

Learning Domain Theories using Abstract Background Knowledge

Peter Clark and Stan Matwin
Ottawa Machine Learning Group
Computer Science, University of Ottawa
Ontario, CANADA K1N 6N5
{pclark,stan}@csi.uottawa.ca

Abstract

Substantial machine learning research has addressed the task of learning new knowledge given a (possibly incomplete or incorrect) domain theory, but leaves open the question of where such domain theories originate from in the first place. In this paper we address the problem of constructing a domain theory itself from more general, abstract knowledge which may be available. The basis of our method is to first assume a structure of the target domain theory, and second to view background knowledge as constraints on components of that structure. This enables a focusing of search during learning, and also produces a domain theory which is explainable with respect to the background knowledge. We present a general framework for this task and describe learning algorithms which can be employed, and then apply a particular instance of it to the domain of economics. In this application domain, the background knowledge is a qualitative model expressing plausible economic relationships, examples are sets of numeric economic data, and the learning task is to induce a domain theory for predicting the future movement of economic parameters from this qualitative background knowledge and data. We evaluate the value of this approach, and finally speculate on ways this method could be extended.

1 Introduction

Machine learning research is now heavily focussed on knowledge-intensive learning methods (eg. [1]), an essential step in the field's development. An important advance has been the development of systems which will learn given an initial (possibly incomplete or incorrect) domain theory (eg. ML-Smart [2], Focl [3], Forte [4], Either [5]).

Despite this, the issue of how such domain theories can be learned in the first place remains a difficult open problem. Although there have been significant recent advances in inductive learning technology, in particular in Inductive Logic Programming (eg. Golem [6], Foil [7]), it is still well-recognised that to learn all but the simplest domain theories some other form of background knowledge is required to constrain search. The purpose of this paper is to present a framework for a simple but general description of this learning task, and describe and evaluate algorithms for its solution.

For the purposes of this paper we define a **domain theory** to be a system of knowledge for solving some specific target task, and **background knowledge** more generally to refer to arbitrary available knowledge. We thus view an idealised domain theory as task-specific, coherent¹

¹loosely meaning internally consistent: A formal definition of coherence is difficult to capture, eg. see [8] and related papers for discussion.

and non-redundant (avoids details irrelevant to the task). In contrast, background knowledge may be over-general (for the performance task), ambiguous and contain more inconsistencies. The learning task is thus partly one of knowledge extraction (from the background knowledge) and partly one of inductive elaboration (using known facts). The learned domain theory can be viewed as a ‘concretisation’ of the more general knowledge.

The main contributions of this paper are to present a simple methodology for formalising this task, and to illustrate and apply a particular instance of it to the domain of economics. Our methodology involves two steps: first, assume a domain-independent *structure* of the target domain theory, and second to view background knowledge as specifying *constraints on components* of that structure. The result allows us to first focus search on hypotheses which are ‘good’ with respect to the background knowledge (and hence learn more accurately or quickly, conditional on the quality of that knowledge), and second to learn a domain theory compatible with background knowledge, and hence explainable by reference to it. This explainability aspect is particularly advantageous if the results of learning are to later be incorporated within a body of existing knowledge, as is becoming increasingly the case in machine learning research.

We also present a simple but general instance of this approach, in which the background knowledge is expressed as a qualitative model (QM) and the examples are sets of numeric data. Terms used in the model (eg. ‘high inflation’) are abstract and have an ambiguous interpretation. The structure of the QM defines ‘plausible’ rules in the domain, expressed in terms which are not necessarily observable (ie. different from those used to describe examples). Here a *language gap* exists between the terminology of the background knowledge and of the examples’ descriptions, with no clear mapping between the two. This problem is a recurring one in learning (eg. [9, 10]). We apply our framework by assuming a ‘two-layered’ structure for the target domain theory, in which the top layer comprises of qualitative prediction rules extracted from the model and the bottom layer defines a mapping between the qualitative terms and quantitative data, thus spanning this gap. We evaluate the application of this background knowledge to the economic prediction problem. Finally we speculate on ways in which a reverse process could be added, by which learning could feed back to improve the quality of the original background knowledge itself. The potential of this feedback offers some exciting possibilities for extending the method presented in this paper.

2 Learning Domain Theories

2.1 Definitions

For the purposes of this paper we define a **domain theory** to be a system of knowledge for solving some specific target task. For a classification of examples task, a domain theory will be able to compute a class value for each example, ie. mapping examples onto the set of possible classes:

$$DT : E \rightarrow C$$

where E and C are the sets of all possible examples and classes respectively. In contrast, we define **background knowledge** more generally as any knowledge available a priori for learning.

This task-specific definition of a domain theory is a more specific notion than is usual: we adopt it because we wish to distinguish a coherent system of knowledge adequate for solving a task from other knowledge which may be available. In our learning framework, the background knowledge does not constitute a domain theory (ie. cannot perform classification) but expresses constraints on how a domain theory can be constructed.

2.2 A Simple Methodology

The general problem of learning domain theories is immense: For many theory representation languages used in ML the space of possible theories is huge or even infinite. A related problem is how to bring available background knowledge, often expressed in a completely different language to that used to describe available data, to bear on the search problem.

Our general methodology is simple, but we hope contributes a useful way of conceptually decomposing the learning task:

1. assume a domain-independent *structure* for the learned domain theory.
2. view background knowledge as specifying *constraints on components* of this structure.

By assuming a domain theory structure, the learning problem can be decomposed into sub-problems, and by interpreting background knowledge as constraints we can define restricted search spaces for solving each sub-problem.

For the rest of this paper we work with a particular instance of this approach, in which the domain theory is assumed to have a ‘two-layered’ structure (defined below) and the background knowledge expresses constraints on each component layer. While we focus on this simple instance of the approach, we note that we could apply the same style of solution to other more complex structures, and hence why we have separated out its general form. Many of the issues, such as interaction between theory components, will be common to other structures also.

2.3 An Instance of this Approach

2.3.1 A Structure for Domain Theories

In general we assume a logic language L for representing a domain theory, defined in a similar way to Prolog, with connectives \wedge and \rightarrow . The assertions admissible in L will be of the type:

$$P_1 \wedge \dots \wedge P_i \rightarrow Q \tag{1}$$

according to the usual syntax of the Horn clause logic languages.

The particular domain theory structure we assume has two ‘layers’, the top layer using the abstract terminology of the background knowledge and the bottom layer relating this terminology to the basic facts known about examples. This two-layer approach thus accounts for background knowledge being expressed in general, abstract terms $T_i \in T$ which have an imprecise or ambiguous mapping onto the facts $F_i \in F$ known about examples. Thus we assume a complete domain theory is a set of clauses of the form (1), consisting of the union of two clause sets as follows:

Prediction rules: A set of clauses of the form

$$T_1 \wedge \dots \wedge T_i \rightarrow C_j \tag{2}$$

where the $T_i \in T$ are abstract terms used to express background knowledge and C_j is a class prediction.

Term Definitions: A set of clauses of the form

$$F_1 \wedge \dots \wedge F_i \wedge G_1 \wedge \dots \wedge G_j \rightarrow T_k \tag{3}$$

where $F_i \in F$ are literals whose truth value on examples is known and $G_i \in G$ are other literals with known definitions (eg. arithmetic tests).

The T_i can be described as ill-defined ‘theoretical’ terms, and the F_i as ‘observational’ terms [11], the two-layer structure distinguishing between these two vocabularies of background knowledge and observation. We call a clause of type (2) a **rule**, and a clause of type (3) a **definition**. A domain theory thus consists of a set of rules and set of definitions, which we will refer to as $RSet$ and $DSet$ respectively.

2.3.2 Background Knowledge

In our learning framework, we do not assume that the rules of a domain theory are given: instead we only assume that we know which rules are ‘plausible’ (ie. can occur in $RSet$) and which are not. Using this particular domain theory structure, we thus take **background knowledge** to be a **specification of the space of rules** $RSpace$ from which an $RSet$ can be extracted. In the economics application the background knowledge is in the form of a qualitative model whose nodes are predicates in T . A path in the model (eg. $\text{sales} \xrightarrow{Q^+} \text{profits} \xrightarrow{Q^+} \text{wages}$) corresponds to a rule (eg. ‘if sales high and profits high then wages increase’), and hence the model specifies the space of ‘plausible’ rules which may be used in $RSet$.

Similarly, we do not assume definitions of the terms T_i are given, but again only the spaces of plausible definitions $DSpace_i$ are specified. We elaborate on this later.

2.3.3 The Learning Task

We can thus state the learning task as applied to this domain theory structure:

Given:

- Background knowledge comprising:
 - A set of plausible rules (ie. a specification of the rule space $RSpace$)
 - A set of plausible definitions for each term in those rules (ie. a specification of the definition space $DSpace_i$ for each term T_i)
- A set of examples E
- A metric of quality of a domain theory
- A limited amount of computational resources available

Find the best domain theory possible comprising:

- A set of rules $RSet$ drawn from $RSpace$
- A set $DSet$ of term-definition pairs $\{T_i, D_{ij}\}$, each D_{ij} drawn from $DSpace_i$ such that all terms T_i used in $RSet$ are defined.

2.3.4 Issues for a Learning System: The Multi-Search Problem

Given a structure for the domain theory, the task for a learning system is to find suitable components to instantiate it. In our case, this comprises:

1. Selection of a rule set, expressed in the terminology of the background knowledge.
2. Construction of definitions of the terms used in this rule set.

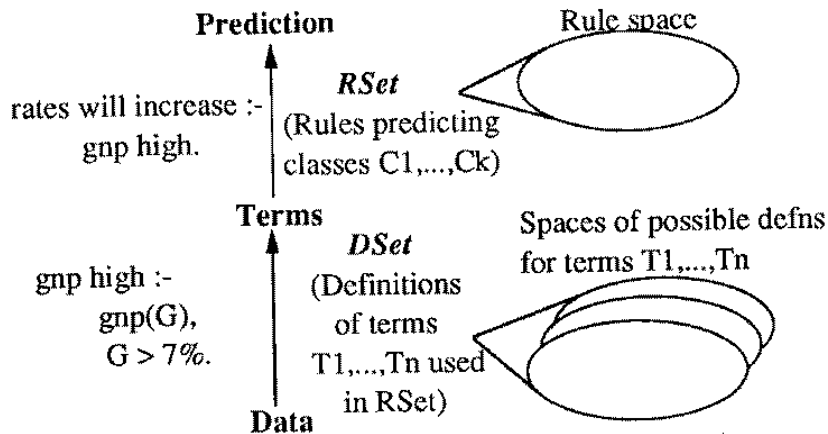


Figure 1: The Two-Layered Theory Structure and Search Problem.

We depict this two-step task schematically in Figure 1.

The key issue for learning is how to handle the interdependence between these different searches. In the two layer problem, the two searches are not independent; in fact it is precisely their interdependence which makes solutions to this problem difficult. This mutual dependency problem can be loosely stated as follows: To search for a good rule set, we need to know the definitions of the terms in those rules so that their accuracy on training data can be computed. However to evaluate which definition of a term maximises a rule set’s quality, we need to already have that rule set selected. We conjecture that this two-search (or more generally multi-search) problem structure will be typical of future larger scale learning systems, in which tractability constraints force decomposition of the learning problem into mutually-dependent smaller search tasks.

3 Related Work and Context

Our system’s goal is to learn a domain theory – a system of knowledge for solving some target task – given abstract background knowledge. In this section we review work related to this goal.

3.1 Theory Construction Systems

In the simplest case, propositional rule learning systems (eg. C4.5 [12], CN2 [13]) can be seen as learning very simple ‘domain theories’ for a classification task, using no background knowledge. The limited expressiveness of these systems’ rule languages, and the labour-intensive task of choosing a suitable representation of examples, are well-known limitations which more recent work has sought to overcome.

Work in constructive induction (CI) has sought to increase the expressive power available for expressing learned knowledge, by allowing systems to introduce intermediate terms in its representation language not present in the original data. This can greatly simplify the structure of useful domain theories, allowing search to locate them more easily. However, the space of intermediate terms which can be introduced is potentially huge and can itself be intractable to adequately explore. Even with simple domains (eg. tic-tac-toe) search can take considerable time (eg. [14, 15]).

Recent research in inductive logic programming (ILP) has also sought to learn domain theories in more expressive languages (in particular Horn clause languages). These languages allow

a greater range of theories to be described, but at the same time the problem of controlling the greatly expanded search becomes more acute. Addressing this problem has been an important thrust of the research. Several ingenious methods have been developed, including techniques such as determinacy constraints ([6, 7]), mode and type declarations, and rule schemas (eg. in CIA [16])². In addition, allowing users to specify ‘background predicates’ which can be included in the domain theory obviates (or at least reduces) the requirement for CI capabilities, a huge benefit for learning. However, even with these constraints the search problem still prevents all but relatively simple domain theories being induced. The work here can be seen as taking these areas of research one step further, allowing abstract, domain-dependent knowledge to further constrain search and resulting in more sophisticated domain theories being induced.

3.2 Theory Modification Systems

A second class of learning problem which has received attention is that of **theory modification**. Here, it is assumed a domain theory *is* available but may contain errors. The learning task is to remove these errors, typically using examples to guide learning (eg. Either [5], Forte [4], Krust [18]). Our work shares some aspects of these systems, but differs in that we do not assume a ‘nearly correct’ theory but instead assume more abstract knowledge. Our concern is thus with removing the vagueness and ambiguity in the abstract knowledge, rather than removing errors of omission or commission in an already given ‘concrete’ theory.

3.3 Knowledge-Based Theory Construction Systems

3.3.1 Operationalisation and EBL

In our framework, the search for a domain theory is constrained by more general background knowledge. We can view this as a process of specialising background knowledge, or, in EBL terms, of *operationalising* the non-operational, abstract knowledge available.

In early EBL work (eg. [19]) it was assumed that the domain theory included precise definitions of non-operational terms; in other words, there was a ‘bridge’ given spanning the gap between non-operational and operational expressions. More recent work on integrating EBL and similarity-based learning (SBL) (eg. [2, 3]) has looked at relaxing the assumption that non-operational terms will have correct definitions. The systems ML-SMART and FOCL both consider the case where there may be disjunctive and possibly erroneous definitions of terms. To learn the ‘correct’ operationalisation, training examples are used to isolate which of the disjunctive definitions is most accurate, followed by inductive learning to improve this definition. (We could loosely call this ‘example-guided operationalisation’).

Our framework here can be seen as a development of this approach, but differing in two important ways:

1. We do not have a non-operational theory to start with: we wish to construct one from a space of rules specified in the background knowledge.
2. Also, we wish to operationalise this entire constructed theory, not just a single predicate. As a result, we must account for global repercussions of local operationalisation choices on the rest of the theory. This complicates the evaluation of operationalisation decisions, and presents a credit assignment problem.

²An excellent overview of this field is given in [17].

Parameter	Details	Units
gnp	Gross National Product	% increase (1 year)
sales	Retail sales	% increase (1 year)
unemp	Unemployment	%
inflatn	Consumer prices	% increase (1 year)
wages	Wages/earnings	% increase (1 year)
stocks	Stock price indices	% increase (1 year)
money	Money supply (broad)	% increase (1 year)
rates	Interest rates (bank prime lending)	%
ca_bal	Current account balance	% increase (1 year)
exchange	Trade-weighted exchange rate	% increase (1 year)

Table 1: The ten economic parameters used.

3.3.2 Refinement of Abstract Models

Our work is also closely related to work on qualitative model refinement (eg. by Mozetic [20]) in which an abstract model is repeatedly instantiated until a fully specified model is completed. Our approach can be seen as a development of this, in which we do not assume a strict top-down (abstract-to-specific) learning approach and in which the learned domain theory may be in a different form (ie. not a qualitative model) to the background knowledge.

4 Application to the Domain of Economics

We now describe the application of this learning model to the domain of economics. We briefly describe the examples, the learning task and the qualitative background knowledge available, and then examine the effectiveness of learning in this application domain.

4.1 Examples and the Learning Task

4.1.1 Raw Economic Data

The raw economic data consists of the values of 10 economic parameters $P_i \in P$ for a particular country at a particular time, taken from an economic magazine (the Economist). The parameters used are as shown in Table 1, and example values shown in Table 2. We take values for 12 countries (Australia, Belgium, Canada, France, Germany, Holland, Italy, Japan, Sweden, Switzerland, UK, USA) over a time-span of 8 consecutive years (1983-91). To simplify the presentation here, we will describe our data set as containing one reading per year (per parameter per country); in fact we took data every six months over this time-span, ie. using 18 time points in total.

4.1.2 The Learning Task

A general learning task would be to predict (for some country) the value of some parameter P_i in year $Y + 1$ given values of all parameters P in years up to and including Y .

For this work, we adopt a slightly simpler learning task: namely to predict the direction of

Param	Country	Value (%)								
		1983	1984	1985	1986	1987	1988	1989	1990	1991
gnp	canada	4.8	5.0	4.4	3.5	4.1	4.0	2.3	0.5	-0.8
sales	canada	4.0	2.6	6.9	3.1	7.7	3.7	-1.1	-4.1	-11.8
:	:	:	:	:	:	:	:	:	:	:

Table 2: The raw economic data.

Example	Attribute-values										Class GNP = inc./dec.?
	current yr				prev yr			prev prev yr			
	gnp	sales	unemp	...	gnp	sales	...	gnp	sales	...	
canada 1985	4.4	6.9	10.2	...	5.0	2.6	...	4.8	4.0	...	decrease
canada 1986	3.5	3.1	9.4	...	4.4	6.9	...	5.0	2.6	...	increase
:	:	:	:	:	:	:	:	:	:	:	:

Table 3: Training examples for classifying future movement of GNP.

change of parameter P_i , ie. **increase** or **decrease**³ from year Y to $Y + 1$. This converts the prediction task into one of symbolic classification.

Rather than predict for one particular parameter, we have chosen to predict for all the 10 parameters' directions of change. Thus the final rule set $RSet$ is in fact the union of 10 separately learned sets (one for each parameter) but still constrained to share common definitions of terms they use ($DSet$).

4.1.3 Training Examples

Positive and negative examples are extracted from the raw data by choosing a year, observing whether the parameter of interest increases or decreases, and recording the values of parameters for that year and previous years. To constrain the task, we only look two years back in the past. This transformation is illustrated in Table 3. Ten training sets are extracted from the raw data in this way, one set for each of the 10 parameters to predict for.

4.2 Background Knowledge

4.2.1 Specifying the Rule Space with a Qualitative Model

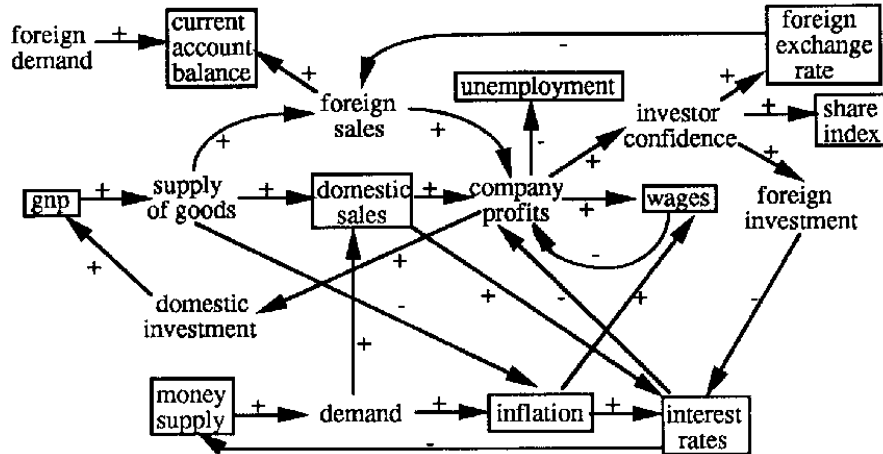
While we do not have enough economic knowledge for parameter prediction directly, we do have *some* knowledge of the relationship between economic parameters. Some potential rules are plausible according to this naive knowledge, whereas others are not. For example, the rule

“if rates high then GNP will decrease.”

has a plausible explanation: high rates reduce companies' profits, reducing future investment and eventually reducing productivity and the country's GNP.

We capture this naive knowledge in the form of a qualitative model, expressing the believed relations between the 10 parameters P and an additional 8 unmeasurable parameters Q

³We do not include **unchanged**, as it is unusual for a parameter to remain precisely the same in two consecutive years. If it does, we assign the class **increase**, ie. strictly **increase** means increase-or-equal-to.



(See Table 1 for explanation of the abbreviations used)

Figure 2: The Economic Qualitative Model used as Background Knowledge.

(confidence, profits, supply, domestic demand, foreign demand, domestic investment, foreign investment, foreign sales). The model can be depicted as a network of nodes and directed arcs, each node representing one of these parameters and each arc representing a qualitative influence of one parameter on another. Each parameter has an associated numeric value (for a given country and year), but in the model we use just two qualitative values, **high** or **low**. As in Qualitative Process Theory [21], we label the arcs $Q+$ to denote a positive influence and $Q-$ a negative influence. If we can find a path from one parameter P_i to another P_j , then we say there is a plausible relationship between P_i and P_j , explainable by the path, which can be used to form a rule in the domain theory. The complete model thus specifies the space of rules $RSpace$ from which a ‘concrete’ domain theory can be extracted, each path in the model corresponding to a different rule.

The model we use is depicted in Figure 2, constructed manually by the authors in the style of Charniak’s economic model [22]. Boxed items are the 10 measurable parameters P , described in Table 1. Unboxed items are the unmeasurable parameters Q , which are not included in rules extracted from the model but can be used for explanation purposes. The algorithm for extracting rules is given in Appendix A. The idea of extracting plausible rules from a QM is similar to DeJong’s Plausible EBL [23]. In our case, though, this is only one component of the overall learning task: the abstract terms used in the model are ill-defined, hence validation of extracted rules cannot proceed independent of the second component of the learning problem, as we now describe.

4.2.2 Specifying the Definition Space

While our qualitative model looks similar to the QMs of Qualitative Process Theory, it differs in one important respect: we do not assume a particular mapping from qualitative values onto quantitative values. For example, the parameter GNP can take qualitative values (**high** or **low**), but what exactly constitutes “high GNP”? This could include:

- GNP > some constant
- GNP > previous year’s GNP
- GNP > average of the n previous years’ GNP
- GNP > world average GNP for this year

- etc.

In this respect the model is incompletely specified, and a “gap” exists between its abstract qualitative terminology and the hard facts of the economic data.

However, while we do not know which definitions of these terms are most suitable, we *do* know some constraints on what they should look like. For example, a definition of “high GNP” should at least refer to the current GNP value, and probably should not refer to some obscure value of a parameter in another country several years previously. This sort of knowledge constitutes the second part of the background knowledge, namely a specification of the space of plausible definitions of terms in the model.

For our economic task, we impose the following constraints on definitions:

- A definition of “high P_i ” should involve some test that the current value of P_i is greater than some other value.
- This other value might be a constant, or some function of previous years’ values of P_i
- Data more than two years old, and from other countries, is probably not relevant to this other value.
- That function should have a simple algebraic structure.

These constraints should be viewed as a working hypothesis for the purposes of this research; we accept that some other definitions outside this scope may also be plausible.

These constraints thus define a space $Dspace_i$ of definitions of the form:

$$v_{iy} \geq f(v_{iy-1}, v_{iy-2}, K) \quad \longrightarrow \quad P_i = \text{high}^4 \quad (4)$$

where **high** is the qualitative value of parameter P_i in year Y (for country C)

where v_{iy} is the known numeric value of parameter P_i in year Y

v_{iy-1} is the same in year Y-1

v_{iy-2} is the same in year Y-2

K is a constant $\in \{1, 1.1, 1.2, 1.3, 1.5, 2, 3, 4, 5, 10, 20, 40\}$

$f()$ is an arbitrary arithmetic expression constructed with the operators

$\{+, -, /, *\}$, and in which the values v_{iy-1} , v_{iy-2} and K appear at most once.

5 Learning Algorithms

5.1 Introduction

The basis of our method is the structuring of an otherwise intractable learning task into separate search problems, through the specification of a structure for the target domain theory. We conjecture that for any large-scale learning task, this style of decomposition will be necessary. This decomposition of a large scale learning task into a multi-layer structure in this way is novel, and raises challenges for its solution.

In this section, we describe algorithms for conducting search in the spaces which our particular assumptions about theory structure produce. In the subsequent section, we investigate their

⁴In Horn clause form this would be expressed as (taking $f = v_{iy-1} + v_{iy-2}/3$ as an example):
 $\text{gnp}(C, Y, \text{high}) :-$

. Y1 is Y-1, Y2 is Y-2, $\text{gnp}(C, Y, V)$, $\text{gnp}(C, Y1, V1)$, $\text{gnp}(C, Y2, V2)$, $V_i = V1 + V2/3$.

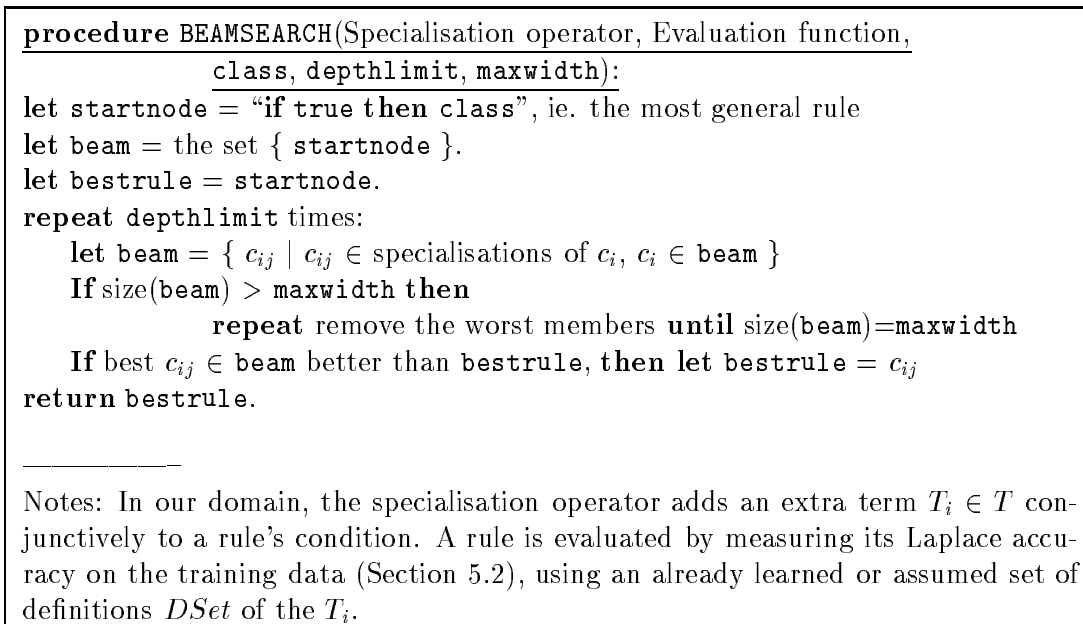


Figure 3: The General-to-Specific Beam Search Algorithm.

application to our economics problem. These algorithms should be viewed as possible tools for addressing search, rather than definitive solutions to the search problem. As we discuss later, several outstanding issues remain for handling this learning task, and in multi-search learning tasks in general.

5.2 Search for a Rule Set

The ‘top layer’ of our domain theory (Figure 1) requires extracting a set of rules from $RSpace$. Given a set of definitions $DSet$ for all terms in $RSpace$, a standard covering algorithm can be applied:

```

let  $RSet = \{\}$ 
foreach class  $C_i$ 
  let  $TrainExs =$  the training examples
  repeat
    find the best rule  $R$  (covers many examples  $E \in TrainExs$  of  $C_i$  and few of
       $C_{j,j \neq i}$ , as specified by some evaluation function)
    remove examples of  $C_i$  covered by  $R$  from  $TrainExs$ 
    add  $R$  to  $RSet$ 
  until all examples of  $C_i$  have been covered or no more rules can be found.
return  $RSet$ .

```

The quality of a hypothesis rule is based on its performance on the training examples⁵. A standard beam search covering algorithm can be used to perform the search in line 5 (**find** the best rule...), and is described in Figure 3.

⁵We use the Laplace estimate $Q = (n_p + 1)/(n_p + n_n + n_c)$ where n_p , n_n and n_c are the number of positive examples covered, negatives covered and total number of classes (= 2, here) respectively.

```

procedure OPTIMISE(DSet, DSpace1,...,DSpacei, Evaluation function):
repeat until quiescence or resources expire:
  for i = 1 to n do:
    forall definitions  $d_{ij} \in \text{DSpace}_i$  (eqn 4) for term  $T_i$  do:
      Evaluate the quality of a modified DSet, =
        DSet but with defn of term  $T_i$  remade as  $d_{ij}$ 
      if the best choice  $d_{i\text{best}}$  improved DSet
      then change defn of term  $T_i$  in DSet to be  $d_{i\text{best}}$ 
return DSet

```

Notes: In our domain, *DSet* is a set of 10 term-definition pairs ($\{T_i, d_{ij}\}$), defining the 10 abstract, qualitative terms T_i in the QM (Figure 2) in terms of the observable facts (Table 3) each according to a formula of the form of equation 4. DSpace_i is the space of all possible formulas of the form equation 4. Given an already learned or assumed rule set *RSet*, a *DSet* is evaluated by measuring the overall performance of *RSet* on training data using *DSet* to define its terms, according the formula in Appendix B.

Figure 4: The Local Optimiser Algorithm.

5.3 Search for a Definition Set

A definition set consists of a set of term-definition pairs for each term T_i used in *RSet*. In our economic domain, there are 10 terms to define and hence the definition set can be represented as a 10 element vector, element i being the definition of term T_i .

Given a rule set, we wish to optimise the choices of definitions of each term. To do this, we employ a local optimisation algorithm which repeatedly optimises choice of d_i , holding the other 9 decisions $d_j, j \neq i$ fixed, for different values of i until quiescence is achieved. This algorithm is shown in Figure 4. One equally valid alternative would be to apply a genetic algorithm, but for the purposes of this paper we have not included this within our scope.

5.4 Choosing a Start Point

These two algorithms present a boot-strapping problem: To induce rules, a set of definitions is required, and conversely to optimise definitions a rule set is first required. To start the process, we experimented with three starting points for a definition set *DSet* as follows:

Random: Select a definition for each term T_i at random.

Normal: Select definitions for all terms corresponding to the the following simple form of equation 4:

$$v_{iy} \geq v_{iy-1} \quad \longrightarrow \quad P_i = \text{high}$$

We select this form as it corresponds to one intuitive interpretation of a ‘high’ parameter value, namely ‘increased with respect to the previous year’.

Entropy: As we wish to find rules which discriminate positive and negative examples, it seems plausible that the terms used in those rules should individually be good discriminators.

For the entropy starting point, we select (for each term) the definition which provides most information about the class – ie. a definition which covers most positives and few negatives (or vice versa). We use the entropy measure to evaluate a definition’s information gain:

$$E(D_{ij}) = E_{D_{ij}} - E_0 = - \sum_k p_{ijk} \log_2(p_{ijk}) - E_0$$

where p_{ijk} is the probability that an example with property T_i (according definition D_{ij}) will be in class C_k , and E_0 is the initial entropy (a similar formula to E_D , but p_{ijk} simply being the prior probability that an example is in C_k independent of T_i).

6 Empirical Investigation

We now present results of applying this background knowledge and these algorithms to this domain. The purposes of the experiments are three-fold: first, to illustrate the methodology and show that a domain theory can be learned which is both predictive and explainable with respect to the background knowledge; second, to examine the applicability of the suggested algorithms to the problem; and third, as a side issue, to comment on how good our qualitative model is as a source of background knowledge for this task.

Before starting, two important points should be made. First, the domain of economics is a notoriously difficult domain to work with. Economic parameters are affected by a potentially unbounded list of factors (eg. politics, general elections, international conflict), making the data appear noisy to any algorithm which cannot represent these factors. Even running a highly predictive induction algorithm (CN2 [24, 13]), an average prediction accuracy of only 57.2% could be achieved (compared with the default accuracy of 50.4%⁶). The rule language of this algorithm is relatively unconstrained, in that an arbitrary number of different numeric tests can be used to construct the rule set. In these runs, an average total of 461 different numeric tests were included in each rule set: in contrast, we wish to learn rules in a different language with considerably more constraint on testing the raw numeric data, allowing only 10 numeric tests total (one for each definition of the 10 parameters) to define the ten possible qualitative terms used in the rules.

Secondly, we wish to emphasise the importance of clearly distinguishing between the two purposes of this section: namely illustration of the approach, versus evaluation of the quality of the particular background knowledge we are using (our QM & definition form, eqn 4). The section stands on its own as an illustration of the method, while the particular results reflect more the particular background knowledge we have been using.

In all the experiments the data was randomly split into 66% for training and 33% for testing, and results were averaged over five trials. Numbers following ‘±’ are the standard errors.

6.1 Constraining the Rule Space with Background Knowledge

Our domain knowledge imposes two constraints on the domain theory: the learned rules must be consistent with the qualitative model, and the definitions of terms used in those rules must be in the simple form expressed in equation 4. Concerning the first of these, the qualitative model dramatically constrains the rule space: limiting search to rules with no more than four tests

⁶Measured experimentally, using 66% of the data for training and 33% for testing. Algorithm simply predicts the most common class in the training data. Result is not exactly 50% due to the class probability distribution not being perfectly equal.

<i>RSpace</i> to search:	Using defns <i>DSet</i> :	Accuracy (%)		runtime (sec)
		Train	Test	
all rules (2.3×10^6)	random	80.9 \pm 1.1	54.1 \pm 0.8	4865 \pm 381
	normal	94.2 \pm 0.1	53.9 \pm 1.2	5141 \pm 318
	entropy	83.8 \pm 1.1	55.3 \pm 0.8	6055 \pm 781
using QM (1666 rules)	random	58.5 \pm 0.4	54.4 \pm 0.6	427 \pm 73
	normal	60.3 \pm 0.4	54.5 \pm 0.6	510 \pm 81
	entropy	60.4 \pm 0.6	54.8 \pm 0.7	493 \pm 101

Table 4: Comparison of learning with and without the QM as background knowledge.

Name	Initial <i>DSet</i>		Optimised <i>DSet</i>		
	Accuracy (%)		No. of <i>D</i> _{<i>s</i>} changed ^a	Accuracy (%)	
	Train	Test		Train	Test
random	58.5 \pm 0.4	54.4 \pm 0.6	9.0 \pm 0.3	61.0 \pm 0.4	55.0 \pm 0.6
normal	60.3 \pm 0.4	54.5 \pm 0.6	7.3 \pm 0.6	62.3 \pm 0.3	54.7 \pm 0.9
entropy	60.4 \pm 0.6	54.8 \pm 0.7	6.0 \pm 0.4	61.7 \pm 0.5	54.5 \pm 1.1

^aie. of the 10 initial defns in *DSet*, the no. that were different in the optimised *DSet*.

Table 5: Application of the optimisation algorithm to improve term definitions.

in conditions, the size of *RSpace* is 1666 rules⁷ compared with 10^5 rules given no constraints⁸. The hope is that these ‘explainable’ rules will be adequate for constructing a predictive domain theory. In addition, if the qualitative tmodel (and hence rules in the constrained space) is particularly good then we hope that search using the QM may find good rules otherwise missed by heuristically searching the entire rule space.

To investigate this, we compared the accuracies of domain theories learned with and without using the background knowledge (the QM). To make this comparison, we assumed definitions of the terms according to the three methods described earlier (Section 5.4). The results are shown in Table 4, applying the beam search algorithm of Figure 3 with a `depthlimit` of 4 and a `maxwidth` of 25.

In fact, in terms of classificational accuracy there was no significant difference found between induction constrained by the QM and induction without. In some ways it is surprising that the unconstrained (and substantially more time-consuming) search did not out-perform the constrained search – there may be predictive rules in this space which the QM ruled out. Conversely, though, we could also have expected that the constrained search would not only be faster but also more accurate, given that it is hopefully ‘biasing’ the search towards better rules. The lack of any significant difference suggests that the model’s main contributions are in providing ‘explainable’ results (ie. compatible with background knowledge), and in helping avoid overfitting of the rules to the data as reflected in the accuracies on the training data. We

⁷This number is a function of the connectivity of the qualitative model, and was computed by exhaustively applying the algorithm in Appendix A to our QM (Figure 2).

⁸Rule condition: conjunct of between 1 and 4 terms. Rule conclusion: one term. Each term is a test on one of 10 possible bi-valued parameters. Thus $\text{size}(RSpace) = (11!/(7! 4!) - 1) \times 10 \times 2^5 \approx 10^5$.

discuss this result further in Section 6.3.

6.2 Exploring the Definition Space

In the first experiments we assumed fixed definitions for the 10 terms in the rule language, ie. assumed a *DSet*. We secondly investigated applying the optimisation algorithm (Figure 4) to try to find better definitions of terms, while keeping the induced rule set fixed. As shown in Table 5, the optimisation algorithm improved the performance of the domain theory on the training data but did not significantly improve its performance on the test data. Thus, in this case, the optimisation algorithm appears only to be contributing to an overfitting of the domain theory to the training data.

This result is also surprising: we expected that the optimisation algorithm would locate better definitions of terms within the *DSpace_i* for the domain theory, improving its overall prediction accuracy. A related unexpected result in Table 4 was that, given randomly selected definitions of terms, the induction system could still learn a domain theory which was not significantly worse than using the other fixed definitions. This is surprising because, at the very least, we could naturally expect definitions with low entropy – ie. which are good separators of positive and negative examples, to outperform randomly selected definitions.

Two factors which may contribute to these findings are as follows. First, it may be that the definition spaces *DSpace_i* defined by equation 4 simply does not include highly discriminating attribute tests, and thus all definitions will perform fairly equally. In fact, subsequent examination of the entropy of the most discriminating definitions within these spaces suggests that this may be the case: The average entropy, even of the best definitions, corresponds to selecting only a 60:40 mix of positive:negative examples⁹. Part of this weak discriminatory power may be due to our domain theory constraints: desiring coherence, we required the *same* definition of terms to be used in rules predicting all the ten different economic parameters. A second factor may be that the space of definitions we selected (equation 4) should be expanded.

6.3 Discussion

From the methodology’s point of view, the most important point is that we have illustrated is that it can be applied to learn a domain theory which is not only predictive but also structured and explainable with respect to the available background knowledge. All the rules ‘make sense’, ie. are explainable in the same style as the example in Section 4.2.1, while non-sensical rules have been naturally excluded as a consequence of our approach. This explainability aspect is particularly significant if the learned knowledge is to be incorporated within a body of existing knowledge, as is becoming increasingly the case in machine learning research. It also offers significant potential for assisting in the labour-intensive task of post-learning rule engineering, an essential part of commercial application of machine learning, in which non-sensical rules have to be identified, removed or edited, and the training data modified. The effort normally involved in this task is reported to be typically of the order of months per application [25], so any assistance which can be provided is potentially valuable. In addition, as illustrated in Section 6.1, use of domain knowledge can substantially reduce the size of the search space involved allowing more focussed search to be conducted. This focussing potentially allows a learning system to identify a better domain theory within given time constraints, as poorly predictive parts of the space can be excluded.

⁹Best definitions had an entropy ≈ 0.97 , corresponding, for example, to a split [30% covered +ves, 20% covered -ves, 20% uncovered +ves, 30% uncovered -ves]

Our particular results in this economics domain were also surprising, in that the background knowledge had little impact on predictive accuracy, its main advantage instead being explainability. This suggests that the information content of our particular qualitative model, for prediction purposes, was more limited than we originally expected. The fact that we can identify this is itself valuable, and suggests the obvious and exciting extension to allow feedback from the results of learning to improve the background knowledge itself: for example by identifying which qualitative relations in our economic QM are most reliable, and which parts of the definition space provide contain best discriminators. Our evaluation above suggests several ways in which this could be done, for example by computing information content of definitions in the definition space or by finding which rules in the entire rule space are most predictive and incorporating those into the qualitative model.

7 Conclusion

We have presented a simple methodology for allowing background knowledge to guide learning of domain theories, based on assuming a structure for the target theory and formulating background knowledge as constraints on components of that structure. This enables a focussing of search during learning, and also produces a domain theory which is explainable with respect to the background knowledge. Both these properties are essential for learning systems which are to work in knowledge-intensive environments and where the learned knowledge is to be integrated back within an existing system of knowledge.

We have also formalised and an instance of this approach, in which background knowledge expressed in abstract, ambiguous, qualitative terms can guide learning, and presented algorithms for addressing the learning task. We have illustrated its application to the task of economic prediction. Our approach allowed us to bring prior qualitative economic knowledge to bear on the task, constraining the search and producing domain theories which were sensible with respect to this knowledge. Surprisingly, this also revealed limitations in the original economic background knowledge itself which we were previously unaware of. We consider this a beneficial finding enabled by our approach, and suggests an exciting development of the approach whereby the results of learning can feed back into the background knowledge itself.

Acknowledgements and Availability

We are particularly grateful to Rob Holte for many insightful discussions on the work presented here, and to all the members of the Ottawa Machine Learning Group for the stimulating research environment they provide. The work described here has been performed at the Knowledge Acquisition laboratory at the University of Ottawa. Activities of the laboratory are supported by the Natural Sciences and Engineering Research Council of Canada, the Canada Centre for Remote Sensing, and the Pacific Forestry Centre. Copies of the learning algorithms (implemented in Quintus Prolog) and the economic data set are available from the authors on request.

Appendix A: Extraction of Rules from the Qualitative Model

A rule is drawn from the qualitative model by extracting a tree from the model's network representation. The root node of the tree is the conclusion of the rule, all other nodes conjunctively form the rule's condition. Each element in the condition is a test whether a parameter has a qualitative value `high` or `low`. The conclusion is a prediction of a parameter's future direction

of change (**increase** or **decrease**). The rule is plausible in the sense that it is explainable by reference to the model. The precise rule extraction algorithm is given below:

To find a rule predicting whether some economic parameter P_C will change in the next year in a chosen direction $v_{PC} \in \{\mathbf{increase}, \mathbf{decrease}\}$:

1. Mark all nodes in the QM as unvisited.
2. Start at node n_{PC} , representing parameter P_C , and follow an arc backwards to some unvisited node n_{Pi} representing parameter P_i . If $v_{PC} = \mathbf{increase}$ and the arc was labelled **Q+** then let $v_{Pi} = \mathbf{high}$ (conversely, let $v_{Pi} = \mathbf{low}$ if $v_{PC} = \mathbf{decrease}$ or the arc was labelled **Q-**). Mark n_{Pi} as visited. Form the rule **R**:

$$P_i = v_{Pi} \quad \longrightarrow \quad P_C = v_{PC}$$

3. Repeat zero or more times:
Follow an arc backwards from an already visited node n_v to an unvisited node n_u representing parameter P_{nu} . If that arc was labelled **Q+**, then let $v_{nu} = v_{nv}$, if **Q-** then v_{nu} = the inverse of v_{nv} (ie. if $v_{nv} = \mathbf{high}$ then $v_{nu} = \mathbf{low}$, and vice versa). Add the term $P_{nu} = v_{nu}$ conjunctively to the condition of the rule **R**, and label n_u as visited.

The consequent of the rule $P_C = v_{PC}$ is interpreted as a prediction about how the numeric value of P_C will change in the next time step.

Appendix B: Evaluation of the Quality of a Domain Theory

We evaluate the quality Q of a domain theory by estimating its future predictive accuracy on test examples, applying the Laplace estimate to rules' coverages on the training data:

$$Q(DT) = \sum_i w_i \left(\frac{c_{pi} + 1}{c_{pi} + n_{pi} + 2} \right) \quad \text{where} \quad w_i = \frac{c_{pi} + n_{pi}}{\sum_i (c_{pi} + n_{pi})}$$

where c_{pi} and c_{ni} are the number of positive and negative examples covered by rule i respectively, and the summation is over the entire rule set $RSet$ plus a 'default' rule assigning the most common class to all remaining uncovered examples.

References

- [1] Derek Sleeman and Peter Edwards, editors. *Proc. Ninth Int. Machine Learning Conference*, Ca, 1992. Kaufmann.
- [2] Francesco Bergadano and Attilio Giordana. A knowledge-intensive approach to concept induction. In John Laird, editor, *ML-88 (Proc. Fifth Int. Machine Learning Conference)*, pages 305–317, Ca, 1988. Kaufmann.
- [3] Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning Journal*, 1992. (To appear).
- [4] Bradley L. Richards and Raymond J. Mooney. First-order theory revision. In *ML-91 (Proc. Eighth Int. Machine Learning Workshop)*, pages 447–451, Ca, 1991. Kaufmann.
- [5] Raymond J. Mooney and Dick Ourston. Constructive induction in theory refinement. In *ML-91 (Proc. Eighth Int. Machine Learning Workshop)*, pages 178–182, Ca, 1991. Kaufmann.

- [6] S. Muggleton and C. Feng. Efficient induction of logic programs. In *First International Conference on Algorithmic Learning Theory*, pages 369–381, Tokyo, Japan, 1990. Japanese Society for Artificial Intelligence.
- [7] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, Aug 1990.
- [8] James C. Lester and Bruce W. Porter. Generating context-sensitive explanations in interactive knowledge-based systems. Tech. Report AI-91-160, Univ. Austin at Texas, TX, 1991.
- [9] Anne v.d.L. Gardner. The design of a legal analysis program. In *AAAI-83*, pages 114–118, 1983.
- [10] Bruce W. Porter, Ray Bareiss, and Robert C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.
- [11] Ranan B. Banerji. Learning theoretical terms. In Stephen Muggleton, editor, *Inductive Logic Programming*. 1992.
- [12] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. Inductive knowledge acquisition: a case study. In *Applications of Expert Systems*, pages 157–173. Addison-Wesley, Wokingham, UK, 1987.
- [13] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In Yves Kodratoff, editor, *Machine Learning – EWSL-91*, pages 151–163, Berlin, 1991. Springer-Verlag.
- [14] Christopher J. Matheus. *Feature Construction: An Analytic Framework and An Application to Decision Trees*. PhD thesis, University of Illinois at Urbana-Champaign, 1990.
- [15] Jerzy W. Bala, Ryszard S. Michalski, and Janusz Wnek. The principle axes method for constructive induction. In Derek Sleeman and Peter Edwards, editors, *Proc. Ninth Int. Machine Learning Conference (ML-92)*, pages 20–29, CA, 1992. Kaufmann. (and personal communication).
- [16] L. de Raedt and M. Bruynooghe. Constructive induction by analogy: A method to learn how to learn? In *Proc. 4th European Machine Learning Conference (EWSL-89)*, pages 189–200, London, 1989. Pitman.
- [17] David Aha. Relating relational learning algorithms. In Stephen Muggleton, editor, *Inductive Logic Programming*. 1992.
- [18] Susan Craw and Derek Sleeman. The flexibility of speculative refinement. In *ML91 (Proc. Eighth Int. Machine Learning Workshop)*, pages 28–32, Ca, 1991. Kaufmann.
- [19] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning Journal*, 1(1):47–80, 1986.
- [20] Igor Mozetic. The role of abstractions in learning qualitative models. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, CA, 1987. Kaufmann.
- [21] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.
- [22] James C. Spohrer and Christopher K. Riesbeck. Reasoning-driven memory modification in the economics domain. Technical Report YALEU/DCS/RR-308, Yale University, May 1984.
- [23] Gerald DeJong. Explanation-based learning with plausible inferencing. In *Proc. 4th European Machine Learning Conference (EWSL-89)*, pages 1–10, London, 1989. Pitman.
- [24] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning Journal*, 3(4):261–283, 1989.
- [25] Jill Houston, 1992. (Senior consultant, Intelligent Terminals Ltd., Personal communication).