Multi criteria decision methods for boosting CBR agents with genetic algorithms

Beatriz López, Carles Pous, Pablo Gay, Albert Pla
University of Girona, Campus montilivi, edifice P4, Girona, Spain
{beatriz.lopez, carles.pous}@udg.edu, {pgay,apla}@eia.udg.edu

ABSTRACT

In this paper we use genetic algorithms to learnt weights in a boosting scenario where several case-based reasoning agents cooperate. In order to deal with the genetic algorithm results, we propose several multi-criteria decision making methods. We experimentally test the methods proposed in a breast cancer diagnosis domain.

Keywords

Ensemble Learning, Case-Based Reasoning, Genetic Algorithms, Multicriteria Decision Making

1. INTRODUCTION

Ensemble learning has been a matter of concern in the last recent years because of its benefits on reducing the bias-variance of classifiers. Bagging, boosting and staging are three very well known ways of addressing this relatively new way of learning. Bagging assigns randomly to each learner a set of examples, so the construction of complementary learners is left to the chance and to the unstability of the learning methods. Boosting actively seek to generate complementary base learners, on the basis of the methods of the previous learners. Staking deals with the combination of models of different algorithms [18, 13].

Ensemble learning has been recently applied to multi-agent systems, so that several learning agents collaborate in a distributed environment. For example, in [11] the authors propose several ensemble schemas for cooperative case-based learners.

The usual way in which bagging and boosting integrate the different learners is under a weighted voting schema. Therefore, the key issue is the weight assigned to each agent. AdaBoost [3] is one of the best known learning algorithms for this purpose, having today a lot of variants. More recently, genetic algorithms (GAs) have also been applied [14], but mainly in non-multi-agent environments. Our research is related to extend the application of genetic algorithms, for boosting purposes, in a multi-agent environment where each classifier is linked to a given agent. Under this perspective, our multi-agent system approaches to trust learning as in [5] and [1].

Cite as: Multi criteria decision methods for boosting CBR agents with genetic algorithms, Lopez, Pous, Gay and Pla, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

In particular, in our approach each classifier follows case-based reasoning (CBR) method, so we are dealing with CBR agents. Moreover, since we are actively seeking for the complementary of learners through a GA, we say that we are boosting CBR agents.

According to the genetic algorithm theory, several runs are required in order to deal with the randomness involved in this kind of algorithms [9]. Thus, two runs with different random-number seeds will generally produce different detailed behaviours. But finally, a single weight should be assigned to a boosting agent. In this paper, we present several alternatives for obtaining this single weight from the outcomes of the different genetic algorithm runs. Our methods have been applied in a breast cancer diagnosis domain, and we show the different results obtained.

This paper is organised as follows. First we introduce the boosting schema in which our CBR agents cooperate. Next, we describe the GA we propose and the methods to manage the outcomes of the different runs. We continue by providing the information about the application domain we are working on and the results obtained in it. Finally, some related work is highlighted and some conclusion and discussion is provided.

2. BOOSTING CBR AGENTS

Our multi-agent system (MAS) consists of n case-based agents that cooperate for solving a problem. Each agent provides its advise or solution about a case, and a coordinator makes a final decision based on a weighted voted schema.

Each agent is trained with a set of examples; however, each agent receives only a part of the examples (as in [10]). That is, if a case is composed by A attributes, each agent receives a subset of it, $A_1, \ldots, A_n \subset A$, with $A_i \bigcap A_j = \emptyset$. This partition correspond to the different agent specialization. For example, in a medical domain, A_1 attributes could correspond to the epidemiological analysis, A_2 to the radiological data, and so on, representing each subset the particular unit in a hospital in which the data have been gathered. Thus, each agent is specialised in a particular field of knowledge of the domain.

Since there is a coordinator agent in charge of dealing with cooperation issues, the system is centralised. The coordinator agent keeps a $weight_i$ on each agent according to the performance provided by the agent. This weights are learned according to the method proposed in this paper in section 3.

When a new case C needs a solution, the coordinator agent broadcasts the case to the CBR agents. CBR agents compute internally a solution for the case following a case-based reasoning process. Next, the CBR agents reply to the coordinator with a tuple containing the class to which the case belongs according to its case-base, and the confidence degree on the solution it has computed. That is, $a_i =< class_i, \delta_i >$, where a_i is the answer of the i agent; $class_i$ the class provided by the agent; and δ_i is a confidence value in [0,1], where 1 means high confident. We are currently considering a diagnosis environment, so only two class values are under evaluation: 1 (positive diagnosis or illness) and 0 (negative diagnosis or healthy).

Afterwards, the coordinator agent combines the different answers in order to find information regarding the positive diagnostic according to the following expression:

$$v = \frac{\sum_{i=1}^{n} class_i * \omega_i}{\omega_i} \tag{1}$$

where n is the number of agents; and ω_i is a combination of the weight of the i agent and δ_i, such that $\omega_i = f(weight_i, \delta_i)$. The f function can be any, as for example the multiplication.

Then, if v is over a given threlhold τ, the final answer of the multi-agent system is positive. Otherwise, negative. This decision procedure follows the reuse step of a case-based system as explained in [12]. See also [6] for further details on the boosting CBR MAS system.

3. MCDM FOR GA RESULTS

The method we propose to learn the agents weight has two phases: genetic algorithm learning and a multi-criteria decision process.

3.1 Genetic algorithm

A genetic algorithm (AG) consists on the following steps [9]:

1. Start with a randomly generated population of chromosomes
2. Calculate the fitness of each chromosome in the population
3. Repeat
   (a) Select a pair of chromosomes from the current population
   (b) With a probability p_c cross over the pair to form two offsprings
   (c) With a probability p_m mutate the two offsprings
4. Replace the current population with the new one
5. Goto step 2 until some ending condition holds..

When applying genetic algorithms to learn the weights in a boosting CBR agents scenario, the key issues are how to represent chromosomes and how to define the fitness function. Particularly, we have defined the chromosome as an array of n values; each value represents the weight of an agent. On the other hand, the fitness of a chromosome is a function of the error of the boosting CBR system it codifies when applied to a data set of examples. So the chromosome is translated to the corresponding boosting CBR MAS, it is run for a given set of examples, and an averaged error over

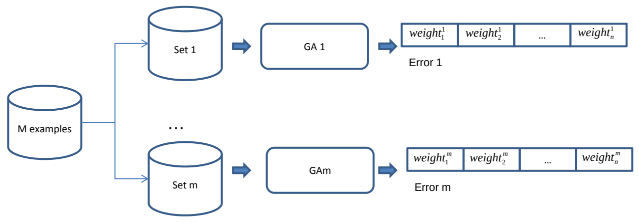

Figure 1: AG runs.

all of the examples is provided. Finally, whenever between a population and the new one there is no improvements (the error does not decrease), 50 additional iterations are performed, and the GA is stopped. Regarding other details about crossover, mutation, and remaining details see [6].

As it is possible to observe, randomness plays a large role in each run; so two different runs can produce different results. Thus, different runs are required, and a final solution should be obtained. For this purpose, we apply a cross-validation methodology, to obtain m folds from the original M examples. Each fold is composed by a training data set and a test data set. Then, the GA is repeated for each training data set, obtaining m sets of weights. After that, the boosting CBR multi-agent system configured with the learnt weights is tested with the test data set and m error rates are obtained, one per each of the GA run (see Figure 1).

Thus, after the m GA runs, we have get m sets of weights and m error rates, from which we need to determine a final set of weights. Note, that it is not the case that the GA with the lowest error rate is the best. As stated above, randomness is involved in the cross-validation procedure (data with which the weights have been obtained) and in the algorithm. So we propose the use of multi-criteria decision making to obtain a final set of weights for our boosting schema.

3.2 Multi-Criteria Decision Methods

Multi-criteria decision making (MCDM) aims at supporting decisions when several alternatives are available according to different criteria [17]. We can order those alternatives, and then choose one. We can also combine all of them to obtain a new solution thanks to either information fusion techniques or aggregation operators. We are interested in the second option. Among the different aggregation operators, there are the mean (M) and the weighted mean (WM).

Thus, after m runs of the GA on boosting a number of n CBR agents, we get the following sets of weights:

run	Agent1	Agent2	...	Agentn
1	$weight_1^1$	$weight_2^1$	...	$weight_n^1$
...	...	...	...	...
m	$weight_1^m$	$weight_2^m$	...	$weight_n^m$

The different runs can be considered alternatives from the MCDM point of view. Thus, a mean that can compute a final weight $weight_i$ for the i agent is the following:

$$weight_i = \frac{\sum_{i=1}^{m} weight_i^j}{m} \tag{2}$$

where m is the total number of weights obtained by the AGs regarding the i agent (see Figure 1).

In the case of a WM, we need to compute the mean values of the different weights $weigth_i^j$ obtained according to another ponderation $\mu_1^1, ...\mu_1^m, \ldots, \mu_n^1, ...\mu_n^m$. Thus,

$$weight_i = \frac{\sum_{j=1}^{m} weight_i^j * \mu_i^j}{\sum_{j=1}^{m} \mu_i^j} \quad (3)$$

So, a new parameter μ_i^j should be determined in order to obtain the final values $weigth_i$. We propose four methods: rated ranking, voted ranking, error based, and mean value.

First, the rated ranking method consist on 1) ranking the different weights for a given sets, and 2) computing the distance to the first position.

We can compute the ranking of each agent in all the runs by sorting them according to a descending order (from the highest to the lowest weight). So, we obtain a set of ranks as follows:

run	Agent1	Agent2	...	Agentn
1	$rank_1^1$	$rank_2^1$	...	$rank_n^1$
...	...	...	...	...
m	$rank_1^m$	$rank_2^m$	...	$rank_n^m$

Next, the μ_i^j is computed as follows:

$$\mu_i^j = \frac{1}{rank_i^j} \quad (4)$$

Second, in the voted ranking method, all the weights obtained by the GA $weight_i^j$ are ranked as in the previous one. However, after obtaining the ranking, we count the times an agent occupies the same rank, obtaining the "voted" rank for each agent vot_i^k. Thus if we have m runs, we have m possible votes per agent. In the next step, all the votes are averaged according to the following expression:

$$\mu_i^j = \frac{\sum_{k=1}^{n} [(n+1) - k] * \frac{vot_i^k}{m}}{n} \forall j \quad (5)$$

Third, the error based method takes advantage of the information provided by the learning algorithm related to the error to which the GA converges. Thus, the distance to the error is used as the μ_i^j, as follows

$$\mu_i^j = 1 - error^j \quad (6)$$

Finally, we define a the mean value method based on the mean weight value obtained for the agents in all the runs. Let mv_i be this mean value. Then, the inverse to the distances to this value is used as μ_i^j. That is,

$$\mu_i^j = 1 - |mv_i - weight_i^j| \quad (7)$$

4. APPLICATION TO DIAGNOSIS

We have tested our methodology in a breast cancer diagnosis scenario. We have used a Breast Cancer data base provided by the team of physicians we are working with. The database was constituted of 612 independent cases, with 239 healthy people. A first selection of the relevant attributes was performed by the physicians and 85 attributes were selected.

Data of each case has been partitioned in 8 groups, following the questionnaire criteria with which physicians have collected them that are related to different medical specialisations (epidemiology data, family information data, etc.). Therefore, we have 8 CBR agents in our system.

4.1 Experimental set up

We have followed a cross-validation procedure, with 90% of the cases for training and 10% for testing. Up to 10 folds were generated, and 10 AG runs have been performed, one per fold. Thus, we finally obtain 10*8 weights.

The following experimental settings have been defined:

- None: no learning has been applied. So all the agents weights have been set to 1.
- Mean: the mean operator has been used
- Ranking: the WM has been used together with the rated ranking method
- Voting: the WM operator has been applied together with the voting method
- Error: the WM operator has been used together with the error-based method
- MeanValue: the WM operator has been used together with the mean value method.

The results obtained in each experimental configuration are detailed in the next section.

4.2 Results

The weights obtained are after the 10 AG runs are the following:

	A0	A1	A2	A3	A4	A5	A6	A7
None	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Mean	0.58	0.02	0.03	0.03	0.27	0.02	0.04	0.02
Ranking	0.55	0.01	0.01	0.01	0.22	0.01	0.01	0.00
Voting	0.54	0.01	0.02	0.01	0.21	0.01	0.03	0.01
Error	0.39	0.02	0.02	0.02	0.19	0.01	0.03	0.01
MeanValue	0.45	0.02	0.03	0.03	0.19	0.02	0.04	0.02

Then, we have implemented the corresponding CBR agents. We have used ROC (*Receiver Operator Characteristics*) curves to plot the results[2]. Figure 2 shows the plot corresponding to the different scenarios. As it is possible to observe, the worst situation is when all of the agents weights are set to 1 (so the boosting voting schema has no weight). The remaining methods perform quite well and in a similar behaviour.

Analysing the weights obtained by all of our methods, we see that in fact, the weights are quite closer. So we are not surprised on obtaining so close results.

5. RELATED WORK

There are several works related to boosting CBR agents in particular, and ensemble learning in general [16, 15, 8]. For example, in [11] two schemas are proposed: bagging and boosting. In this work, the authors focus on how cases should be shared among agents. We are not so worried about that, but in how to set up the weights assigned to the agents thanks to a GA methodology. We are using the complete set of examples to train all the agents according to a cross-validation methodology as in [7].

A work closer to us is [10], in which the authors propose a corporate memory for agents, so that each agent knows about a piece of the case, as in our case. In [10], however,

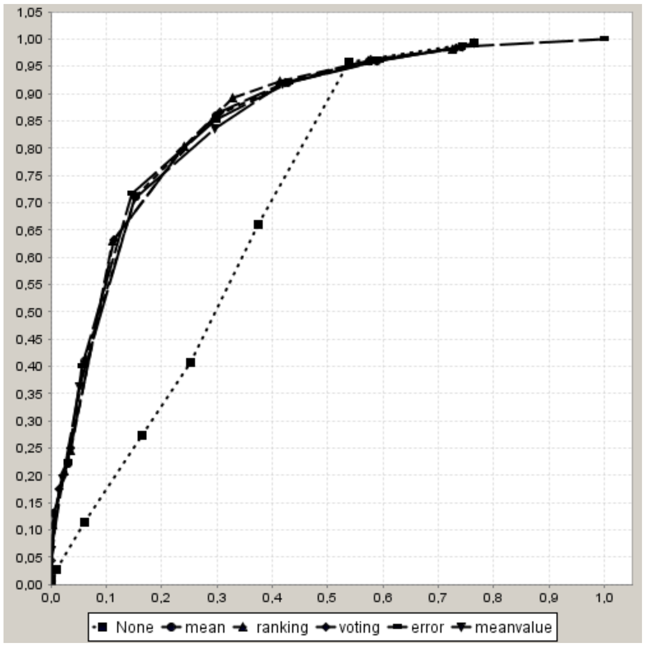

Figure 2: Comparison of the different scenarios.

the authors propose a negotiated retrieval method based on distributed constraint optimisation techniques. We are using the basic weighting voting schema for combining CBR agents.

Regarding research works on to use GA in a boosting environment, it is important to distinguish the approach followed in [14]. Here, the authors analyse the greedy behaviour of Adaboost and suggest to use GAs to improve the results. Another interesting work is [4], in which the AGs are used to search on the boosting space for sub-ensembles of learners.

6. CONCLUSIONS

Boosting mechanism are a promising paradigm for multiagent systems. In this paper we have described a boosting mechanism based on CBR agents, in which the final result of the system is the weighted voting results of the different agents. In order to determine the weights, we are using genetic algorithms. Due to the randomness involved in GA, it is necessary to run several times the GAs, obtaining different results. In this paper we present a analyse different multicriteria decision making methods in order to deal with the different GA results, and that allows to determine a single weight for each agent.

The methods have been applied to a breast cancer diagnosis data base. The results shown that MCDM methods obtain weights that are more robust to changes on the examples. Among all the methods presented, the one based on the ranking of the agents in each AG is the one that outperforms the other, although the results are quite close.

7. ACKNOWLEDGMENTS

This research project has been partially funded by the Spanish MEC projects DPI 2006-09370, CTQ2008-06865-C02-02/PPQ, TIN2008-04547/TIN, and Girona Biomedical Research Institute (IdiBGi) project GRCT41.

8. REFERENCES

[1] A. Birk. Boosting cooperation by evolving trust. *Applied Artificial Intelligence*, 14:769–784, 2000.

[2] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boostingg. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[4] D. Hernández-Lobato, J. M. Hernández-Lobato, R. Ruiz-Torrubiano1, , and A. Valle. Pruning adaptive boosting ensembles by means of a genetic algorithm. In *IDEAL (LNCS 4224)*, pages 322–329, 2006.

[5] K. Komathyk and P. Narayanasamy. Trust-based evolutionary game model assisting aodv routing againsts selfishness. *Journal of network and computer-application*, 31(4):446–471, 2008.

[6] B. López, A. Pla, P. Gay, and C. Pous. Boosting cbr agents with genetic algorithms. In *ICCBR*, submitted, 2009.

[7] E. Lozano and E. Acuña. Parallel computation of kernel density estimates classifiers and their ensembles. In *Proceedings of the International Conference on Computer, Communication and Control Technologies*, 2003.

[8] F. J. Martin, E. Plaza, and J. L. Arcos. Knowledge and experience reuse through communication among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering*, 9(3):319–341, 1999.

[9] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1998.

[10] M. V. Nagendra-Prasad and E. Plaza. Corporate memories as distributed case libraries. In *10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop*, pages 1–19, 1996.

[11] S. Ontañon and E. Plaza. Cooperative multiagent learning. In *ALAMAS, LNAI 2636*, pages 1–17, 2003.

[12] C. Pous, P. Gay, A. Pla, J. Brunet, J. Sanz, and B. López. Modeling reuse on case-base reasoning with application to breast cancer diagnosis. In *AIMSA, LNAI 5253*, pages 322–332, 2008.

[13] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach (second edition)*. Prentice Hall, 2003.

[14] İsmet Yalabik, F. T. Yarman-Vural, G. Uçoluk, and O. T. Sehitoglu. A pattern classification approach for boosting with genetic algorithms. In *22th International Symposium on Computer and Information Sciences*, pages 1–6, 2007.

[15] Z. Sun and G. R. Finnier. Case based reasoning in multiagent systems (chapter 7). In *Intelligent techniques in E-commerce: A case-based reasomning perspective, Springer*, 2004.

[16] E. I. Teodorescu and M. Petridis. An architecture for multiple heterogeneous case-based reasoning employing agent technologies. In *CIMAS*, 2008.

[17] V. Torra and Y. Narukawa. *Modeling Decisions: Information Fusion and Aggregation Operators*. Springer, 2007.

[18] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques,2nd Edition*. Morgan Kaufmann, 2005.