Extensions of Ant-Miner Algorithm to Deal with Class Imbalance Problem

Murilo Zangari, Wesley Romão, and Ademir Aparecido Constantino

Universidade Estadual de Maringá, Brazil
{murilo.zangari,wesley.uem,ademir.uem}@gmail.com

Abstract. A database has class imbalance when there are more cases of one class then the others. Classification algorithms are sensitive of this imbalance and tend to valorize the majority classes and ignore the minority classes, which is a problem when the minority classes are the classes of interest. In this paper we propose two extensions of the Ant-Miner algorithm to find better rules to the minority classes. These extensions modify, mainly, how rules are constructed and evaluated. The results show that the proposed algorithms found better rules to the minority classes, considering predictive accuracy and simplicity of the discovered rule list.

Keywords: Data Mining, classification, class imbalance, Ant-Miner.

1 Introduction

Database with class imbalance problem is one of the relatively new problems that emerged when machine learning matured from an embryonic science to an applied technology, broadly used in the worlds of business, industry and scientific research [4]. This increase in interest gave rise to two workshops held in 2000 at AAAI (Association of the Advancement of Artificial Intelligence) conference [2] and other in 2003 at ICML (International Conference on Machine Learning) conference [3] and a special edition of ACM SIGKDD Explorations in 2004 [4] [5] [6].

In the classification task, a database is deemed imbalanced when there are many more cases of some class then the others. Many classification algorithms (e.g., C4.5) tend to ignore the minority classes. In real-world scenarios, the ratio of the small to the large classes can be drastic such as 1 to 100, 1 to 1.000, 1 to 10.000 and sometimes even greater [4] [5] [7].

Several techniques to solve the class imbalance problem have been proposed both at the data and algorithmic levels. At data levels, these solutions include many forms of re-sampling [4] [5] [6] [8]. At the algorithmic level, solutions include adjusting the cost of the various classes so as to counter the class imbalance or learning from one class [3] [4].

This paper proposes two extensions to the Ant-Miner classification algorithm [1] that address the class imbalance issue, thus finding better rules to minority classes with a better predictive accuracy and comprehensibility.

H. Yin et al. (Eds.): IDEAL 2012, LNCS 7435, pp. 9–18, 2012.

2 Related Work

The Ant-Miner [1] is a classification algorithm that aims to solve the classification task of Data Mining. This algorithm is based on the ACO (ant colony optimization) metaheuristic [10] [11], in which each ant incrementally builds/modifies a solution to a certain problem, in this case the problem is the classification rule discovery.

Parpinelli et al. [1] evaluated the Ant-Miner with six databases from UCI (University of California at Irvine) [19]. The algorithm was compared with other classification algorithms and achieves good results. The results showed that the Ant-Miner is a promising technique for classification rule discovery. Several researches and modifications have been proposed to improve efficiency: Ant-Miner2 [12], Ant-Miner3 [13], Ant-Miner+ [14], cAnt-Miner [15], Multiple pheromone types Ant-Miner [17].

3 The Algorithms

Most of Ant-Miner extensions aim to improve the predictive accuracy. In this paper we present two Ant-Miner extensions that aim to find better rules when applied on databases with class imbalances. Such algorithm should find rules to minority classes, since classification algorithms are sensitive of this kind of imbalance and tend valorize the majority classes, which is a problem when the minority classes are the interest classes. Both are developed in Java on Eclipse development environment. The only difference between the two algorithms proposed is how the rule quality is calculated.

3.1 The Ant-MinerCI Algorithm

The Ant-MinerCI (Ant-Miner Class Imbalance) receives databases in the ARFF file format, the same format used in Weka [18]. Figure 1 shows the pseudo-code of the Ant-MinerCI algorithm.

```
Ant-MinerCI:
TrainingSet = {all the training cases};
DiscoveryRuleList = []; //initialized with an empty list
Calculate the heuristic value for each term
<attribute=value> related to interesting_class;
WHILE (TrainingSet > Max_uncovered_cases)
  i = 1; //ant index
  j = 1; //convergence test index
  Initialize all terms with the same amount of pheromone;
  REPEAT
    IF (First iteration of WHILE)
      Ant_i add the interesting class on the consequent;
    END IF
    WHILE (No. of antec. on rule Ant_i ≤ max_num_antec)
      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;
    END WHILE
```

Fig. 1. Pseudo-code *Ant-MinerCI*