

The Communication of Inductive Inferences

Winton Davies and Peter Edwards

Department of Computing Science,
King's College,
University of Aberdeen,
Aberdeen,
Scotland, UK, AB24 3UE

{wdavies, pedwards}@csd.abdn.ac.uk

Abstract We propose a new approach to communication between agents that perform inductive inference. Consider a community of agents where each agent has a limited view of the overall world. When an agent in this community induces a hypothesis about the world, it necessarily reflects that agent's partial view of the world. If an agent communicates a hypothesis to another agent, and that hypothesis is in conflict with the receiving agent's view of the world, then the receiving agent has to modify or discard the hypothesis.

Previous systems have used voting methods or theory refinement techniques to integrate these partial hypotheses. However, these mechanisms risk destroying parts of the hypothesis that are correct. Our proposal is that an agent should communicate the bounds of an induced hypothesis, along with the hypothesis itself. These bounds allow the hypotheses to be judged in the context from which they were formed.

This paper examines using version space boundary sets to represent these bounds. Version space boundary sets may be manipulated using set operations. These operations can be used to evaluate and integrate multiple partial hypotheses. We describe a simple implementation of this approach, and draw some conclusions on its practicality. Finally, we describe a tentative set of KQML operators for communicating hypotheses and their bounds.

1. Introduction

In this paper we address the question: "*How should agents communicate inductive inferences?*" This question is important because of a growing interest in systems that distribute a learning task amongst a community of agents (Weiss and Sen, 1995). Such systems are often referred to as multi-agent learning systems (Brazdil, 1991). The question is also relevant to the development of the Knowledge Query and Manipulation Language (Finin, 1993). KQML currently does not define standards for the communication of inductive inferences. Inductive inference differs from deductive inference in that it is logically unsound. This means that a logical sentence derived

using inductive inference with respect to partial knowledge of a world may be inconsistent with the world as a whole. This is the central problem when communicating an induced hypothesis. In a multi-agent learning system this requires agents to evaluate or refine hypotheses proposed by other agents. The major problem with this is that the refinement process may change the rule so that whilst it is correct for the refining agent, it may no longer be correct with respect to the original agent's view of the world.

We were initially motivated by a desire to extend KQML so that it could be used for communication within multi-agent learning systems. All previous systems have used a variety of messages to facilitate communication between learning agents. We had intended to rationalise these into a core set of KQML message primitives (or performatives). These would have been performatives such as "learn", "refine", and "test". However, the following question arose: "How could inductive inferences be communicated, whilst simultaneously maintaining a logically sound distributed knowledge-base?" This led us to propose a solution based on the communication of the bounds to the inductive inference."

Version spaces (Mitchell, 1978) seem the natural candidate to describe these bounds, as they represent all possible hypotheses that are consistent with an agents' view of the world. In Section 2.2 we provide an overview of Mitchell's work and review more recent extensions to it. Briefly, Mitchell showed that for a given supervised inductive learning problem, that the space of consistent hypotheses may be represented by two sets, one containing the most general hypotheses and the other containing the most specific hypotheses. These are jointly called the version space boundary sets.

This paper examines the feasibility of using version space boundary sets to represent the bounds of an inductive hypothesis. Thus an agent that receives an induced hypothesis and the bounds now knows the space of hypotheses that was consistent for the sending agent, as well as the particular hypothesis that was selected. The receiving agent is now able to modify the specific hypothesis if it is inconsistent with its own world view, but can constrain the modifications it makes to be consistent with the set of hypotheses that was valid for the originating agent. We shall now give a simple example of the problem and the solution.

1.1 A Simple Example

In order to demonstrate the problem of agents communicating inductive hypotheses, we consider the simple goal of learning a propositional binary relation. Even this trivial task (there being just 16 possible hypotheses) demonstrates the essential elements involved. We describe the learning task, how it is represented as a version space and then show how it is affected when distributed between two agents. In this example we are only considering distribution of the training examples. This is referred to as horizontal distribution. However, other aspects of the learning problem can be distributed. These include vertical distribution of examples (agents each have different facts about all the examples), as well as the distribution of language biases and selection biases.

1.2.1 A Simple Supervised Learning Problem

The task is to learn an intensional definition of a relation $f(X)$ in terms of two unary predicates (e.g. $p(X)$ and $q(X)$) and the full range of Boolean operators (i.e. \neg, \wedge, \vee). We are given positive and negative extensional examples of $f(X)$, and the Boolean value of $p(X)$ and $q(X)$. Thus there are 4 distinct possibilities for any particular example (corresponding exactly to the traditional propositional truth table shown in Table 1).

X	p(X)	q(X)	f(X)
0	F	F	?
1	F	T	?
2	T	F	?
3	T	T	?

Table 1 A truth table for features p(X) and q(X)

There are 2^4 (i.e. 16) possible hypotheses for the definition of $f(X)$. However, once we have seen a positive or negative example for each one of the four truth table rows, then the relation will be known. However, given less than the four distinct examples, an inductive leap must be made. This simple example shows the inputs to an inductive learning algorithm:

1. The conceptual bias is simply $p(X)$ and $q(X)$.
2. The language bias is all possible logical combinations of the conceptual bias. Together 1 and 2 constitute the concept description language (CDL).
3. The selection bias is the preference the algorithm has in selecting one of the remaining set of possible hypotheses which are consistent with the training examples. Typically Occam's Razor is used, which selects the shortest syntactic hypothesis (e.g. given a choice of $p(X) \Rightarrow f(X)$ or $p(X) \wedge q(X) \Rightarrow f(X)$ it would chose the former).
4. The training examples are of the form $f(object-1) \in \{T, F\}$, $p(object-1) \in \{T, F\}$, $q(object-1) \in \{T, F\}$.

1.2.2 Representing the Problem as a Version Space

We can now show the version space corresponding to this example (see Fig 1). Each node of the version space represents a hypothesis. A hypothesis can be represented by the extensional definition of the relation $f(X)$, i.e. as a set that contains the distinct values of X such that $f(X)$ is true. The hypothesis is equivalently represented by an intensional description in terms of $p(X)$ and $q(X)$. In Fig 1, each node contains both forms. Each node contains a description of the form $\{2,3\} = P$, where $\{2,3\}$ represents the extensional form, and P represents the intensional form of the hypothesis.

A hypothesis $h1$ is more general (or more specific) than a hypothesis $h2$, iff $h1 \subset h2$ (or $h1 \supset h2$). The lattice connecting the nodes is a partial order of generality between hypotheses. This partial order is most clearly seen in the extensional form. With this particular concept description language (CDL), there are only 4 distinct objects in the universe, which are denoted by $n \in \{0..3\}$. Thus the hypothesis $\neg p(X) \wedge \neg q(X)$, covers a single object ($\{0\}$), which is more general than bottom node (described as F , for all false), covering no objects ($\{\}$), and directly more specific than the hypotheses $\neg p(X)$, $\neg q(X)$, and $(\neg p(X) \wedge \neg q(X)) \vee (p(X) \wedge q(X))$, which cover $\{0, 1\}$, $\{0, 2\}$, and $\{0, 3\}$ respectively.

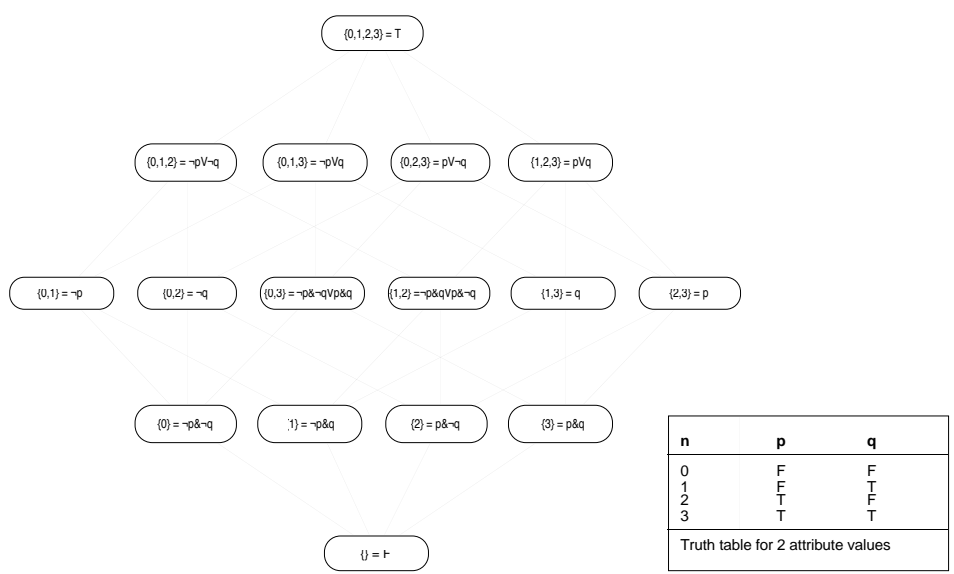


Fig 1: A version space for features $p(X)$ and $q(X)$.

A version space allows us to express the set of valid hypotheses in terms of boundary sets consisting of the most specific generalisations (G) and most general specialisations (S). Normally G and S will be described intensionally. The ability to represent a set of hypotheses by the boundary sets is the essence of the power of the version space approach.

Let us now look at how the version space changes with training examples (see Fig 2). Initially, G is set to T (standing for “all true”, the top node), and S is set to F (the bottom node). Now these describe the initial version space, because all 16 hypotheses are valid with respect to no training examples. Imagine now, that we have a negative training example $f(\text{object-1})$ where $p(\text{object-1})$ and $q(\text{object-1})$ are both true. This changes G from T to $\neg p(X) \vee \neg q(X)$, leaving only 8 possible hypotheses.

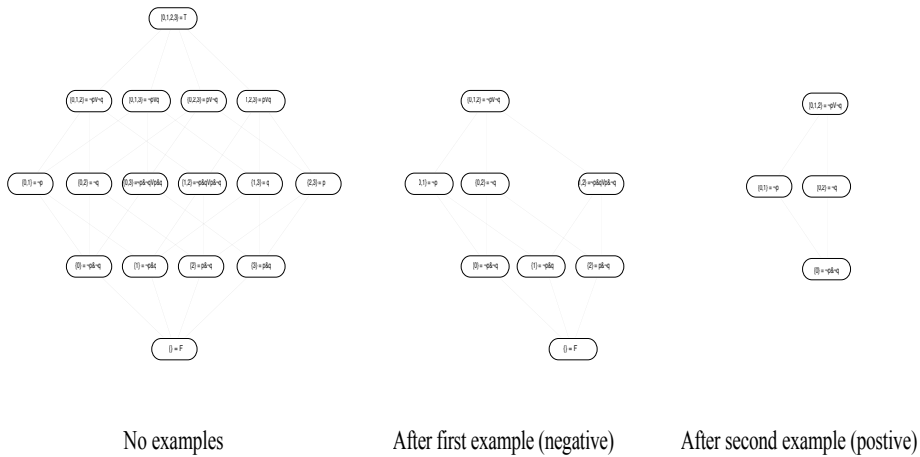


Fig 2: Updating the version space.

Now if we have a positive example where $p(X)$ and $q(X)$ are false, then the S set is changed to $\neg p(X) \wedge \neg q(X)$. There are now only 4 possible hypotheses left (G , S , $\neg p(X)$, and $\neg q(X)$). Note that if G predicts an unclassified training example to be false, then it is, but if it classifies it as positive, then it may be wrong. S is the inverse, and until we have $G = S$, then we must guess if an unclassified example falls into the space between G and S . This basis for this guess is the selection bias. If Occam's razor is used, then we would select the shortest one, in which case is a randomly choice between $\neg p(X)$ or $\neg q(X)$.

Traditionally, a supervised concept formation learning algorithm such as ID3 (Quinlan, 1986), FOIL (Quinlan 1990), or FOCL (Pazzani, 1992), directly calculates a hypothesis, without considering G & S . However, by definition, even a directly calculated hypothesis has a version space. The basis of proposed approach is, in fact, that agents make this version space explicit.

1.2.3 Distributing the Learning Problem

Let us now turn our attention away from the version space representation of the hypothesis space, and examine distributed learning in the context of this simple example. Imagine two learning agents, each of which has two distinct sets of training examples. If all these examples were given to one agent, then the relation $f(X)$ would be fully defined, and the intensional definition would be sound. However, with only 2 examples each, the best each agent can do is to generate the set of 4 possible hypotheses each, and chose one according to its selection bias. For example, let us consider a situation in which the agents have the training examples shown in Table 2 and Table 3 below.

X	p(X)	q(X)	f(X)
2	T	F	F
3	T	T	T

Table 2: Examples for Agent 1.

X	p(X)	q(X)	f(X)
0	F	F	T
1	F	T	F

Table 3: Examples for Agent 2.

This example was generated using the FOCL program (Pazzani, 1991). For agent 1, the actual version space is $G_1 = \neg p(X) \vee q(X)$ and $S_1 = p(X) \wedge q(X)$. The selected hypothesis using FOCL, F_1 , is $q(X)$. For agent 2, the actual version space is $G_2 = p(X) \vee \neg q(X)$ and $S_2 = \neg p(X) \wedge \neg q(X)$. The selected hypothesis, F_2 is $\neg q(X)$. So, what is the problem? Quite simply, each agent generates a hypothesis that is inconsistent with respect to all the examples. The correct (and sound) inference for the undistributed examples should be $(\neg p(X) \wedge \neg q(X)) \vee (p(X) \wedge q(X))$. If either hypothesis (F_1 or F_2) was tested against the other agent's two examples, it classify both examples incorrectly. So what is the solution ?

1. The simplest solution would be to send all the training examples to one agent. However, we are only dealing with distinct examples, but there could be a large number of duplicate examples. For example, let X be US Social Security ID's, and p(X) represent male or female, and q(X) represent employed or not. The hypothesis we are learning from examples may be female unemployed and male employed, and yet there could be approximately 2.2×10^8 training examples (the US population), of which there are only 4 possible distinct examples given this conceptual bias.
2. The second solution is for one agent to send the induced hypothesis to the other agent, and then for this agent to perform theory revision on it, with respect to its own examples. We have attempted this using the theory revision mechanism of FOCL (Pazzani, 1991). Agent 1 sends its induced hypothesis ($q(X) \Rightarrow f(X)$) to agent 2, which uses it as background knowledge. Even given this information, FOCL returns $\neg q(X) \Rightarrow f(X)$ as the hypothesis for agent 2. What went wrong? There is a fundamental problem facing theory revision: the theory reviser cannot know what version space the background knowledge was selected from. The FOCL revision algorithm has to make an assumption as to the version space the rule is drawn from. A natural approach is to assume the theory is over specific and/or over general. In this case, we can see that the hypotheses the theory reviser can consider are $\{F, \neg p, \neg q, \neg p \vee q, p \vee q, T\}$, (see Fig 3). None of which fall within the version space of the second agent's examples.

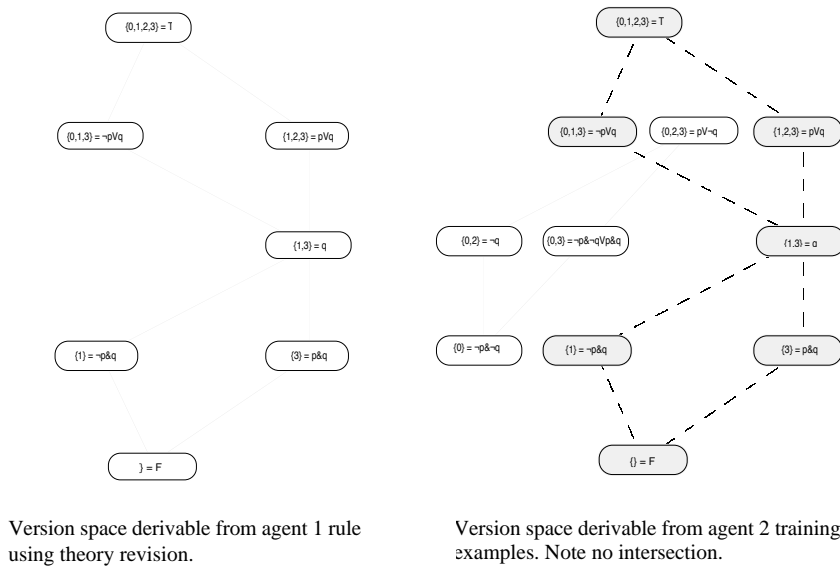


Fig 3: Theory Revision and Version Spaces.

3. The third solution forms the basis of our approach. Agent 1 must send the version space that bounds its chosen hypothesis, as well as the hypothesis itself. Now the second agent may search this for a hypothesis consistent with its example (perhaps using the induced hypothesis as a starting point), or it may use version space intersection [Hirsh 89] to find the solution.

1.2.4 Intersecting Version Spaces

Let us finally demonstrating how the version space of agent 1 can be intersected with that of agent 2. A version space is a set, and thus can be intersected. There is a simple algorithm to do this if the CDL is unrestricted logically. This is simply to conjoin the G sets, and disjoin the S sets. Fig 4 shows an example of the intersection of the two version spaces from the previous example.

The first two version spaces show the set of possible hypothesis for each agent, and the one preferred by FOCL is highlighted. The final version space is the result of intersecting the two sets, which in this case is a singleton, containing the only possible hypothesis, $(\neg p(X) \wedge \neg q(X)) \vee (p(X) \wedge q(X))$. If the intersection is not a single hypothesis, then one again could calculate the preferred hypothesis, according to a particular selection bias.

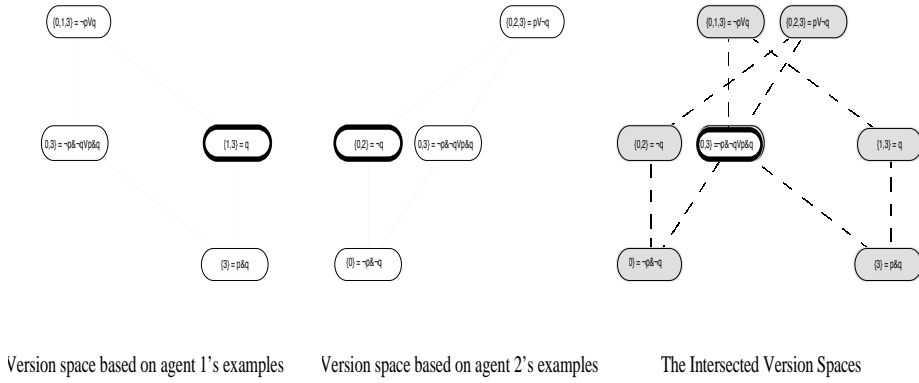


Fig 4: Intersecting Version Spaces

For completeness' sake, the symbolic form of the intersection is as follows:

$$\begin{aligned}
 G_{1 \wedge 2} &= (\neg p(X) \vee q(X)) \wedge (p(X) \vee \neg q(X)) \\
 &= (\neg p(X) \wedge p(X)) \vee (p(X) \wedge q(X)) \vee (\neg p(X) \wedge \neg q(X)) \vee (q(X) \wedge \neg q(X)) \\
 &= (p(X) \wedge q(X)) \vee (\neg p(X) \wedge \neg q(X)) \\
 S_{1 \vee 2} &= (p(X) \wedge q(X)) \vee (\neg p(X) \wedge \neg q(X))
 \end{aligned}$$

As can be seen, G and S are identical, and so we detect the version space is a singleton.

1.3 Structure of the Remainder of the Paper

In Section 2 we describe some previous multi-agent learning systems and provide an overview of version space theory. We also review the Inferential Theory of Learning which we plan to use as the basis of our communication language (Michalski, 1993).

In Section 3 we describe our initial implementation, and discuss the practicality of the approach. We also discuss a number of features of boundary set representations which may prove useful.

Section 4 attempts to outline the KQML performatives to be used in communicating inductive inferences.

Section 5 lays out the future direction of this work. In particular we discuss how to increase the efficiency of our approach, how to handle integrating different learning biases, and how to deal with limited relational (i.e. first order) learning tasks.

2. Related Work

2.1. Multi-Agent Learning Systems

Brazdil et al. (1991) discuss a number of early multi-agent learning systems and attempt a classification of the general types. They defined a multi-agent learning system as one in which two or more agents communicate during learning. Their simplest class of system is when distributed data-gathering agents communicate with a single learning agent. Given that this trivially requires an algorithm to modify a concept with respect to new examples, then this class can be thought of as abstractly including all incremental learning algorithms, e.g. ID5 (Utgoff, 1989) and theory refinement systems, e.g. EITHER (Mooney & Ourston, 1991). The second class of systems integrate individual agent's hypotheses into a single theory. One method is to simply order the hypotheses (Gams, 1989). Another method is for the agents to vote for the best hypothesis (Brazdil & Torgo, 1990). The final class of system discussed by Brazdil et al. are hybrids, in which agents both refine and vote on hypotheses during the inductive process; see for example (Sian, 1991). It should also be noted that agents may communicate knowledge induced by one agent which is then used as background knowledge for another learning goal; see, for example (Davies, 1993).

Since this initial attempt to survey activity in the area, a number of additional systems and methods have been developed: Provost & Hennessey, 1995; Silver et al., 1991; Svatek, 1994; Chan & Stolfo, 1995. There have also been a number of workshops, as well as a comprehensive survey of learning in Distributed Artificial Intelligence (Weiss and Sen, 1995) Researchers in the field of computational learning theory have also analysed aspects of the area of multiple learners. These include Kearns and Seung (1995) who model inductive inference using an "oracle". Normally, each call to the oracle returns a single classified example. They show that to model a system with multiple learners, each call to the oracle should return a summary of the examples seen by a single agent. Cesa-Bianchi et al. (1995) also address a similar problem and present algorithms for integrating the results of "experts" (single agents that make a series of predictions over time). Finally, Jain & Sharma, (1993) address the issue of team learning, in which just one agent of a team must learn the correct concept.

2.2. Version Spaces

Mitchell (1978) first introduced the version space model of inductive inference, by demonstrating how inductive inference could be seen as a search for and through a set of hypotheses consistent with the available knowledge. This set of candidate concept definitions is called a *version space*. The basic components used to construct a version space are: positive and negative examples of the concept and representational biases (also called the Concept Description Language or CDL).

The first representational bias is the set of terms that can be used to describe the concept (often called the conceptual bias). The second representational bias is a

restriction on the logical combination of these terms (often called the logical language). The examples together with the representational biases uniquely determine the version space of an inductive inference. The examples are used to eliminate all inconsistent hypotheses. The representational biases constrain the available hypotheses. The conceptual bias determines the maximum size of the version space. The logical language can decrease the size of this space, by disallowing certain hypotheses. It should be noted that background knowledge can be used deductively at any stage in this process: whether it is to determine the classification of an example or to deduce values of the terms used in the concept description language.

Mitchell demonstrated that a partial ordering exists over the hypotheses, based on the subset relation between the extensional definitions of hypotheses. This partial ordering defines a lattice which allows the version space to be represented using two *boundary* sets (G and S), which contain the most general and most specific hypotheses. As training examples are presented, the boundary sets are updated using the Candidate Elimination Algorithm (Mitchell 1978).

To complete the inductive inference a selection bias is applied to the version space. This imposes a preference order over the candidate hypotheses. An example of a selection bias is Occam's Razor, where hypotheses with shorter descriptions are preferred. However, it should be noted that all the hypotheses in the version space are valid concept definitions for the available examples. This is why maintaining version spaces keeps an inductive inference logically sound. If we select one hypothesis (using the selection bias), then we are uncertain if that concept definition is sound. However, if we maintain the set of all consistent concept definitions, it can be stated that exactly one of these definitions is sound. As more examples and/or representational biases are added, then the version space can converge to a singleton hypothesis, which is the logically sound definition of the concept. However the agent should never communicate a selected candidate hypothesis as if it had been soundly deduced until this point is reached.

It is also possible to obtain the same convergence effect by intersecting version spaces (Hirsh, 89). For a given learning goal, the examples and representational biases can be partitioned between several agents. Each agent then learns a version space consistent with the available knowledge. The resulting version spaces can then be intersected to form a new version space. This version space is identical to the one that would have been generated by a single agent with all the examples and representational biases available to it.

Hirsh extended the original version space model in several ways. Firstly, he defines the mathematical conditions (*convexity* and *definiteness*) that are necessary to represent the hypothesis space using boundary sets. Convexity occurs if a subset of a partially ordered set does not contain any "holes" with respect to the partial order. Definiteness occurs if a set has a finite number of elements at the upper and lower limits of the partial order. These properties taken together allow a hypothesis space to be represented using boundary sets (and thus the space is a version space). Hirsh proves

that all finite Concept Description Languages are representable as version spaces, and we will therefore restrict our discussion to these languages.

Secondly, Hirsh proves that two version spaces may be intersected to form a new version space that is consistent with the two original version spaces. He defines an algorithm to perform the intersection. Hirsh suggests that this algorithm be used to incrementally learn a version space, by creating version space from single examples, and then intersecting them together. He also demonstrates how background knowledge can be incorporated, by generating a version space consistent with the knowledge, which can then be intersected with the version space being learned from the examples. This allows version spaces to be used for explanation-based learning (Mitchell et.al. 1986), as well as for concept learning. It is important to note here that any two version spaces can be intersected. By proving this property, Hirsh allows us to distribute the task of inductive inference, as described earlier.

Smith (1995) adopts a similar approach to Hirsh, but describes the integration process as a constraint optimisation problem. His aim is to simulate any inductive algorithm using a set of constraints and preferences. We will not examine his work in detail here, except to note that it addresses the issue of formally specifying the inductive biases. He shows how biases may be described using regular grammars (or finite state automata).

2.3. The Inferential Theory of Learning

Michalski's (1993) Inferential Theory of Learning describes a unified theory of learning. Simply put, the theory states that there are two types of inference: deductive and inductive, which give rise to other forms of reasoning such as analogy and abduction. These occur when measures of similarity are introduced. Michalski argues that a learning problem can be regarded as the application of inference-based operators to knowledge. In this model, there are two basic sets of knowledge: the reference set and the descriptor set. The reference set defines the input to the inference process (i.e. the training examples), whilst the descriptor set defines the output model (which we refer to here as the concept description language).

Operators may manipulate the reference set or descriptor set. Eight pairs of operators are defined which transform the knowledge in the system, and three pairs of operators which manipulate the knowledge (but do not transform it). The transformation operators include the following pairs which relate to inductive inferences:

- *Generalise & Specialise* change the number of objects covered by a concept definition;
- *Concretion & Abstraction* changes the number of hypotheses by varying the descriptor set;
- *Associate* and *Disassociate* increase or decrease the size of a reference set;
- *Characterise* and *Discriminate* define the target concept in terms of the reference set(s);

- *Select* and *Generate* either select a candidate hypothesis, or generate a set of hypotheses.

The three remaining operators relate to clustering, analogy, similarity based learning, and ad-hoc modifications of existing knowledge.

Michalski's central argument is that a learning agent can be designed which applies any of these operators to knowledge to achieve a given learning goal. We agree with his thesis, but have modified it to fit the version space model in a distributed environment. Thus the ITL combined with version spaces forms the basis of our proposed KQML operators in Section 4.

3 An Initial Implementation of Distributed Learning using Version Space Boundary Sets

This section describes a prototype learning agent that uses version space boundary sets to bound induced hypotheses. It both creates the boundary sets for communication, and uses received boundary sets to bound the refinement of hypotheses. Its basic learning algorithm is an attribute value version of FOIL (Quinlan, 1990). That is, it uses training examples in the form *attribute(example-id, value)*, and generates horn clauses that describe the given concept. We use the EPILOG theorem prover (Genesereth, 1995) to provide deductive inference capabilities.

Our intended model of distributed learning is as follows:

- Agents generate the hypothesis (F) that best fits their view of the training examples and boundary sets G & S that describe the version space of the training examples (and of course the CDL).
- One of the agents then receives all the other hypotheses and bounds and then performs the following steps:
 1. Conjoin all the G bounds and S bounds to produce G' and S'
 2. Check to see if any of the hypotheses obey the following relationship:

$$S' \rightarrow F \rightarrow G'$$
 3. If one does, then this is the best hypothesis (F').
 4. Else use G' and S' to generate F'.
- The triple G', S', F' now represents the best hypothesis for the community as a whole.

Step 2 follows from the fact that the implication relational operator is the inverse of subsumption, and that the relationship defines membership of the version space. Step 3 follows from the definition of selection bias as a partial ordering of the hypotheses. Step 4 makes use of a heuristic, that the shortest F' in a DNF CDL problem, is describable in terms of the disjuncts of G'. An informal proof suggests that every disjunct in G' keeps negative examples out of the hypothesis space, but not each one is needed to cover the positive examples (S'). Therefore to simulate an Occam's Razor

selection bias, we need only try combinations of G's disjuncts. Note that we do not yet say that it is identical to the F that would be produced by an approach employing an Information Gain metric (Quinlan, 1986) or Minimum Description Length principle (Quinlan, 1994)

Our agent's basic inductive learning algorithm is as follows:

1. Initialise two sets P and N according to whether we are generating G, S, F from examples, or using G and S to generate F.
2. Create an empty clause.
3. Add literal to the clause and remove from P and N training examples that do not follow from the clause.
4. Repeat 3 until N is empty
5. Save the clause, then reset N and repeat from 2 until P is empty.
6. Return the clauses.

This is basically the Separate and Conquer approach employed by FOIL and FOCL. However, the difference lies in how P and N are represented. If we are generating G, then N is set to the negative examples available, whilst P is initially defined intensionally as a set of size $sizeof(all-true) - sizeof(N)$. The reverse is true if we are generating S. We maintain P (or N) by calculating $sizeof(current\ clause)$. If we are generating F (from examples), then P and N are set to the examples. If we are generating F from G and S, then we set P to be S' and N to be the complement of G'.

We implement the function $sizeof(X)$ based on the fact we know the concept description language contains a finite set of literals. Consider a sentence consisting of a disjunction D of conjunctions $C_1..C_n$ of literals $L_1..L_n$. Thus $sizeof(X)$ is defined as:

$$sizeof(D) = (\sum_{C \in D} sizeof(C)) - sizeof(\bigcap_{C \in D} C)$$

$$sizeof(C) = \frac{1}{\prod_{L \in C} sizeof(L)} \prod_{L \in C} sizeof(L)$$

$$sizeof(L) = |values_of_L|$$

3.1 Initial Observations

For small languages, the procedure works well. However it scales poorly. An analysis (prompted by initial experiments) of $sizeof(X)$ suggests it is NP-Hard. This is based on the observation that the algorithm is attempting to count the number of examples that would satisfy the G or S description. This is effectively the SAT problem (Floyd and Beigel, 1994) which is known to be NP-Complete. This is a disappointing result, but should have been expected. However, it is not grounds for giving up the approach.

Firstly, learning descriptions that are 3-DNF and above is known to be NP-Complete. Therefore we should not expect distribution to ameliorate the problem.

Secondly, there are a number of ways to deal with our problem:

1. Optimise the *sizeof(X)* algorithm so that is as efficient as possible, given the class it is in. An example of this would be to use bit vectors. Using this approach we have moved from being able to handle 4 binary attributes to 16 binary attributes with a reasonable response time (< 1 minute, on a 100Mhz RISC chip).
2. Approximate the G and S descriptions and calculations of *sizeof(X)*. This approach might also be desirable in order to handle noise in the training examples.
3. Reduce the Concept Description Language to a formally less hard problem (for example, learning descriptions which are 1-DNF).

In terms of evaluating the approach we will adopt the principle that the initial calculations are done “off-line”. Thus we will compare the cost to generate F’ from G’ and S’, with the cost of generating F’ from all the examples plus the cost of the transmission.

Note that this initial agent can only deal with a fixed concept description language and selection bias. We discuss this restriction in section 5. However despite currently being restricted to a fixed CDL, the bounds G and S may be used with other selection biases. However, such biases (or rather the algorithms that implement them) will have to be modified to use the intensional definition of G and S.

4. The Proposed KQML Operators

Michalski’s model of inductive inference is as follows: induction is the process of tracing backwards the tautological rule of universal specialisation; abduction is the result of tracing backward domain rules. Michalski notes that the first type of inductive inference is unsound, whilst the second may be unsound, depending on the strength of the reverse implication. However, it is possible to cast both forms of inductive inference within the version space model. It is also possible to place deductive inference within the version space model (see the previous section).

So our first modification to Michalski’s theory is to replace his model of inference with the version space model. The second step is to make the operators refer to operations over version spaces. The final step is to cast them as KQML performatives that can be sent as requests between agents.

The second step is straightforward. In Michalski’s terminology, the reference set defines the training examples (either intensionally or extensionally), and the descriptor set describes the representational biases. Michalski does not have a specific term for the selection bias, but offers the appropriate operator.

The final step is to define the KQML performatives in terms of the operators, their version space parameters. We also have to formally specify the semantics of these performatives, although currently, they are informally stated. An agent is then able to

send and receive requests for inductive and deductive inferences. They are able to integrate the replies (which are in terms of hypotheses bounded by G and S) using version space intersection.

We are proposing the following basic extensions to KQML. This is not an exhaustive list of the new performatives, but it is a representative sample. The first performative we identify is *discriminate*. This is the basic request that specifies a version space. The actual version space can either be the set of all concepts, or the boundary sets. An agent receiving such a request can use its own knowledge to generate the version space, or it may “sub-contract” the task to other agents.

(DISCRIMINATE <goal-concept>,<positive-reference-set>,<negative-reference-set>, <descriptor-set>)

The next two pairs of performatives allow an agent to modify an inference request. *Associate* increases the membership of the original reference set, whilst *disassociate* (not shown), reduces it.

Abstract reduces the number of hypotheses that can appear in the version space. For example, by removing concepts that may be used, or constraining the language. *Concrete* (not shown) increases the number of different hypotheses.

(ASSOCIATE <goal-concept>, <reference-set>)

(ABSTRACT <goal-concept>, <descriptor-set>)

The *select* performative applies the selection bias to a given version space. Initially it will return a single hypothesis, but it may be possible to return an ordered set as well.

(SELECT <goal-concept>, <version-space>, <selection-bias>)

The final performative, *intersect*, is not one of Michalski’s operators, but is required to integrate multiple version spaces.

(INTERSECT <goal-concept>, <version-space1>,...<version-spaceN>)

These performatives will be nested in usage. For example, an agent could send:

(REQUEST (SELECT g(?X) (INTERSECT g(?X) (DISCRIMINATE_{agent1} g(?X))..)))

to an integrating agent in order to achieve the effect of distributed induction. Note that this formalism does not permit the use of version space and selected hypothesis triples of the form <G,F,S>. Thus an agent must ask (or tell) *Discriminate* and *Select* if it wishes to transmit the bounds and the hypothesis.

The parameters to these performative are informally defined as follows:

1. <goal-concept> : The name of the target concept.
2. <reference-set>: A definition of the examples. It could be an extensional set, or a set of knowledge that define the examples, etc. In many cases there will be two reference sets, representing positive and negative training examples.

3. `<descriptor-set>`: This defines the representational biases. As the representational bias is a language, we assume we will specify this using a grammar of an appropriate class. For a finite language a regular expression (Floyd and Beigel, 1994) will suffice.
4. `<version-space>`: The operators imply the generation of a version space, either enumerated fully, or defined as a boundary set. Therefore the *reply* to an operator request consists of a version space. It is also used as the input to and output from an *intersect* performative.
5. `<selection-bias>`: It is not yet certain how we can represent this. It is generally a procedural bias, therefore we could just name a procedure (e.g. *'most-general'*). However, it could also be specified declaratively (Smith, 1995).

These performatives and parameters are still under investigation, and will be more refined as we implement learning agents.

5. Discussion & Future Work

This is very much work in progress, and to date we have not fully expanded all the details. Nor have we evaluated the approach empirically. However, a number of issues have already been raised, which we will discuss here.

We believe our method subsumes two of the previous approaches to multi-agent learning: the voting mechanism can be thought of as a group of agents attempting to find at least one hypothesis that is contained in each of their version spaces for the problem; theory revision is equivalent to modifying a hypothesis so it is in the version space of the agent performing the revision. However, there are two problems with these methods. Firstly, it might not be possible to find one hypothesis consistent with all the version spaces even if one does exist, given that many algorithms use a hill climbing approach. Secondly, they result in the selection of a single hypothesis, which might be incorrect. The version space approach avoids both these problems. It is not certain whether our approach will have the efficiency of these general methods; or that of specialised distributed learning algorithms (or inherently incremental learners). However, we do believe that it is a general approach which should allow inductive learning algorithms to be modified to used in agents. It is also perceivable that this approach can be specialised (for example, a neural network learning algorithm could send weight vectors that correspond to G and S), or can be generalised along the lines of (Smith 95) where all the biases are declarative.

The specific concerns so far are:

1. The approach adopted so far involves an NP-Complete algorithm.
2. We have not investigated how to handle noise and uncertainty, nor have investigated how to accommodate shifts in bias (Des Jardins and Gordon, 1995).
3. Our implementation is an attribute-value learner.

4. We do not have a mechanism for either integrating version spaces of different concept description languages or manipulating different reference or descriptor sets.

Future work will seek to address these issues in part. The first two points may well be jointly solved by approximating the version space boundary sets, using statistical information. The third point will be pursued by first taking the approach employed in LINUS (Lavrac and Dzeroski, 1994), and then examining the possibility of modifying an ILP algorithm such as GOLEM (Muggleton, 1992) to generate and use version spaces. Recent work by Nienhuys-Cheng and De Wolf (1996) suggests that certain classes of ILP problems are representable with boundary sets. We will not initially seek to address the fourth point because of the limited usage it would have, as version spaces of different languages can only be integrated on a lowest common denominator basis.

Acknowledgments

The authors would like to acknowledge funding by the UK Engineering and Physical Sciences Research Council for the initial phases of this research. They also acknowledge the support and encouragement of Mike Genesereth at Stanford University, and helpful comments from Nils Nilsson, Anna Patterson, Josefina Sierra, Lise Getoor, Ofer Matan, Ken Brown and Will Schuman.

Bibliography

- P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. Van de Velde (1991). Learning in Distributed Systems and Multi-Agent Environments, In *Proceedings of the European Working Session on Learning (EWSL91)*, Springer-Verlag, pages 424-439, Porto, Portugal.
- P. Brazdil and L. Torgo (1990). Knowledge Acquisition via Knowledge Integration, in *Current Trends in AI*, B. Wielenga et al.(eds.), IOS Press, Amsterdam.
- N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth (1995). *How to Use Expert Advice*, Technical Report UCSC-CRL-95-19, University of California, Santa Cruz, CA.
- P. K. Chan and S. J. Stolfo (1995). A Comparative Evaluation of Voting and Meta-Learning on Partitioned Data, In *Proceedings of the Twelfth International Conference on Machine Learning (ML95)*, Morgan-Kaufmann, pages 90-98, Lake Tahoe, CA.
- W. Davies (1993). *ANIMALS, An Integrated Multi-Agent Learning System*, M.Sc. Thesis, Department of Computing Science, University of Aberdeen, UK.
- M. Des Jardins and Diana F. Gordon (1995). Evaluation and Selection of Biases in Machine Learning, *Machine Learning*, 20:1-17.

- T. Finin, J. Weber, G. Wiederhold, M. Geneserth, R. Fritzson, D. MacKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck (1993). *Draft Specification of the KQML Agent-Communication Language*, Unpublished Draft.
- R. W. Floyd and R. Beigel (1994). *The Language of Machines*, Computer Science Press, NY.
- M. Gams (1989). New Measurements Highlight the Importance of Redundant Knowledge, In *Proceedings of the 4th European Working Session on Learning (EWSL89)*, Pitman-Morgan Kaufmann, pages 71-80, Montpellier, France.
- M. Geneserth (1995). *Epilog for Lisp 2.0 Manual*. Epistemics Inc., Palo Alto, CA.
- H. Hirsh (1989). *Incremental Version Space Merging: A General Framework for Concept Learning*, Ph.D. Thesis, Stanford University.
- S. Jain and A. Sharma (1993). *Computational Limits on Team Identification of Languages*, Technical Report 9301, School of Computer Science and Engineering, University of New South Wales, Australia.
- M. Kearns and H. S. Seung (1995). Learning from a Population of Hypotheses, *Machine Learning*, 18:255-276.
- N. Lavrac and S. Dzeroski (1994). *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, Herts, UK.
- R. S. Michalski (1993). Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning, *Machine Learning*, 11:111-151.
- T. M. Mitchell (1978). *Version spaces: An Approach to Concept Learning*, Ph.D. Thesis, Stanford University.
- T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabell (1986). Explanation-Based Generalization: A Unifying View, *Machine Learning*, 1:1-33.
- R. J. Mooney and D. Ourston (1991). A Multistrategy Approach to Theory Refinement, In *Proceedings of the International Workshop on Multistrategy Learning*, pages 115-130, Harper's Ferry, WV.
- S. Muggleton (1992). *Inductive Logic Programming*, Academic Press, London, UK.
- S. H. Nienhuys-Cheng and R. De Wolf (1996). Least Generalizations and Greatest Specializations of Sets of Clauses, *Journal of Artificial Intelligence Research*, 4:341-363
- M. Pazzani and D. Kibler (1992). The Utility of Knowledge in Inductive Learning, *Machine Learning*, 9: 57-94.
- F. J. Provost and D. N. Hennessy (1995). Distributed Machine Learning: Scaling up with Coarse Grained Parallelism, In *Proceedings of the Second International*

Conference on Intelligent Systems for Molecular Biology (ISMB94), AAAI Press, pages 340-348, Stanford, CA.

- J. R. Quinlan (1986). Induction of Decision Trees, *Machine Learning*, 1:81-106
- J. R. Quinlan (1990). Learning Logical Definitions from Relations, *Machine Learning*, 5:239-266
- J. R. Quinlan (1994). The Minimum Description Length Principle and Categorical Theories, In *Machine Learning, Proceedings of the 11th International Workshop (ML94)*, Morgan Kaufmann, pages 233-241, New Brunswick, NJ.
- S. S. Sian (1991). *Learning in Distributed Artificial Intelligence Systems*, Ph.D. Thesis, Imperial College, UK.
- B. Silver, W. Frawley, G. Iba, J. Vittal, and K. Bradford (1990). ILS: A Framework for Multi-Paradigmatic Learning, In *Machine Learning, Proceedings of the 7th International Workshop (ML90)*, Morgan Kaufmann, pages 348-356, Austin, TX.
- B. D. Smith (1995). *Induction as Knowledge Integration*, Ph.D. Thesis, University of Southern California.
- V. Svatek (1994). *Integration of Rules from Expert and Rules Discovered in Data*, Unpublished Draft, Prague University of Economics, Czech Republic.
- P. E. Utgoff (1989). Incremental Induction of Decision Trees, *Machine Learning*, 4:161-186.