Improving individual learning capabilities in Multi Agent Systems

Eloi Puertas
IIIA - Artificial Intelligence Research Institute,
CSIC - Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia
(Spain).
eloi@iiia.csic.es

Eva Armengol
IIIA - Artificial Intelligence Research Institute,
CSIC - Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia
(Spain).
eva@iiia.csic.es

ABSTRACT

In multi-agent systems, individual learning capabilities can be improved thanks to the interaction with other agents. In the task of classification problem solving each agent is able to solve the problems alone, but in a collaborative scenario, an agent can take advantage of the knowledge of the others. In our approach when an agent decided to collaborate with other agents, it reaches the solution taking into account the justifications that other agents done about the solution they propose. Moreover, the agent learns to solve new problems using the justifications from the other agents and checking their answers as well as to solve further problems by its own.

1. INTRODUCTION.

Multi-agent systems are an usual approach to face up complex but decomposable problems. In Machine Learning the idea of cooperation between entities appears on the formation of *ensembles*. An ensemble is composed of several base classifiers (using inductive learning methods), being each one of them capable of completely solve a problem. Because the classifiers can provide different solutions for the same problem, the key issue of ensembles is how to aggregate the different solutions proposed by the different classifiers. Perrone and Cooper [**?**] proved that to aggregate the solutions obtained by independent classifiers improves the accuracy of each classifier separately. In that approach the cooperation among entities is done by both sharing the results for the same problem and reaching an aggregated solution.

This same idea is taken by Plaza and Ontañón [**?**] but they adapt it to agents. These authors define a *commitee* as an ensemble of agents where each agent has its own experience and it is capable of completely solve new problems. Each agent in a commitee can solve problems but it also can collaborate with other agents in order to improve its accuracy. The difference among this approach and the most common approaches to multi-agent learning systems (MALS) is that here agent is able to completely solve a problem whereas in most common approaches of MALS each agent solves a part of a problem.

In the current paper we taken the same idea of Plaza and Ontañón but we propose that the agents can take benefit of the collaboration with other agents by learning domain knowledge. Our point is that if the agents are able to justify the solution they provide, then each agent can use that justification as new domain knowledge. Therefore, an agent learns from the collaboration with other agents. This approach is different from that taken by Provost and Hennessy [**?**] since in such approach the goal is to build a common domain theory from distributed data. Instead, our approach consists on independent agents able to solve problems by its own and that can, in turn, collaborate when the reached solution is not confident enough. Our goal is not to build a common domain theory but improve the performance of each agent taken benefit of the collaboration with other agents.

The metaphor of our approach is that of a set of experts (f.e. physicians) in some domain, each of which has different experience (each physician has treated different patients) and needs advice from other experts to solve some particular cases. As long as an expert interacts with others for solving problems that initially are outside of his experience he acquires, in turn, experience on this kind of problems and, consequently, the interaction with other experts is reduced. We implemented this metaphor using a multi-agent system in which each agent is able to completely solve a classification problem. Each agent communicates with the other agents when it is not confident with the solution that it has reached. In such situation, the agent asks the other agents for solving the problem and it collects different solutions about that problem. In addition to the solution class, the other agents give an explanation about the solution they proposed. This explanation, that we call *justification*, is used by the agent to check the agents answers as well as to solve further problems by its own. The idea would be: whether the agent has asked collaboration for solving a problem before, can it now predict the solution for a similar problem without collaboration? The approach we introduce in this paper gives an affirmative answer to that question.

The paper is organized as follows. In section 2 we explain our multi-agent architecture and the algorithm followed by the agents. Then in section 3 we define the justifications, how an agent can asses its confidence on the justifications,

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

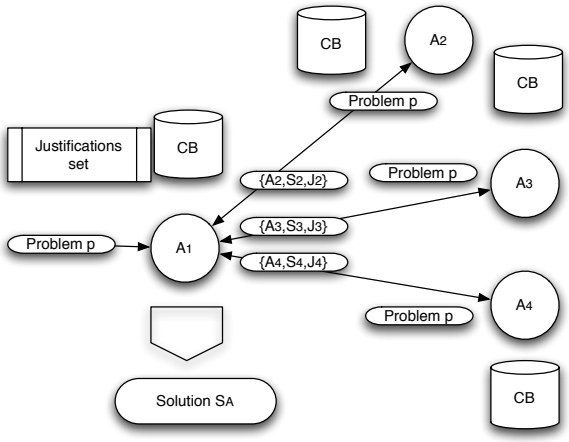


Figure 1: Multi-agent Architecture.

procedure **Collaborative-Algorithm** (p)
(1) $S_i \leftarrow$ **ProblemSolving** (p)
(2) **if** ($HasEnoughSupport(S_i)$)
(3) **then Return:** S_i
(4) **else** $SJp = UseJustificationTable(\mathcal{T}, p)$
(5) **if** (SJp **Is Not** empty)
then Return:$AggregateSolutions(SJp)$
(6) **else** $Send(p,AllAgents)$
(7) AS = *set of all the answers*
(8) CA = $ConsistentAnswersSet(AS)$
(9) **if** (CA **Is** empty)
(10) **then Return:** S_i
(11) **else Return:**
$AggregateSolutions(CA)$

Figure 2: Collaborative Algorithm used by the agents for solving problems

and how to aggregate the solutions of the other agents. Finally in section 4 the experimental results are shown. Related work is appointed in section 5 and the main conclusions and the future work are presented in section 6.

2. MULTI-AGENT ARCHITECTURE

In this section we propose a multi-agent scenario where there are several agents that are capable to solve problems of a certain domain. Our goal is twofold: on one hand the agents have to collaborate for solving problems and on the other hand they would take benefit of the collaboration to learn about the domain knowledge. In this way the agents reduce the number of collaborations while they learn new domain knowledge, and finally only hard problems should be solved with collaboration.

In the architecture we propose Fig. 1 the agents hold the following properties:

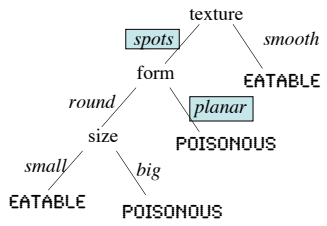
1. they are cooperative, i.e. they always try to solve the problems;
2. the experience (case base) of each agent is different;
3. each agent is capable of completely solving a problem.

The collaboration among the agents begins when one of the agents has to solve a new problem but it is not able to reach a confident enough solution by its own. Let us suppose an architecture formed by 4 agents *(A_1, A_2, A_3* and *A_4)* as in Fig. 1. When agent A_1 needs help to solve a problem it starts a very simple "query and answer" protocol. It sends the problem p to the other agents who answer the agent A_1 with its own solution for that problem. Now agent A_1 should be able to give a more confident solution for p.

Figure 2 shows in more detail the algorithm followed by the agents for solving new problems. Let us suppose that agent A_1 has to solve a problem p. The first step is to use some problem solving method on its case-base for solving p. The only requirement for the problem solving method is that it has to be able to give a justification of the solution (see section 3). This justification will be used by the agent A_1 to assess the support of this solution. When a solution S_i has enough support (step 2), this means that the agent does not need the collaboration of other agents for solving p, therefore the process finishes giving S_i as solution for p (step 3 in Fig. 2). When the solution S_i has not enough support means that the agent A_1 needs to collaborate with the other agents in order to achieve a solution for p. For simplicity in the explanation, let us suppose that p is the first problem for which the agent needs to start the collaboration process. In such situation, the step 5 fails (since the set of previous justifications is empty) and the agent A_1 sends the problem to the other agents (step 6 of the algorithm). Each agent solves p using its own problem solving method and case base, and returns (step 7) a tuple $\{A_j, S_j, J_j\}$ where A_j is the agent who solved the problem (A_2, A_3 and A_4 in our example), S_j is the solution for p and J_j is the justification of A_j for that solution. Each justification J_j is checked against the case base of agent A_1 and it builds the set CA of consistent justifications (see section 3). If CA is empty (step 9) then the solution S_i predicted by the agent A_1 is returned as solution for p (step 10). Otherwise, the agent A_1 uses an aggregation method to predict the class for p (step 11).

The agent A_1 stores in a table $\mathcal{T}$ the consistent justifications from the other agents as well as its own justifications. This table allows the agent to increase its domain experience, i.e. to learn new domain knowledge. Thus, when the agent A_1 reaches a solution for a problem with low support, it can look up the table $\mathcal{T}$ before it starts the collaboration protocol. Let us suppose that A_1 has solved the problem p with low confidence. In such situation, A_1 retrieves from the table $\mathcal{T}$ a set of justifications (SJ_p) satisfied by problem p (step 4 of the algorithm in Fig. 2). If the subset SJ_p is not empty (step 5) then the agent A_1 can classify p according to the justifications in SJ_p using the same aggregation method as in step 5. Otherwise the agent A_1 is forced to start the collaboration process (step 6).

Two important issues in the algorithm are how an agent can estimate the confidence on a justification and when a justification is considered consistent with the experience of an agent. The confidence on a justification determines how reliable the solution associated to that justification is. Consistent justifications allow the agent to enrich its experience since these justifications are incorporated to the domain knowledge available to the agent. In the next section



M_1 = {texture = *spots*, form = *planar*, habitat= *forest*}

Figure 3: Decision tree for classifying mushrooms. Marked edges can form a justification of why m_1 could be classified as poisonous.

we define the justifications and how they can be used by the agents. Moreover, in section 3.2 we explain how an agent can reach a final solution for a new problem from the solutions proposed by either other agents or justifications.

3. JUSTIFICATIONS

We define a *justification* as a symbolic description formed by all the aspects of the problem that have been used to achieve the solution. How to reach this symbolic description depends on the method used for solving the problem. Thus, when a decision tree is used for solving a problem, a justification could be formed by the path from the root to the leaf that classified the problem. For instance, let us suppose the decision tree in Fig. 3 useful to classify mushrooms as eatable or not and the example e_1 described as $\{texture = spots, form = planar, habitat = forest\}$. Using the decision tree, e_1 will be classified as poisonous and the reason may be the path used to classify the example, i.e. $\{texture = spots, form = planar\}$. Similar, for systems using relational representation (for instance ILP systems [**?**]) the justification could be the rules used to classify an example. In our experiments we used a lazy learning method called LID [**?**].

A justification is a proof reinforcing the solution given, thus an agent could take benefit of associating both solution and justification. If an agent is able to assert that a justification is a good domain rule, this justification can be directly applied to new problems. In the same way, the agent use the confident justifications that the other agents have already sent to it before asking them for a new problem.

Each agent collects justifications from its own (solving problems alone) and from other agents. These justifications are stored in order to use this knowledge in future situations. In our approach an agent A keeps all justifications in a table $\mathcal{T}_A$. A row of this table contains the following information:

- a justification (J_i): A symbolic description.
- a class solution (S_i): The class predicted by J_i.
- a set of precedents ($\mathcal{P}_{Ji}$): List of problems from the case base of A covered by the justification J_i.
- a set of agents ($\mathcal{A}_{Ji}$): List of agents that used J_i to predict the same class solution S_i.
- a significance score (λ_{Ji}): A score assessing the importance of J_i.

Our point is that this table supports the agent in acquiring experience in solving problems that are, initially, quite different from those in its case base. In other words, the interaction with the other agents and the use of the justifications they propose for problems that the current agent is not able to solve with enough confidence, allows the agent to improve its experience on the domain. As a consequence, that agent can solve new problems using previous justifications of other agents but without interact with them.

Let us suppose that an agent A has to solve a problem p_1. Using some problem solving method, A achieves S_i as the solution for p_1 and this solution is justified by J_i. In particular, this means that the aspects included in J_i are shared by a subset of cases $\mathcal{P}_{Ji}$ that belong to S_i. Depending on the support of $\mathcal{P}_{Ji}$, the agent A will decide whether or not to collaborate with the other agents (see section 3.1).

When the agent A has not enough confidence in its own solution, it asks other agents who return to A tuples $\{A_j, S_j, J_j\}$ meaning that the agent A_j proposes S_j as solution for p_1 and this solution is justified by J_j. Then, the agent A checks each J_j with its own case base, extracting the set of precedents $\mathcal{P}_{J_j}$ covered by J_j. There are several possible situations. The first one is that the cases in $\mathcal{P}_{J_j}$ belongs to different classes, this means that J_j is *not consistent* with the case base of A. A second situation is that there are no cases in the case base of A satisfying J_j (i.e. $\mathcal{P}_{J_j} = \emptyset$). The third situation is when all cases in $\mathcal{P}_{J_j}$ belong to the same class, this means that J_j is *consistent* with the cases base of A. The agent A associates to each consistent justification a score λ_{Jj} and stores them in the table $\mathcal{T}_A$.

Finally the agent A aggregates the class associated to each confident justification to reach a final solution for p_1 (see section 3.2). When A needs to solve a new problem, say p_2, it can check $\mathcal{T}_A$ for all justifications that are satisfied by p_2. In that way, thanks to the learned justifications, the agent A can predict the solution for p_2 without asking other agents.

3.1 Confidence of the justifications: an estimation

The collaboration among the agents is established when one of the agents has not enough confidence in the solution achieved for a problem using its own experience. One of the causes of this low confidence is that the problem solving method has produced more than one solution for the problem. In such situation, the agent needs collaboration in order to decide which of the solutions could be the correct one for the problem at hand. Nevertheless, when the problem solving method produced only one solution, how the agent can assess the confidence in that solution? In Machine Learning there are several measures commonly used to assess that confidence [**?**]. In our experiments we used the *recall* measure and the *significance*.

Let us suppose that an agent A_1 believes that the solution of the problem p is the class S_j because of the justification J_j. Let $\mathcal{P}_{J_j}$ be the set of cases from the case base of A_1 belonging to S_j covered by the justification J_j. In such situation, the confidence of A_1 in the solution S_j is computed using the *recall* measure which expression is the following:

$$recall(J_j) = \frac{\#\mathcal{P}_{J_j}}{\#CB_{A_1}[S_j]} \quad (1)$$

The recall is a ratio among the number of cases covered by a justification ($\#\mathcal{P}_{J_j}$) and the total number of cases of

the class S_j in the case base of the agent A_1. A justification has a high recall if this justification covers a high number of cases of the same class. Justifications with high recall should be more general than the others with low recall. Those justifications covering cases that belong to the same class and having a high recall are assessed with high confidence by the agent A_1. This means that A_1 only accepts non ambiguous and class representative justifications. High recall is defined as an index above a certain domain-dependent threshold parameter defined into the interval [0,1]. A high threshold forces the agent to collaborate most of the times, but when the threshold is too low the agent never asks for collaboration (it will solve the problems by its own).

Let us suppose now that A_1 asks for collaboration to the other agents. In such situation, how can A_1 assess the confidence to the answers produced by the other agents? In our scenario the agents do not exchange cases, therefore when A_1 gets a justification J_i from other agent, A_1 can only check J_i with its own case base, extracting the set of precedents $\mathcal{P}_{J_i}$ covered by J_i. As explained before, according to the classes owned by the cases in $\mathcal{P}_{J_i}$ (when $\mathcal{P}_{J_i} \neq \emptyset$) the justification J_i is either *not consistent* or *consistent*. Not consistent justifications are discarded by A_1 because they are too general since they are satisfied by cases belonging to several classes. Consistent justifications are stored and the confidence of A_1 on them is evaluated.

Notice that $\mathcal{P}_{J_i} = \emptyset$ means that there are no cases in the case base of A satisfying J_i, therefore in such situation the confidence of J_i cannot be evaluated. Nevertheless, J_i cannot be discarded, since it is correct for A_2. In such situation, A_1 stores J_i with $\lambda_{J_i} = 0$. This means that eventually, J_i could be used by A_1 although with low confidence.

Thus, agent A_1 only assesses the confidence on consistent justifications. A possible situation is that there are only few cases in the case base of A_1 covered by a consistent justification J_i. In other words, the recall ratio for J_i is low, but this not necessarily means that the justification is wrong. For this reason we need a measure that identifies significant justifications independently where they come from.

One of the first developed rule induction algorithms, CN2 [**?**], evaluates the quality of rules using a test of significance based in the likelihood ratio statistic [**?**]. The significance λ_j for the justification J_j is calculated as follows:

$$\lambda_j = 2\#\mathcal{P}_{J_j} \sum_{i=1}^{k} q_i \log\left(\frac{q_i}{\#CB_{A_n}[S_i]}\right) \quad (2)$$

where k is the number of classes, $\#\mathcal{P}_{J_j}$ are all cases covered by the justification J_j, q_i is the relative frequency calculated using the Laplace correction (Eq. 3) and $\#CB[S_i]_{A_n}$ is the relative frequency of all cases of class S_i in the case base of the agent A_n. The result is distributed as Chi-Square approximation (χ^2) with k-1 degrees of freedom.

We have applied the Laplace correction to the relative frequencies [**?**]. With this correction we smooth the big changes in the relative frequency due to have few or no examples in some classes. This correction is calculated as follows:

$$q_i = \frac{(\#\mathcal{P}_{J_j}[S_i] + 1)}{(\#\mathcal{P}_{J_j} + k)} \quad (3)$$

where $\#\mathcal{P}_{J_j}[i]$ are the cases in the predicted class S_i covered by the justification; $\#\mathcal{P}_{J_j}$ is the total number of cases covered by the justification and k is the number of classes.

The significance measure λ_j compares the class probability distributions in the set of covered examples with the distribution over the whole training set. We guess that a justification is reliable if these two distributions are significantly different. However it is not possible to assure that a justification is really significant or not, it can only be said that is unlikely that this justification is not significant according to the experience.

3.2 Aggregating justifications

Let us suppose that agent A_1 has already collected all justifications from the other agents and that it has selected the subset of justifications $J_1 \dots J_n$ consistent with its experience. Now A_1 uses the equation (2) to calculate the significance scores $\lambda_{J1} \dots \lambda_{Jn}$ for each justification. Finally, A_1 has to determine the solution class where p belongs to.

Let $SC = \{S_1 \dots S_k\}$ be the set of solution classes where p can be classified. Each class $S_i \in SC$ is supported by justifications that are consistent with the experience of A_1. Therefore A_1 can decide the class for p using a voting system [**?**] as usual in both multi-agent systems and ensemble learning. There are several voting systems that could be used to aggregate the solutions given by the agents. In our approach, the vote for a class $S_i \in SC$ takes into account the significance score of the justification λ_{Ji} in the following way:

$$V(S_i) = \sum_{j=1}^{n} \lambda_{Jj[S_i]}$$

where $\lambda_{Ji[S_i]}$ is the significance score associated to the justifications that predict the class S_i. Using this voting system, the agent A_1 classifies p as belonging to the class S_i such that $V(S_i)$ is maximum.

4. EXPERIMENTS

We performed several experiments implementing the multi-agent architecture described in section 2. Each agent has its own case base (disjoint from the other agents) and its own method of problem solving. There is also a query-answer protocol that allows to an agent to ask to the others for solving a problem. In our experiments, all the agents use the problem solving method called LID [**?**]. Given a new problem, LID gives as result the solution class(es) where the problem can be classified and a justification of that solution. Although in our experiments all the agents have the same problem solving method, this is not a requirement for the architecture we propose, the only condition is that the problem solving method has to produce justifications. Thus, an agent that is asked for classifying a problem produces as result, independently of the problem solving method, a tuple (p, $\{c_1 ... c_s\}$, λ_p) where p is the problem to be solved, $\{c_1 ... c_s\}$ is the classes in which the agent classifies p, and λ_p is the justification of that classification.

The experiments have been performed on three domains from the UCI repository [**?**]: Soybean, TicTacToe and Car. The goal of the experiments is to prove that an agent can learn from the collaboration with other agents. This learning can be seen in two aspects: the agent improves its accuracy on a test set and the collaboration with other agents diminishes as long as the agent uses the justifications obtained from previous collaborations (the table of justifications). To achieve this goal we performed the following process:

Domain	Total of cases	Training set	Test set
TicTacToe	958	862	96
SoyBean	307	246	61
Car	1728	1556	172

Table 1: characteristics of domains: total of cases and number of cases once they have been split in training and test set

i) given a domain D, the total number of cases was randomly split in two parts: the test set (20% in Soybean domain and 10% in the rest) and the training set (80% in Soybean domain and 90% in the rest).

ii) The training set, in turn, was divided in N parts uniformly distributed, where N is the number of agents in the architecture.

iii) One agent is arbitrarily chosen as the inquirer agent. This agent receives all the test set and starts the algorithm shown in Fig. 2.

Table 1 shows the total number of cases for each domain and how they have been distributed between the training set and the test set according to the point *i*) above.

Concerning the accuracy, Fig. 4 shows the result of several experiments where the multi-agent system is composed of 2, 3, 5, 8 and 10 agents. These results are the average of 7 ten-fold cross-validation trials for both the Car and TicTacToe and 7 five-fold cross-validation trials for Soybean.

Left column of each group shows the accuracy of an agent solving problems using only its own case base. Notice that this accuracy decreases as long as the number of agents increases. This is due to the point *ii*) above, i.e. the training set is uniformly divided among the agents, therefore as more agents compose the system smaller is the case base of each agent. Right column of each group shows the accuracy reached by an agent using the algorithm explained in section 2 taking advantage from the multi-agent system. The accuracy achieved by an agent solving problems in collaboration is higher than solving problems alone. This result was expected due to the *ensemble effect* [**?**] proved for a set of classifiers and which effect can also be translated to a set of agents. In short, the ensemble effect states that when agents are minimally competent (individual error lower than 0.5) and they have uncorrelated errors (this is true in our experiments because agents have disjoint case bases), then the error of the combined predictions of the agents is lower than the error of the individual agents.

Let us to analyze in more detail the accuracy of an agent solving problems in collaboration with other agents. Figure 5 shows for each multi-agent configuration (3, 5, 8 and 10 agents) and each domain, the percentage of cases that an agent solves by its own, i.e. step 3 of the algorithm in Fig.2 (label *not collaborative*); using the justification table, i.e. step 5 of the algorithm (label *using justification*); and in collaboration with other agents, i.e. step 9 of the algorithm (label *collaborative*). For example, whe the system is composed of three agents and solves problems in the Soybean domain, around the 40% of cases are solved by the agent alone. This means that the remaining 60% should not be solved with enough confidence. Nevertheless, using the justification table the number of cases that the agent is able

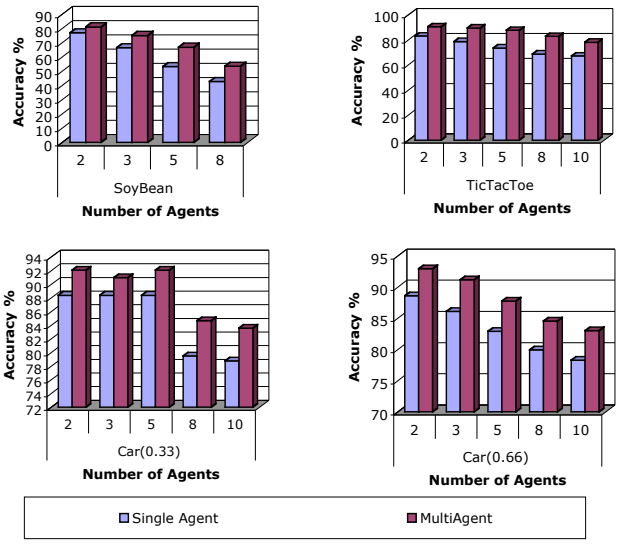


Figure 4: Accuracy exhibited by multi-agent systems with 2, 3, 5, 8 and 10 agents in three domains: soybean, Tic Tac Toe and Car. Graphics compare the accuracy of an agent using only its own case base (left columns) with the accuracy of that agent solving problems in collaboration with the other agents in the system (right columns). These results are the average of seven trials of 5-fold cross-validation for soybean and 10-fold crossvalidation for the other domains.

to solve without collaborations increases around the 60%. This shows that the agent is reusing some justifications that have been useful in previous cases. Notice that, for all the configurations and domains number of cases solved without collaboration decreases as long as the number of agents increases due to the decrement of the number of cases in the case base of the agents. Also, the number of cases solved using $\mathcal{T}_A$ increases when there are more agents in the system due to two main reasons: 1) the number of cases in each case base is low, and 2) the self-confidence of an agent decreases because the recall measure does not reach the confidence threshold. As a consequence the agent has not enough experience in the form of cases and uses the justifications, obtained from the other agents in previous collaborations, as domain rules for solving new problems before establish a new collaboration with the agents.

It is important to remark here the utility of the justification table since without it the inquirer agent would ask for solving in collaboration around the 50% of cases in the car domain (*Using justification* plus *Collaborative*). Using the justification table, the agent is able to solve without collaboration (in the sense of sending a message) approximately the 65% of test cases (*Not collaborative* plus *Using justication*). This improvement is more clear in the Soybean domain with a configuration of 5 and 8 agents. In both configurations, the percentage of cases solved with enough confidence by the agent alone is under 20%, whereas using the table of justifications this percentage increases to around the 60%. This proves that the agent learns from the collaboration with other agents and, in some sense, increases its experience in

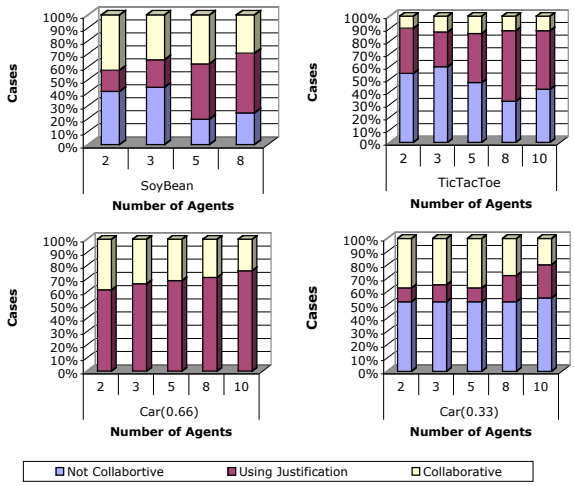


Figure 5: For each domain and each configuration of the multi-agent system (2, 3, 5, 8 and 10 agents), graphics show the percentage of cases that an agent solves by using its case base (label *not collaborative*), using the justification table (label *using justification*) and in collaboration with other agents (label *collaborative*). These results are the average of seven trials of 5-fold cross-validation for soybean and 10-fold crossvalidation for the other domains.

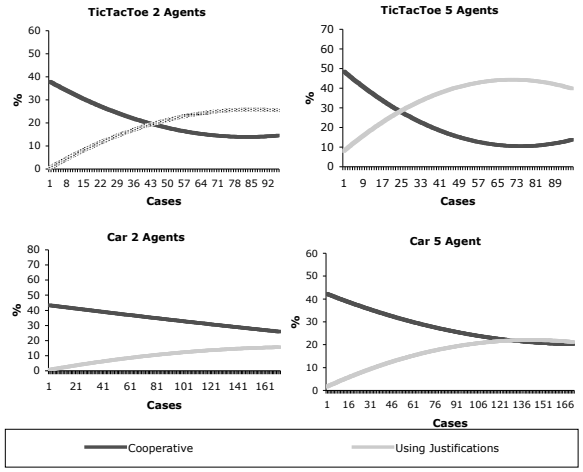


Figure 6:

the domain. Figure 6 shows the evolution of the agent experience along the time on Tic Tac Toe and Car domains for configurations having 2 and 5 agents. These graphics show that initially the agents asks for collaboration and does not uses the table of justifications. Depending on the domain and the number of agents configuring the system, the agent takes benefit of the table of justifications to solve problems without collaboration. Graphics representing the behavior of the agent in the Tic Tac Toe domain, show that there is a point (around the 40th case in the configuration with two agents) where the most common way for solving a problem is without collaboration, i.e using either the case base or the table of justifications. The behavior on the Car domain follows the same tendency, i.e. the use of the table of justifications increases and the collaboration with other agents decreases, nevertheless there is not an intersection point between both lines. This is due to the fact that the case case of each agent has enough cases for solving most of problems and the process to build the table of justifications is slow. Notice that when the system is configured by 5 agents there is intersection between the two lines since the case base of the agents is smaller than in the configuration with 2 agents and this allows to more frequently establish collaboration among the agents; consequently, the current agent can build the table of justifications.

The experiments with the Car domain show how the threshold parameter can influence the collaboration among the agents. The training set of Car is composed of 1556 cases. Even when the multi-agent system is composed of 10 agents, this means that the case base of each agent contains more than 150 cases. In other words, the experience of the agents is enough to solve all the cases without collaboration. For this reason, we performed several experiments using several thresholds for the recall. Graphics labeled as $Car(0.33)$ in Figs. 4 and 5 are experiments where the agent asks for collaboration when the recall is lower than 0.33. Figure 5 shows that almost all the problems solved without collaboration and Fig. 4 shows an accuracy of around 92% for systems with 3, 5 and 8 agents. Graphics referred as *Car(0.66)* correspond to experiments performed taking 0.66 as threshold. All cases have been solved in collaboration with the other agents, because the threshold parameter is too high. This means that the agent has always been forced to ask for collaboration until it has enough knowledge in the justifications table.

5. RELATED WORK

The architecture introduced in this paper is related with works on *ensemble learning* [**?**]. The goal of ensemble learning is to aggregate the solutions proposed by a set of independent classifiers. Each classifier uses an inductive learning method to induce domain knowledge and it can solve completely a problem using this domain knowledge. This same idea was taken by Plaza and Ontañón [**?**] to define commitees as ensembles of agents. In that work the authors analyze different collaboration strategies focused on a lazy behavior of the agents, i.e. an agent has as goal to solve the current problem with collaboration when necessary. Nevertheless agents do not extract any information from the collaboration. Instead, the goal of our approach is twofold: on one hand an agent has to solve new problems but, on the other hand, the agent has to learn more about the domain. This means that the collaboration with the other agents has to be reduced as well as the agent gains experience.

In [**?**] we performed a first attempt to model a multi-agent system that improves the individual capabilities of an agent. In that work, an agent A with poor experience (small case base) on a domain performs a learning phase by asking other agents for solving problems from its own case base. Since A knows the correct solution of these problems, it can induce a domain theory from the problems correctly solved by the other agents. In the current paper, we defined a similar scenario but now the agent A not necessarily has poor expe-

rience in the domain. Now, A can gain experience thanks to the explanations (justifications) that the other agents made about the solutions they propose. The agent A does not induce domain theory but it stores consistent justifications that can be used as domain rules. The concept of justification has also been used by Plaza and Ontañón in [**?**] but these authors use the justifications to assess the weight of the vote of each agent when aggregating the solutions.

The idea of reusing justifications as a kind of domain rules was also introduced in C-LID [**?**]. The C-LID method was used by a knowledge-based system (KBS) and it is built on top of the lazy learning method called LID. The goal of C-LID was to use the justifications from LID as patterns that support solving similar problems (those satisfying the justifications) without use the lazy learning method. In the current paper we extended this idea of caching the justifications of a KBS to a multi-agent learning architecture. The main difference is that now an agent has two kinds of justifications: those generated when it solves problems by its own (like the KBS), and those coming from the other agents.

6. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a multi-agent architecture where each agent can completely solve a problem. When the problems are not solved with enough confidence, the agent can ask for collaboration to the other agents. The collaboration is established by means of the justifications, i.e. an asked agent sends to the inquirer agent the solution and a justification of that solution. The justifications that are consistent with the knowledge of the inquirer agent can be stored and used in the future for solving similar problems without collaboration. The experiments we performed show that an agent learns from the collaboration since thanks to the justifications, it can solve problems similar to other problems that initially were solved with low confidence.

There are two possible future research lines. On one hand new measures for assessing the confidence on justifications should be analyzed. On the other hand we plan to use association rules to dynamically discover the expertise of the agents. With this knowledge about the other agents an agent could detect the more appropriate team of agent for solving each new problem.

Acknowledgements

This work has been supported by the SAMAP project (TIC 2002-04146-C05-01).