Extending Learning to Multiple Agents: Issues and a Model for Multi-Agent Machine Learning (MA-ML)

SATI S. SIAN
Dept. of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ
Email: sss@doc.ic.ac.uk

Abstract

Many real world situations are currently being modelled as a set of cooperating intelligent agents. Trying to introduce learning into such a system requires dealing with the existence of multiple autonomous agents. The inherent distribution means that effective learning has to be based on a cooperative framework in which each agent contributes its part. In this paper we look at the issues in multi-agent machine learning and examine what effect the presence of multiple agents has on current learning methodologies. We describe a model for cooperative learning based on *structured dialogue* between the agents. MALE is an implementation of this model and we describe some results from it.

Keywords Multiple Agents, Distributed Learning, Support Combination.

1 Introduction

Current Machine Learning (ML) research deals primarily with one single learning agent [Michalski et. al., 1983; Michalski et. al., 1986; Kodratoff, 1988]. However many real world problems are modelled better as a set of cooperating intelligent agents. In addition, even agents who are not explicitly part of cooperative framework will, in many real world situations, have to deal with other agents in their operating environment. Recognising this situation, recent work in the field of Distributed Artificial Intelligence (DAI) [Huhns, 1987; Bond & Gasser, 1988; Gasser & Huhns, 1989] has examined the problems of coordination and cooperation associated with such multi-agent systems. The problem of learning when multiple agents are present, however, has not received much attention.

Our primary objective in this paper is to look at *multi-agent machine learning* (MA-ML) by examining:

- the motivation for multiple agents in learning
- the problems introduced by the presence of multiple agents
- the effect on current methodologies
- a framework and model for MA-ML based on structured dialogue between the agents

In addition we have carried out some investigations with an implementation of the model called MALE (Multi-Agent Learning Environment) and some results from this are described.

2 Attributes of Multi-Agent Systems

We will start by looking at the various attributes of multi-agent learning systems such as the reasons for trying to construct such systems, the problems that occur and the specific issues that have to be tackled.

2.1 The Motivation

The reasons for modelling a system using multiple intelligent agents range from improvements in speed to autonomous agents providing a better 'fit' to the problem [Bond & Gasser, 1988]. At the very least therefore, the existence of such systems and the requirement of having learning within them motivates the study of MA-ML. More positively however multiple agents may bring the following benefits to the learning task itself :

- *Scalability* : Individual agents in a system will have bounds on the resources available to them. Scaling up the learning task beyond a certain level will require the use of multiple agents and cooperation between these agents.
- *Speed* : Where it is possible to parallelise the learning process, the use of multiple agents may give advantages of speed and efficiency. However this has to be weighed against the overhead due to cooperation.
- *Fault Tolerance* : Distributed systems in general provide a more graceful degradation in performance in the presence of failures. In addition the use of cross-checking of results between agents may provide more reliable results.
- *Encapsulation* : A system of multiple agents allows the encapsulation of specific learning knowledge or expertise in particular agents. Such encapsulation gives advantages in development, management, understandability and reliability.

The increased availability of distributed platforms has allowed these rationales to be tested in distributed processing and increasing now in distributed AI. This work has led to the development of numerous methodologies some of which will be useful when building MA-ML systems.

2.2 The Problems

The problems associated with multi-agent systems all arise from the fact that the existence of multiple agents implies a distribution of information. Where the solution of a particular problem requires the availability of all such information (or as much as possible), effective ways of overcoming the distribution are required. Learning is one such problem where this distribution has important consequences.

Other than the learning algorithm, the main factor in deciding the quality of what an agent learns is the quality of the data on which the learning is based. For a learning agent, distribution of this data implies two possible problems:

- *Completion* : Consider a situation where the distribution is based on a partitioning of information into parts which are distinct but coupled. If.an agent has access to data only from one part and what has to be learned requires knowledge from other parts then this agent is unable to learn as its local data is incomplete. We shall call this the *completion problem.* An example of this would occur in a distributed manufacturing system in which individual agents are assigned the construction of distinct sub-parts and the learning task consists of finding attributes of sub-solutions that make a good overall component. Since these parts have interrelationships between them, the learning process of individual agents is constrained by the information held by other agents.
- *Confidence* : A well known problem when using induction is that the resulting hypothesis cannot be completely validated. However the confidence in a particular hypothesis increases as we obtain more and more data that is correctly explained by the hypothesis. Distribution of this data may therefore mean that an agent is unable to reach a level of confidence in its hypothesis sufficient to allow it to make use of it even though such data exists in the system. This we shall call the *confidence problem.*

These problems necessitate that we develop some means of cooperation between the agents in the system.

2.3 The Issues

We can quickly discount the possibility of simply collecting the source information of the multiple agents at one special point and using available methods of learning as one solution to the distribution. Other than the sheer volume of data, we would loose all the advantages that we previously outlined. What is required is a more cooperative framework where the participating agents perform local processing as far as possible and cooperate when necessary. Such an approach means the following issues must be handled:

- Recognising when to cooperate : One may adopt the approach that the agents always cooperate. This will be necessary when the agents do not have knowledge of the specific abilities of the other agents. Rather what exists in the system is implicit knowledge that there are other agents and that these agents are potentially useful. Where agents are aware of the other agents' abilities, more focused cooperation may be possible. The goal of such cooperation is always to overcome the completion and/or confidence problems.

- How to cooperate : Cooperation based on communication requires the use of some interaction language, a protocol for structuring the use of this language and associated semantics to allow the agents to make sense of the interaction. In complex systems such a language may hide representational differences between the agents and require agents to find a translation mechanism for expressing hypotheses in a common form. This is necessary to allow other agents to evaluate these hypotheses. The cooperation scheme also needs a mechanism for integrating the learning of the participating agents.
- Dealing with conflict : Having autonomous intelligent agents introduces the possibilities of conflict in the views of these agents. This requires having methods for recognising and resolving conflict.

In general we have to revise the prevalent procedure for a learning agent from a (get data $\rightarrow$ form best generalisation) to one of (get data $\rightarrow$ form best generalisation $\rightarrow$ confer with other agents $\rightarrow$ revise hypothesis).

3 The Effect on Current Methodologies

We can now look at the existing paradigms in ML [Michalski, 1987] and see what affect the presence of multiple agents has, viewing the changes necessary on both the algorithmic and structural level.

Learning by being told

The presence of multiple agents may be reflected as an extension to one or both parts of the teacher/learner scenario in learning-by-being-told i.e multiple teachers and/or multiple learners. The multiple learners case poses no new problems. The one-to-one interaction is simply extended to a one-to-many interaction. Additional problems may however occur for the teaching agent if it is biasing its teaching based on knowledge of the learning agent. In this case the existence of multiple learners means that the teaching agent has to maintain multiple contexts.

The multiple teachers scenario requires additional capabilities on the part of the learning agent. The main problems arise when there is a conflict between the knowledge received from two teachers. Since the assumption in this form of learning is that the teacher's knowledge is correct, conflict has to be attributed to contextual differences. However if the assumption of correctness is relaxed then some framework for conflict resolution has to exist between the teaching agents.

Learning by Deduction

In analytical or explanation-based learning [Mitchell et. al, 1986] the presence of multiple agents becomes significant when, as in many real-world domains, the domain knowledge is distributed amongst more than one agent. In such a situation the first step in the learning process, that of constructing a logical proof of why an instance is an example of a concept, may not be within the means of a single agent. What is required is a process of distributed search. [Kitamura & Okumoto, 1990] present a method that they call *diffused inference* in which each agent performs local search as far as possible and solicits help from the other agents when necessary. The subsequent step of generalisation may also require the use of more than one agent.

One useful attribute of the use of multiple agents in EBL is that one may be able to specify different operationality criteria for the different agents. As is quite often the case, an abstract domain theory is useful to different agents in different ways. Consider for example a domain theory describing the operation of a vehicle. The operationality criteria for this theory will be different for an agent that wishes to use it for learning how the vehicle functions from one who wishes to use it for diagnosis of faults. Such differences may be effectively captured in a multi-agent scenario.

Learning by Analogy

Analogical learning transfers well between single and multi-agent systems. The process of recognising a similarity between two problems at some abstract level is the same inter-agent as it is intra-agent. Derivational analogy is more problematic as derivations are local to an agent.

Learning by Induction

In the presence of multiple agents, the inductive learning suffers from the two problems of completeness and confidence that we described earlier. These problems occur in learning-from-examples where due to the examples being distributed each agent can only form a partial hypothesis and in learning-from-observation where different agents may form different concepts due to different foci of attention.

4 A Cooperative Framework for MA-ML

From the large space of problems in multi-agent learning we have concentrated our attention on the problem of learning by induction in peer group agents. Using models of human peer group learning as a basis, a framework for cooperative learning is proposed. An analyses of how the compositional structure of multi-agent systems affects the role of cooperation in learning has been reported by us in [Sian, 1990a] and the use of this cooperative framework to get adaptation in Distributed AI systems is reported in [Sian, 1990b]. Here we will give details of the model itself and how the model can be used to get group induction. Results from an implementation of the model are also shown. Our model consists of the following components:

- Agents that learn from their experience
- An interaction board for negotiation between agents
- An interaction language for talking about hypotheses
- An integration function for combining the opinions of the agents

In the model each agent first learns locally. When an agent has constructed a hypothesis in which it has reasonable confidence (a parameter to the system) it proposes it to the other agents via the interaction board. Other agents use their own experience to evaluate the hypothesis and may make changes to it. The net confidence value of each such hypothesis is used to select the one that should be accepted by the agents. The following sections give details of this process and the components of the model.

We make the following two assumptions:

- The cooperation paradigm is one of consensus with agents accepting the hypotheses that have the greatest support.
- The hypothesis representation and generation is the same in all agents.

4.1 The Learning Agent

Agents in our model consist of the following parts:

- Performance component : This is the part that is responsible for carrying out whatever problem solving activity the agent is assigned within a multi-agent system. This component will contain domain knowledge and expertise about the agent's task. A significant attribute of this knowledge is that it is necessarily non-monotonic so that the learning sub-system can add or modify information contained within it. Such changes are monitored by a simple belief maintenance component.

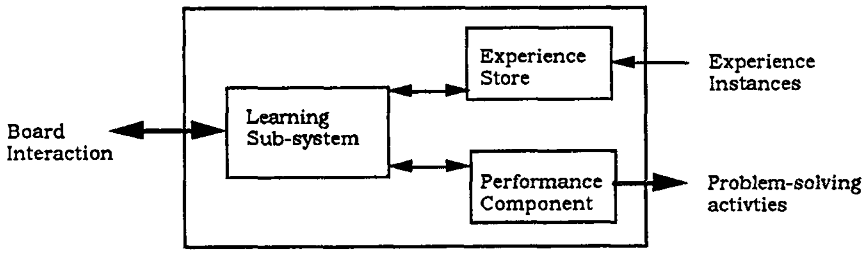


Fig 1: Agent Structure

- Experience store : During its problem-solving activities the agent receives information about the events and state of the external environment. This forms the source data for the learning system. The learning may be restricted to certain issues by filtering this data so that only data regarding that which has to be learnt arrives in the experience store. The store is organised as a hierarchy in which the instances form the lowest level and successively higher levels contain generalisations that classify these instances at the first level and classify lower level generalisations at successive levels. Maintenance of the hierarchy is controlled by the learning algorithm.

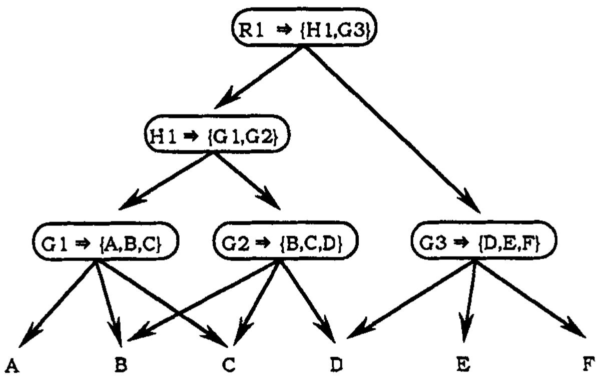


Fig 2: Experience Store Hierarchy

- Learning sub-system : This has three primary tasks; learning by induction on the data in the experience store, maintaining the store and interaction with the board for cooperative learning. The learning algorithm is incremental and is described in detail in the next section. The experience store is the 'workspace' of the algorithm and the learning process makes changes in this hierarchy. Additional maintenance of the store essentially consists of removing instances that are covered by a concept description which has been agreed and in which the agent has sufficient confidence. A full-memory model is impractical in most domains. Interaction using the board is the process by which the agents cooperate with the other agents to get the other agents' opinions about hypotheses that the proposer has learnt locally. This serves to both modify a hypothesis based on another agent's experience and increase confidence in it based on concurrence from the other agents.

Note that there is no direct connection between the experience store and the performance component. The data in the experience store must be processed by the learning sub-system before being made available to the performance component.

Learning Algorithm

The learning process consists of an incremental learning algorithm that constructs a generalisation hierarchy with successively lower levels being more specific. The lowest level consists of individual instances. Each generalisation has associated with it two values; *in-conf* and *out-conf*. The value in-conf is a measure of how many instances the generalisation correctly covers and out-conf is a measure of how many instances argue against the generalisation (*net-conf* is the difference between the two). The algorithm proceeds as follows:

```
For each new instance I
   If I is correctly explained by current knowledge
      Then increase confidence in this knowledge
   Else
      For all generalisations G in heirarchy that cover this instance
         Check( G,I )

   For all instances not now covered
      Create new generalisation and insert in heirarchy

   If net-conf > threshold for any generalisation G
      Propose G on board

Check( G,I ):
   If +ve instance
      Then increase in-conf of G
   If -ve instance
      Then If specialisation( G,I ) does not succeed
            Then increase out-conf of G
```

The basic goal of the algorithm is to create a generalisation hierarchy with maximum *net-conf* values at each node. Each new instance is first checked against what the agent already knows. This is to stop the agent from spending effort in learning previously agreed hypotheses. Instances that agree with these hypotheses simply increase their confidence. All other instances are added to the hierarchy. The first step is to check the current generalisations starting with the most specific. An instance correctly covered by one of these generalisation need not be checked against those at a higher level. If an instance is incorrectly covered then the we first try and specialise the generalisation so as to include as many as possible of the originally covered set but exclude the new instance. The specialisation procedure uses both attributes of the instances and domain knowledge to change the generalisation. If no specialisation can be found then the out-conf value of the generalisation is increased. Generalisations with a net-conf value too low are removed. At the end of this stage we may have a number of instances not covered by any generalisation. Where possible new generalisations are created to cover these instances and linked into the hierarchy. The preference when creating these generalisations is to find more generalised versions of existing ones in an incremental fashion to cover these instances.

Additional changes to the hierarchy may occur as a result of the interaction with the other agents and we shall discuss these after looking at the the cooperative activity in the system. This algorithm has some similarity to the UNIMEM algorithm [Lebowitz, 1987] in that it also constructs a hierarchy of generalisations. However in the UNIMEM hierarchy each instance may only have one parent. Each instance is unique and consists of attribute-value pairs. UNIMEM attaches confidence values to each attribute rather then the generalisation as a whole.

Hypothesis evaluation

In addition to learning from its own environment, a cooperative framework requires agents to evaluate other agents' hypotheses with respect to their local data. This may consist in the first instance of agreement or disagreement with a hypothesis. However a more useful form is for the evaluating agent to itself propose ways of making the hypothesis more acceptable. This leads to a negotiation process that we will examine in the next section. Here we will look at how an agent evaluates a hypothesis proposed by another agent.

The goal of the evaluation as we have said is for the evaluating agent to check if the proposed hypothesis is consistent with its own data. From the evaluating agent's perspective it would prefer to do this as efficiently as possible.Trying to check a hypothesis against individual instances is prohibitive. Instead the agent can use the generalisation hierarchy to get much more effective evaluation. The result of this evaluation can be one the following four cases. If P is the proposal then

- *Totally Consistent* if the agent has no data against P
- *Totally Inconsistent* if the agent has data against P and can find no way of specialising P to exclude this data
- *Partially Consistent* if a specialisation of P is consistent with the agent's data
- *Disjoint* if the agent has no relevant data regarding P

The result will decide what response an agent gives to a proposal. We will see this when looking at the operators available to an agent for talking about hypothesis.

4.2 Interaction Board and Language

The interaction board is a multi-level shared structured which the agents use to communicate. Peer agents are connected to one level. Higher levels are necessary to cope with hierarchical organisations in multi-agent systems. Agents at one level are able to influence the interaction at lower-levels. The structural similarity to a blackboard architecture [Erman et. al., 1980] is obvious. The role of our agents is not however to opportunisticly transform a problem but to express an evaluated opinion as regards proposed hypotheses. Such a centralised structure is appropriate in peer groups as it is not known *a priori* which agent will be of benefit.

The interaction language consists of a set of operators for hypotheses. These operators allow agents to introduce/remove hypotheses, express the results of local evaluation and change the status of hypotheses based on the integration of the evaluations.

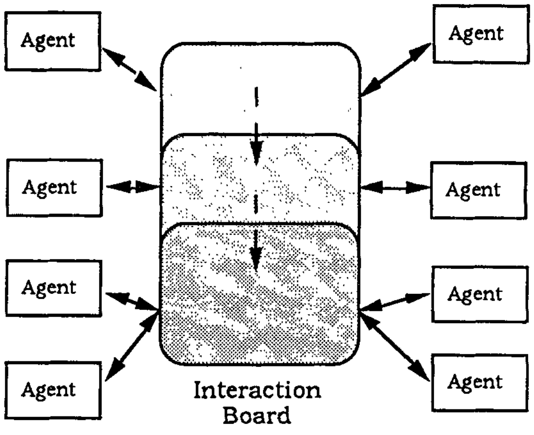


Fig 3: Shared Interaction Board Structure

The following is the definition of the operators:

- Introduction & Removal :
  PROPOSE(H,C), ASSERT(H), WITHDRAW(H), ACCEPT(H)
- Evaluation :
  CONFIRM(H,C), DISAGREE(H,C), MODIFY(H,H',C,S), NOOPINION(H)
- Status modifiers :
  AGREED(H,T)

where	H is the hypothesis in question
	H' is a modified H
	C is the confidence value (range 0..1)
	T is the resultant confidence value (range 0..1)
	S is a similarity measure (range 0..1)

As can be seen from the definitions, most of these operators have an associated confidence value. This allows the agents to give a measure of how many instances the response is based on. A CONFIRM based on lots of instances is obviously more powerful that one based on just a few. The resultant confidence value for H is given by the integration function described in the next section. The similarity measure S is a measure of the number of instances that motivate the specialisation of H to H'. The mapping from the number of instances to the confidence value is dependent on the context of the system. So, for example, if within a particular situation the existence of 5 instances indicates near certainty then this would map to a value close to 1. Other situations may need hundreds of instances to approach this value. The exact numbers are less important then the mapping being consistent across agents.

The protocol that structures the use of this language consists of a set of rules that specify the order in which the operators must occur. Essentially once an agent proposes a hypothesis it waits for the other agents to respond. Each agent evaluates the proposal and returns a CONFIRM if it is totally consistent, DISAGREE if it is totally inconsistent, MODIFY if it is partially consistent and NOOPINION if it is disjoint relative to its data as defined in the previous section. These responses are combined using their associated confidence values and the integration function to give T. Any modified H (H') has to be reevaluated by the other agents. Once this process is complete, the version with the highest T value is agreed and the subsequent ACCEPT from the agent removes it from the board. The other versions are withdrawn by the proposers. If the highest T value is below a threshold then no hypothesis is accepted.

4.3 The Integration Function

In order to judge the relative merits of a hypothesis against other versions of that hypothesis we have to be able to give each hypothesis a net value based on the responses and their associated confidence values that it received from the participating agents. In general this can be a difficult problem since the confidence values represent a measure of belief on the part of the agent. Use of Bayesian functions is inappropriate as these values are certainly not probabilities. A confidence value of C in H does not imply a confidence value of 1−C in ¬H. Similar problems resulting from trying to combine the belief values of two rules were encountered in the MYCIN project [Shortliffe & Buchanan, 1980]. The situation is even more difficult in our case as the more recent work on the Shafer-Dempster theory [Shafer, 1976] (which deals adequately for the MYCIN problems) requires knowing the complete set of possible hypotheses suggested by the evidence which is a condition that we cannot meet in our general situation.

We have adopted an extension of the approach in [Stefanyuk 87] in which he presents an axiomatisation of a combination function and derives the function from these axioms. This work considers only confidence values in support of a hypothesis and our extension has been to use Stefanyuk's Formula for the combination of confidence values of one kind (either in support-of or support-against), provide an axiomatisation of a combination function that handles both kinds of confidence values and finally to derive this function from these axioms. Let x_1 and x_2 be two confidence values of one kind, $c(x_1,x_2)$ be a function that combines these values, α_n and β_n be the total confidence for and against a hypothesis respectively and $C(\alpha_n,\beta_n)$ be the function that gives the resultant confidence in the hypothesis. Then the following axiomatic definitions apply:

A1: $0 \le x_1 < 1, 0 \le x_2 < 1, 0 \le c(x_1,x_2) < 1, 0 \le C(\alpha_n,\beta_n) < 1$

A2: $c(x,0) = x$

A3: $c(x_1,x_2) = c(x_2,x_1)$

A4: $C(0,\beta_n) = 0, C(\alpha_n,0) = \alpha_n$

A5: As $\alpha_n \to 1$ then $C(\alpha_n,\beta_n) \to 1-\beta_n$, as $\beta_n \to 1$ then $C(\alpha_n,\beta_n) \to 0$

A6: $c(x_1,x_2)$ and $C(\alpha_n,\beta_n)$ can be expanded as a power series in x_1,x_2 and α_n,β_n respectively

From these we derive the following functions ([Sian, 1990c] gives the rational for the axioms and the derivation):

$c(x_1,x_2) = x_1 + x_2 - x_1 x_2$ (Stefanyuk's Formula)
$C(\alpha_n,\beta_n) = \alpha_n - \alpha_n\beta_n$

Within our interaction framework CONFIRM represents confidence in a hypothesis, DISAGREE confidence against the hypothesis, MODIFY partial confidence in the hypothesis (with the similarity measure determining the level of partiality and NOOPINIONs representing neither. The combination function is therefore defined as follows:

Net-Val(H) = Total_Confidence(supporting,H) [1 - Total_Confidence(against,H)]

Let Count(T) give the number of responses of type T. Then given n agents if:

Count(CONFIRM) = c, Count(DISAGREE) = d,
Count(MODIFY) = m, Count(NOOPINION) = p and
$n = c + d + m + p$

then

Total_Confidence(supporting,H) $= V_{c+m}$
Total_Confidence(against,H) $= V_d$

where

$V_x = V_{x-1} + C_x(1-V_{x-1})$
$V_0 = 0.$

in which

C_x = Confidence of xth agent giving response.

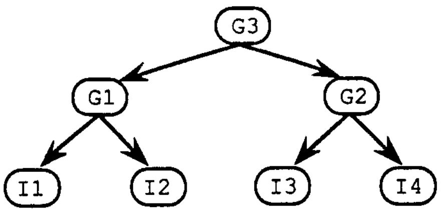
Consider an example with 4 responses; 2 confirms with values c_1 and c_2,, 1 modify with value m_1 and 1 disagree with value d_1. If the similarity measure of the modify response is s_1 then the response can be considered a confirm $c_3 = m_1 s_1$. Then:

Total_Confidence(supporting,H) $= c_1 + c_2 + c_3 - c_1c_2 - c_1c_3 - c_2c_3 + c_1 c_2c_3.$
Total_Confidence(against,H) $= d_1$
Net-Val(H) = $= (c_1 + c_2 + c_3 - c_1c_2 - c_1c_3 - c_2c_3 + c_1 c_2c_3)(1-d_1)$

4.4 An Example

We can now tie all these parts together and illustrate the functioning of the model with an example. We will consider a system with three agents operating in the domain of commodities trading. Each agent is responsible for trading within a particular area. Each agent receives information on events that occur in the area and how the prices of various commodities change with these events. The goal of the learning is to create generalised descriptions of how prices fluctuate due to various classes of events. To aid the learning process agents have domain knowledge about these commodities available to them. First let us see how one agent learns locally. Given the following data agent 1 constructs the generalisation hierarchy shown below:

```
I1:   Price(Coffee,Rising),Weather(Kenya,Frost)
I2:   Price(Coffee,Rising),Weather(Kenya,Drought)
I3:   Price(Tea,Rising),Weather(Kenya,Frost)
I4:   Price(Tea,Rising),Weather(Kenya,Drought)
```

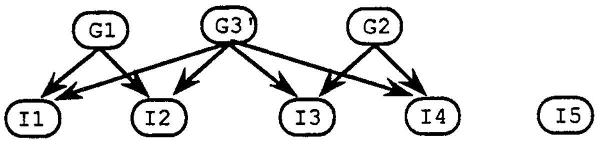


where

```
G1:   Price(Coffee,Rising) if Weather(Kenya,Adverse)
G2:   Price(Tea,Rising) if Weather(Kenya,Adverse)
G3:   Price(Crop,Rising) if Weather(Kenya,Adverse)
```

The relationship between (coffee,tea) and crop and between (frost,drought) and adverse is part of a is-a hierarchy that forms part of the agent's domain knowledge. When a new instance I5 arrives the hierarchy changes as shown:

```
I5:   Price(Cocoa,Steady),Weather(Kenya,Flood)
G3':  Price(Crop,Rising) if Weather(Kenya,Adverse),Affects(Adverse,Crop)
```



G3 has had to be specialised since it incorrectly covered I5. This is achieved by finding an attribute that distinguishes I5 from as many as possible of the original set. The agent uses its domain knowledge to find that cocoa is unaffected by floods whereas the others are affected by frost and drought.

Other agents use this same procedure to construct their own hierarchies based on the data they receive. Let us assume that Agent 3 has enough confidence in one of its generalisations to propose it on the board. The other agents will then use the evaluation procedure described previously to check this proposal against their own experiences. Note that the objective of the interaction is to find the most specific version of the proposal that has the maximum support. Therefore agents not only indicate confirmation or disagreement with a proposal but also modifications that make it more consistent with their experiences. In this case the interaction proceeds as follows:

Agent 3: **PROPOSE**(P1,0.7) [P1 = Price(Cocoa,Rising) if Weather(Brazil,Adverse)]
Agent 2: **MODIFY**(P1,P2,0.6,0.6) [P2 = Price(Cocoa,Rising) if Weather(Country,Adverse)]
Agent 1: **MODIFY**(P1,P3,0.8,0.63) [P3 = Price(Crop,Rising) if Weather(Country,Adverse), Affects(Adverse,Crop)]
Agent 1: **MODIFY**(P2,P3,0.8,0.63)
Agent 3: **CONFIRM**(P2,0.7)
Agent 3: **CONFIRM**(P3,0.7)
Agent 2: **MODIFY**(P3,P4,0.6,0.54) [P4 = Price(Crop,Rising) if Weather(Country,Adverse), Affects(Adverse,Crop), Produces(Country,Crop)]
Agent 1: **CONFIRM**(P4,0.8)
Agent 3: **CONFIRM**(P4,0.7)

[Net-Val(P1) = 0.9556, Net-Val(P2) = 0.9556, Net-Val(P3) = 0.9724, Net-Val(P4) = 0.976]

Agent 2: **AGREED**(P4,0.976)
Agent 3: **WITHDRAW**(P1), **ACCEPT**(P4)
Agent 1: **WITHDRAW**(P3), **ACCEPT**(P4)
Agent 2: **WITHDRAW**(P2)

Agent 2 generalises P1 with regard to country. Agent 1 generalises with regard to crop but adds the specialisation of 'Affects' which its data has suggested. The same response is also appropriate for P2. The original proposer Agent 3 has no data against either and therefore confirms both with its original confidence value. The interesting response is Agent 2's MODIFY to P3. This is based on it having an instance in which the price of tea is unaffected by drought in its region. Since it only has a very small number of instances to back a modification its the difference between its similarity measure and proposal value is very low (0.06). However on evaluating P4 the other agents find it consistent with their data and its resultant net value is highest. Note that our main interest is in the relative values rather than the actual numbers. Studies in psychology show that such convergence to the correct hypothesis once it has been suggested by one agent also occurs in human group induction [Laughlin & Shippy, 1983]. The other proposals are withdrawn and P4 accepted (an agreed hypothesis such as P4 is automatically removed once it has been accepted by all the other agents).

The net result of this cooperation is that agents are amalgamating their experience at a high level. Each agent is benefiting from the experience of the other agents. This results in hypotheses being corrected when necessary and further confirmed when possible.

4.5 Some experiments

The above model has been implemented in a system called MALE (multi-agent learning environment) in PROLOG with agents as processes that interact with the interaction board via message passing. We have tested the model using examples from a commodities trading domain as the source data. This data contains various sorts of events such as changes in weather, changes in political situation, announcement of alternatives to some commodities, changes in economic conditions plus how these events were followed by changes to the prices of various commodities such as crops, metals and oil. The goal of the system is to get the maximum average performance from the agents where performance is measured by the ability to correctly predict the price change given an event.

We have tried to evaluate the performance based on a number of scenarios:

1) Single agent receiving all the data compared to a multi-agent system with the data spread equally (both systems using the learning algorithm defined in the model)
2) Multi-agent system with single agent learning only (no interaction)
3) Multi-agent system no-learning
4) The effect of varying the number of agents participating in the interaction
5) Varying the spread of the data with some agents getting more data

For this paper the significant test is the first (results of the others are more relevant to the design of multi-agent systems and are reported in [Sian, 1990b]). The predictive ability of single versus multi-agent systems with the same data showed no difference if the agents interact after receiving each new instance. Otherwise the multi-agent system catches up the single agent after each interaction session.

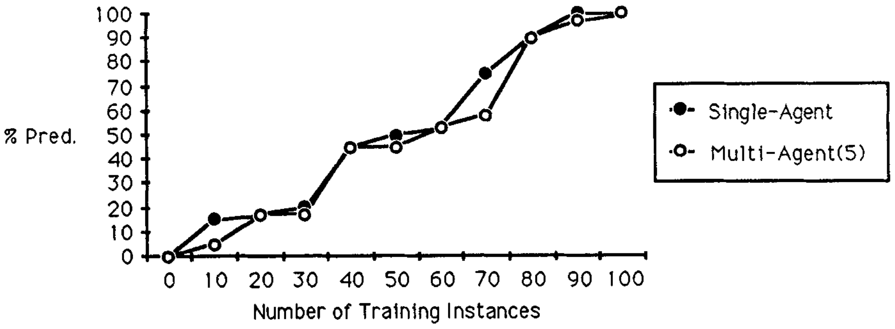


Fig 4: Predictive performance of single versus multi-agent learning

Predictive performance was measured against a test set. The multi-agent systems had 5 agents which interacted after receiving 2 new instances each. This result leads us to propose the following for our model:

Proposition 1 : The predictive ability of a learning system is unaffected by the use of multiple agents.

The difference in the two systems occurs in the speed with which they perform the inductive generalisation. In tests the single agent system outperforms the multi-agent system in the initial stages as the later has the overhead of cooperation. However as the number of instances increases the parallelisation of the learning process rapidly overcomes this overhead. On average with 5 agents the multi-agent system on a SUN 4 for this data was 3 times faster.

From this we propose the following:

Proposition 2: In the presence of large amounts of data, a multi-agent learning system will give advantages in speed over a single agent system.

We have are now trying MALE with the soya-bean disease data used in incremental AQ [Riene & Michalski, 1987]. A multi-agent scenario can be envisaged for this data if we assume that the data was collected from different autonomous agricultural centres. A cooperative approach to finding discriminant descriptions for these diseases can then be applied.

5 Conclusion

Consideration from both a pragmatic point of view i.e. multi-agent systems exist and therefore we must study learning within these systems and from a technological point of view i.e. looking for benefits to the learning process from the use of multiple agents warrants the study of the hitherto unexplored area of MA-ML. In this paper we have described the problems and issues in MA-ML and how the presence of multiple agents may affect current paradigms in ML. We have proposed a cooperative framework and studied its use in a sample domain. As the results show, MA-ML can give considerable benefits to learning systems. Further work needs to look at the use of different algorithms in the agents, richer forms of dialogue between the agents and ways of formulating the belief combination in agents.

References

[Bond & Gasser, 1988]	- Bond A. & Gasser L. [Eds], *Readings in DAI*, Chapter 1, Morgan Kaufmann Publishers, San Mateo, CA 1988.
[Erman et. al, 1980]	- Erman L. et. al, *The Hearsay II Speech Understanding System*, Computing Surveys 12(2), p213-253, June 1980
[Gasser & Huhns, 1989]	- Gasser L. & Huhns M. [Eds], *Distributed AI II*, Pitman, London, 1989.
[Huhns, 1987]	- Huhns M. [Ed], *Distributed AI*, Pitman, London, 1987.
[Kitamura & Okumoto, 1990]	- Kitamura Y. & Okumoto T., *Diffusing Inference: A Inference Method for Distributed Problem Solving and Its Property*, in Proceeding of

	First International Working Conference on Cooperating KBS, University of Keele, Oct 1990.
[Kodratoff, 1988]	- Kodratoff Y., *Introduction to Machine Learning*, Pitmann Publishers, London, 1988.
[Laughlin & Shippy, 1983]	- Laughlin P. & Shippy T., *Collective Induction*, Journal of Personality and Social Psychology 45(1), p 94-100, 1983.
[Lebowitz, 1987]	- Lebowitz M., *Experiments with Incremental Concept Formation: UNIMEM*, Machine Learning 2(2), p103-138, 1987.
[Michalski et. al, 1983]	- Michalski R. et. al., *Machine Learning: An AI Approach*, Morgan Kaufmann Publishers, Los Altos, CA, 1983.
[Michalski et. al., 1986]	- Michalski R. et. al., *Machine Learning: An AI Approach Vol 2*, Morgan Kaufmann Publishers, Los Altos, CA, 1983.
[Michalski, 1987]	- Michalski R., *Learning Strategies and Automated Knowledge Acquisition: An Overview*, in Computational Models of Learning, Bolc L. [Ed.], Springer-Verlag, Germany, 1987.
[Mitchell et. al., 1986]	- Mitchell T. et. al., *Explanation-based Generalisation: A Unifying View*, Machine Learning 1(1), p 47-80, 1986.
[Riene & Michalski, 1987]	- Riene R. & Michalski R., *Incremental Learning of Concept Descriptions*, in Machine Intelligence Vol 11, Hayes J. et. al. [Eds.], p263-288, Morgan Kaufmann, San Mateo, 1987.
[Shafer, 1976]	- Shafer G., *A Mathematical Theory of Evidence*, Princeton, New Jersey, Princeton University Press, 1976.
[Shortliffe & Buchanan, 1980]	- Shortliffe & Buchanan, *Rule-Based Expert Systems*, MIT Press, 1980.
[Sian, 1990a]	- Sian S., *The Role of Cooperation in Multi-Agent Learning*, in Proceeding of First International Working Conference on Cooperating KBS, University of Keele, Oct 1990.
[Sian, 1990b]	- Sian S., *Adaptation based on Cooperative Learning in Multi-Agent Systems*, in Proceedings of Second Workshop on Modelling Autonomous Agents (MAAMAW 90), ONERA, Paris, August 1990.
[Sian, 1990c]	- Sian S., *Learning in Distributed AI Systems*, Ph. D Thesis, Imperial College, University of London, London (forthcoming).
[Stefanyuk 87]	- Stefanyuk V. L., *Some Aspects of the Theory of Expert Systems*, Soviet Journal of Computing Science 25(5), p110-116, 1987.