Optimizing Stacking Ensemble by an Ant Colony Optimization Approach

Yijun Chen
Dept. of Computing and Decision Sciences
Lingnan University
Tuen Mun, Hong Kong
yijunchen@ln.edu.hk

Man Leung Wong
Dept. of Computing and Decision Sciences
Lingnan University
Tuen Mun, Hong Kong
mlwong@ln.edu.hk

ABSTRACT

An ensemble is a collective decision making system which applies some strategy to combine the predictions of classifiers to generate its prediction on new instances. Stacking is a well-known approach among the ensembles. It is not easy to find a suitable ensemble configuration for a specific dataset. Ant Colony Optimization (ACO) is a popular metaheuristic approach which could be a solution to find configurations. In this work, we propose a new Stacking construction method which applies ACO in the Stacking construction process to generate domain-specific configurations. The experiment results show that the new approach can achieve promising results on 18 datasets compared with some well-known ensemble approaches.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

ACO, Ensemble, Stacking, Metaheuristics

1. ACO-STACKING APPROACH

In an ACO-Stacking construction task, given the base classifiers and meta classifiers, the approach selects a configuration which contains a subset of the base classifiers combining with a meta classifier to achieve the best performance. Prior to the execution of the major process of ACO-Stacking, the pool C of base classifiers is generated, which contains m classifiers generated by the learning algorithms, $C = \{c_1, \cdots, c_m\}$. For each classifier c_i, its local information η_i is initialized from a pre-test on the whole training set. The metric: *precision* of each classifier from the pre-test is selected as the local information η_i in this approach and it would be kept during the searching process. There are k ants in the colony and each one is given a learning algorithm as its meta-combining scheme. Thus each ant is a Stacking configuration. S_j represents the configuration constructed by the j^{th} ant, $j \leq k$. After all these settings are finished, the ACO searching process will iteratively execute. In the first iteration, each ant is initialized with a base classifier randomly and the accuracy α_{S_i} of this configuration is calculated on an independent validation set. In the following iterations, when the j^{th} ant begins its search, it selects a classifier *c'* from the pool C to its current configuration S_j by using the products of the pheromone and the local information of the ants. The possibility of a classifier c_i to be selected by the j^{th} ant is given in equation 1.

$$p_i = \begin{cases} \frac{\mu_i * \eta_i}{\sum_{t=1, c_i \notin S_j}^{m} \mu_t * \eta_t} & \text{if } \mathrm{c_i} \notin \mathrm{S_j}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Using the roulette wheel selection of the possibilities, c_i is selected and a new configuration S'_j is generated, where $S'_j = S_j \cup c_i$. Then S'_j is tested on the same validation set. If the $\alpha_{S'_j}$ is better than α_{S_j}, it will replace S_j and the ant continues to generate a new S'_j by using the same strategy. If S'_j cannot improve α_{S_j}, this ant keeps the current configuration and its search is frozen in the iteration. The next ant in the colony starts its searching until all ants finish their search. During the process, once a classifier c_i is chosen and successfully added to any S_j to generate a new S'_j, the pheromone of c_i will accumulate. The improvement of accuracy from S_j to S'_j is used to update the pheromone of c_i. The update rule is given in equation 2.

$$\mu'_i = \mu_i * (1 - \tau) + CC * \mu_i * \frac{\alpha_{S'_j} - \alpha_{S_j}}{\alpha_{S_j}} \quad (2)$$

where CC is a constant number and τ is the evaporation rate. τ and CC are introduced to adjust the historical knowledge and the current knowledge.

After all iterations finish, the best configuration S_{best} of the k ants will be chosen as the final Stacking configuration. The parameters of ACO-Stacking are given in Table 1.

Table 1: Parameters of ACO-Stacking and GA-Ensemble

ACO-Stacking	Value	GA-Ensemble	Value
Colony Size	30	Population Size	30
Iterations	10	Generations	10
Evaporation Rate	0.1	Elite Rate	0.1
CC	10	Cull Rate	0.1
		Crossover Operation	Uniform
		Crossover Rate	0.5
		Mutation Rate	0.1

2. EXPERIMENTS AND RESULTS

The experiments are conducted in the Waikato Environment for Knowledge Analysis (WEKA) [6]. 18 datasets

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

Table 2: The classification accuracies of the ensembles

Dataset	Bagging	AdaBoost	Random Forest	StackingC	GA-Ensemble	ACO-Stacking
Balance-Scale	71.68	76.48	76.96	86.08	92.44	**98.56**
Breast-W	95.1359	96.4235	95.9943	**97.2818**	96.1373	95.1359
Chess	99.437	**99.499**	98.905	99.437	99.1865	99.343
Colic	67.9348	70.9239	71.4674	64.1304	75	**76.9022**
Credit-A	86.3768	84.3478	84.3478	**86.8116**	85.8116	82.3188
Credit-G	74.0	69.6	74.1	74.7	73.8	**75.0**
Glass	73.8318	**79.4393**	73.3645	69.1589	71.9626	76.1682
Heart-C	78.8779	76.8977	79.2079	**84.1584**	78.8779	74.5875
Heart-Statlog	80.0	80.3704	78.1481	**84.1584**	80.7407	75.9259
Hepatitis	83.2258	85.8065	80.6452	81.9355	83.871	**87.7419**
Ionosphere	**93.4473**	93.1624	**93.4473**	90.8832	91.453	89.1738
Iris	95.3333	93.3333	95.3333	95.3333	**96.0**	**96.0**
Labor	84.2105	**89.4737**	87.7193	**89.4737**	84.2105	87.7193
Lymphography	79.0541	81.0811	81.0811	83.1081	81.0811	**85.8108**
Sonar	74.5192	77.8846	80.7692	81.7308	85.0962	**87.9808**
Vehicle	76.5957	76.2411	**77.0686**	74.1135	75.8865	74.2317
Vote	96.3218	95.8621	95.8621	**96.7816**	94.9425	94.2529
Wine	94.9438	96.6292	97.191	96.0674	97.7528	**98.3146**
w/t/l	10/1/7	8/0/10	10/1/7	10/0/8	10/1/7	-
RAI	70.54%	32.95%	35.13%	21.4%	2.59%	-

from the UCI machine learning repository [4] are used. The datasets are **Balance-Scale**, **Breast-W**, **Chess**, **Colic**, **Credit-A**, **Credit-G**, **Glass**, **Heart-C**, **Heart-Statlog**, **Hepatitis**, **Ionosphere**, **Iris**, **Labor**, **Lymphography**, **Sonar**, **Vehicle**, **Vote** and **Wine**. The ten-fold cross validation scheme is used in the experiments.

2.1 Learning Algorithms

Ten different learning algorithms in WEKA are used to generate base classifiers. The algorithms are **Naive Bayes (NB)**, **Logistic**, **IB1**, **IBk (k =5)**, **KStar**, **OneR**, **PART**, **ZeroR**, **Decision Stump** and **C4.5 Decision Tree (DT)**. The details of the algorithms could be found in [6]. These algorithms are also used as the meta classifier candidates for ACO-Stacking.

2.2 Compared Approaches

In the experiments, ACO-Stacking is compared with the following ensemble methods.
AdaBoost [5] with C4.5 DT as its learning algorithm;
Bagging [1] with C4.5 DT as its learning algorithm;
Random Forest [2];
StackingC [3] with NB, IBk and C4.5 DT as its base classifiers and Multi-Response Model Tree (MRMT) as the meta classifier;
GA-Ensemble [7]. The pool of base classifiers of GA-Ensemble is the same as that of ACO-Stacking. The meta-combiner is either a MRMT or a majority voting scheme. The parameters of GA-Ensemble are listed in Table 1.

2.3 Experiment Results

Table 2 summarized the accuracies of the approaches on the datasets and two empirical test results. The first one is the *w/t/l* test, where *w* means ACO-Stacking outperforms the other approach, *t* means their performances are the same and *l* means ACO-Stacking is not as good as the other approach. It can be observed that ACO-Stacking outperforms Bagging, Random Forest, StackingC and GA-Ensemble. On the other hand, ACO-Stacking outperforms AdaBoost in only eight datasets while it loses in the other ten datasets.

The second empiricical test, Relative Improvement (RAI), is conducted to evaluate different approaches. The RAI is calculated by using the equation 3

$$p = \sum \frac{\alpha_i - \alpha_i'}{\alpha_i} \tag{3}$$

where α_i refers to the accuracy of ACO-Stacking in the i^{th} data set and α_i' refers to the accuracy of the approach being compared with.

From the test results in Table 2, ACO-Stacking gains improvement over all the other approaches.

3. CONCLUSIONS

From the experiments and empirical tests, ACO-Stacking outperforms Bagging, AdaBoost, Random Forest and StackingC and is slightly better than GA-Ensemble. In summary, ACO-Stacking is a promising approach and it will be modified to improve its performance.

4. REFERENCES

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] L. Breiman. Random forest. *Machine Learning*, 45(1):5 – 32, Oct 2001.

[3] S. Džeroski and B. Ženko. Stacking with multi-response model trees. In *International workshop on multiple classifier systems*, volume 2364, pages 201–211. Springer, June 2002.

[4] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[5] Y. Freund and R. E. Schapire. Desicion-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Science*, 55(1):119–139, 1997.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10 –18, 2009.

[7] F. J. Ordóñez, A. Ledezma, and A. Sanchis. Genetic approach for optimizing ensembles of classifiers. In *Proceedings of the Twenty-First International FLAIRS Conference*, pages 89–94, 2008.