

Argumentation techniques for multi-agent concept learning

Bachelorscriptie cognitieve kunstmatige intelligentie
7,5 ECTS

Author:
Ruurdje Procee

Supervisor:
dr. Gerard A.W.
Vreeswijk

May 4, 2011

Contents

1	Introduction	1
1.1	Relevance to the field of AI	1
1.2	Overview	2
2	Learning in multi-agent systems	2
2.1	Generalization and Inductive (concept) learning	2
2.2	Multi-agent learning	3
2.3	Feature terms and case bases	4
3	Generating rules and arguments	5
3.1	Rules and examples	6
3.2	Argumentation-based bottom-up induction	6
3.3	Belief Revision	7
4	Argumentation about rules	7
4.1	Argumentation frameworks	8
4.2	Argumentation in A-MAIL	8
5	A-MAIL Interaction protocol	9
6	Conclusion	10
6.1	Results	10
6.2	Further research	11

1 Introduction

In this bachelor thesis, we will discuss an article ([5]) that describes a method for learning concepts: multi-agent inductive learning combined with argumentation techniques. It is a combination of multi-agent learning, inductive learning, case-based learning and argumentation techniques. These different themes will be discussed in the following sections. The question will be whether it is worth the effort to use an algorithm with multiple agents instead of just applying some learning algorithm.

1.1 Relevance to the field of AI

Artificial intelligence is about trying to construct entities that can express some form of intelligent behaviour. Learning is considered as some form of intelligent behaviour, but there are more interesting subjects that are relevant.

The bachelor cognitive artificial intelligence combines linguistics, philosophy, computer science and psychology. All these items will return in the coming sections, although some will only be mentioned briefly. As mentioned above, (concept) learning is interesting for a psychologist and philosopher. From a psychology perspective, it is interesting to try to model human behaviour, like learning, because it can give a better understanding of human functioning.

In the discussed paper, learning is combined with argumentation techniques.

Argumentation is an important quality among philosophers, and it also gives insight in how human reasoning works. From a computer science-perspective, it is a good way to communicate among different entities or agents. It is intuitive from a human-perspective since it resembles human reasoning.

To conclude, many interesting perspectives are involved in learning and argumentation. In the discussed paper, the focus will be on how the learning of concepts can be done by a computer. However, I will elaborate more on some subjects involving the previously discussed perspectives when possible.

1.2 Overview

In the next sections, all the different components of A-MAIL (*Argumentation for Multi-agent Inductive Learning*) will be discussed. Before the start of the in-depth analysis, a short overview is given. In A-MAIL, there are 2 different agents. Each is given half of the examples and will perform induction on it. The agent will produce a hypothesis and will communicate it to the other agent. Then the argumentation rounds will start. In each round, one agent can either give an argument to defeat the other agent's argument or give an example e that is not consistent with the hypothesis at that moment. When the other argument or example is valid (i.e. it defeats the argument), the other agent will change its beliefs. The argumentation ends when no more new arguments are added or when the hypotheses are consistent with both case bases.

Now that we have a short overview, the algorithm will be discussed in more detail. In the coming sections, the different techniques will be described first and next the implementation in A-MAIL is given. In the following section, the relevance of using a multi-agent system is discussed (section 2) and a short overview of inductive concept learning is given. We will continue in section 3 with an overview how learning is applied in A-MAIL. A brief explanation of argumentation techniques and how it is used in A-MAIL is given in section 4. Next, these different components are combined in section 5 and a more specific description on A-MAIL will be given. Finally in the conclusion, section 6, the advantages and problems will be discussed along with ideas for further research.

2 Learning in multi-agent systems

In this section, inductive learning will be explained in more detail followed by an outline of the advantages and problems involved when expanding the learning to multiple agents. Finally, the way the examples and rules are represented in the program is discussed.

2.1 Generalization and Inductive (concept) learning

Inductive learning is learning from examples and making a generalization of these examples. The assumption made is that a hypothesis that approximates the target function in the set of training examples, will also approximate the target function over unobserved examples [3]. An example of inductive learning is:

Every swan we have seen until now is white.
All swans are white.

We have made a generalization that from the observation that all the swans we have seen are white, we can derive that every swan is white.

Concept learning is about learning these derivation rules. When given a set of examples, (here: a set of descriptions of swans), we want to learn a concept (here: swan). Then, when we get a new example, we can infer that it must be a swan or not. Concept learning is a form of *supervised learning*: a lot of examples that are properly classified are given, and rules have to be made to describe the data. Because of the assumption made above, we can apply these rules to unobserved examples.

This is an easy example involving only one property (color), so the rule is quite simple. However, more extensive rules can be made about more difficult concepts that have more properties. The problem with multiple properties is that not every example fits in the rules perfectly. For example: a rule of the concept 'tree' can be: 'a plant with a wooden trunk and leaves'. But there are trees without leaves, like a pine-tree. But it is a tree. When we want to make rules that cover each example perfectly, you have the risk of over fitting: the rules are too much specified to the training examples, so you end up saving all the examples. Therefore, the rules can cover some negative examples or not covering positive examples too. To determine if a rule is good enough, we can assign values to the rules in some way. For example: with 80% statistical confidence, we can say that every swan is white.

2.2 Multi-agent learning

Learning algorithms can be applied directly to the data, but a system with multiple agents involved can be used too. These systems need agents that can learn in some way, a way to talk and communicate about the hypothesis and a way to integrate the opinions [6].

However, it seems easier to just apply induction on the data directly, because no communication and integration methods are necessary. However, according to Sian (1991), learning with multi-agent systems has some advantages:

- Speed: when the example space is really big, it will take a lot of time to learn from a hypothesis. When the examples can be split among the different agents, the learning process will run parallel.
- Scalability: learning from an example space is a task that can easily be split into different components. This makes it easy to add more agents when there is more data.
- Fault tolerance: learning is divided among different agents and they will communicate about this. Because the results will be checked by different agents, this may result to more reliable results.

However, there are some issues as well:

- The information is distributed among the agents, some way is needed to share this information
- It is possible that an agent can't learn some parts of the data because there is not enough information or that the confidence cannot be confirmed.
- Tools are needed to deal with conflict. This means that there should be some way to communicate about it and find a solution for the problem.

Now we have discussed some advantages and problems, we will look at possible solutions. For example, the first problem can be solved in different ways [1]:

1. Centralize the data and apply machine learning
2. Exchange information while learning the data
3. Learn locally and then share the results

The first solution doesn't have any of the advantages mentioned above, but doesn't have all the issues. The second solution needs specific learning algorithms and the third solution is intuitively the most logical solution. In the paper [5] that will be discussed, the third solution is chosen. It has the advantages mentioned above, but also the issues. The chosen solution for dealing with distributed information is using an argumentation framework. Using this, agents can communicate about the things they learned and also discuss it.

As mentioned above, the question is whether it is a problem that the data is distributed among the agents and if it isn't easier to just apply some learning algorithm on the data. In the next sections, the algorithm will be discussed, and in the conclusions this question will be answered.

2.3 Feature terms and case bases

This section contains a short explanation about how the different examples are stored in the program used to execute A-MAIL. In this program, feature terms are used to represent the different examples. The most important aspect that feature terms have, is that the name is relevant instead of the order. For example in

```
mother(x,y)
```

the order of the variables defines who the mother and who the child is. In feature terms, this will be described like:

```
mother[child=x, mother=y]
```

There is also a sort hierarchy with a \leq -relation, for example *family* \leq *mother* because 'family' is more general than 'mother'. This hierarchy is used in describing the data that has to be learned. Data is given from one of the data-sets on zoology, soybeans or sponges.

The data was represented in the representation language NOOS [4]. A case base consists of 3 parts:

- Ontology, a hierarchy of sorts and what features it can have (e.g a robot has a description that includes whether it has a tie, that it can hold an object and other properties)

- Domain model, defines instances a sort or feature can have (e.g. the objects a robot can hold are flags and swords or a robot can wear a tie or not)
- Terms, describing the individual in a specific domain (e.g. robot R1 is wearing a tie and holding a sword)

Using these parts, a new set of data can be described. An example is the following term:

```
(define-episode (robot-description :id R2)
  (smiling true)
  (holding flag)
  (has-tie true)
  (head-shape octagon)
  (body-shape octagon)
)
```

```
(define-episode (robot-problem :id PR2)
  (description R2)
  (solution friendly)
)
```

In this example, the robot is smiling, holding a flag, having a tie and is octagon-shaped. It is a friendly robot.

The aim for the inductive learning algorithm is to make rules to determine whether a robot is friendly (or unfriendly, depending on what the target concept is). An example is the following rule:

```
(define (robot-description)
  (head-shape (define (shape))))
  (has-tie false)
  (body-shape (define (shape)))
  (smiling (define (boolean))))
```

This rule describes an unfriendly robot: it says that a robot without a tie is unfriendly.

3 Generating rules and arguments

Before continuing with the generation of rules and arguments, some notation in A-MAIL is mentioned first. In A-MAIL, there are two agents. Every agent A_i has a case base E_i consisting of examples: $E_i = \{e_1, \dots, e_n\}$. These examples have a target concept C that is either positive or negative, denoted as $+$ or $-$. A target concept is for example ‘unfriendly robot’ or ‘swan’. There is also a hypothesis space \mathcal{H} which is a set of hypotheses. These hypotheses consist of different rules. These sets are initially empty. The goal for this agent is to learn a hypothesis H_i such that $H_i(e) = C(e)$ for all $e \in E_1 \cup \dots \cup E_m$: the hypothesis has to be consistent with all the local case bases. A hypothesis consists of different rules h . The concept to be learned is given.

Between the rules, a *more-general-than* relation exists. This is the same relation as the previously described \leq -relation. $h_1 \sqsubseteq h_2$ means that h_1 is more general than h_2 (h_1 contains less information than h_2) and that there is a feature in both rules, and the feature of h_1 subsumes (\sqsubseteq) the feature in h_2 . The examples in the case base should be subsumed in the rules ($h_1 \sqsubseteq e$).

3.1 Rules and examples

In A-MAIL, two kinds of arguments can be used. For clarity, I will be using a slightly different notation than in the article. Instead of using \bar{C} for the target concept, $C(e)$ will be used when it is the target concept of a positive example and $\neg C(e)$ when it is a negative example. The two types of arguments are:

- Example argument $\alpha = \langle e, C(e) \rangle$ (e is an example, and $C(e)$ is the concept). These arguments are only used to determine the confidence of rules or to attack rules of the other agent when no rule can be found. The terms example and case are used interchangeably.
- Rule argument $\alpha = \langle h, C(e) \rangle$ (h is a rule).

For each argument, the confidence B is determined. The confidence is determined by the ratio of examples covered by h and supporting the same concept from A_i 's case base divided by the total number of examples of A_i . When we look back at our example of the Ferrari, 'red car' is probably applicable to many other examples, since there are a lot red cars that are not Ferrari's. So this rule probably doesn't have a high confidence. Whether a rule is acceptable depends on whether it exceeds a certain value, τ . This value is defined by the user. Every example argument is acceptable by definition.

3.2 Argumentation-based bottom-up induction

In A-MAIL, an algorithm called ABUI, *Argumentation-based Bottom-up Induction* is used to perform induction on the given data set. It will output a set of rules. An agent has a local case base E_i and when an argument is defeated, it has to change its beliefs.

ABUI is used for two things: 1) generate a hypothesis from examples and 2) generate attacks to arguments. The input of ABUI is a collection of examples E , a target concept $C(e)$ a set of arguments Q and a generalization g . The goal of the algorithm is to output a rule h such that h supports $C(e)$, h is more specific than g , h is τ -acceptable in E and $\langle h, C(e) \rangle$ is not under attack in Q .

First, a set of seeds is computed. A seed consists of the positive examples that are covered by the generalization g . Next, the algorithm ABUI(E , C , Q , g) generalizes each seed to generate rules and arguments:

1. Initialize the current rule c with the seed e .
2. Generalize this rule with generalization method γ
3. Add the rules that are more specific than g and not under attack to set G'
4. Add all rules that are τ -acceptable to hypothesis H .

5. Select the rule with the highest confidence from G' to be the current rule in the next round and go to 2.
6. When the set of seeds is empty, all the seeds are generalized. Return the rule with the highest confidence.

A hypothesis consists of different rules. When a hypothesis is generated, different rules have to be derived from the examples. ABUI is called with the parameters $\text{ABUI}(E_i, +, \emptyset, \perp)$. E_i is a set of positive examples from the agents case base, $+$ the target concept, \emptyset because there are no counterarguments yet and \perp because there are no rules yet, so the generalization is the most general rule. The result is a rule for some positive examples. ABUI is called again with the examples that are not yet covered.

ABUI can also be used to generate attacks to other arguments. For example, an attack for $\alpha = \langle h, + \rangle$ can be searched for with $\beta = \text{ABUI}(E_i, -, Q, h)$. The reverse of the target function is used. When an argument is returned, this will be used. If no argument can be found, the agent will look for an example attacking α . If that fails, the agent is unable to attack α . (See section 4 for more information on attacking arguments)

Only the positive examples are used. Assuming that the case base is correct, applying induction on only the positive examples will be enough to generate consistent rules.

3.3 Belief Revision

The agent has beliefs in the form of the local case base E_i and hypothesis H_i for target concept C . These beliefs can change when an argument that is not consistent with the local case base is acceptable. The belief revision will occur after all the arguments and examples are exchanged, so at the end of an interaction round. This is done in the following way:

- When it is an example argument, this is added to the case base and the acceptability of the arguments is reevaluated.
- When it is a rule argument, the agent tries to attack this new argument. When this is not possible, the rule will be added and rules that become not acceptable will be removed. New rules will be generated for examples that are not covered anymore. This is done by using ABUI.

Because of belief revision, the local case bases can change and this is an important element of A-MAIL.

4 Argumentation about rules

When the rules have been generated, A-MAIL requires the agents to communicate about them. By doing this, their case bases will become consistent with each other. The communication is done in an argumentation framework, as explained in the next section.

4.1 Argumentation frameworks

Argumentation can be used to determine which arguments are acceptable and unacceptable. For example, if one person says something, another person can have an argument to attack what the first person is saying, making it false (or unacceptable). This can continue for a long time, but it is orderly and strict. When the first person doesn't have a counterargument anymore, it means that his initial claim becomes unacceptable.

Argumentation between agents works the same. Arguments can be attacked and the agent that cannot defend his last argument, loses the argument. An example:

1. Agent A: A robot always wears a tie
2. Agent B: I have seen a robot that doesn't wear a tie.
3. Agent A: You must have mistaken a robot with a human.

Agent A states something (1), agent B attacks this argument with (2), making (1) untrue, but A has another argument (3) that makes (2) untrue and because of that, makes (1) true again because it is no longer under attack.

To describe it more formal: An argumentation framework is a pair $AF = \langle AR, attacks \rangle$ [2]. AR is a set of arguments and *attacks* a relation between two arguments. For example, $attacks(\alpha, \beta)$ means that α is an attack on β .

An argument is acceptable when it is not under attack. When an argument is under attack, it is unacceptable. However, when another argument attacks the attacking argument, the defeated argument becomes acceptable again, since nothing attacks it anymore. An example will make it clearer:

First, B attacks A:

$A \leftarrow B$

Now A is unacceptable because it is under attack. When another argument C is added that attacks B:

$A \leftarrow B \leftarrow C$

C attacks B, and because B becomes unacceptable, it can't attack A anymore and A becomes acceptable.

4.2 Argumentation in A-MAIL

In the discussed paper [5], \rightarrow is used instead of \longrightarrow . However, both symbols can be used in the same way.

In A-MAIL, an attack relations holds when:

1. $\langle h_1, C(h_1) \rangle \rightarrow \langle h_2, C(h_2) \rangle \iff C(h_1) = \neg C(h_2) \wedge h_2 \sqsubset h_1$
A rule can attack another rule when the one is a negative example of the other and is more general. It must be *strictly* more general because all

the examples covered by h_1 are also covered by h_2 and if they support different concepts, they must be in conflict. It also prevents cycles.

2. $\langle e, C(e) \rangle \rightarrow \langle h, C(h) \rangle \iff C(e) = \neg C(h) \wedge h \sqsubseteq e$
When an example wants to attack a rule, it has to have a reversed target concept.

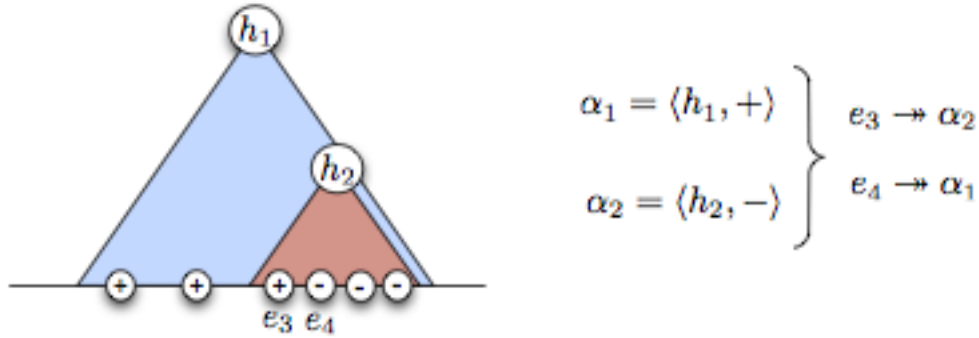


Figure 1: An example of different argument types and relations. The (+) and (-) are examples. [5]

An example can be found in figure 1. There are two rules. h_1 subsumes h_2 so h_1 more general. Because of rule 1, $\alpha_1 \rightarrow \alpha_2$ because $h_2 \sqsubseteq h_1$. In this example, e_3 can attack h_1 , because it supports a different target function.

All these arguments can be represented in an argumentation line. This is a sequence of acceptable arguments. In an argumentation line $\alpha_n \rightarrow \alpha_{n-1} \rightarrow \dots \rightarrow \alpha_1$, α_1 is the first argument (the root) and α_1 attacks α_{n-1} . The first argument and the other odd-numbered arguments are generated by the agent whose hypothesis is under attack. The even-numbered arguments are generated by the attacking agent. When the final argument is even-numbered, all the even-numbered arguments are true and the odd-numbered are false because they are under attack by the even arguments. The reverse happens when the final argument is odd-numbered. Example arguments can only be used to end a line because examples can only attack rules and rules can only attack other rules.

5 A-MAIL Interaction protocol

A-MAIL is an multi-agent learning-protocol with the use of argumentation techniques. In the previous sections, we have looked at inductive learning (section 3) and how agents can communicate with the help of argumentation techniques (section 4). In this section, we will combine these techniques and describe how this protocol works.

First, some remarks about the representation. In the program, there are two agents. The status of argumentation on a moment t is represented as $\langle R_1^t, R_2^t, G^t \rangle$

where $R_i^t = \{\langle h, + \rangle \mid h \in \{h_1, \dots, h_n\}\}$ is the set containing an argument for each rule. This is the hypothesis an agent holds. G^t contains all the arguments that belong to some argumentation tree rooted in an acceptable argument of one of the agents. There are other sets the agent knows about: $Q^t \subseteq G^t$, the subset of undefeated arguments generated before t , and $I_i^t \subseteq G^t$, the collection of undefeated arguments generated before t by the other agent A_j , but not acceptable for agent A_i .

The interaction protocol is divided in different rounds. Each round, one of the agents has the token. This agent can make the move: attack arguments or add new arguments.

When there are two agents, the protocol works as follows:

1. It starts at round $t = 0$. Each agent performs induction on the given set using $\text{ABUI}(E_i, +, \emptyset, \perp)$. The returned hypothesis H_0 is communicated to the other agent. The token is given randomly to one agent.

The following steps will be repeated until no new argument was added during the last 2 rounds. When this happens, it means that either the hypotheses are consistent or no new argument can be generated anymore.

2. When the agent A_i with the token has changed his hypothesis during the last round, the new hypothesis will be communicated to the other agent.
3. The agent with the token checks which of his arguments are defeated, and tries to defend these arguments by sending new attacks. He also tries to find the arguments in set I_i^t , the set of for this agent unacceptable arguments, and tries to generate an argument β (using ABUI) attacking argument $\alpha \in I_i^t$. (Only one argument can be attacked) When there was an attack, go to 5. Otherwise, go to 4.
4. When there is an example that is not covered in the hypothesis of the other agent, A_i will send this example to the other agent. This agent will put this in its local case base.
5. Both agents update their beliefs (*Belief revision*). The token goes to the other agent and a new round starts in 2.

To ensure that this protocol ends, an argument can't be send more than 2 times by the same agent. When it has ended, the agents have an agreed-upon hypothesis.

6 Conclusion

6.1 Results

In section 2, the question whether it was an advantage to use multiple agents was discussed briefly. In the discussed paper, some experiments were done to answer this question. In these experiments, a comparison was done between:

- A-MAIL
- Single agent that performed learning individually. Used to show the benefits of multi-agent learning
- Centralized learning on all examples. This is thought to do a better job but slower. Used to measure the quality of the outcome.

In the experiment, the training set was divided randomly amongst the two agents and, given a target concept, the goal of the agents was to find hypotheses for that concept. The precision (*how many rules are correct?*) and the recall (*how much of the positives does the model return?*) were measured and different levels of confidence (τ) were used. The outcomes of the experiments indicated that with $\tau = 0.85$ A-MAIL is as precise as centralized learning and has the same recall as centralized learning (and better than individual learning). The time needed to learn the hypothesis in A-MAIL is less than centralized learning and approximately the same for individual learning.

To conclude, A-MAIL works as good as centralized learning, but faster. So using multiple agents has advantages.

6.2 Further research

In the discussed paper [5], the author gives some suggestions for further research. I will briefly discuss these suggestions.

1. **Expanding the framework to multiple agents.** The framework now works with 2 agents and when using more agents, the data can be distributed among more agents, thus increasing the speed. Using more agents requires more communication and organization. For example, the agent has only to defend one argument that is under attack, since there is only one attacking agent. When there are multiple agents, more arguments have to be defended and when attacking an argument, a choice has to be made whether still only one argument can be attacked or one for every agent.
2. **Expanding the framework to learning multiple concepts.** Now, only one concept can be learned (for example: that animal is an elephant). It will be more informative to know in what class the item is ('The animal is a fish') instead of just knowing in which it is *not* ('The animal is not an elephant').
3. **Explore other ways to determine and handle confidence** In the current framework, the confidence of a rule is determined when ABUI is generating new rules, and only when this is high enough, a rule can be acceptable. It might be quicker to determine the confidence inside the argumentation framework. This way, when an example or rule is given, the confidence can be calculated immediately on the agents local case base. This doesn't require ABUI to run.
4. **Explore whether the framework can be speed-up** When implementing suggestion 1 and 3, the speed will probably increase. The question is whether there are other ways to increase the framework.

These suggestions are all very useful and might lead to faster concept learning. However, as already mentioned in the introduction, there are other perspectives to consider as well. It would be interesting to see whether there are similarities between concept learning on a computer and for a human. But it would be more interesting to see if the communication between the agents in this framework resembles the dynamics of a social group. When people are learning together, does it work the same? Is the way that they discuss new hypotheses and eventually find them acceptable the same as in A-MAIL? When looking into these questions, it can lead to more insight in learning in groups. But it also works the other way around: maybe people use some shortcut to get quicker conformity. These shortcuts can be used to increase the speed of some learning algorithm.

References

- [1] W. Davies and P Edwards. Distributed learning: An agent-based approach to data-mining. *Proceedings of Machine Learning-95 Workshop on Agents that learn*, 1995.
- [2] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
- [3] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [4] Santiago Ontañón. *Ensemble Case Based Learning for multi-agent systems*. PhD thesis, Universitat autònoma de Barcelona, 2005.
- [5] Santiago Ontañón and Enric Plaza. Multiagent inductive learning: an argumentation-based approach. In *Proc. ICML-2010, 27th International Conference on Machine Learning*, pages 839–846, 2010.
- [6] Sati S. Sian. Extending learning to multiple agents: Issues and a model for multi-agent machine learning (ma-ml). In Yves Kodratoff, editor, *Machine Learning EWSL-91*, volume 282. Springer-Verlag, 1991.