Argumentation-based Example Interchage for Multiagent Induction

Santiago ONTAÑÓN [1], and Enric PLAZA
Artificial Intelligence Research Institute, IIIA-CSIC

Abstract. Argumentation can be used by a group of agents to discuss about the validity of hypotheses. In this paper we propose an argumentation-based framework for multiagent induction, where two agents learn separately from individual training sets, and then engage in an argumentation process in order to converge to a common hypothesis about the data. The result is a multiagent induction strategy in which the agents minimize the set of examples that they have to exchange (using argumentation) in order to converge to a shared hypothesis. The proposed strategy works for any induction algorithm which expresses the hypothesis as a set of rules. We show that the strategy converges to a hypothesis indistinguishable in training set accuracy from that learned by a centralized strategy.

Keywords. Multiagent learning, induction, argumentation

Introduction

Multiagent induction is the problem of learning a hypothesis or model (such as a set of rules, or a decision tree) from data when the data is distributed among different agents. Some real-life domains involve such forms of distributed data, where data cannot be centralized due to one or several of the following reasons: storage size, bandwidth, privacy, or management issues. Storage size and bandwidth are less a problem nowadays, however, in large data sets they might still be an issue. In this paper we will propose a framework in which agents will use a simple form of argumentation in order to arrive to a model of all the data while minimizing the communication, and specially minimizing the amount of examples exchanged, and ensuring that the hypothesis found is as good as if centralized induction with all the data was used.

Argumentation frameworks can be used in multi-agent systems for different purposes such as joint deliberation, persuasion, negotiation, and conflict resolution [10]. Previous work [7] has shown how argumentation can be used by agents that use lazy learning or case-based reasoning (CBR) techniques. In this paper we introduce a framework where agents that use inductive learning argue about learnt hypotheses. In this framework, agents generate hypotheses locally, and then argue about them until they agree.

Formalizing agent communication as argumentation allows us to abstract away from the induction algorithm used by the agents. Thus, all the strategies presented in this

[1]Corresponding Author: IIIA (Artificial Intelligence Research Institute), CSIC (Spanish Council for Scientific Research), Campus UAB, 08193 Bellaterra, Catalonia (Spain), santi@iiia.csic.es.

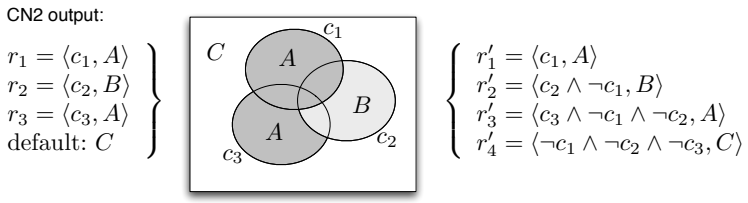

Figure 1. Postprocessing of the rules generated by CN2 in order to remove the order dependencies, and thus fit in our argumentation framework.

paper can work with any induction algorithm that learn hypotheses expressed as a set of independent rules. Algorithms such as ID3 [9] can also be used, since a tree can be easily flattened into a set of rules. Algorithms such as CN2 [4] that learn an *ordered* set of rules also fit in this framework, but rules require some preprocessing to remove the dependencies that the ordering introduces (as elaborated in Section 1). Moreover, the framework is also agnostic in regards to the representation formalism.

This paper is organized as follows. Section 1 presents our multi-agent learning framework. Section 2 presents two strategies for multiagent induction based on argumentation, and Section 3 empirically evaluates them, comparing them to other strategies in the literature. Section 4 provides a quick overview of the related work, and finally the paper closes with conclusions and future work.

1. A Framework for Multi-Agent Learning

Let A_1 and A_2 be two agents who are completely autonomous and have access only to their individual collections of examples, or training sets T_1, and T_2. A training set $T_i = \{e_1, ..., e_n\}$ is a collection of examples. Agents can individually use induction in order to infer a hypothesis (or model) of the data; we will use the terms "model" and "hypothesis" indistinguishably. These hypotheses will be used during argumentation and, given the dynamic nature of the training sets, old hypotheses will be discarded and new ones will be inferred as need be.

Examples, *hypotheses* and *rules* are the three key concepts of the learning framework proposed in this paper. We will restrict ourselves to classification tasks, therefore, an *example* $e = \langle P, S\rangle$ is a pair containing a problem P and a solution S. In the remainder of this paper, we will use the dot notation to refer to elements inside a tuple; e.g., to refer to the solution of an example e, we will write $e.S$.

Our framework is restricted to hypotheses H that can be represented as a set of *rules*: $H = \{r_1, ..., r_m\}$. A rule $r = \langle D, S\rangle$ is composed of a *body* $r.D$, and a *solution*, $r.S$. When a problem P matches the body $r.D$ of a particular rule r, the rule predicts that the solution to the problem P is $r.S$. When a problem matches the body of a rule $r.D$, we say that the rule *subsumes* the problem: $r.D \sqsubseteq P$. A large number of induction algorithms can generate hypothesis that can be represented using rules. Moreover, the framework introduced in this paper does not specify which representation formalism agents use to represent examples. In principle, any data representation (propositional, relational, or any other) could be used.

When using algorithms such as CN2, that produce an ordered set of rules, the rules produced have to be postprocessed in order to remove the order relationship among them.

The left hand side of Figure 1 shows a set of three rules generated by CN2 (plus the default solution assigned by CN2 when no rule covers a problem). The center of Figure 1 shows a graphical representation of the way these rules partition the problem space among the three different solutions A, B, and C (the three circles represent the subset of problems that are subsumed by each of the three conditions in the body of the rules: c_1, c_2 and c_3). Notice for instance that rule r_2 states that all the problems that are subsumed by c_2 have solution B. However, r_2 is only considered if r_1 is not fired. Therefore, that rule is postprocessed and converted into rule r'_2, which states that all examples that are subsumed by c_2, but not by c_1 have solution B. In general, a rule is postprocessed by adding the negations of all the previous rules to its body. Finally, the default solution computed by CN2 is converted also into a rule containing the conjunction of the negation of the body of all the rules. The result of this process is a set of independent rules, which can be used in our framework.

In order to use argumentation, two elements must be defined: the *argument language* (that defines the set of arguments that can be generated), and an *attack relation*. In our framework, the argument language is composed of two kinds of arguments:

- A *rule argument* $\alpha = \langle A, r \rangle$, is an argument generated by an agent A stating that the rule r is true.
- A *counterexample argument* $\beta = \langle A, e, \alpha \rangle$, is an argument generated by an agent A stating that e is a counterexample of (an example contradicting) argument α.

Including additional types of counterarguments, such as "rule counterarguments" is part of future work (see Section 5).

To define the relation among arguments, we have to take into account all the possible different situations that can arise while comparing two arguments consisting of rules or examples. Figure 2 shows all these situations. The top row of Figure 2 considers all the possible comparisons of two rule arguments, r_1 and r_2 such that $r_1.S = r_2.S$. Only three situations might arise: a) r_1 and r_2 are totally unrelated, b) the sets of problems covered by r_1 and r_2 have a non empty intersection, and c) one is more general than the other. The middle row of Figure 2 considers all the possible comparisons of two rule arguments, r_1 and r_2 but this time $r_1.S \neq r_2.S$. The same three situations arise (unrelated, non-empty intersection, and one more general than another). Notice that in the non-empty intersection situation we also require that no rule is more general than another (we don't include the extra restriction in the figure for clarity). Thus, when comparing any two rule arguments, only 6 situations might arise. Situations a), b), c) and d) represent rule arguments that are *compatible*, whereas situations e) and f) represent *conflicting* arguments. Situation c) is a special situation and we say that r_1 *subsumes* r_2.

The third row of Figure 2 shows all the possible situations that arise when comparing a rule argument with a counterexample argument: g) both the counterexample and the rule support the same class, h) in which the counterexample, although supporting the same class, is not covered by the rule, i) where the counterexample supports a different solution than the rule, and the rule covers the counterexample, j) in which the counterexample, although supporting a different class, is not covered by the rule. In our framework, we assume that a counterexample cannot be attacked. Out of the four situations, the counterexample argument only attacks the rule in situation i), where it is called an *attacking counterexample* of r_1.

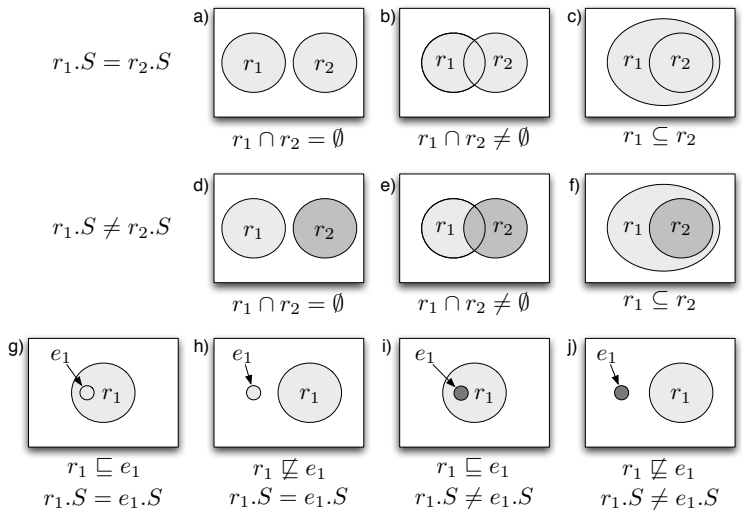

Figure 2. All the possible different situations that can arise while comparing two arguments consisting of rules or counterexamples.

Using these two types of arguments and the *compatible*, *conflicting*, *subsumed*, and *attack* relations among arguments, next section introduces two different multiagent induction strategies.

2. Argumentation-based Multiagent Induction

In this section we will present two strategies, AMAI (Argumentation-based Multiagent Induction) and RAMAI (Reduced Argumentation-based Multiagent Induction). Both strategies are based on the same idea, and share the same high level structure.

1. A_1 and A_2 use induction locally with their respective training sets, T_1 and T_2, and obtain initial hypotheses H_1 and H_2 respectively.
2. A_1 and A_2 argue about H_1, obtaining a new H_1^* derived from H_1 that is consistent with both A_1 and A_2's data.
3. A_1 and A_2 argue about H_2, obtaining a new H_2^* derived from H_2 that is consistent with both A_1 and A_2's data.
4. A_1 and A_2 obtain a final hypothesis $H^* = H_1^* \cup H_2^*$. Remove all the rules that are subsumed by any other rule (situation c) in Figure 2).

Thus, both agents perform induction individually in step 1 and then, in steps 2 and 3 (which are symmetric), the agents use argumentation to refine the individually obtained hypotheses and make them compatible with the data known to both agents. Basically, one agent proposes rules, and the other agent either *accepts* them or sends a counterexample. Finally, when both hypotheses are compatible, a final global hypothesis H^* is obtained as the union of all the rules learned by both agents while removing redundant rules. Notice that, unless the induction algorithms are not able to learn rules with 100% accuracy in

the training set, there should not be any conflicting rules in H^*. AMAI and RAMAI only differ in the way steps 2 and 3 are performed. Step 2 in AMAI works as follows:

2.a Let $H_1^0 = H_1$, and $t = 0$.
2.b If there is any rule $r \in H_1^t$ that has not yet been *accepted* by A_2, then send the argument $\alpha = \langle A_1, r \rangle$ to A_2. Otherwise (all the rules in H_1^t have been accepted) the protocol goes to step 2.e.
2.c A_2 analyzes $\alpha.r$ and tries to find a counterexample that attacks it. A_2 sends the counterargument $\beta = \langle A_2, e, \alpha \rangle$ to A_1 if a counterexample e is found; otherwise r is accepted and the protocol goes back to step 2.b.
2.d When A_1 receives a counterexample argument β, the counterexample $\beta.e$ is added to the training set T_1, and A_1 updates its hypothesis[2] obtaining H_1^{t+1}. The protocol goes back to step 2.b, and $t = t + 1$.
2.e The protocol returns H_1^t.

The main idea is that A_1 infers rules according to its individual training set T_1, and A_2 evaluates them, trying to generate counterarguments to the rules that do not agree with its own individual training set T_2. Step 3 in AMAI is the dual situation where A_2's rules are attacked by A_1's counterexamples. Notice that only one counterexample is exchanged at a time in AMAI. Agents are not allowed to send the same counterexample twice to ensure the convergence of the protocol in case of noisy data.

The second strategy, RAMAI, improves over AMAI in trying to minimize the number of times the hypothesis has to be updated while trying to keep a low number of exchanged counterexamples. Step 2 in RAMAI works as follows:

2.a Let $H_1^0 = H_1$, and $t = 0$.
2.b Let $R^t \subseteq H_1^t$ be the set of rules in the hypothesis of A_1 not yet *accepted* by A_2. If empty, then the protocol goes to step 2.e, otherwise A_1 sends the set of arguments $\mathcal{R}^t = \{\langle A_1, r \rangle | r \in R^t\}$ to A_2.
2.c For each $\alpha \in \mathcal{R}^t$, A_2 determines the set of examples C_α in its training set that are attacking counterexamples of $\alpha.r$: $C_\alpha = \{e \in T_2 | \alpha.r.D \sqsubseteq e.P \wedge \alpha.r.S \neq e.S\}$. For each argument $\alpha \in \mathcal{R}^t$ such that $C_\alpha = \emptyset$, A_2 accepts rule $\alpha.r$. Let $I^t \subseteq \mathcal{R}^t$ be the subset of arguments for which A_2 could find attacking counterexamples. A_2 computes the minimum set of counterexamples B^t such that $\forall \alpha \in I^t,\ C_\alpha \cap B^t \neq \emptyset$, i.e. the minimum subset of examples that can attack all arguments in I^t. A_2 sends the set of counterexample arguments $\mathcal{B}^t$ consisting of a counterexample argument $\beta = \langle A_2, e, \alpha \rangle$ for each pair e, α such that $e \in B^t$, $\alpha \in I^t$, and β attacks α.
2.d When A_1 receives a set of counterexample arguments $\mathcal{B}^t$, it adds their counterexamples to its training set T_1, and updates its inductive hypothesis, obtaining H_1^{t+1}. The protocol goes back to step 2.b, and $t = t + 1$.
2.e The protocol returns H_1^t.

As before, Step 3 in RAMAI is just the dual of Step 2. The idea behind RAMAI is that an example can be an attacking counterexample of more than one rule at the same time. RAMAI computes the minimum set of examples that attacks all the rules in I^t and sends them all at once. Therefore, the number of times the hypothesis has to be updated is likely reduced (see experiments in Section 3).

[2] If the induction algorithm of A_1 is not incremental, then A_1 can use induction from scratch with the new extended training set that includes e.

3. Experimental Evaluation

In order to evaluate our approach, we tested the multiagent induction strategies in four different data sets from the Irvine machine learning repository: three propositional ones (soybean, zoology, cars), and a relational one (demospongiae). For demospongiae, we used a subset consisting of the axinellida, adromerida and astrophorida classes. Moreover, we tested it using three different induction algorithms: ID3 [9], CN2 [4] and INDIE (a relational inductive learner [2]). This base techniques for induction are applied to three multiagent induction strategies: Individual (where agents just do induction individually), Union (where agents do induction individually, and then they put together all the rules they learn into one common hypothesis), and DAGGER [5] (the only other distributed induction technique independent of the learning algorithm to the best of our knowledge, see Section 4 for a brief explanation of DAGGER). We also compared the results against Centralized induction (one sole agent having all data). We evaluated convergence, time, number of examples exchanged, number of rules exchanged, number of induction calls, and both training and test set accuracy. All the results presented are the average of 10 fold cross validation runs.

Since demospongiae is a relational data set, it has to be converted to propositional so that ID3 and CN2 can use it. In the demospongiae data set, examples are represented as trees, we computed the set of all possible different branches that the examples have, and each one is converted to a feature (70 different features are defined in this way). Each example consists of about 30 to 50 features each, so there is a large amount of missing values in the resulting propositional representation. Thus, both ID3 and CN2 have troubles learning in this domain. CN2 does, in fact, a better job, but ID3 achieves a very low classification accuracy. Additionally, since the basic ID3 cannot handle missing values, all missing values where considered to have the special value "missing" when the data set was used by ID3. For CN2, a beam size of 3 was used in all the experiments.

Table 1 presents the classification accuracy comparison. We ran each combination of induction algorithm (ID3, CN2, INDIE) with multiagent induction strategy (Centralized, AMAI, RAMAI, Individual, Union and DAGGER) with all the data sets (except the combination of INDIE-DAGGER, that is not possible, since DAGGER assumes propositional data sets, and INDIE requires them in relational form). In each experimental run the training set was randomly split among the two agents, forming their individual training sets (except in the case of the Centralized strategy, where there was only one agent). Accuracy is measured in the original training set (with 90% of the examples), and also in the remaining 10% of the test set.

The training set accuracy results confirm is that the hypotheses learnt by AMAI and RAMAI are indistinguishable in training set accuracy from those learnt by using Centralized induction, achieving a 100% accuracy every time where Centralized induction also does. When agents perform Individual induction, having less data, accuracy diminishes; agents using the Union strategy improve their accuracy with respect to an individual strategy, but still it is not guaranteed to be as good as that of Centralized accuracy. DAGGER shows good accuracy (although not guaranteeing that of Centralized induction).

Analyzing test set accuracy (accuracy over unseen examples), we observe that, except in a few cases where DAGGER achieves higher accuracy (and one where Union does), AMAI and RAMAI achieve same or higher accuracy than the Centralized approach. Table 1 shows the highest results for each induction algorithm in boldface (when the difference was not statistically significant, more than one result is highlighted).

Table 1. Training and test accuracy measurements of different multiagent induction strategies combined with different induction algorithms.

	Training				Test			
	Soyb.	Zool.	Cars	Demosp.	Soyb.	Zool.	Cars	Demosp.
ID3	100.00	100.00	100.00	99.44	85.00	**99.00**	88.95	58.57
ID3-AMAI	100.00	100.00	100.00	99.70	**88.50**	**99.00**	88.95	58.21
ID3-RAMAI	100.00	100.00	100.00	99.74	87.67	**99.00**	89.24	58.21
ID3-individual	85.67	93.85	93.84	80.20	76.50	90.00	86.84	55.54
ID3-union	90.25	94.73	97.73	94.05	81.00	94.00	**90.99**	60.36
ID3-DAGGER	99.57	100.00	76.36	99.76	80.67	92.50	68.95	**62.50**
CN2	100.00	100.00	100.00	100.00	**84.66**	94.00	**80.64**	**78.57**
CN2-AMAI	100.00	100.00	100.00	100.00	**84.90**	**93.50**	**80.61**	**79.11**
CN2-RAMAI	100.00	100.00	100.00	100.00	**84.66**	**93.50**	80.17	**78.93**
CN2-individual	87.82	94.62	89.90	88.29	77.83	87.50	**80.84**	74.46
CN2-union	54.91	91.65	80.41	70.71	63.66	86.00	80.00	68.20
CN2-DAGGER	99.49	99.65	95.86	99.88	79.33	92.50	75.34	**78.93**
INDIE	99.64	100.00	100.00	100.00	83.00	**94.00**	81.80	95.00
INDIE-AMAI	99.64	100.00	100.00	100.00	**84.33**	93.00	**91.25**	**95.89**
INDIE-RAMAI	99.64	100.00	100.00	100.00	**84.50**	**94.00**	**91.37**	94.11
INDIE-individual	89.21	94.07	93.93	96.45	77.50	85.50	87.76	54.11
INDIE-union	91.44	96.48	97.42	97.90	78.00	90.00	**91.80**	94.29

Table 2. Time (in seconds) required to complete the induction process per agent, number of examples shared per agent (as a percentage of the number of examples owned by an agent), number of rules sent per agent, and number of times the base induction algorithm had to be invoked per agent (notice that in the Centralized case, there is only one agent). Results are average over all the induction algorithms and all the data sets.

	time	Examples	Rules	Induction calls
Centralized	2.8	100.00%	0.00	1.00
Individual	1.5	0.00%	0.00	1.00
Union	1.5	0.00%	67.63	1.00
DAGGER	3.5	68.56%	64.75	1.50
AMAI	155.4	19.04%	3748.70	58.90
RAMAI	18.2	21.52%	679.34	5.77

Another effect that can be seen is that ID3 and CN2 cannot properly handle the complexity of the demospongiae data set, since, although they can achieve high training set accuracy, the rules they learn do not generalize and achieve very low test set accuracy. INDIE, however, being a relational learner, can handle demospongiae in its native representation formalism, and thus learn much more general rules, that generalize properly, achieving high test set accuracy.

Table 2 shows the amount of time used by each of the different multiagent induction strategies per agent (averaged over all the data sets and induction algorithms), also the percentage of the examples that had to be shared, the number of rules exchanged, and

also the number of times that the agents had to call the base induction algorithm. Notice that time is dominated by the slower learning algorithm (CN2) and the most complex data set (demospongiae), while the fastest algorithm (ID3) required less than a tenth of a second for any strategy except AMAI and RAMAI (where it still required less than a second for any data set). Table 2 shows that AMAI and RAMAI, are the most computationally expensive strategies, AMAI taking 155.4 seconds and RAMAI 18.2, while Centralized accuracy required only 2.8 seconds. However, most of the additional time consumed by AMAI and RAMAI corresponds to multiple invocations of the base induction algorithm after receiving new examples, and RAMAI greatly reduces computational time from AMAI. If an incremental induction algorithm such as ID5R or ITI [12] was used, the amount of time consumed could be further reduced.

Table 2 shows that among all the multiagent induction strategies, DAGGER is the one that requires exchanging the highest percentage of examples, 68.56%, while AMAI and RAMAI exchange only 19.04% and 21.52% respectively. The Union strategy, of course, does not force agents to exchange any example. However, AMAI and RAMAI require the exchange rules to be performed repeatedly, resulting in a larger number of rules being exchanged, whereas other strategies, such as DAGGER, or Union only require exchanging rules once. Comparing AMAI and RAMAI, notice that AMAI exchanges a slightly lower amount of examples, but RAMAI reduces other aspects: requires only a tenth of the time, a fifth of the rules, and a tenth of the number of induction calls.

Summarizing the results, we can conclude that different multiagent induction strategies have different strengths and weaknesses. Performing centralized induction has the problem of having to share all the examples, but achieves a high accuracy. Next in line is DAGGER, which forces the agents to exchange most of their examples, achieving a high accuracy (although not guaranteed to be as high as centralized). On the other extreme, we have the Individual and Union strategies, that have the minimum computational cost, zero example exchange, but also the lowest classification accuracies.

AMAI and RAMAI sit in the middle, requiring the agents to share a small percentage of examples (around 20%), while ensuring the same or higher classification accuracy than centralized induction (especially in the test set, where the hypotheses learnt by AMAI or RAMAI have less overfitting). Thus, the usefulness of of using argumentation to regulate the examples to be interchanged is shown: results a better selection of examples to be exchanged since accuracy is maintained (with respect to the Centralized baseline) by exchanging fewer examples. On the other side, argumentation requires the agents to dynamically reviser their inductive hypotheses, resulting in a higher computational cost. Finally AMAI and RAMAI, we can conclude that RAMAI is the most well balanced strategy, since it requires about a tenth of the computational cost, while only sharing a very small number of additional examples. Consequently, RAMAI shows de feasibility of performing induction on a multiagent scenario, across different inductive techniques and data sets; the Centralized option of sharing all data may not always be always feasible, while in any scenario where sharing a small portion of data is acceptable, then RAMAI is a feasible option.

4. Related Work

Distributed induction has been attempted with four different approaches: computing statistics in a distributed fashion and then aggregating, sharing examples, sharing hy-

potheses or viewing induction as search and distributing the search process. One approach [3] is performing induction from a set of distributed sources, computing a collection of statistics locally in each of the sources, and then aggregating them to learn a model. Some learning algorithms, such as ID3, can be distributed in this way while still guaranteeing that the decision tree found is exactly the same that would be found if all the data were centralized. This approach is restricted to attribute-value representations, while our approach works also for relational learning. Another difference is that they assume a single agent trying to learn from scratch from a collection of distributed sources, while in our framework we assume a multi-agent system with agents that already have an initial hypothesis and improve them by arguing with other agents. Additionally, our research focuses on finding multiagent induction strategies that can be built around standard induction algorithms without modifying them.

The DAGGER approach [5] performs distributed induction by selecting a reduced set of examples from each of the distributed sources, and then performing centralized induction with the union of these sets of examples. DAGGER's proposes a one shot approach that does not ensure preserving classification accuracy, while our strategies do.

Another approach to distributed induction [11], proposes to learn individual models in each of the sources, and then combine them by using a genetic algorithm that uses specialized mutation and crossover operators for being able to merge the hypothesis. The goal of this approach is to distribute the induction task among several agents, so that this parallelization becomes more efficient. Our goal is not to make the induction process more efficient, but to allow groups of agents to perform individual induction while putting together their results by argumentation with the goal of obtaining a joint inductive hypothesis that preserves a high quality. Another example of distributing induction for efficiency is that of distributing the search process of finding rules among a series of distributed processors. Provost and Hennessy [8] propose to perform distributed search for rule learning, where each individual processor only searches with a subset of the data and proposes each candidate rule to the rest for verification.

Concerning argumentation, the idea that argumentation might be useful for machine learning has been proposed by several authors [1,6]. The idea is that hypotheses induced from data can be considered as arguments, and then by defining a proper attack and defeat relations, sound hypotheses can be found. More specifically, the work presented in this paper is complementary to the AMAL argumentation framework for case-based learning [7]. While in AMAI and RAMAI agents collaborate during induction, and then they solve problems individually, in AMAL, agents learn separately, and only collaborate during problem solving. Thus, AMAL is an argumentation model of multi-agent learning based on "solution merging", where as AMAI and RAMAI are based on "hypothesis merging".

5. Conclusions and Future Work

In this paper we have presented AMAI and RAMAI, two different multiagent induction strategies that can be used on top of any induction algorithm capable of learning hypotheses represented using sets of rules. AMAI and RAMAI ensure that the hypothesis learnt will be undistinguishable in terms of training set accuracy from that produced by a centralized approach. The main idea behind AMAI and RAMAI is to let each agent perform induction individually, then argue about the learnt hypotheses to remove inconsistencies, and finally merge both hypotheses.

Experimental results show that, in addition to achieve the same training set accuracy as a centralized method, AMAI and RAMAI obtain hypotheses that are less prone to overfitting, achieving slightly higher test set accuracy. Moreover, AMAI and RAMAI require sharing only a small part of all existing examples (about 20% in our experiments). AMAI and RAMAI also require that the agents perform induction several times, so Incremental techniques for induction could be used to speed the process, but we did not use them to make clear this is not a necessary requirement for our approach.

AMAI and RAMAI use counterexamples as the only form of counterargument. However, we plan to investigate more complex argumentation protocols that let agents use rules also generalizations as counterarguments. The problem of that, is that the base learning algorithms would have to be modified to be able to take rules into account, in addition to the examples in the training sets. This is related to the research in "argument based machine learning" by Možina et al. [6] where they modify the CN2 algorithm to take into account specific rules (arguments) in addition to examples for learning purposes. Additionally, we intend to tackle more complex scenarios: the committee scenario (with n agents deliberating) and k-issue scenario (where learning and argumentation is not about one concept or issue but over a collection of interrelated concepts).

Acknowledgements

This research was partially supported by projects Next-CBR (TIN2009-13692-C03-01) and Agreement Technologies (CONSOLIDER CSD2007-0022).

References

[1] Leila Amgoud and Mathieu Serrurier. Arguing and explaining classifications. In *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007*, pages 164–177, 2007.

[2] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(1):259–294, 2000.

[3] Doina Caragea, Adrian Silvescu, and Vasant Honavar. Decision tree induction from distributed, heterogeneous, autonomous data sources. In *Proc. Conf. on Intelligent Systems Design and Applications (ISDA 03)*, pages 341–350. Springer Verlag, 2003.

[4] Peter Clark and Tim Niblett. The CN2 induction algorithm. In *Machine Learning*, pages 261–283, 1989.

[5] Winston H. E. Davies. *The Communication of Inductive Inference*. PhD thesis, University of Aberdeen, 2001.

[6] Martin Mozina, Jure Zabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10-15):922–937, 2007.

[7] Santiago Ontañón and Enric Plaza. Learning and joint deliberation through argumentation in multiagent systems. In *Proc. AAMAS-07*, pages 971–978, 2007.

[8] Foster John Provost and Daniel N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proc. AAAI-96*, pages 74–79. AAAI Press, 1996.

[9] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[10] Iyad Rahwan, Simon Parsons, and Chris Reed, editors. *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007*, volume 4946 of *LNCS*. Springer, 2008.

[11] Michael J. Shaw and Riyaz Sikora. A distributed problem-solving approach to rule induction: Learning in distributed artificial intelligence systems. Technical Report ADA232822, Carnagie-Mellon University, 1990.

[12] Paul E. Utgoff. An improved algorithm for incremental induction of decision trees. Technical Report 94-072, UMass, 1994.