Learning from cooperation using justifications [1]

Eloi Puertas [a,2], and Eva Armengol [a]
[a] *Artificial Intelligence Institute (IIIA-CSIC)*

Abstract. In multi-agent systems, individual problem solving capabilities can be improved thanks to the interaction with other agents. In the classification problem solving task each agent is able to solve the problems alone, but in a collaborative scenario, an agent can take advantage of the knowledge of others. In our approach, when an agent decides to collaborate with other agents, in addition to the solution for the current problem, it acquires new domain knowledge. This domain knowledge consists on explanations (or *justifications*) that other agents done for the solution they proposed. In that way, the first agent can store these justifications and use them like some kind of domain rules for solving new problems. As a consequence, the agent acquires experience and it is capable to solve on its own problems that initially were outside of his experience.

Keywords. Machine Learning, Multi-agent System, Cooperation, Justifications

1. Introduction.

In Machine Learning the idea of cooperation between entities appears with the formation of *ensembles*. An ensemble is composed of several base classifiers (using inductive learning methods), each one of them being capable of completely solving a problem. Perrone and Cooper [9] proved that aggregating the solutions obtained by independent classifiers improves the accuracy of each classifier on its own: it is the *ensemble effect*. 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 [10] 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 solving new problems. In addition, each agent in a commitee can collaborate with other agents in order to improve its problem solving capabilities. The difference between this approach and the most common approaches to multi-agent learning systems (MALS) is that in the former case the agent is able to completely solve a problem whereas in most common approaches of MALS each agent only solves a part of a problem.

In the current paper we take the same idea of Plaza and Ontañón but we propose that the agents can take benefit from the collaboration with other agents by learning domain

[1] This work has been supported by project CBR-PROMUSIC (TIC2003-07776-C02-02)
[2] Artificial Intelligence Research Institute (IIIA - CSIC) Campus UAB, 08193 Bellaterra, Catalonia (Spain) Tel.: +1 222 333 4444; Fax: +1 222 333 0000; E-mail: eva@iiia.csic.es.

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. The idea of take benefit from cooperation between learners was appointed by Provost and Hennessy [13]. In their work authors show how individual learners can share domain rules in order to build a common domain theory from distributed data and how this improves the performance of all the system. In our work we are interested on show how one individual agent can improve its own domain knowledge by means of other agents. Thus, we propose that, in addition to the classification of a problem, agents have to be able to give an explanation of why they proposed such solution. This explanation, that we call *justification*, is used by the agent as new domain knowledge since it can use it for solving further problems on its own.

The paper is organized as follows. In section 2 we explain our multi-agent architecture and the algorithm followed by the agents and then, in section 3, justifications are defined. 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 of solving 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 from the collaboration to learn new domain knowledge. In the architecture we propose 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 on its own. In such situation, a simple "query and answer" protocol is used to send the problem to other agents who answer with its own solution for that problem.

Figure 1 shows the algorithm followed by the agents for solving new problems. Let us suppose that agent A 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 for the solution (see section 3). This justification will be used by the agent A to assess its confidence on that solution.

In Machine Learning there are several measures commonly used to assess the confidence on a solution [3]. In our experiments we used the *recall* measure that is a ratio between the number of cases covered by a justification and the total number of cases of the class S_j in the case base of the agent A. The recall of a justification J_j is computed using the following expression:

$$recall(J_j) = \frac{\#\mathcal{P}_{J_j}}{\#CB_A[S_j]}$$

where $CB_A[S_j]$ is the set of cases in the case base of agent A that belong to the class S_j, and $\mathcal{P}_{J_j}$ is the set of cases of S_j covered by the justification J_j. High recall is defined

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

Figure 1. Collaborative Algorithm used by the agents for solving problems

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 and a low threshold produces no collaboration.

When a solution S_i has enough support (step 2 in Fig. 1), 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. When the solution S_i has not enough support, A asks other agents for solving p. For simplicity, let us suppose that p is the first problem for which the agent asks for collaboration. In such situation, the step 4 fails (since there are not previous justifications) and the agent A sends the problem to the other agents (step 5 of the algorithm). Each agent solves p using its own problem solving method and case base, and returns (step 6) a tuple $\{A_j, S_j, J_j\}$ where A_j is the agent who solved the problem, S_j is the solution for p and J_j is the justification of A_j for that solution.

The agent A checks each one of these tuples against its case base and builds the set CA of consistent justifications (see section 3). When CA is empty (step 8) then the solution S_i predicted by the agent A (although with low confidence) is returned as solution for p. Otherwise, A uses a voting system to predict the class for p (step 9). In particular, this voting system takes into account the confidence that A has on the justifications given by the other agents.

Agent A stores consistent justifications (set CA) into a table $\mathcal{T}$. This table allows the agent to increase its domain experience, i.e. to learn new domain knowledge. Thus, when A reaches a solution for a problem on his own with low support, it can use the table $\mathcal{T}$ (step 3 of the algorithm) before it starts the collaboration protocol. Let SJ_p be the set of justifications in $\mathcal{T}$ satisfied by p, if SJ_p is not empty (step 4) then A can classify p according to the justifications in SJ_p using the aggregation method as in step 4. Otherwise the agent A is forced to start the collaboration process (step 5).

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 we define the justifications and how they can be used by the agents.

3. Justifications

A *justification* is defined 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 problem solving. Thus, when a decision tree is used for problem solving, a justification could be formed by the path from the root to the leaf that classified the problem. Similarly, in methods using relational representation (for instance ILP systems [7]), a justification could be formed by the rules used to classify an example.

In our approach, each agent collects justifications coming both from 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 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 of A covered by J_i.
- a set of agents ($\mathcal{A}_{Ji}$): List of agents that used J_i to predict S_i.
- a significance score (λ_{Ji}): A score assessing the confidence on J_i.

The table $\mathcal{T}$ supports an agent in acquiring experience when it solves problems. In that way, the agent improves its experience on the domain by means of interactions with other agents and using justifications other agents propose for those problems that the agent was not able to solve with enough confidence. Let $\mathcal{P}_{J_j}$ be the set of precedents belonging to the cases base of an agent A that satisfy the justification J_j. There are several possible situations: 1) cases in $\mathcal{P}_{J_j}$ belong to different classes, i.e. J_j is *not consistent* with the case base of A; 2) $\mathcal{P}_{J_j} = \emptyset$, i.e there are no cases in the case base of A satisfying J_j; and 3) all cases in $\mathcal{P}_{J_j}$ belong to the same class, i.e. J_j is *consistent* with the case base of A. Only consistent justifications are stored in $\mathcal{T}$.

When an agent A want to assess the confidence on a consistent justification J_i, a possible situation is that only few cases in the case base of A are covered by J_i. This produces a low recall for J_i but this does not necessarily mean that the justification is wrong. For this reason we need a measure that identifies significant justifications independently where they come from. CN2 [4], evaluates the quality of rules using a test of significance based in the likelihood ratio statistic [6]. The significance λ_j for the justification J_j is calculated as follows:

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

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 and $\#CB_A[S_i]$ is the relative frequency of all cases of class S_i in the case base of agent A. The result is distributed as Chi-Square approximation (χ^2) with k-1 degrees of freedom.

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 it is unlikely that this justification is not significant according to the experience.

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

Table 1. Total number of cases of each data set and how they have been partitioned in the experiments.

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. All the agents use the relational lazy learning method called LID [1] as problem solver method. 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 experiments have been performed on three domains from the UCI repository [8]: 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 proved because 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:

i) given a domain D, the total number of cases was randomly splitted 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) (table 1).
ii) The training set, in turn, was divided in N parts uniformly distributed, where N is the number of agents in the architecture.

Concerning the accuracy, Table 2 shows the result of several configurations of the multi-agent system (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. As it is expected, the accuracy of the ensemble increases due to *ensemble effect* [5] proved for a set of classifiers and that 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 agents is lower than the error of individual agents. Notice that the accu-

	Tic Tac Toe		Soybean		Car	
n. Ag.	Single	Multi	single	Multi	Single	Multi
2	83.03	90.47	76.81	80.98	88.40	92.10
3	78.60	89.38	66.41	74.94	88.40	91.00
5	73.56	87.59	53.16	66.70	88.40	92.10
8	68.67	82.76	42.66	53.44	79.49	84.65
10	67.16	78.03	–	–	78.78	83.58

Table 2. Accuracy exhibited by configurations with 2, 3, 5, 8 and 10 agents in Soybean, Tic Tac Toe and Car.

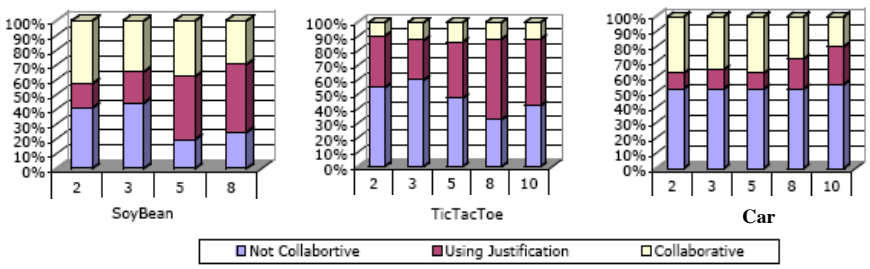

Figure 2. Given a number of agents (X-axis), graphics show for each domain the percentage of cases (Y-axis) solved by an agent using 1) its case base (label *not collaborative*), 2) the justification table (label *using justification*) and 3) collaboration with other agents (label *collaborative*).

racy decreases as long as the number of agents increases due to point $ii)$ above, i.e. the training set is uniformly divided among the agents, therefore as more agents compose the system, the smaller the case base of each agent is.

Figure 2 shows for each multi-agent configuration and each domain, the percentage of cases that an agent solves on its own (label *not collaborative*), i.e. step 3 of the algorithm in Fig.1; using the justification table (label *using justification*), i.e. step 5 of the algorithm; and collaborating with other agents (label *collaborative*), i.e. step 9 of the algorithm. For example, when the system is composed of three agents and solves problems in the Soybean domain, around 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 to solve without collaborations increases around 60%. This shows that the agent is reusing some justifications that have been useful in previous cases.

It is important to remark here the utility of the justification table, since without it the agent would ask for collaboration around 50% of cases in the car domain (*Using justification* plus *Collaborative*) whereas using the justification table, the agent is able to solve without collaboration approximately 65% of the 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 60%. This proves that the agent learns from collaboration with other agents increasing its experience in the domain.

Figure 3 shows the evolution of the agent experience along the time on Tic Tac Toe and Car domains for a configuration with 5 agents. The x-axis shows the problem number, and the y-axis shows, in percentage, how many times a method has been used for solving each problem. Therefore, the graphics show how the decisions taken by the agent change from the first to the last problem.

5. Related Work

The architecture introduced in this paper is related with works on *ensemble learning* [12]. The goal of ensemble learning is to aggregate solutions proposed by a set of inde-

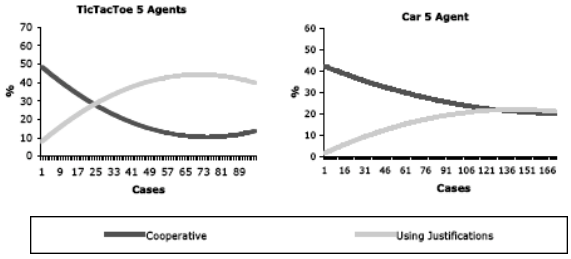

Figure 3. Evolution of the agent experience on Tic Tac Toe and Car for configurations with 2 and 5 agents

pendent 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 [10] to define commitees as ensembles of agents. In that work, 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.

In the same way appointed by Weiß in [14], agents in our system improve their performance by means of communication. In turn, communication itself is reduced as long as agents acquire more knowledge. In our approach, agents learn using high-level communication, in form of justifications that are used for solving the current problem and also future problems. The idea of learning from high-level communication was introduced firstly by Sian [15] in his approach called *consensus learning* and, more recently, by Plaza and Ontañón [11]. In the framework proposed by Sian each agent learns from its experience and obtains hypothesis. When an hypothesis has reasonable confidence, it is proposed to the other agents by means of a shared blackboard. Other agents use their own experience to evaluate the published hypothesis and may make changes to it. Finally, agents accept the hypotheses that have greatest support by consensus. Conversely, our agents do not share a blackboard, but justifications of a solution are stored by only one agent (that asking the others for solving the current problem). Therefore, stored justifications are only consistent with the knowledge of that agent.

The reuse of justifications as domain rules was previously introduced in C-LID [2]. The goal of C-LID was to use the justifications from LID as patterns that support solving similar problems (those satisfying the justifications) without using 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.

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 could not be solved with enough confidence, the agent can ask other agents for collaboration. The collaboration is established by means of justifications that agents give in addition to the solution they propose. The inquirer agent

stores those justifications consistent with its knowledge in order to use them for solving further problems. The experiments show that an agent learns from the collaboration since thanks to the table of justifications, an agent acquires domain knowledge allowing it to solve more problems than without that collaboration.

In the future we plan to analyze other measures for assessing confidence on justifications. On the other hand, we also plan to use association rules to dynamically discover the expertise of agents. With this knowledge about other agents an agent could detect the most appropriate team of agents for solving each new problem.

References

[1] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Machine Learning: ECML-2001*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[2] E. Armengol and E. Plaza. Remembering similitude terms in case-based reasoning. In *3rd Int. Conf. on Machine Learning and Data Mining MLDM-03*, Lecture Notes in Artificial Intelligence 2734, pages 121–130. Springer Verlag, 2003.

[3] M. Berthold and D. J. Hand, editors. *Intelligent Data Analysis: An Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[4] P. Clark and T. Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3(4):261–283, 1989.

[5] L. K. Hansen and P. Salomon. Neural network ensembles. *IEEE Transactions on Pattern analysis and machine intelligence*, 12:993–1001, 1990.

[6] J. Kalbfleish. *Probability and Statistical Inference, volume II*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1979.

[7] S. Muggleton. Inductive logic programming. *New Generation Comput.*, 8(4):295–, 1991.

[8] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[9] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall, 1993.

[10] E. Plaza and S. Ontañón. Ensemble case-based reasoning: Colaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *CBR Research and Development: ICCBR-2001*, volume 2080, pages 437 – 451, 2001.

[11] E. Plaza and S. Ontañón. Justification-based multiagent learning. In *ICML-*, 2003.

[12] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In *Book on Advances of Distributed Data Mining, editors Hillol Kargupta and Philip Chan, AAAI press, 2000.*, 2000.

[13] F. J. Provost and D. N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *AAAI/IAAI, Vol. 1*, pages 74–79, 1996.

[14] S. Sen and G. Weiß. Chapter 6: Learning in multiagent systems. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 259–299, Cambridge, MA, USA, 1999. MIT Press.

[15] S. S. Sian. Extending learning to multiple agents: Issues and a model for multi-agent machine learning. In Y. Kodratoff, editor, *EWSL*, volume 482 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 1991.