Arguments and Couterexamples in Case-based Joint Deliberation

Santiago Ontañón[1] and Enric Plaza[2]

[1] MAIA, Department of Applied Mathematics and Analysis
UB, University of Barcelona, Gran Via de les Corts Catalanes, 585
08007 Barcelona, Catalonia, Spain.
santi@maia.ub.es
[2] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
enric@iiia.csic.es

Abstract. Multiagent learning can be seen as applying ML techniques to the core issues of multiagent systems, like communication, coordination, and competition. In this paper, we address the issue of learning from communication among agents circumscribed to a scenario with two agents that (1) work in the same domain using a shared ontology, (2) are capable of learning from examples, and (3) communicate using an argumentative framework. We will present a two fold approach consisting of (1) an argumentation framework for learning agents, and (2) an individual policy for agents to generate arguments and counterarguments (including counterexamples). We focus on argumentation between two agents, presenting (1) an interaction protocol (AMAL2) that allows agents to learn from counterexamples and (2) a preference relation to determine the joint outcome when individual predictions are in contradiction. We present several experiment to asses how joint predictions based on argumentation improve over individual agent's prediction.

1 Introduction

Argumentation frameworks for multiagent systems can be used for different purposes like joint deliberation, persuasion, negotiation, and conflict resolution. In this paper, we focus on argumentation-based joint deliberation among learning agents. Argumentation-based joint deliberation involves discussion over the outcome of a particular situation or the appropriate course of action for a particular situation. Learning agents are capable of learning from experience, in the sense that past examples (situations and their outcomes) are used to predict the outcome for the situation at hand. However, since individual agents experience may be limited, individual knowledge and prediction accuracy is also limited. Thus, learning agents that are capable of arguing their individual predictions with other agents may reach better prediction accuracy after such an argumentation process.

In this paper we address the issue of joint deliberation among two learning agents using an argumentation framework. Our assumptions are that these two agents work in the same domain using a shared ontology, they are capable of learning from examples, and they interact following a specific interaction protocol. In this paper, we will propose an argumentation framework for learning agents, and an individual policy for agents to generate arguments and counterarguments.

Existing argumentation frameworks for multiagent systems are based on deductive logic. An argument is seen as a logical statement, while a counterargument is an argument offered in opposition to another argument. However, these argumentation frameworks are not designed for learning agents, since they assume a fixed knowledge base. Learning agents, however may induce several generalizations that are consistent with the examples seen at a particular moment in time; the *bias* of the generalization technique used determines which of the valid generalizations is effectively hold by a learning agent.

Agents having learning capabilities allows a new form of counterargument, namely the use of *counterexamples*. Counterexamples offer the possibility of agents learning *during* the argumentation process, and thus improving their performance (both individual and joint performance). Moreover, learning agents will allow us to design individual agents policies to generate adequate arguments and counter arguments. Existing argumentation frameworks focus on how to deal with contradicting arguments, but few address the problem of how to generate adequate arguments. Thus, their focus is on the issue defining a preference relation over two contradicting arguments; however for learning agents we will need to address two issues: (1) how to define a preference relation over two conflicting arguments, and (2) how to define a policy to generate arguments and counterarguments.

In this paper we present a case-based approach to address both issues. The agents use case-based reasoning (CBR) to learn from past cases (where a case is a situation and its outcome) in order to predict the outcome of a new situation; moreover, the reasoning needed to support the argumentation process will also be based on cases. In particular, both the preference relation among arguments and the policy for generating arguments and counterarguments will be based on cases. Finally, we propose an interaction protocol called AMAL2 to support the argumentation process among two agents to reach a joint prediction over a specific situation or problem.

In the remainder of this paper we are going to introduce the multi-agent CBR ($\mathcal{M}$AC) framework in which we perform our research (Section 2). In this framework, Section 2.1 introduces the idea of justifications. After that, Section 3 provides a specific definition of arguments and counterarguments that we will use in the rest of the paper. Then, Section 4 defines a preference relation between induced arguments. Sections 5 and 5.1 presents specific policies to generate both arguments and counterarguments. Using the previous definitions, Section 6 presents a protocol called AMAL2 to allow two agents to solve a problem in a collaborative way using argumentation. Finally, Section 7 presents an empir-

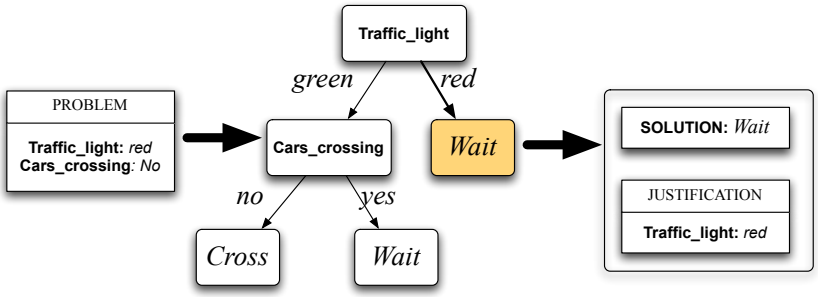

Fig. 1. An example of justification generation using a decision tree.

ical evaluation of the argumentation protocol presented. The paper closes with related work and conclusions sections.

2 Multi-Agent CBR Systems

In this section we are going to define the multi-agent learning framework in which our research is performed [8].

Definition 1. *A* Multi-Agent Case Based Reasoning System *(*$\mathcal{M}$AC*)* $\mathcal{M} = \{(A_1, C_1), ..., (A_n, C_n)\}$ *is a multi-agent system composed of* $\mathcal{A} = \{A_i, ..., A_n\}$*, a set of CBR agents, where each agent* $A_i \in \mathcal{A}$ *possesses an individual case base* C_i*.*

Each individual agent A_i in a $\mathcal{M}$AC is completely autonomous and each agent A_i has access only to its individual and private case base C_i. A case base $C_i = \{c_1, ..., c_m\}$ is a collection of cases. Each agent has (in general) its own CBR method(s) to solve problems using the cases stored in its individual case base. Agents in a $\mathcal{M}$AC system are able to individually solve problems, but they can also collaborate with other agents to solve problem in a collaborative way.

In this framework, we will restrict ourselves to analytical tasks, i.e. tasks, like classification, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Therefore, a case $c = \langle P, S \rangle$ can be defined as a tuple containing a case description P and a solution class $S \in \mathcal{S}$. In the following, we will use the terms *problem* and *case description* indistinctly. Therefore, we can say that a case consists of a case description plus a solution class, or that a case is a problem/solution pair. Moreover, we will use the dot notation to refer to elements inside a tuple. e.g., to refer to the solution class of a case c, we will write $c.S$.

2.1 Justifications in Multi-Agent Systems

Many expert and CBR systems have an explanation component [13]. The explanation component is in charge of justifying why the system has provided a

specific answer to the user. The line of reasoning of the system can then be examined by a human expert, thus increasing the reliability of the system.

Most of the existing work on explanation generation focuses on generating explanations to be provided to the user. However, in our approach we will use explanations (or justifications) as a tool for improving communication and coordination among agents. In our work, we are interested in justifications since they can be used as arguments. For that purpose, we take benefit from the ability of some machine learning methods to provide more information than just the solution class, i.e. the ability to provide justifications.

Definition 2. *A* justification *built by a CBR method to solve a problem P that has been classified into a solution class S_k is a description that contains the relevant information that the problem P and the retrieved cases $C_1, ..., C_n$ (all belonging to class S_k) have in common.*

For example, Figure 1 shows a justification build by a decision tree for a toy problem. In the figure, a problem has two attributes (*traffic_light*, and *cars_crossing*), after solving it using the decision tree shown, the predicted solution class is *wait*. Notice that to obtain the solution class, the decision tree has just used the value of one attribute, *traffic_light*. Therefore, the justification must contain only the attribute/value pair shown in the figure. The values of the rest of attributes are irrelevant, since whatever their value the solution class would have been the same.

In general, the meaning of a justification is that all (or most of) the cases in the case base of an agent that satisfy the justification (i.e. all the cases that are *subsumed* by the justification) belong to the predicted solution class. In the rest of the paper, we will use $\sqsubseteq$ to denote the subsumption relation. In our work, we use LID [1], a CBR method capable of building symbolic justifications. LID uses the feature term formalism to represent cases (*Feature Terms* (or ψ-terms) are a generalization of the first order terms).

When an agent provides a justification for a prediction, the agent generates a *justified prediction*:

Definition 3. *A* justified prediction *is a tuple $\langle A, P, S, D \rangle$ containing the problem P, the solution class S found by the agent A for the problem P, and the justification D that endorses S as the correct solution for P.*

Justifications can have many uses for CBR systems [6, 7]. In this paper, we are going to use justifications as arguments, in order to allow agents to engage learning based argumentation processes.

3 Argumentation in Multi-agent Learning

Let us start by presenting a definition of argument, that we will use in the rest of the paper:

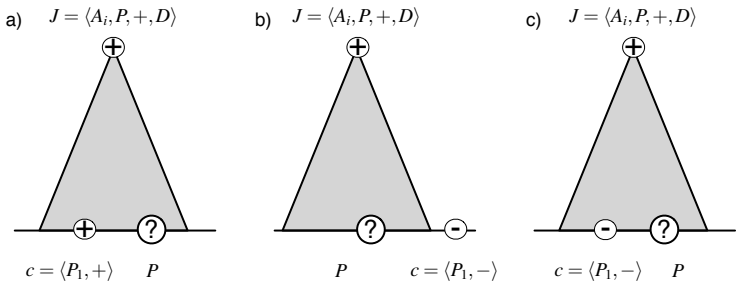

Fig. 2. Relation between cases and justified predictions. The case c is a counterexample of the justified prediction J in c), while it is not in a) and b).

Definition 4. *An* argument α *generated by an agent* A *is composed of a statement* S *and some evidence* E *that endorses that* S *is true.*

In the remainder of this section we will see how this general definition of argument can be instantiated in specific kind of arguments that the agents can generate. In the context of $\mathcal{M}$AC systems, agents argue about the correct solution of new problems and can provide information in two forms:

- A specific case: $\langle P, S \rangle$,
- A justified prediction: $\langle A, P, S, D \rangle$.

In other words, agents can provide specific cases or generalizations induced from that cases, since we are in an learning framework. Using this information, and having in mind that agents will only argue about the correct solution of a given problem, we can define three types of arguments: justified predictions, counterarguments, and counterexamples.

- A *justified prediction* α is generated by an agent A_i to argue that A_i believes that the correct solution for a given problem P is $\alpha.S$, and the evidence provided is the justification $\alpha.D$. In the example depicted in Figure 1, an agent A_i may generate the argument $\alpha = \langle A_i, P, Wait, Traffic\_light = red \rangle$, meaning that the agent A_i believes that the correct solution for P is $Wait$ because the attribute *Traffic_light* equals *red*.
- A *counterargument* β is an argument offered in opposition to another argument α. In our framework, a counterargument consists of a justified prediction $\langle A_j, P, S', D' \rangle$ generated by an agent A_i with the intention to rebut an argument α generated by another agent A_j, that endorses a different solution class than α for the problem at hand and justifies this with a justification D'. In the example depicted in Figure 1, if an agent generates the argument $\alpha = \langle A_i, P, Walk, Cars\_crossing = no \rangle$, an agent that thinks that the correct solution is *Stop* might answer with the counterargument $\beta = \langle A_i, P, Stop, Cars\_crossing = no \wedge Traffic\_light = red \rangle$, meaning that while it is true that there are no cars crossing, the traffic light is red, and thus the street cannot be crossed.

– A *counterexample* c is a case that contradicts an argument α. Specifically, for a case c to be a counterexample of an argument α, the following conditions have to be met: $\alpha.D \sqsubseteq c$ and $\alpha.S \neq c.S$. Figure 2 illustrates the concept of a counterexample. In this figure, justified predictions are shown as triangles and the bottom of the triangles represents the specific cases subsumed by the justified predictions. Figure 2 presents three illustrations: In a) c is not a counterexample of α since the solution of c is the solution predicted by α; in b) c is not a counterexample of α since c is not subsumed by the justification $\alpha.D$; finally, in c) c is a counterexample of α).

Moreover, when two agents provide conflicting arguments (i.e. justified predictions that endorse different solution classes), a preference relation is required to decide which of both is considered the valid one. A typical preference relation used in argumentation is the *specificity* criterion [9], that basically says that the argument that contains more information should be preferred. However, in this paper we are going to present a preference relation specifically designed for learning agents (explained in Section 4).

Finally, notice that when an agent generates a counterargument β to an argument α, it does so with the expectation that the counterargument β is preferred to α. Thus, the agents need a specific policy to generate preferable counterarguments. In this paper we are going to propose a specific policy to generate such counterarguments (explained in Section 5.1).

By exchanging arguments, counterarguments and counterexamples, agents can argue about the correct solution of a given problem. However, in order to do so, they need a specific interaction protocol, a preference relation between arguments, and a decision policy to generate counterarguments. In the following sections we will present these three elements.

4 Preference Relation

The argument that an agent provides might not be consistent with the information known to other agents (or even to some of the information known by the agent that has generated the justification due to noise in training data). For that reason, we are going to define a preference relation over contradicting justified predictions based on cases. Intuitively, we will define a *confidence* measure for each justified prediction (that takes into account the cases owned by each agent), and the justified prediction with the highest confidence is the preferred one.

The confidence of justified predictions is assessed by the agents via an examination procedure. The idea behind examination is to count how many of the cases in an individual case base *endorse* the justified prediction, and how many of them are counterexamples of that justified prediction. The more endorsing cases, the higher the confidence; and the more the counterexamples, the lower the confidence.

Specifically, to examine a justified prediction α, an agent obtains the set of cases contained in its individual case base that are subsumed by $\alpha.D$. The more

of these cases that belong to the same solution class $\alpha.S$ predicted by α, the higher the confidence will be. After examining a justified prediction α, an agent A_i obtains the *aye* and *nay* values:

- $Y_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S = c.S\}|$ is the number of cases in the agent's case base *subsumed* by the justification $\alpha.D$ that belong to the solution class $\alpha.S$ proposed by α,
- $N_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S \neq c.S\}|$ is the number of cases in the agent's case base *subsumed* by justification $\alpha.D$ that *do not* belong to that solution class.

When two agents A_1 and A_2 want to assess the confidence on a justified prediction α made by one of them, each of them examines the predictions and sends the aye and nay values obtained to the other agent. Then, both agents have the same information and can assess the confidence value for the justified prediction as follows:

$$\mathsf{C}(\alpha) = \frac{Y_\alpha^{A_1} + Y_\alpha^{A_2} + 1}{Y_\alpha^{A_1} + Y_\alpha^{A_2} + N_\alpha^{A_1} + N_\alpha^{A_2} + 2}$$

i.e. the confidence on a justified prediction is the number of endorsing cases divided by the number of endorsing cases plus counterexamples found by each of the two agents. Notice that we add 1 to the denominator, the reason is to avoid excessively high confidences to justified predictions whose confidence has been computed using a small number of cases (in this way, a prediction endorsed by 2 cases and with no counterexamples has a lower confidence than a prediction endorsed by 10 cases with no counterexamples). Notice that this correction follows the same idea than the Laplace correction to estimate probabilities (only that we are just interested on preventing overestimation of the confidence).

Thus, the preference relation used in our framework is the following one: a justified prediction α is preferred over another one β if $C(\alpha) \geq C(\beta)$.

5 Generation of Arguments

In our framework, arguments are generated by the agents using learning algorithms. Any learning method able to provide a justified prediction can be used to generate arguments. For instance, decision trees and LID [1] are suitable learning methods.

Thus, when an agent wants to generate an argument endorsing that a specific solution class is the correct solution for a given problem P, it generates a justified prediction as explained in Section 2.1.

5.1 Generation of Counterarguments

When an agent A_i generates a counterargument β to rebut an argument α, A_i expects that β is preferred over α. With that purpose, in this section we are

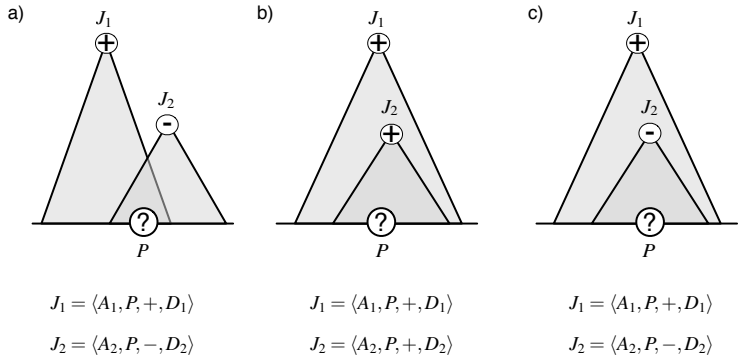

Fig. 3. Relation between arguments.

going to present a specific policy to generate counterarguments based on the *specificity* criterion [9].

The specificity criterion is widely used in deductive frameworks for argumentation, and states that between two conflicting arguments, the most specific should be preferred since it is, in principle, more informed. Thus, counterarguments generated based on the specificity criterion are expected to be preferable (since they are more informed) to the arguments they try to rebut. However, notice that since in this work we use a preference relation based on confidence we cannot guarantee that counterexamples generated based on specificity are always going to be preferred.

Therefore, when an agent wants to generate a counterargument β to an argument α, it will generate a counterargument that it is more specific than α. Figure 3 illustrates this idea. In Figure 3.c) β is a counterargument of α, and is more specific than α. However in Figure 3.a) β is not more specific than α and in Figure 3.c) both arguments endorse the same solution, and thus β is not a counterargument of α.

The generation of counterarguments using the specificity criterion impose some restrictions over the learning method, although LID or ID3 can be easily adapted to generate counterarguments. For instance, to adapt LID we can do the following: LID is an algorithm that generates a description starting by the empty term and heuristically adding features to that term. Thus, at every step, the description is more specific, and the number of cases that are subsumed by that description is reduced. When the description only covers cases of a single solution class, LID terminates and predicts that solution class. To generate a counterargument to an argument α LID just has to use as starting point the description $\alpha.D$ instead of the empty term. In that way, the justification provided by LID will always be subsumed by $\alpha.D$, and thus the resulting counterargument will be more specific than α. However, notice that LID may sometimes not be able to generate counterarguments, since may be the description $\alpha.D$ cannot be

specified any further, or because the agent does not contain any cases subsumed by $\alpha.D$ to run LID.

Moreover, notice that agents can also try to rebut the other agent's arguments using counterexamples. Specifically, in our experiments, when an agent A_i wants to rebut an argument α, uses the following policy:

1. The agent A_i tries to generate a counterargument β more specific than α (in our experiments agents use LID). If the A_i succeeds, β is sent to the other agent as a counterargument of α.
2. If not, then A_i searches for a counterexample $c \in C_i$ of α in its individual case base C_i. If a case c is found, then c is sent to the other agent as a counterexample of α.
3. If no counterexamples are found, then A_i cannot rebut the argument α.

Next section presents the interaction protocol we propose to perform argumentation in our learning framework.

6 Argumentation-based Multi-Agent Learning

In this section we will present the Argumentation-based Multi-Agent Learning Protocol for 2 agents (AMAL2). The idea behind AMAL2 is to allow a pair of agents to solve a problem using learning methods, and that the joint solution provided is coherent with the information known by both agents.

At the beginning of the protocol, both agents will make their individual predictions for the problem at hand. Then, the protocol establishes rules so that if one of the agents does not agree with the prediction of the other, it can provide a counterargument. Then, the other agent can respond with another counterargument, and so on.

In the remainder of this section we will present all the elements of the AMAL2 protocol. First, we will formally present the specific performatives that the individual agents will use in the AMAL2 protocol, that will allow them to *state* a prediction, to *rebut* an argument, and to *withdraw* an argument that the other agent's arguments have rendered invalid. Then, we will present the AMAL2 protocol.

6.1 Protocol Performatives

During the AMAL2 protocol, each agent will propose arguments and counterarguments to argue about which is the correct solution for a specific problem P. The AMAL2 protocol consists on a series of rounds. In the initial round, both agents state with are their individual predictions for P. Then, at each iteration an agent can try to rebut the prediction made by the other agent, or change its own prediction. Therefore, at each iteration, each of the two agents holds a prediction that it believes is the correct one.

We will use $H_t = \langle \alpha_1, \alpha_2 \rangle$ to note the pair of predictions that each agent holds at a round t. When at a certain iteration an agent changes its mind and

changes the prediction it is holding (because the counterarguments of the other agent has convinced him), it has to inform the other agent using the withdraw performative.

At each iteration, agents can send the following performatives to the other agent:

- $assert(\alpha)$: meaning that the justified prediction that the agent is holding for the next round will be α.
- $rebut(\alpha, \beta)$: meaning that the agent has found a counterargument or a counterexample α to the prediction β.
- $withdraw(\alpha)$: meaning that the agent is retiring a justified prediction α, since the counterarguments presented by the other agent have render it invalid.

In the next section the AMAL2 protocol is presented that uses the performatives presented in this section.

6.2 Argumentation Protocol

The AMAL2 protocol among two agents A_1 and A_2 to solve a problem P works in a series of rounds. We will use t to denote the current round (initially $t = 0$). The idea behind protocol is the following one: initially, each agent makes its individual prediction. Then, the confidence of each prediction is assessed, and the prediction with the highest confidence is considered the winner. However, if the agent that has provided the prediction with lower confidence doesn't agree, it has the opportunity to provide a counterargument. Agents keep exchanging arguments and counterarguments until they reach an agreement or until no agent is able to generate more counterexamples. At the end of the argumentation, if the agents have not reached an agreement, then the prediction with the highest confidence is considered the joint prediction.

Notice that the protocol starts because one of the two agents receives the problem to be solved, and that agent sends the problem to the other agent requesting him to argue about the correct solution of the problem. Thus, after both agents know the problem P to solve, round $t = 0$ of the protocol starts:

1. Initially, each one of the agents individually solves P, and builds a justified prediction (A_1 builds α_1^0, and A_2 builds α_1^0). Then, each agent A_i sends the performative $assert(\alpha_i^0)$ to the other agent. Thus, both agents know $H_0 = \langle \alpha_1^0, \alpha_2^0 \rangle$.
2. Then, the agents use the preference relation (presented in Section 4) to determine which of the two arguments in H_t is the preferred argument. After that, the agent that has provided the non preferred argument may try to rebut the other agent's argument. Each individual agent uses its own policy to rebut arguments:
   - If an agent A_i generates a counterargument α_i^{t+1}, then it sends the following performatives to the other agent, A_j, in a single message: $rebut(\alpha_i^{t+1}, \alpha_j^t)$, $withdraw(\alpha_i^t)$, $assert(\alpha_i^{t+1})$. This message starts a new round $t + 1$, and the protocol moves back to step 2.

- If an agent A_i selects a counterexample c of the other agent's justified prediction, then, A_i sends the following performative to the other agent, A_j: $rebut(c, \alpha_j^t)$. The protocol moves to step 3.
- If no agent provides any argument, the protocol ends, and the prediction of H_t with the higher confidence is considered the joint prediction.

3. The agent A_j that has received the counterexample c retains it, and generates a new argument α_j^{t+1} that takes into account the new case. To inform A_i of the new argument, A_j sends A_i the following performatives: $withdraw(\alpha_j^t)$, $assert(\alpha_j^{t+1})$. This message starts a new round $t+1$, and the protocol moves back to step 2.

Moreover, in order to avoid infinite iterations, if an agent sends twice the same argument or counterargument, the protocol ends.

7 Experimental Evaluation

In this section we empirically evaluate the AMAL2 argumentation protocol. We have made experiments in two different data sets: *sponge*, and *soybean*. The sponge data set is a marine sponge classification problem, contains 280 marine sponges represented in a relational way and pertaining to three different orders of the Demospongiae class. The soybean data set is a standard data sets from the UCI machine learning repository. Specifically, the the soybean data set has 307 examples pertaining to 19 different solution classes.

In an experimental run, training cases are distributed among the agents without replication, i.e. there is no case shared by two agents. In the testing stage problems arrive randomly to one of the agents. The goal of the agent receiving a problem is to identify the correct solution class of the problem received.

Each experiment consists of a 10-fold cross validation run. Specifically, an experimental run consists of the following steps:

1. The training set is distributed among the two agents.
2. The testing cases are send one by one to one of the two agents at random.
3. Notice that during argumentation, each of the agents may learn new cases (send as counterexamples by the other agent). In our experiments, each agent forgets all the cases learned during the solution of a problem after each argumentation protocol. This is done since the retention of cases in the initial cases of the test set may alter the answers that the agents provide for the ulterior cases of the test set (it remains as future work to see how the agents evolve as they learn new cases by solving problems together).

Moreover, we have made experiments in four different scenarios: in the first scenario, a 100% of the cases of the training set are distributed among the agents; in the second scenario, the agents only receive a 75% of the training cases; in the third scenario, they only receive a 50%; finally in the fourth scenario agents only receive a 25% of the training cases. We have made those experiments to

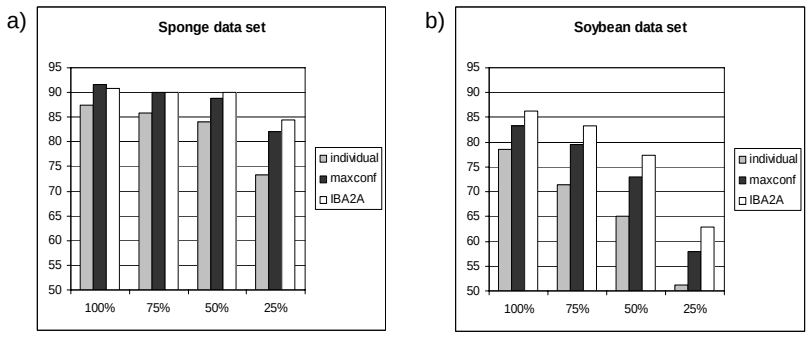

Fig. 4. Classification accuracy results in the Sponge and Soybean domains.

see how the argumentation protocol (and how the argument generation policies) work when the agents have different amount of data.

Figures 4.a and 4.a show the classification accuracy achieved by agents using the AMAL2 argumentation protocol in the sponge and soybean data sets. For each of the 4 scenarios (100%, 75%, 50% and 25%) three bars are shown: *individual*, *maxconf* and *AMAL2*. The individual bar represents the classification accuracy achieved by agents solving problems individually, the maxconf bar represents classification accuracy of the two agents using the following simple strategy: both agents solve the problem individually, then they evaluate the confidence of both predictions, and the prediction with the highest confidence is selected (notice that this is equivalent to using the AMAL2 protocol without any agent providing any counterargument). Finally, the AMAL2 bar represents the classification accuracy of the two agents using the AMAL2 protocol.

Figures 4.a and 4.b show several things. First, that using collaboration is always beneficial, since both maxconf and AMAL2 systematically outperform the individual agents in terms of accuracy. Moreover, both figures also show that the accuracy achieved by AMAL2 is higher than the accuracy achieved by maxconf in most of the experiments (in fact, AMAL2 is better or equal than maxconf in all the experiments except in the 100% scenario of the sponge data set). Moreover, the less data the individual agents have the greater the benefits of AMAL2 are. When each individual agent has enough data, then predictions and confidence estimations are reliable, and thus little or nothing is gained form the argumentation. However, when agents have access to limited data, the argumentation process helps them finding predictions that take into account more information, thus making them more accurate.

8 Related Work

Research on MAS argumentation focus on several issues like a) logics, protocols and languages that support argumentation, b) argument selection and c) argument interpretation. Approaches for logic and languages that support ar-

gumentation include defeasible logic [2] and BDI models [12]. Although argument selection is a key aspect of automated argumentation (see [11] and [12]), most research has been focused on preference relations among arguments. In our framework we have addressed both argument selection and preference relations using a case-based approach.

Concerning CBR in a multiagent setting, the first research was on "negotiated case retrieval" [10] among groups of agents. Our work on multiagent case-based learning started in 1999 [4]; later Mc Ginty and Smyth [5] presented a multi-agent collaborative CBR approach (CCBR) for planning. Finally, another interesting approach is *multi-case-base reasoning* (MCBR) [3], that deals with distributed systems where there are several case bases available for the same task and addresses the problems of cross-case base adaptation. The main difference is that our $\mathcal{M}$AC approach is a way to distribute the *Reuse* process of CBR (using a voting system) while *Retrieve* is performed individually by each agent, while the other multi-agent CBR approaches focus on distributing the Retrieve process.

9 Conclusions and Future Work

In this paper we have presented a learning framework for argumentation. Specifically, we have presented AMAL2, a protocol that allows two agents to argue about the solution of a given problem. Finally, we have empirically evaluated it showing that the increased amount of information that the agents use to solve problems thanks to the argumentation process increases their problem solving performance, and specially when the individual agents have small amount of information.

The main contributions of this work are: a) an argumentative framework for learning agents; b) a case based preference relation over arguments, based on computing an joint confidence estimation of arguments (this preference relation has sense in this learning framework since arguments are learnt from examples); c) a specific and efficient policy to generate arguments and counterarguments based on the specificity relation (commonly used in argumentative frameworks); and d) a principled usage of counterexamples in the argumentation process e) a specific argumentation protocol for pairs of agents that collaborate to decide the correct solution of a given problem.

Moreover, the work presented in this paper concerns only pairs of agents. However, as future work we plan to generalize the AMAL2 protocol to work with a larger number of agents. A possibility to do that is a token based protocol where the agent owner of the token engages on 1-to-1 argumentation with every other agent that disagrees with him. When all this 1-to-1 argumentation have finished, the token passes to the next agent. This should continue until no agent engages any 1-to-1 argumentation. Then, from the outcome of all the 1-to-1 argumentation processes, a joint solution will be predicted.

Acknowledgments. This research was partially supported by the CBR-ProMusic project TIC2003-07776-C02-02.

References

[1] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Proceedings of the 10th European Conference on Machine Learning, ECML'2001*, pages 13–24, 2001.

[2] Carlos I. Chesñevar and Guillermo R. Simari. Formalizing Defeasible Argumentation using Labelled Deductive Systems. *Journal of Computer Science & Technology*, 1(4):18–33, 2000.

[3] D. Leake and R. Sooriamurthi. Automatically selecting strategies for multi-case-base reasoning. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning: Proceedings of the Fifth European Conference on Case-Based Reasoning*, pages 204–219, Berlin, 2002. Springer Verlag.

[4] Francisco J. Martín, Enric Plaza, and Josep-Lluis Arcos. Knowledge and experience reuse through communications among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering*, 9(3):319–341, 1999.

[5] Lorraine McGinty and Barry smyth. Collaborative case-based reasoning: Applications in personalized route planning. In I. Watson and Q. Yang, editors, *ICCBR*, number 2080 in LNAI, pages 362–376. Springer-Verlag, 2001.

[6] Santi Ontañón and Enric Plaza. Justification-based multiagent learning. In *Int. Conf. Machine Learning (ICML 2003)*, pages 576–583. Morgan Kaufmann, 2003.

[7] Enric Plaza, Eva Armengol, and Santiago Ontañón. The explanatory power of symbolic similarity in case-based reasoning. *Artificial Intelligence Review*, 24(2):145–161, 2005.

[8] Enric Plaza and Santiago Ontañón. Ensemble case-based reasoning: Collaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *In Case-Based Reasoning Research and Development: ICCBR-2001*, number 2080 in LNAI, pages 437–451. Springer-Verlag, 2001.

[9] David Poole. On the comparison of theories: Preferring the most specific explanation. In *IJCAI-85*, pages 144–147, 1985.

[10] M V Nagendra Prassad, Victor R Lesser, and Susan Lander. Retrieval and reasoning in distributed case bases. Technical report, UMass Computer Science Department, 1995.

[11] K. Sycara S. Kraus and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence Journal*, 104:1–69, 1998.

[12] N. R. Jennings S. Parsons, C. Sierra. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8:261–292, 1998.

[13] Bruce A. Wooley. Explanation component of software systems. *ACM CrossRoads*, 1998.