Multi criteria decision methods for boosting CBR agents with genetic algorithms

Beatriz López
University of Girona
Campus montilivi, edifice P4
Girona, Spain
beatriz.lopez@udg.edu

Carles Pous
University of Girona
Campus montilivi, edifice P4
Girona, Spain
carles.pous@udg.edu

Pablo Gay
University of Girona
Campus montilivi, edifice P4
Girona, Spain
pgay@eia.udg.edu

Albert Pla
University of Girona
Campus montilivi, edifice P4
Girona, Spain
apla@eia.udg.edu

ABSTRACT

There is an increasing interest on ensemble learning since they reduce the bias-variance problem of several classifiers. In this paper we approach an ensemble learning method in a multi-agent environment. Particularly, 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 multicriteria decision making methods. We experimentally test the methods proposed in a breast cancer diagnosis database.

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 [21, 16].

Ensemble learning has been recently applied to multi-agent systems, so that several learning agents collaborate in a distributed environment. For example, in [13] 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 agents.

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.

AdaBoost [4] 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 [17], 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.

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 [11]. 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 are cooperate for solving a problem. Each agent provides its advise or solution about a case, and a coordinator that makes a final decision based on a weighted voted schema.

Each agent is trained with the same set of examples; however, each agent receives only a part of the examples (as in [12]). Thus, each agent is specialised in a particular field of knowledge of the domain as shown in Figure 1.

Since there is a coordinator agent in charge of dealing

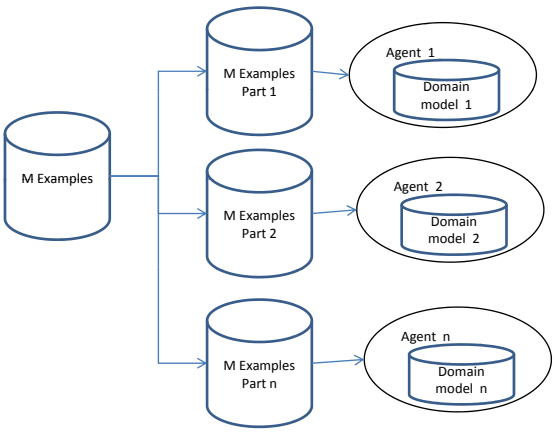

Figure 1: Agents specialisation

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 confident it is on that solution (see Figure 2). That is:

$$a_i =< class_i, \delta_i > \quad (1)$$

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:

$$v = \frac{\sum_{i=1}^{n} class_i * \omega_i}{\omega_i} \quad (2)$$

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.

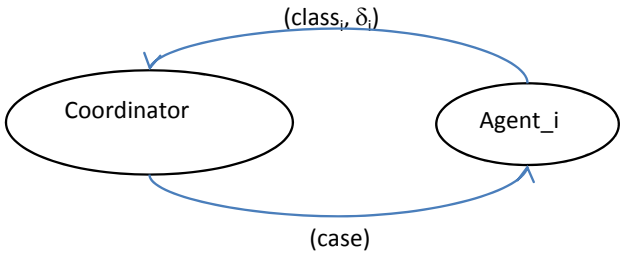

Figure 2: Agent's interaction.

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 [14]. See also [8] for further details on the boosting CBR MAS system.

3. MULTI-CRITERIA DECISION MAKING METHODS FOR GENETIC ALGORITHMS 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 [11]:

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 χ iterations.

As it is possible to observe, randomness plays a large role in each run; so two different runs can produce different results. Thus, averaged results on several runs should be obtained.

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 all of the examples is provided. Finally, 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.

Given a set of M examples, m data sets can be generated according to a cross-validation methodology. Then, the GA is repeated for each set, obtaining m sets of weights together with the m error rates, one per each of the GA run (see Figure 3.)

Regarding other details about crossover, mutation, and remaining details see [8].

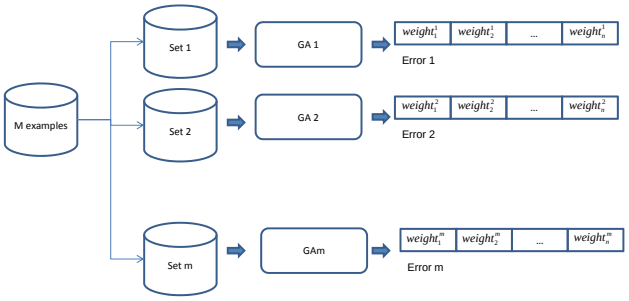

Figure 3: AG runs.

3.2 Multi-Criteria Decision Methods

Multi-criteria decision making (MCDM) aims at supporting decisions when several alternatives are available according todifferent criteria [20]. 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	$weight_1^1$	$weight_2^1$	...	$weight_n^1$
2	$weight_1^2$	$weight_2^2$	...	$weight_n^2$
...	...	...	...	...
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:

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

where m is the total number of weights obtained by the AG regarding the i agent (see Figure 3).

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 $\mu_1, ...\mu_n$. Thus,

$$weight_i = \frac{\sum_{j=1}^{m} weight_i^j * \mu_i^j}{\sum_{j=1}^{m} \mu_i^j} \quad (4)$$

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.

3.2.1 Rated raking method

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 sets 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	A0	A1	A2	A3	A4	A5	A6	A7
1	0.74	0.06	0.05	0.01	0.04	0.00	0.07	0.03
2	0.63	0.03	0.01	0.00	0.27	0.00	0.04	0.01
3	0.42	0.01	0.06	0.04	0.35	0.02	0.08	0.03
4	0.79	0.01	0.04	0.01	0.08	0.02	0.02	0.02
5	0.02	0.00	0.01	0.09	0.79	0.05	0.02	0.01
6	0.83	0.02	0.03	0.01	0.03	0.03	0.03	0.01
7	0.90	0.02	0.01	0.01	0.01	0.01	0.04	0.01
8	0.35	0.04	0.03	0.04	0.42	0.04	0.06	0.03
9	0.24	0.02	0.02	0.03	0.64	0.01	0.04	0.01
10	0.85	0.01	0.04	0.03	0.03	0.01	0.02	0.01

Table 1: Example of weights obtained for n agents after m GA runs .

run	A0	A1	A2	A3	A4	A5	A6	A7
1	1	3	4	7	5	8	2	6
2	1	4	6	8	2	7	3	5
3	1	8	4	5	2	7	3	5
4	1	7	3	8	2	5	6	4
5	5	8	7	2	1	3	4	6
6	1	6	4	7	2	3	5	8
7	1	3	6	5	8	4	2	7
8	2	6	8	5	1	4	3	7
9	2	6	5	4	1	8	3	7
10	1	6	2	4	3	7	5	8

Table 2: Rankings of the example.

run	Agent1	Agent2	...	Agentn
1	$rank_1^1$	$rank_2^1$	...	$rank_n^1$
2	$rank_1^2$	$rank_2^2$	...	$rank_n^2$
...	...	...	...	...
m	$rank_1^m$	$rank_2^m$	...	$rank_n^m$

Finally, the μ_i^j is computed as follows:

$$\mu_i^j = \frac{1}{ranking_i^j} \quad (5)$$

In order to illustrate with an example this method, suppose that we have 8 agents (n=8) and we have run 10 times the GA (m=10). The weights obtained by the GA are shown in table 1. Then, after sorting the values of table 1, we get the ranks shown in table 2. So agent $A0$ has been the agent with the highest weight in runs 1-4,6-7 and 10; while it occupies the second position in runs 8 and 9, and the fifth position in the 5th run.

Afterwards, the μ_i^j is computed according to equation 5. Finally, the weight of each agent, $weight_i$, is computed according to equation 4, obtaining the following results:

A0	A1	A2	A3	A4	A5	A6	A7
0.55	0.01	0.01	0.01	0.22	0.01	0.01	0.00

3.2.2 Voted raking method

In this 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 "voting" rank for each agent vot_i^k. Thus if we have n agents, we have n possible votes per agent. In

the next step, all the votes are averaged according to the following expression:

$$\mu_i^j = \frac{\sum_{k=1}^{n}[(n+1)-k] * \frac{vot_i^k}{n}}{n} \forall j \quad (6)$$

Observe, that μ_i^j as it is, can also be used as the $weight_i$, and we analyse this consideration in the results section.

Following the example of table 1 and rankings from table 2, we obtain the following votes from :

Rank	A0	A1	A2	A3	A4	A5	A6	A7
1	7	0	0	0	3	0	0	0
2	2	0	1	1	4	0	2	0
3	0	2	1	0	1	2	4	0
4	0	1	3	2	0	2	1	1
5	1	0	1	3	1	1	2	1
6	0	4	2	0	0	0	1	3
7	0	1	1	2	0	3	0	3
8	0	2	1	2	1	2	0	2

Finally, the μ_i^j values are then the following ones:

A0	A1	A2	A3	A4	A5	A6	A7
0.93	0.41	0.51	0.44	0.79	0.43	0.68	0.33

These are then combined according to equation 4, and the following agent weights are obtained:

A0	A1	A2	A3	A4	A5	A6	A7
0.54	0.01	0.02	0.01	0.21	0.01	0.03	0.01

3.2.3 Error based method

In this method we want to take 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

$$\mu_i^j = 1 - error^j \quad (7)$$

In our example, we got the following error rates for every GA run:

1	2	3	4	5	6	7	8	9	10
0.29	0.30	0.27	0.31	0.29	0.33	0.38	0.26	0.26	0.34

Therefore, the μ_i^j obtained for our agents with this method are the following ones:

A0	A1	A2	A3	A4	A5	A6	A7
0.39	0.02	0.02	0.02	0.19	0.01	0.03	0.01

3.2.4 Mean value method

Now, we define a 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,

$$\mu_i^j = 1 - |mv_i - weight_i^j| \quad (8)$$

In our example, we get the following mv^j values (from table 1):

A0	A1	A2	A3	A4	A5	A6	A7
0.58	0.02	0.03	0.03	0.27	0.02	0.04	0.02

Finally, the weights obtained for each agent are the following:

A0	A1	A2	A3	A4	A5	A6	A7
0.45	0.02	0.03	0.03	0.19	0.02	0.04	0.02

4. APPLICATION TO BREAST CANCER DIAGNOSIS

In this section we provide a brief description of the data base used for experimentation. Afterwards, some details about the experimental setup are presented. Next, the results obtained are shown.

4.1 Experimental set up

We have used a Breast Cancer data base provided by the team of physicians we are working with. It consists of 871 cases, with 628 corresponding to healthy people and 243 to women with breast cancer. There are 1199 attributes for each case. Since there are redundant, wrong and useless information a preprocess was carried out. Also, the preprocess was used to obtain data corresponding to independent individuals, since there are patients in the database that are relatives. As consequence, the database was constituted of 612 independent cases, with 239 healthy people.

A first selection of the relevant attributes was performed by the physicians. According to their knowledge, 85 attributes were selected, being 46 of them numerical and the remaining categorical.

Data 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.). Each group of data has been assigned to an agent. Therefore, we have 8 CBR agents in our system.

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 eXiT*CBR tool has been used [15] since it allows to easily define CBR boosting agents in a parameter-based way.

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 the ones shown in the previous section, in the illustrative example, that we collect here for convenience, and to which we have added the W results (equation 3):

	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

Regarding the outputs of the boosting CBR MAS, we have used ROC curves. ROC (*Receiver Operator Characteristics*) curves depict the tradeoff between hit rate and false alarm rate [3].

Figure 4 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 in detail the results, we obtain the following AUC (Area Under the Curve) values:

Scenario	AUC
None	0.706
Mean	0.851
Ranking	0.854
Voting	0.852
Error	0.853
MeanValue	0.848

The AUC values are quite similar too, being the ranking method the one that outperforms the others. 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. However, this does not happen with the weights obtained in each AG run, as shown in table 1.

We have also analysed the results of our methods compared to a single AG run. The different situations are being analysed are the following:

- run1: the weights of the first GA run have been set in the boosting CBR MAS, and run over all the folders used for all the GA.
- run1_newfolder:the weights of the first GA run have been set in the boosting CBR MAS, and run the MAS over a new set of folders.

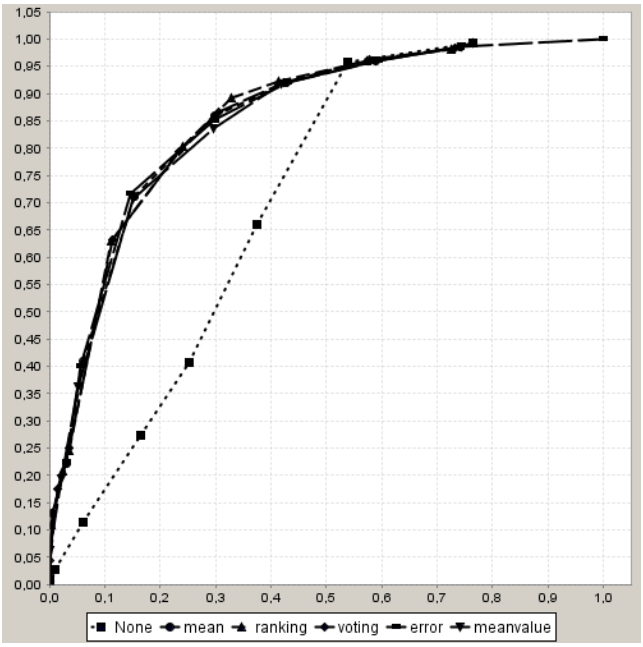

Figure 4: Comparison of the different scenarios.

- ranking: the weights according to our ranking method, and run using the same folders than the GA (the same than in the run1).
- ranking_newfolder: the weights obtained with our ranking method, and run the MAS over the new set of folders (the same than in the run1_newfolder)

In Figure 5 top shows the results. We can see that the performance of the learnt weights are similar. However, we in Figure 5 bottom, we have changed the weights obtained in the first run by the ones obtained in the fourth run. Both runs stop with a close error value: the first run at 0.29, and the fourth run at the error value of 0.31, quite closer to the previous one. So these results confirm the need of running the AG several times and finding an aggregated value. The aggregated results maintain the behaviour.

We have repeated the experimentation increasing the number of folds up to 50. Results, however, were not so good. The AUC obtained are the following:

Scenario	AUC
None	0.706
Mean	0.833
Ranking	0.827
Voting	–
Error	–
MeanValue	0.848

The reason for that is that the variance on the weights obtained by the AG is higher. Figure 6 top shows the mean an standard deviation of the weights obtained with the 10 folds, while in Figure 6 bottom the ones obtained for the 50 folds. As it is possible to observe, the variance with 50 folder has increased. So, in a future work we need to determine the adequate number of runs required in order to obtain the appropriate results.

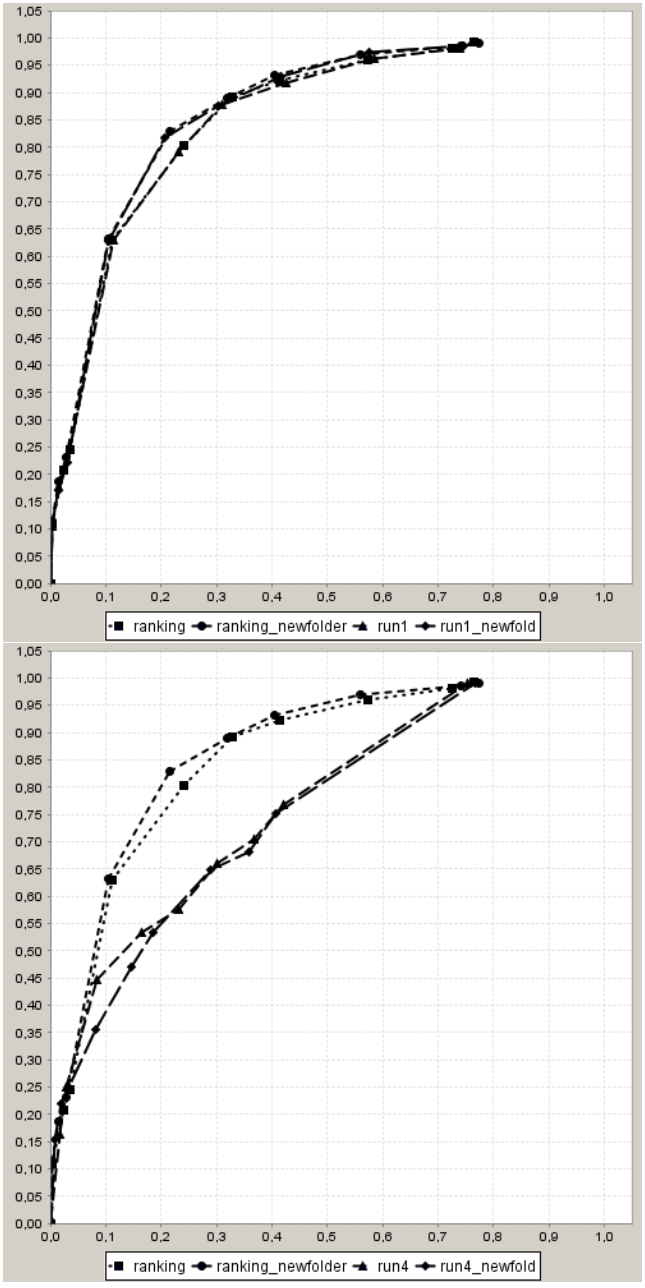

Figure 5: Comparison of boosting CBR performance when examples change.

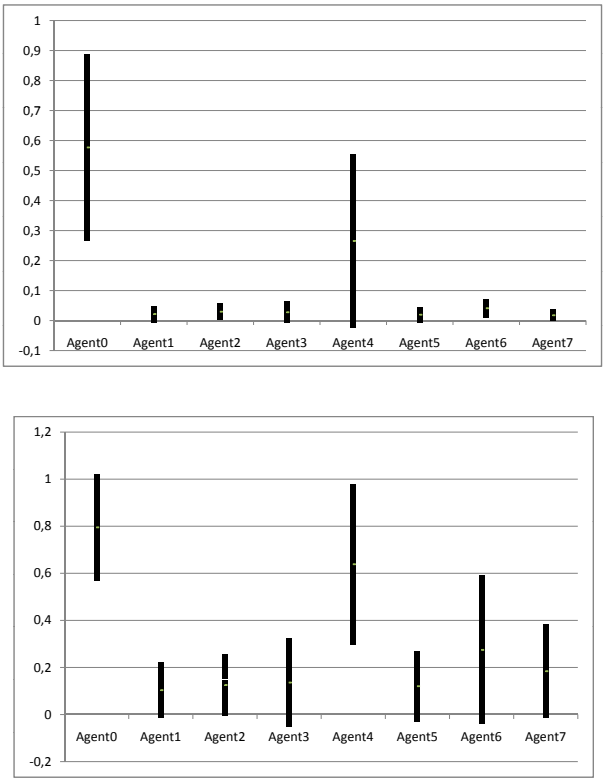

Figure 6: Mean and standard deviation a) *top*: after 10 runs; b) *bottom*: after 50 runs.

Finally, we have also used the value of μ_i^j as the weights $weight_i$, since it is the same for all the runs. That means, that we do not weight the AG waits, but directly we are using the qualitative information about the rank the agent has in the results. Results obtained are show in Figure 7, in which the excl_voting line is the one corresponding to this experiment, and compared to the previous voting results. We can see that the voting schema, as we have proposed in section 3, is the best.

5. RELATED WORK

There are several works related to boosting CBR agents in particular, and ensemble learning in general [19, 18, 10]. For example, in [13] 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, as in [9].

A work closer to us is [12], 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 [12], however, 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

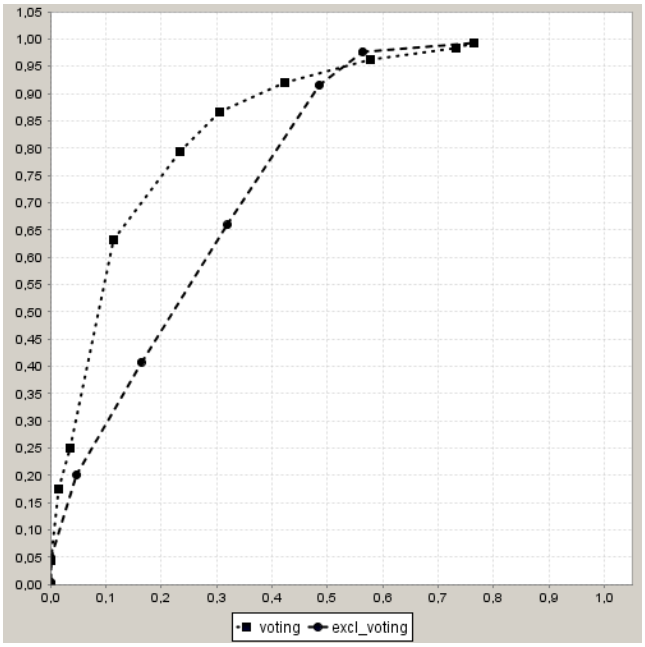

Figure 7: Consideration of a voting based exclusively on the rank.

in [17]. Here, the authors analyse the greedy behaviour of Adaboost and suggest to use GAs to improve the results. Another interesting work is [5], in which the AGs are used to search on the boosting space for sub-ensembles of learners.

They are also some approaches that boost GAs, as [7, 2, 22]. For example, in [22] a two stage method is proposed, in which in a first step, genetic fuzzy classifiers provide sets of rules, and in a second boosting stage, these classifiers are combined. Nevertheless, our goal is the opposite to these works: we are not boosting GAs but using GA to combine classifiers.

Finally, some interested studies to which our research work is related is [6]. In this case, agent's trust is studied under the evolutionary paradigm. We believe that our approach based on specialised agents is equivalent to it. This view of ensemble weights as trust has also been studied in [1].

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 DPI2006-09370, Girona Biomedical Research Institute (IdiBGi) project GRCT41 and DURSI AGAUR SGR 00296 (AEDS).

8. ADDITIONAL AUTHORS

9. REFERENCES

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

[2] A. Eiben, M. Schut, and A. de Wilde. Boosting genetic algorithms with self-adaptive selection. In *IEEE Congress on Evolutionary Computation*, pages 1584–1589, 2006.

[3] T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4., HP Labs, 2003.

[4] 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.

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

[6] 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.

[7] B. Liu, B. McKay, and H. A.Abbass. Improving genetic classifiers with a boosting algorithm. In *Congress on Evolutionary Computation, volume 4*, pages 2596–2602, 2006.

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

[9] 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*, page xx, 2003.

[10] 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.

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

[12] 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.

[13] S. Ontañon and E. Plaza. Cooperative multiagent learning. In *Alonso, E., D. Kazakov, D. Kudenko (Eds.) Adaptive Agents and Multi-Agent Systems, LNAI 2636, Springer Verlag*, pages 1–17, 2003.

[14] 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 *D. Dochev, M. Pistore and P. Traverso (Eds.): Artificial Intelligence: Methodology, Systems, and Applications (AIMSA), Lecture Notes in Computer Science (LNAI 5253), Springer*, pages 322–332, 2008.

[15] C. Pous, P. Gay, A. Pla, and B. López. Collecting methods for medical cbr development and experimentation. In *M. Schaaf (editor): Workshop Proceedings of the 9th European Conference on Case-Based Reasoning, CBR in the Health Sciences (ECCBR-HC), Tier, Tharax-Verlag*, pages 89–98, 2008.

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

[17] Ïsmet 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.

[18] 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.

[19] E. I. Teodorescu and M. Petridis. An architecture for multiple heterogeneous case-based reasoning employing agent technologies. In *1st International Workshop on Combinations of Intelligent Methods and Applications (CIMAS)*, 2008. On line: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-375/.

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

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

[22] T. Özyer, R. Alhajj, and K. Barker. A boosting genetic fuzzy classifier for intrusion detection using data mining techniques for rule pre-screening. In *Design and application of hybrid intelligent systems, IOS Press*, pages 983–992, 2006.