Expert Systems with Applications 41 (2014) 2688–2702

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Applying Ant Colony Optimization to configuring stacking ensembles for data mining

YiJun Chen, Man-Leung Wong *, Haibing Li

Department of Computing and Decision Sciences, Lingnan University, Tuen Mun, Hong Kong

ARTICLE INFO

Keywords:
ACO
Ensemble
Stacking
Metaheuristics
Data mining
Direct marketing

ABSTRACT

An ensemble is a collective decision-making system which applies a strategy to combine the predictions of learned classifiers to generate its prediction of new instances. Early research has proved that ensemble classifiers in most cases can be more accurate than any single component classifier both empirically and theoretically. Though many ensemble approaches are proposed, it is still not an easy task to find a suitable ensemble configuration for a specific dataset. In some early works, the ensemble is selected manually according to the experience of the specialists. Metaheuristic methods can be alternative solutions to find configurations. Ant Colony Optimization (ACO) is one popular approach among metaheuristics. In this work, we propose a new ensemble construction method which applies ACO to the stacking ensemble construction process to generate domain-specific configurations. A number of experiments are performed to compare the proposed approach with some well-known ensemble methods on 18 benchmark data mining datasets. The approach is also applied to learning ensembles for a real-world cost-sensitive data mining problem. The experiment results show that the new approach can generate better stacking ensembles.

1. Introduction

Over years of development, it has become more and more difficult to improve significantly the performance of a single classifier. Recently, there has been growing research interest in the method to combine different classifiers together to achieve better performance. The combining method is referred to as Ensemble. In early research, ensembles were proved empirically and theoretically to perform more accurately than any single component classifier in most cases. If an ensemble is generated by a set of classifiers which are trained from the same learning algorithm, this ensemble is a homogeneous ensemble. If an ensemble is generated by a set of classifiers, which are trained from different learning algorithms, this ensemble is a heterogeneous ensemble (Dietterich, 2000). For example, Bagging (Breiman, 1996) and Boosting (Schapire, 1990) are homogeneous ensembles, while stacking (Wolpert, 1992) is a heterogeneous ensemble.

To generate an ensemble to achieve expected results, two important things should be considered carefully. The first is to introduce enough diversity into the components of an ensemble. The second is to choose a suitable combining method to combine the diverse outputs to a single output (Polikar, 2006). The diversity is the foundation of an ensemble. However, as the diversity increases, the marginal effect decreases after a certain threshold. The memories and computing cost increase significantly while the performance does not improve steadily. For early Bagging and Boosting methods, the diversity is achieved by using the resample strategy. The classifiers included in Bagging are trained with the data subsets, which are randomly sampled from the original dataset. A majority voting scheme is applied as the combining method to make a collective decision. Boosting uses a weighted resample strategy. The weights of all instances are initialized equally. If an instance is misclassified, its weight will be increased. Thus it will be more likely to select the misclassified instances into the next training subset. The diversity generating process stops when the errors are too small. The combining scheme of Boosting is a weighted majority voting. Compared to Bagging and Boosting, stacking does not manipulate the training dataset directly. Instead, an ensemble of classifiers is generated based on two levels. In the base level, multiple classifiers are trained with different learning algorithms. The diversity is introduced because different learning algorithms make different errors in the same dataset. A meta-classifier is applied to generate the final prediction. The meta-classifier is trained with a learning algorithm using a meta-dataset which combines the outputs of base-level classifiers and the real class label.

One problem of stacking is how to obtain an "appropriate" configuration of the base-level classifiers and meta-classifier for each

* Corresponding author. Tel.: +852 26168093.
E-mail addresses: yijunchen@ln.edu.hk (Y. Chen), mlwong@ln.edu.hk (M.-L. Wong), haibingli@ln.edu.hk (H. Li).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.10.063

domain-specific dataset. The number of base-level classifiers and the kinds of learning algorithms are closely related to the diversity. The kind of meta-classifier is also important to the fusion of the base-level classifiers. However, such configuration is still "Black Art" (Wolpert, 1992). Some researchers have proposed different methods to determine the configuration of stacking. Ting and Witten solved two issues about the type of meta-classifier and the kinds of its input attributes (Ting & Witten, 1999). Džeroski and Ženko introduced Multi-Response Model Trees as the meta-classifier (Džeroski & Ženko, 2002). Zheng and Padmanabhan (2007) and Zhu (2010) proposed their Data Envelopment Analysis (DEA) approaches respectively. Ledezma et al. and Ordóñez et al. proposed approaches which search the ensemble configurations using Genetic Algorithms (GAs) (Ledezma, Aler, & Borrajo, 2002; Ordóñez, Ledezma, & Sanchis, 2008).

In this work, we propose an approach using Ant Colony Optimization (ACO) to optimize the stacking configuration. ACO is a meta-heuristic algorithm which is inspired by the foraging behaviour in real ant colonies. Some approaches were proposed recently to apply ACO in data mining. Parpinelli et al. proposed Ant Miner to extract classification rules (Parpinelli, Lopes, & Freitas, 2002). Some approaches apply ACO in feature subset selection tasks (Al-Ani, 2006; Zhang, Chen, & He, 2010).

The rest of this paper is organized as follows. In Section 2, the background of this work, including the related ensemble approaches and the Ant Colony Optimization method, is introduced. In Section 3, the details of our approach are presented. In Section 4, a number of conducted experiments are described to compare our approach with other ensemble methods. Further, the experiment results are presented and discussed in this section. In Section 5, our approach is applied to solve a real-world data mining problem. In the last section, a conclusion is given.

2. Background

2.1. Ensembles

2.1.1. Bagging

Bagging, short for bootstrap aggregating, is considered one of the earliest ensemble scheme (Breiman, 1996). Bagging is intuitive but powerful, especially when the data size is limited. Bagging generates a series of training subsets by random sampling with replacement from the original training set. Then the different classifiers are trained by the same classification algorithm with different training subsets. When a certain number of classifiers are generated, these individuals are combined by the majority voting scheme. Given a testing instance, different outputs will be given from the trained classifiers, and the majority will be considered as the final decision.

A Random Forest is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman, 2001). Random Forest can be considered a special type of Bagging.

2.1.2. Boosting

In 1990, Schapire's weak learning framework was proposed (Schapire, 1990). An elegant algorithm, Boosting, which boosts any given weak learners to a strong learner was also provided in this work.

Boosting also applies re-sampling of training data set and majority voting. However, Boosting does not treat all the instances equally, but focuses on the more informative instances which are important to the classification decision. The algorithm generates three classifiers using the same weak learner. The first learner C_1 is trained with a random subset of the training set. The second learner C_2 is trained with a more informative dataset by iteratively flipping a fair coin to decide which instances to add. If a head comes up, some samples are selected from the training set and presented to C_1 until an instance is misclassified by C_1. This instance is added to the training set of C_2. If a tail comes up, a similar process is conducted whereas the first correctly classified instance is selected. The third learner C_3 is trained with the instances which are differently classified by C_1 and C_2 by filtering the whole training set. Finally, a three-way majority voting scheme is used to combine the three classifiers.

AdaBoost is a popular variation of the original Boosting scheme (Freund & Schapire, 1997). AdaBoost maintains a weighted distribution of instances, trains a series of classifiers of the same weak learner with different instances drawn according to the distribution and finally combines the weak learners through a weighted majority voting scheme to generate the final decision. At the beginning of the process, all the instances are initialized with the same weight. For each training iteration, a training subset is drawn from the instances distribution D_t. Then the classification error of this weak learner is calculated and used in changing the weight updating parameter α_t to manipulate the sample distribution to enlarge the probabilities of the currently misclassified instances to be used in the next training iteration. After the weight updating and normalization, the new instances distribution D_{t+1} is generated. α_t is also used as the weight of the weak learner in the weighted majority voting procedure. Some variations of AdaBoost, such as AdaBoost.M1 and AdaBoost.R, have been proposed (Freund & Schapire, 1996, 1997).

2.1.3. Stacking

In the previous ensemble schemes, the individual weak learners are the same. On the other hand, stacking has a two-level structure: level-0 (base-level) classifiers and a level-1 (meta) classifier (Wolpert, 1992). The base-level classifiers are trained with the training set and generate their predictions. Then the meta-classifier is trained with the meta-data to map the outputs of the base-level classifiers to the actual class label. The meta-data could be $((y_i^1, y_i^2, \ldots, y_i^m), y_i)$, where y_i^m means the prediction given by the m^{th} base-level classifier on the i^{th}instances, and y_i is the actual class label. During the process of classifying a new instance, the trained base-level classifiers will give their individual predictions, and the predictions will be considered as the input of the meta-classifier to generate the final decision.

GA-Ensemble was proposed by Ordóñez et al. as an extension of their previous approach (Ordóñez et al., 2008). GA-Ensemble applies a genetic algorithm in searching the configurations according to different datasets without *a priori* assumptions. At the beginning, a set of candidate base-level classifiers is trained to generate a pool of base-level classifiers thus to improve the efficiency without losing accuracy. The candidate set must be encoded in a chromosome, which represents a potential configuration. Binary encoding is used to accompany the canonical GA, where a 0 in the gene means that the classifier of this gene will not be used in the configuration and a 1 means the classifier will be used. The last gene in a chromosome represents two different stacking combining schemes: multi-response model tree or majority voting. This GA search process will iterate for several generations. For each generation, the classification accuracies on validation sets are used as the fitness values to evaluate the chromosomes. Some elite chromosomes will be kept for the next generation and some poor ones will be eliminated. Mutation and crossover operations will be applied to some chromosomes to generate new chromosomes. After all generations are finished, the best chromosome will be chosen as the final configuration.