number of attributes.

b_i : Total number of values in attribute field.

I: represents the attributes that are not yet explored by the ant.

```
TrainingSet = all training examples;
InducedRuleSet = Ø;
WHILE (No. of uncovered examples in the TrainingSet >
        MaxUncoveredExamples)
        i = 0;
        Repeat
          i = i + 1;
          Ant (i) constructs a classification rule Ri;
          Prune the rule c;
          Update pheromone of attribute terms;
        UNTIL (i >= NoAnt) OR (the same rule Ri has been
constructed MaxRulesConverge times);
            Select the best rule among all constructed rules;
            Add the best rule to InducedRuleSet;
            Remove training examples covered by the best
rule from the TrainingSet;
END WHILE
Create default rule;
Output InducedRuleSet;
```

Fig. 1 pseudo-code of Ant-Miner

The normalized heuristic function is given by:

$$\eta_{ij} = \frac{\log_2(k) - H_{ij}}{\sum_{i}^{a}\sum_{j}^{b_i}\log_2(k) - H_{ij}} \qquad (2)$$

The entropy of the term ($A_i = v_j$) is the amount of information associated with this term. It is given by the Eq. *(3)*:

$$H_{ij} = -\sum_{w=1}^{k} p(c = c_w / A_i = v_j) * \log_2(p(c = c_w / A_i = v_j) \qquad (3)$$

k: Number of classes.

The pheromone amount and the heuristic value are associated to each term ($A_i = v_j$) that corresponds to a track which may be followed by an ant. All terms are initialized by the same amount of pheromone:

$$\tau_{ij(t=0)} = \frac{1}{\sum_{i=1}^{n} b_i} \qquad (4)$$

n: Number of attributes.

b_i : Number of values in attribute field.

To update the pheromone amount of this term, Eq. (5) is used:

$$\tau_{ij(t+1)} = \tau_{ij(t)} + \tau_{ij(t)} * Q \qquad (5)$$

Where Q is the rule quality defined by:

$$Q = \frac{TP}{TP + FN} * \frac{TN}{FP + TN} \qquad (6)$$

TP (true positive): Number of examples covered by the rule that have the class predicted by the rule.

FP (false positive): Number of examples covered by the rule that have a different class from that predicted by the rule.

FN (false negative): Number of examples not covered by the rule that have the class predicted by the rule.

TN (true negative)*:* Number of examples not covered by the rule that have a different class from that predicted by the rule.

The higher is Q value; the better is the quality of the rule. The pheromone quantity is iteratively updated according to the quality of the rule established by the ant.

Once the ant has finished constructing the rule, the algorithm attempts to prune it [1]. The goal of this pruning is to eliminate some terms in order to enhance the quality of the rule. So the rule becomes easy to interpret by the user. After the first iteration has taken place, we have the entire rule. Then, we attempt to eliminate the terms of the rule one at a time and we calculate the quality of the resulting rule.

Once the construction of the premise rule is finished, the consequent rule is the most frequent class among those covered by the rule in the learning set of examples.

The process of rule construction is repeated until a user-specified number of iterations is reached, or the best rule of the ongoing iteration is identical to the previous one.

The best rule discovered during that iterative process is added to the list of rules and the learning examples covered by this rule (i.e. that satisfy the best rule) are removed from the learning set.

The major weakness of the Ant-miner algorithm is that it deals with nominal attributes only. Continuous attributes (the most frequent in real application domains) are discretized in a pre-processing step.

3. Extension of Ant-Miner: cAnt-Miner

cAnt-Miner [8] is an extension of Ant-Miner that accepts continuous attributes as well as nominal ones. In a domain of continuous values of an attribute, a certain value x is chosen to dynamically partition the example set into two intervals $A_i < x$ and $A_i >= x$.

The best value of x is the one that minimizes the entropy of the ongoing partition:

$$H(A_i, x) = \frac{|S_{A_i<x}|}{|S|} * H(S_{A_i<x}) + \frac{|S_{A_i \geq x}|}{|S|} * H(S_{A_i \geq x}) \quad (7)$$

Where:

- $S_{A_i<x}$ is the total number of examples in the partition $A_i < x$
- $S_{A_i \geq x}$ is the total number of examples in the partition $A_i >= x$
- $|S|$ is the total number of examples in the learning set.
- The entropy values of $S_{A_i<x}$ and $S_{A_i \geq x}$ are calculated by :

$$H(S) = \sum_{w=1}^{k} - p(c = c_w / S).\log_2 p(c = c_w / S) \quad (8)$$

- p(c/S) is the examples proportion of S which has class c.
- k is the total number of classes.

After the best value has been found using Eq. (7), the method uses Eq. (9) to calculate the entropies of the two partitions ($A_i < x$) and ($A_i >= x$). The partition with the minimum entropy value is the one which will be elected by the ant. The term (A*i, operator, x*) is added to the current partial rule of the ant (for ex. Age<18).

cAnt-Miner partitions the example set into two partitions $Ai<x$ and $Ai>=x$ which does not give a precise estimation of the continuous attribute. Another inconvenience of cAnt-Miner is that it has a crisp discretization: A value close to x can easily switch from a partition to another. Thus this method may generate rules with unpredictable classes for values close to the threshold.

4. New method of fuzzy classification : Fuzzy Ant-Miner

4.1 Extension of cAnt-Miner algorithm

In order to avoid the crisp discretization used by cAnt-Miner, our method extends the crisp partition by discretizing the values of an attribute into fuzzy partitions. First a threshold x is determined in the same way as cAnt-Miner [10] (the one that leads to the minimum entropy). We assume that the threshold is a fuzzy number around x. Then the values of continuous attribute are transformed into membership degrees of the fuzzy values Ai approximately $\leq x$ and approximately $\geq x$.

The continuous attribute's membership function (figure 2) of a fuzzy value "A_i *approximately* $\leq x$" is calculated by:

$$\mu_{A_i \text{ approximately} \leq x}(v) = \begin{cases} 1 & \text{si v} \leq \text{a} \\ \dfrac{b-v}{b-a} & \text{si a}\langle v\langle b \\ 0 & \text{si v} \geq \text{b} \end{cases} \quad (9)$$

And $\mu_{A_i \text{ approximately} \geq x}(v) = 1 - \mu_{A_i \text{ approximately} \leq x}(v)$

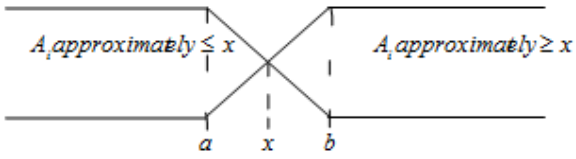

Fig. 2 Fuzzy discretization of a continuous attribute

Where parameters (*a* and *b)* are calculated by:

$$a = x - \frac{\max_{i=1...l}(x_i) - \min_{i=1...l}(x_i)}{\alpha} \quad (10)$$

$$b = x + \frac{\max_{i=1...l}(x_i) - \min_{i=1...l}(x_i)}{\alpha} \quad (11)$$

- $(x_i)_{i=1..l}$ values of attribute in training sets.
- α: parameter to be adjusted for better performance.

Note that other ways of generating the fuzzy partitions [5], [6] can be used in our method.

4.2 Fuzzy Ant-Miner Algorithm

In order to deal with fuzzy concepts we have extended the algorithm Ant-miner to treat these concepts (fig 3):

```
TrainingSet = all training examples;
InducedRuleSet = Ø;
WHILE (No. of uncovered examples in the TrainingSet >
        MaxUncoveredExamples)
        Fuzzy discretization depending on the value α;
        i = 0;
        Repeat
          i = i + 1;
          Ant (i) constructs a fuzzy rule Ri;
          Ant (i) constructs a fuzzy rule class of Ri
          depending on the value β;
          Prune the fuzzy rule c;
        Update fuzzy pheromone of attribute terms;
      UNTIL (i >= NoAnt) OR (the same rule Ri has been
constructed);
          Select the best fuzzy rule;
          Add the best fuzzy rule to InducedRuleSet;
          Remove training examples covered by the best
fuzzy rule from the TrainingSet;
END WHILE
Create default fuzzy rule;
Output InducedRuleSet;
```

Fig. 3 pseudo-code of Fuzzy Ant-Miner

By analogy with Eq. 2, the fuzzy entropy of the fuzzy term $A_i = v_j$ (v_j is approximately $\leq$ or $\geq$ to x) is as follows:

$$H^*_{ij} = Entropy^*(c=c_w / A_i = v_j)$$

$$= -\sum_{w=1}^{k} p^*(c_w / A_i = v_j) * \log_2(p^*(c_w / A_i = v_j)) \quad (12)$$

Where:

- $p^*(c = c_w / A_i = v_j)$ represents the fuzzy probability [13] defined by:
- $$p^*(c_w / A_i = v_j) = \frac{p^*(c_w, A_i = v_j)}{p^*(A_i = v_j)}$$

$$p^*(A_i = v_j) = \sum_{e=1}^{l} \mu_{v_j}(x_e)\, p(x_e) \quad (13)$$

- l: number of examples in the learning set
- μ_{v_j} : is the membership degree of x_e *to* v_j
- $p(x_e)$: is estimated by the frequency of x_e *in the learning set.*

$$p^*(c = c_w, A_i = v_j) = \sum_{e=1}^{l} \min(\mu_{c_w}(c_t), \mu_{v_j}(x_e)) p(c_t, x_e) \quad (14)$$

$p(c_t, x_e)$: is estimated by the frequency of (c_t, x_e) in the learning set.

The fuzzy entropy will be normalized in the same way as the classic entropy *(4)* applied to nominal attributes. The fuzzy heuristic function of the j[th] partition belonging to i[th] attribute is then defined by:

$$\eta^*_{ij} = \frac{\log_2(b_i) - H^*_{ij}}{\sum_{j=1}^{n_p} (\log_2(b_i) - H^*_{ij})} \quad (15)$$

- b_i *: Number of fuzzy values of attribute* A_i

By analogy with (1), the fuzzy probability that an ant choose the fuzzy partition of a continuous attribute among all possible fuzzy partitions is defined by:

$$P_{ij}(t) = \frac{\tau_{ij}(t) * \eta^*_{ij}}{\sum_{i}^{a} \sum_{j}^{b_i} \tau_{ij}(t) * \eta^*_{ij}} \quad (16)$$

4.3 Rule pruning

After obtaining the complete rule, the method proceeds to pruning it. The pruning process is based on the quality of the fuzzy rule. Indeed the quality is computed with TP^* (fuzzy true positive), FP^*(fuzzy false positive), FN^* (fuzzy false negative) and TN^* (fuzzy true negative) that are determined by calculating the membership degree of each term in the rule.

The e[th] example O_e of the training set is defined by:

$$O_e = < (x_i^e)_{i=1..n}, c_{O_e} >$$

Where:

- $(x_i^e)_{i=1....n}$ represents the attributes values of the example.
- c_{O_e} : Class of the example.

Each rule r_m obtained by our method is defined by

$$r_m = < (A_i = v_j)_{i=1..n_r}, c_{r_m} >$$

Where:

- v_j is the fuzzy value

- n_r is the number of terms in the rule.
- c_{r_m} : Class predicted by the best rule obtained.

To determine the quality of the fuzzy rule, the examples are injected into the rule:
The filter $f_{O_e r_m}$ of the example O_e in the rule is defined by the degree membership of a term x_i^e to a fuzzy value v_j and the degree membership of a class example to a class rule:

$$f_{O_e r_m} = <(\mu_{A_i=v_j}(x_i^e))_{i=1..n_r}, \mu_{C_{r_m}}(c_{O_e})>$$

A fuzzy true positive value of example O_e is calculated by:

$$TP^*_{O_e} = \min_{i=1,..l}[\mu_{A_i=v_j}(x_i^e)] * \mu_{C_{r_m}}(c_{O_e}) \quad (17)$$

A TP^* value of a training set is:

$$TP^* = \sum_{e=1}^{l} TP^*_{O_e} \quad (18)$$

Similarly we calculate FP^*, TN^* and FN^*:

$$FP^*_{O_e} = \min_{i=1,..l}[\mu_{A_i=v_j}(x_i^e)] * \mu_{\bar{C}_{r_m}}(c_{O_e}) \quad (19)$$

Where $\mu_{\bar{C}_{r_m}}(c_{O_e}) = 1 - \mu_{C_{r_m}}(c_{O_e})$

$$FP^* = \sum_{e=1}^{l} FP^*_{O_e} \quad (20)$$

$$FN^*_{O_e} = \max_{i=1,..l}[\mu_{\overline{A_i=v_j}}(x_i^e)] * \mu_{C_{r_m}}(c_{O_e}) \quad (21)$$

Where $\mu_{\overline{A_i=v_j}}(x_i^e) = 1 - \mu_{A_i=v_j}(x_i^e)$

$$FN^* = \sum_{e=1}^{l} FN^*_{O_e} \quad (22)$$

$$TN^*_{O_e} = \max_{i=1,..l}[\mu_{\overline{A_i=v_j}}(x_i^e)] * \mu_{\bar{C}_{r_m}}(c_{O_e}) \quad (23)$$

$$TN^* = \sum_{e=1}^{l} TN^*_{O_e} \quad (24)$$

The fuzzy quality Q^* is then calculated by:

$$Q^* = \frac{TP^*}{TP^* + FN^*} * \frac{TN^*}{FP^* + TN^*} \quad (25)$$

Before pruning the rule, the class assigned to it is the most common among those covered by this rule in the training set. Each example O_e covers the rule if $TP^*_{O_e}$ Eq .*(17)* is higher than a parameter β . This parameter is adjusted for better performance.

In each rule pruning iteration, every term is temporarily removed from the rule (note that a change in the antecedent rule may result in a change of the consequent rule), and the quality of the rule is re-evaluated. At the end of the iteration, only the term removal of which leads to an improvement of the rule quality is actually removed. The process of rule pruning stops when there is only one term left, or each term removal no longer leads to the improvement of a rule quality. Once the rule pruning has been performed, the artificial ant increases the level of pheromone of the terms still presented in the rule antecedent according to the fuzzy quality of the rule:

$$\tau^*_{ij(t)} = \tau^*_{ij(t)} + \tau^*_{ij(t)} * Q^* \quad (26)$$

5. Results and discussions

In order to confirm our approach, a comparative study has been done. In that study we have tested our method against standard databases obtained from UCI data repository [16].
All tests have been performed with 3 folds cross-validation. Fuzzy rules generated by Fuzzy Ant-Miner from the learning database are exploited to classify new examples. This classification is done according to the principle of fuzzy inference. In fact each generated rule involving a fuzzy term deriving from the continuous attribute, has a degree of validity that depends on fuzzy probabilities. By consequence, an example filtered by this rule will have that rule's class with a degree of uncertainty which is calculated using the rule of the generalized modus ponens. Parameters values α and β are varied to obtain better performance.

The comparison of our results with those obtained by Ant-miner and by cAnt-miner has shown that our method gives promising results of the predictive accuracies especially for Wisconsin breast cancer (α =40) and diabetes (α =18) databases Tab .1. For Tic-tac-toe and Ljubljana breast cancer, predictive accuracies are not influenced by parameters α and β because their attributes are nominal.
The number of terms per rule Tab .2 is lower in our algorithm and easy to interpret by user.

Table 1: Predictive accuracy (mean ± standard deviation)

Dataset	Ant-Miner	cAnt-Miner	Fuzzy Ant-Miner
Tic-tac-toe	73.04±2.53	73.52±2.81	73.64±2.21
Diabetes	72,84	-	75,03±2,60 (α =18, β=0.7)
Wisconsin breast cancer	96.04 ± 2.80	96.84±2.13	98.05±2.62 (α =40, β=0.6)
Ljubljana breast cancer	75.28±2.24	75.91 ±8.38	78,85

Table 2: Number of terms per rule

Dataset	Ant-Miner	cAnt-Miner	Fuzzy Ant-Miner
Tic-tac-toe	1.18	1.17	1.10
Wisconsin breast cancer	1.97	1.09	1.25 (α =40, β=0.6)
Ljubljana breast cancer	1.28	1.06	1.00

6. CONCLUSION

In this paper we have proposed an extension to the learning algorithms based on ant colonies in order to process continuous valued attributes. These parameters are discretized on the fly according to the concepts of fuzzy logic. Our method takes under account the fuzzy quality of a rule and the fuzzy entropy of a term during the process of rule construction by ants. The fuzzy rules obtained are exploited according the principle of fuzzy inference.

The results obtained with fuzzy Ant-Miner showed an obvious improvement compared to those of Ant-Miner and cAnt-Miner.

References

[1] G A.Chan ,and A.Freitas, "A new classification-rule pruning procedure for an ant colony algorithm", Lecture Notes in Artificial Intelligence 2005, 3871 25–36.

[2] P.Clark, and T.Niblett, "The CN2 rule induction algorithm". Machine Learning, 1989, 3(4) 261–283

[3] M.Dorigo,and T.Stutzle, "Ant Colony Optimization", MIT Press, 2004.

[4] A.Freitas, R.Parpinelli, and H.Lopes, "Ant colony algorithms for data mining", Sci. & Tech. 2nd Ed, 2008.

[5] H.Liu, F.Hussain, C.Tan, and M.Dash, "Discretization: An enabling technique", Data Mining and Knowledge Discovery 2002, 6 393–423.

[6] C.Marsala, «Apprentissage inductif en présence de données imprécises : Construction et utilisation d'arbres de décision flous », Thèse de doctorat, Université Paris 6, 1998, pages 83-115.

[7] D.Martens, M.Backer, R.Haesen, J.Vanthienen, , M.Snoeck, and B.Baesens, "Classification with ant colony optimization", IEEE Transactions on Evolutionary Computation 2007, 11(5) 651–665.

[8] F.Otero, A.Freitas, and C.G.Johnson, "cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes, in Ant Colony Optimization and Swarm Intelligence", LNCS 5217, 2008 Springer, pp. 48–59.

[9] R.Parpinelli, H.Lopes, and A.Freitas, "Data mining with an ant colony optimization algorithm," IEEE. Transactions on Evolutionary Computation, 2002, vol. 6, no. 4, pp. 321–332.

[10] J.R.Quinlan, "Bagging, boosting, and c4.5", in Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press and the MIT Press, pp. 725-730, 1993.

[11] M.Ramdani, « Système d'induction formelle à base de connaissances imprécises », Thèse de doctorat, Université Paris 6, 1994.

[12] Zadeh.A.Lotfi, "Fuzzy sets. Information and control", 1965, 8:338-353;

[13] Zadeh.A.Lotfi, "Probability measures of fuzzy events", Journal Math. Anal. Applic., (1968), 23.

[14] M.Hamlich, and M.Ramdani, «Nouvelle méthode de classification des données médicales par algorithmes de colonies de fourmis », Première Journée de l'Informatique Décisionnelle (JID'10), El Jadida 11 Mars 2010.

[15] M.Hamlich,and M.Ramdani, "Fuzzy classification method by ant colonies", International Conference on Discrete Mathematics & Computer Science DIMACOS'11, 2011, pp. 69.

[16] UCI Machine learning Repository, http://archive.ics.uci.edu/ml/datasets.html.

Mohamed Hamlich is affiliated with Computer science Lab. of FST, Mohammadia, Morocco. His research interests include Data Classification, knowledge extraction and artificial intelligence. He affords scientific advice to a group of research in cardiology at CHU, University Hassan II, Casablanca, Morocco.

Mohammed Ramdani is affiliated with Computer science Lab. of FST, Mohammadia, Morocco. His research interests include Fuzzy logic, Data Classification, knowledge extraction and artificial intelligence.