Using Qualitative Models to Guide Inductive Learning

Clark, P.; Matwin, S.

In: Proc. 10th International Machine Learning Conference (ML93),
pp49-56, (1993) Ed: Paul Utgoff, CA:Kaufmann.

Using Qualitative Models to Guide Inductive Learning

Peter Clark
Knowledge Systems Laboratory
National Research Council
M-50 Montreal Road
Ottawa, Canada
pete@ai.iit.nrc.ca

Stan Matwin
Ottawa Machine Learning Group
Computer Science
University of Ottawa
Ottawa, Canada
stan@csi.uottawa.ca

Abstract

This paper presents a method for using qualitative models to guide inductive learning. Our objectives are to induce rules which are not only accurate but also explainable with respect to the qualitative model, and to reduce learning time by exploiting domain knowledge in the learning process. Such explainability is essential both for practical application of inductive technology, and for integrating the results of learning back into an existing knowledge-base. We apply this method to two process control problems, a water tank network and an ore grinding process used in the mining industry. Surprisingly, in addition to achieving explainability the classificational accuracy of the induced rules is also increased. We show how the value of the qualitative models can be quantified in terms of their equivalence to additional training examples, and finally discuss possible extensions.

1 INTRODUCTION

1.1 OVERVIEW

This paper presents and evaluates a technique for using qualitative models to guide inductive learning from examples. Our objective is to induce rules which are not only accurate but also explainable using this qualitative background knowledge, a requirement both for practical application of machine learning and for integrating the results of learning back into a wider body of existing knowledge. The research can be viewed as developing and evaluating a special case of the general theory-guided learning paradigm (e.g. [Bergadano and Giordana, 1988, Pazzani and Kibler, 1992]), in which the theory is a qualitative model and the learning technique is rule induction from data. Our method is based on defining a notion of consistency of a rule with a qualitative model, and then restricting the specialisation operator in an induction system (CN2) to only investigate specialisations consistent with the QM during search. We describe the application of this method to two learning problems in process control. Our evaluation shows that this method, in addition to achieving consistency of learned knowledge with background knowledge, can also improve overall classificational accuracy. We show how a metric can be defined which quantifies the value of the qualitative model in terms of its equivalence to extra training examples, and finally speculate how empirically learned knowledge might feed back to modify the qualitative model itself.

1.2 MOTIVATION

It is now well recognised that applying standard inductive learning tools such as ID3, C4.5 or CN2 is somewhat of a skill. Their inability to exploit background knowledge leaves the knowledge engineer with substantial work to perform in order to generate rules which both perform well and which are sufficiently 'sensible' that they can enhance the knowledge of domain experts, and be relied upon for real-world performance tasks. Gillian Mowforth, a former employee of Intelligent Terminals Ltd. and with substantial experience of commercially applying rule induction, estimates that in typical commercial applications of ExTran (an ID3 derivative) around 30% of the final decision tree installed for the customer would have been hand-engineered rather than induced [Mowforth, 1992]. She reports typical applications would involve data collection, rule induction, and then analysis of the induced tree in collaboration with the experts to see if it "made sense". This latter process was time consuming, and would be followed by modifying the induction procedure e.g. by removing/adding training examples, by modifying the example description language, or by re-running the induction system in interactive mode to force certain attribute tests to be included/excluded in parts of the tree. Then a new tree would be induced and the process iterated until the tree was acceptable to the experts, the whole application taking several months to complete. Similar experiences have been reported by others involved in machine learning applications.

The complete process is thus interactive, involving

substantial domain expertise in addition to use of an inductive tool. In this process, statistically justified rules are being compared against domain knowledge, and the results used to refine learning. Domain knowledge can be viewed as a compiled version of many training examples (i.e. all previous empirical evidence), above and beyond the data set immediately available. Ideally, this knowledge will prune out rules which by chance perform well on the training data, but in general have poor performance.

In this paper we model this process using a qualitative model to represent background knowledge and restrict the choices available to an inductive engine. At the end of the paper we also speculate on extending our method to perform the reverse process, which Mowforth also reports was common: namely where strong statistical evidence may cause experts to revise their domain knowledge.

2 CONTEXT & RELATED WORK

While it is widely accepted that background knowledge is necessary for all but the simplest learning tasks, we note that there are two principle ways in which background knowledge can be used:

1. To expand the hypothesis language by introducing extra terms (e.g. in Foil [Quinlan, 1990] and Golem [Muggleton and Feng, 1990]).
2. To constrain search (our objective in this paper).

These two methods have significantly different outcomes: in the first case background knowledge actually aggravates the search problem as the search space is expanded, whereas in the second the hypothesis space is restricted, reducing search. We highlight this to clearly distinguish this work from other systems which use background knowledge in the former sense.

The general paradigm of using domain knowledge to guide learning has been advocated by numerous authors (e.g. [Bergadano and Giordana, 1988, Pazzani and Kibler, 1992, Clark and Matwin, 1993, Flann and Dietterich, 1990]). Our work here can be viewed as developing and evaluating a special case of this theory-guided learning paradigm, in which the theory is a qualitative model (QM) and the learning is rule induction from data. Within the general paradigm, abstract background knowledge specifies constraints on which hypotheses should be explored during inductive search. We apply this to a qualitative model by defining a notion of consistency of a rule with the model, and then constraining search to examine only consistent rules. The qualitative model can thus be viewed as indirectly specifying a domain-specific grammar for induced knowledge [Cohen, 1992, DeJong, 1989], or as encoding a set of 'rule models' for the inductive component to search [Kietz and Wrobel, 1992].

We finally note that this work of course differs from machine learning research in compiling qualitative models into rules (in this paradigm, there is no independent training data and the QM does not directly constrain induction e.g. [Bratko et al., 1989]), and in learning QMs themselves from examples (e.g. [Bratko et al., 1991, Mozetic, 1987]).

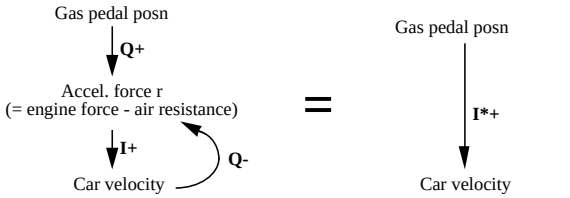

Figure 1: The $I*$ relation, a syntactic shorthand for a self-stabilising feedback loop.

3 LEARNING METHOD

3.1 KNOWLEDGE REPRESENTATION

Our learning method takes as input a set of training examples and a qualitative model, and as output produces classification rules explainable by that model.

A QM comprises nodes, representing parameters of the application domain, and arcs, representing their relationships (arrows indicating temporal precedence). As in Qualitative Process theory (QPT) [Forbus, 1984], we label arcs as either $I+$, $I-$, $Q+$ or $Q-$. The link $X \xrightarrow{Q+} Y$ denotes that Y varies monotonically with X (e.g. if X increases then so does Y), while the link $X \xrightarrow{I+} Y$ denotes that Y's rate of change dY/dt varies monotonically with X. Similarly, the $\xrightarrow{Q-}$ and $\xrightarrow{I-}$ links denote inverse monotonic relationships.

As a syntactic shorthand, we introduce a third label I^*, shown in Figure 1, to denote a self-stabilising feedback loop. $X \xrightarrow{I*+} Y$ also denotes that if X is increased then initially Y will rise; however, as Y subsequently increases, it's rate of increase dY/dt will eventually fall until Y reaches a new constant value. For example, the gas pedal position P in a car is related to the car's speed S by $P \xrightarrow{I*+} S$. Initially depressing the pedal causes dS/dt to rise; however, the car will not increase speed indefinitely but eventually reach a new, higher constant speed. Thus at short time-scales the I^* relationship behaves as $P \xrightarrow{I+} S$, while for long time-scales it behaves as $P \xrightarrow{Q+} S$ (every gas pedal position eventually produces a corresponding speed for the car).

While similar to QPT models, it should be noted that our QMs differ in that they are *incompletely specified*. We have not stated (i) the distinguished or 'landmark' values for each parameter, nor (ii) how to resolve conflicting influences during simulation. As a result, our models *on their own* cannot be used for simulation or prediction. Instead, their role is to constrain induction of quantitative rules from examples, and to provide explanations of those rules. The QM concisely represents the space of relationships which are considered credible in the domain by the model's constructor.

3.2 USING THE QUALITATIVE MODEL TO CONSTRAIN INDUCTION

The application of the QM to rule induction is simple; rather than the inductive tool searching the space of all possible rules, it searches only those which are consistent with the QM. The inductive tool is thus constrained to search only a subset of its original search space. To effect this, we first define when a rule is 'consistent with the QM'. Second, we modify the search operator in the inductive tool to only search rules which satisfy this definition.

Our implementation is as follows. First, we define a rule extraction algorithm which exhaustively enumerates (schemata for) all rules (up to some maximum length) consistent with the QM. This enumeration is stored in a lookup table. Second, an induction tool is used to induce classification rules using a set of training examples, while prevented from searching rules not represented in this lookup table. To do this, we modify the learner so that each time it generates a new hypothesis rule to test, it additionally checks that it is in this table. If it is not, the hypothesis is discarded without further work.

The inductive tool we used was CN2, which induces (in unordered mode) a set of "if...then..." rules given a set of training examples. CN2 executes a greedy set covering algorithm, performing a general-to-specific beam search for a rule at each step [Clark and Niblett, 1989, Clark and Boswell, 1991]. It was modified so that as it specialises hypothesis rules in its beam, it additionally performs this check on specialisations generated. A similar approach can be envisaged for ID3; rather than evaluating all possible attribute tests when expanding a node in the tree, evaluate only those such that the resulting decision tree branch was contained in the table of consistent rules.

3.3 EXTRACTING RULES FROM THE QM

Before defining a decision procedure to identify which rules are consistent with a QM, we first note that this notion of 'consistency' is not as easy to formalise as might be expected. Informally, the decision procedure should identify all and only those rules which an expert will consider 'sensible', given the QM. This requires an interpretation of what should be considered acceptable evidence for a prediction, given the QM. Below we describe our definition of which rules are 'consistent' with a QM, while noting that alternatives might also be acceptable.

We define a **rule** as a structure:

if T_1 **and** ... **and** T_n **then** C

where each T_i is a test on some observable parameter P_i (testing either $P_i > k$ or $P_i < k$, where k is some constant), and the conclusion C asserts either "P_{conc} will increase" or "P_{conc} will decrease" for some observable parameter P_{conc}. The interpretation of the rule is that if the conditions hold at some time T, then P_{conc} will have increased/decreased by time $T + \Delta T$ (where ΔT is a constant, representing how far ahead the user wishes to predict). A **rule schema** is a rule with the constants k_i replaced by universally quantified variables, representing a set of rules.

We wish to know which conjuncts of tests T_i 'sensibly' predict a change in C, given the QM. For example, given the two-node QM for a car: "gas $\xrightarrow{\texttt{I*+}}$ speed", we consider rules or the form

"**if** gas $> k_1$ **then** speed will increase"

consistent with this QM, while rules of the form

"**if** gas $< k_1$ **then** speed will increase"

would not be (where k_1 is some constant).

In general, considering the three different qualitative relations I, I* and Q in isolation, the corresponding structures of consistent rules are:

Reln	Corresponding rule schema
A $\xrightarrow{\texttt{I+}}$ B	**if** A $> k_A$ **then** B will increase.
A $\xrightarrow{\texttt{I*+}}$ B	**if** A $> k_A$ **and** B $< k_B$ **then** B will increase.
A $\xrightarrow{\texttt{Q+}}$ B	(no rule).

The rule schema for I* above expresses a condition of disequilibrium, resulting (by definition, Section 3.1) in a rise in B to re-establish equilibrium. For the Q relation, knowing the value of A alone does not tell us how B might change in future.

We now generalise these schemata to apply to QMs which contain more than just two nodes and one arc. To find a plausible explanation of why our target P_{conc} will change, we simply find a path in the QM from some node (which we call the **source** of the change) to P_{conc} which traverses at least one I or I* arc. One of these I/I* arcs is then nominated as responsible for P_{conc}'s future change; nodes upstream of this arc are considered **causes** of this future change, in that they are either the source or correlated with the source. These nodes together correspond to the A node in the three schemata mentioned earlier. Nodes downstream of the I/I* arc are called the **effects**, and together correspond to the B node in the earlier mentioned schemata. Rules which are consistent with this path are thus those which:

1. test that some subset of observable parameters upstream of the nominated I/I* arc are greater than some constant,
2. (for I* only) test that some subset of observable parameters downstream of the arc are less than some constant,
3. conclude that P_{conc} will increase.

Thus there would be 9 rules[1] consistent with the following path from a QM of a car:

fp (foot position) $\xrightarrow{\texttt{Q+}} gas \xrightarrow{\texttt{I*+}} revs \xrightarrow{\texttt{Q+}} speed$

[1] i.e. rule schemata; we will just use the word 'rule' from now on to simplify the presentation.

namely: {
if $fp > k_{fp}$ and $revs < k_r$ then speed $\uparrow$,
if $gas > k_g$ and $revs < k_r$ then speed $\uparrow$,
if $fp > k_{fp}$ and $gas > k_g$ and $revs < k_r$ then speed $\uparrow$,
: : : : : : : : : : : : : : :
if $fp > k_{fp}$ and $speed < k_s$ then speed $\uparrow$,
: : : : : : : : : : : : : : :
if $fp > k_{fp}$ and $gas > k_g$ and $revs < k_r$ and $speed < k_s$ then speed $\uparrow$ }.

This example uses just positive arcs (e.g. `Q+`). Negative arcs are handled in the obvious way, namely by inverting the greater-than and less-than tests as each negative arc is traversed. Rules predicting a decrease in the target parameter are generated by inverting the greater-than and less-than tests in rules predicting an increase.

This method of extracting consistent rules is still incomplete:

1. It ignores parameters not in the path used, but nevertheless correlated with parameters on the path (e.g. via `Q` relations). These off-path parameters might provide useful evidence of values of on-path parameters, particularly if none of the on-path parameters are directly observable.
2. It assumes just one source.

To overcome the first point, the full rule extraction algorithm also allows off-path parameters, affected by on-path parameters via a chain of `Q` or `I*` relations, to be included in a rule's condition. The net result is to extract a tree from the QM, whose root node is the source and with P_{conc} as one of its leafs. To allow multiple sources, we combine rules together (by conjoining their conditions) to produce new rules, checking that we do not treat any parameter in the new rule as both a cause and an effect simultaneously.

4 EXPERIMENTAL EVALUATION

4.1 APPLICATION DOMAINS

We evaluated our method by applying it to two process control problems: a water tank system containing feedback, and a real-world process of ore grinding, in which rock is crushed into small particles for mineral extraction. For each of these systems, there is a particular **parameter of interest** whose movement we wish to predict (the water level in the lowest tank, and the efficiency of the grinding process respectively). To generate training data, numeric simulators of the real physical processes were constructed.

4.2 THE WATER TANK SYSTEM

The water tank system is shown in Figure 2. Water enters the circuit through the upper pipe, and fills the first tank (with level `L1`). The flow of water out of a tank is proportional to the tank's water level (the higher the level, the greater the pressure at the base of the tank and the faster the out-flow; there is no reflux). In addition, there is feedback from some tanks to

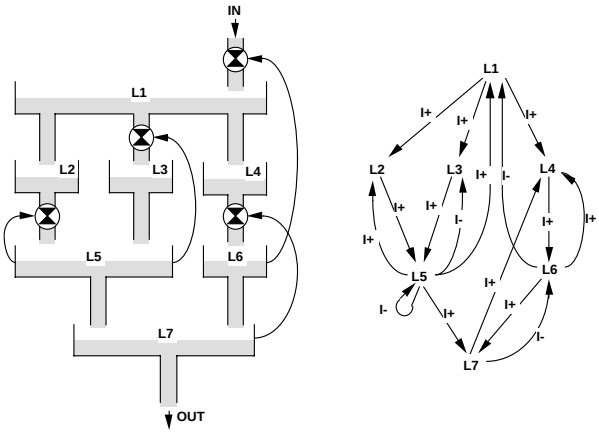

Figure 2: The water tank system (left), and its QM (right). (See text for description).

control valves earlier in the circuit as illustrated. As a valve-controlling tank becomes full, the controlled valve closes thus reducing water flow earlier in the pipe network. There are seven observable parameters, namely the levels in each of the seven tanks. The learning task is to predict whether the level `L7` in the last tank will have increased or decreased by some time $T + \Delta T$ in the future given observations at time T. There is one operator-controllable parameter to this system, namely the flow rate of water into the system. A simple numeric simulator was used to model the behaviour of the system with time. The qualitative model of the system which we constructed, intended to reasonably describe the (simulated) physical system, is also shown in Figure 2.

4.3 THE ORE GRINDING CIRCUIT

The grinding circuit (Figure 3) is substantially more complex, and is a simplified version of a similar circuit used in the mining industry [JKTech Ltd., 1991]. Ore enters along a feed conveyor belt, and accumulates in ball mill one. A ball mill is a large, rotating drum which breaks up the rock into smaller components. A fraction of the contents of the ball mill leaves during each time step, and arrives at the screen. The screen is a metal mesh with holes in, allowing smaller rock to pass through while larger rock is fed back into ball mill one. Ore which passes through the screen reaches and accumulates in a large centrifuge called a cyclone. The smaller contents of the centrifuge are filtered out and leave the system. Larger ore in the centrifuge is also removed (the underflow) and enters a second ball mill, where it is further crushed and then returns to the cyclone. Water can be added to both ball mills, increasing the out-flow from the mills but also reducing the mills' efficiency as energy is then spent 'grinding' water. There are four operator-controllable parameters, namely: the feed rate and size distribution of ore into the system, and the rate of water addition into the two ball mills.

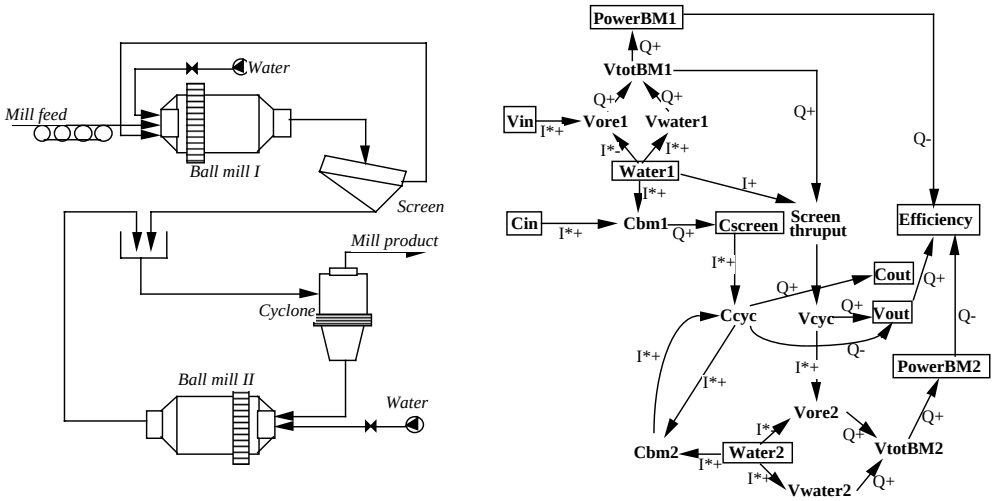

Figure 3: The ore grinding circuit (left), and its QM (right). (See text for description).

The grinding circuit simulator was a simplified version of a more complex, commercial simulator used in the mining industry [JKTech Ltd., 1991]. The QM we constructed of this process is shown in Figure 3. As in the water circuit, the QM is to a large extent our 'guess' at a reasonable qualitative description of the (simulated) physical circuit. The ten observable parameters of the physical system, also contained in the QM (shown in boxes), are: the coarseness and feed rate of ore into the system ($\mathtt{C_{in}}$ and $\mathtt{V_{in}}$), the rate of water addition to each ball mill ($\mathtt{Water_1}$ and $\mathtt{Water_2}$), the power drawn by each ball mill ($\mathtt{Power_{BM1}}$ and $\mathtt{Power_{BM2}}$), the coarseness of ore at the screen ($\mathtt{C_{screen}}$), the coarseness and output rate of ore leaving the system ($\mathtt{C_{out}}$ and $\mathtt{V_{out}}$) and the overall volume/power efficiency of the circuit (`Efficiency`). The learning task is to predict if the overall efficiency will increase or decrease by time $T + \Delta T$ given values of the observables at time T.

4.4 GENERATING DATA SETS

For both applications, data sets were generated using the numeric simulators (NB. not the QMs). Each example in a data set is a snapshot of the system's state at some time T, described by values of the observable parameters, plus an extra qualitative value ('`increase`' or '`decrease`') stating whether the parameter of interest was observed to have increased or decreased by time $T + \Delta T$. ΔT was taken as 10 and 100 time steps for the water and ore systems respectively, corresponding to approximately 1 second and 1 minute real-time.

To generate an example of each process in a random but still physically plausible state, the simulator was run for 500 time steps with the controllable parameter(s) being randomly perturbed at intervals. After 500 time steps, the perturbations were stopped, values of observable parameters recorded, and the simulation continued for another ΔT steps to see if the parameter of interest increased or decreased. These observations formed one example. This process was repeated approximately 500 times for each application to generate two data sets.

4.5 RULE INDUCTION

For each of the two applications, the data set was split randomly into a training and testing set of controlled sizes. Rules were induced by CN2 using the training data, and then tested on the testing data. In normal (no qualitative model) mode, CN2 heuristically searches the rule space for good rules. In constrained (qualitative model) mode, only rules consistent with the qualitative model were explored as described in Section 3.2.

CN2 has two parameters which control the extent of search conducted, namely the beam width and the depth limit of search. CN2 was run with beam widths of 1, 3, 5 and 7, and with depth limits of 2 and 3 (water tanks) or 2, 3 and 4 (ore circuit)[2], and the results averaged. Experiments with five different training set sizes were conducted, the algorithm run in both no-QM and QM modes, and the experiments repeated 30 times (for the water tanks) and 10 times (for the ore circuit). This represents a total of 2400 runs for the water tanks and 1200 runs for the ore circuit. We recorded the CPU time and improvement in classificational accuracy compared with the default accuracy (61.8% water, 61.5% ore) for each run of the algorithm.

4.6 RESULTS

The results (Tables 1 and 2) show averages and their standard errors (denoted by $\pm$). The column 'ex-

[2] corresponding to the maximum lengths of 3 (water) and 4 (ore) we imposed when pre-enumerating rule schemata consistent with the QMs (Section 3.2).

Table 1: Effect of qualitative knowledge on learning (water tank application).

No. training examples	Accuracy increase (%) no QM		Accuracy increase (%) QM		CPU time (sec) no QM		CPU time (sec) QM		'Explainability' no QM	'Explainability' QM
20	13.4	±0.5	13.6	±0.5	5.3	±0.2	2.2	±0.1	39%	100%
40	18.9	±0.4	21.0	±0.3	12.8	±0.3	5.9	±0.1	35%	100%
81	23.9	±0.2	26.3	±0.2	34.3	±0.8	16.0	±0.3	37%	100%
121	25.8	±0.1	27.7	±0.2	61.4	±1.5	29.3	±0.6	39%	100%
162	26.9	±0.2	28.8	±0.1	93.5	±2.1	44.4	±1.0	41%	100%

Table 2: Effect of qualitative knowledge on learning (grinding circuit application).

No. training examples	Accuracy increase (%) no QM		Accuracy increase (%) QM		CPU time (sec) no QM		CPU time (sec) QM		'Explainability' no QM	'Explainability' QM
26	5.7	±0.6	6.0	±0.6	8.2	±0.4	8.9	±0.4	48%	100%
53	12.0	±0.3	12.2	±0.4	22.6	±0.8	26.3	±1.1	53%	100%
107	14.8	±0.3	16.3	±0.2	68.1	±2.2	79.2	±3.1	57%	100%
161	16.0	±0.3	17.0	±0.3	128.6	±4.1	154.3	±6.5	58%	100%
215	17.7	±0.2	18.3	±0.2	205.5	±6.0	240.4	±10.0	54%	100%

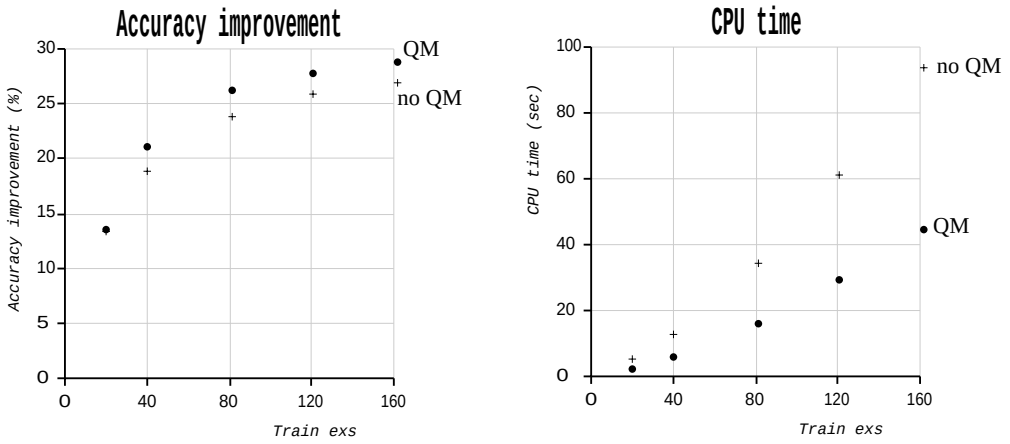

Figure 4: **Water Tank System:** Variation of classificational accuracy and CPU time with number of training examples, comparing learning without and with the QM (plot of data from Table 1).

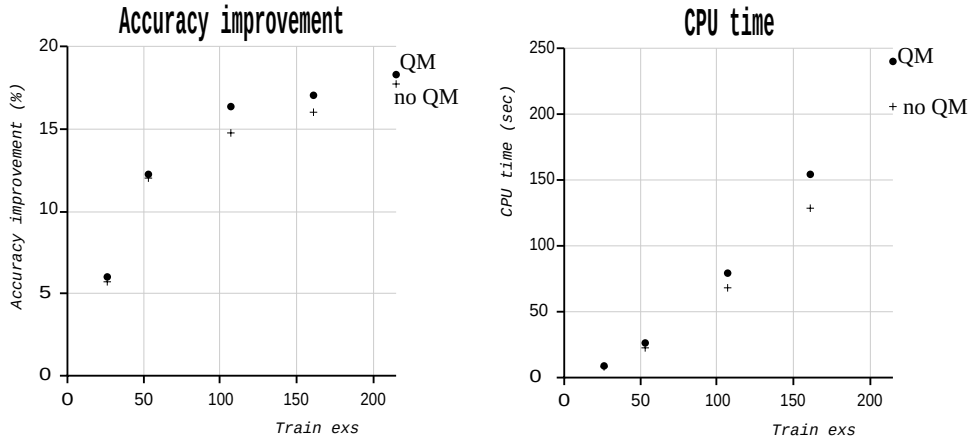

Figure 5: **Ore Grinding Circuit:** Variation of classificational accuracy and CPU time with number of training examples, comparing learning without and with the QM (plot of data from Table 2.

plainability' shows the (average) percentage of induced rules which were consistent with, and hence deemed explainable by, the QM (by definition, this figure will be 100% for QM-constrained learning). The same results are plotted in Figures 4 and 5.

4.7 ANALYSIS AND DISCUSSION

We consider one of the most significant benefits of this method is the explainability of induced rules. All rules induced with the QM 'make sense' (compared with only about 50% without the QM), in that an expert (or indeed the computer itself, using the QM) can construct a causal explanation describing how the state of the system (as revealed by observable parameter values) might cause the parameter of interest to change in the way the rule describes. Such explainability is an essential aspect of learning, as discussed earlier.

Having said this, several other surprising and positive results were observed. First, in both applications classificational accuracy is increased by use of the qualitative models, even though the QM restricts rather than expands the space of rules available to the learner. It thus appears that, without the QMs, the learning algorithm sometimes selects rules which by chance perform well on the training data, but which do not predict well and which are not consistent with the QMs. This result thus illustrates that the QM is injecting extra knowledge into the final rule sets produced, improving performance.

In the water tank application, CPU time is also reduced using the QM, as the induction algorithm has fewer possibilities to explore. In the ore grinding application, however, CPU time is slightly increased by using the QM. This is also surprising, as the QM reduces the size of the total search space. However, two factors may contribute to the increased CPU. First, CN2's specialisation operator has to perform an extra lookup operation to verify a specialisation is in the set of specialisations consistent with the QM. Second, the high connectivity of the ore grinding QM results in a large number of rules being considered consistent, thus only imposing moderate constraint on search. Third, constraining the total search space size does not necessarily constrain the size of the space heuristically searched. CN2's beam search follows the $N_{beamwidth}$ best hypotheses in parallel. So long as there are at least $N_{beamwidth}$ possible options, the CPU time will be unaffected by constraining the space. In addition, the QM may guide the algorithm into richer portions of the space where many possibilities merit exploration, whereas without the QM several 'dead ends' may be explored where search is abandoned earlier.

4.8 QUANTIFYING THE QM'S VALUE

From the plots in Figures 4 and 5, we can define a useful metric of the qualitative models' value, based on its equivalence to an increased number of training examples. This allows us to compare the value of different qualitative models against a common scale.

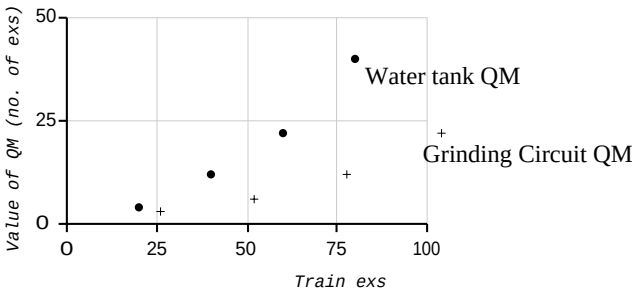

Figure 6: Equivalence of the two QMs to extra training examples.

If a classificational accuracy A can be achieved using N training examples with the QM, or with $N + \Delta N$ examples without the QM, then we define ΔN as the **example-equivalence** of the QM (for classificational accuracy, for N training examples). We similarly define the **CPU example-equivalence** of the QM as the number of extra examples which the QM-constrained algorithm can process in the same time.

Using the plots in Figures 4 and 5, we can plot the example-equivalence of the two QMs as shown in Figure 6. Approximating these curves as straight lines, we can take their slopes as an overall value metric for each QM:

	Approximate example-equivalence ($0 \leq N_{train\_exs} \leq \sim 200$)
Water QM	+0.5 examples per training example
Ore QM	+0.2 examples per training example

(Similarly, the CPU example-equivalences are +0.6 and −0.1 examples per training example respectively). While this provides an appealing metric for a QM, it should be interpreted carefully:

1. As is well known, training set size and accuracy improvement are not linearly related. Thus the monotonically increasing example-equivalence with training size does not imply a monotonically increasing accuracy improvement. In fact, accuracy improvement rises, reaches a maximum and then falls again as training set size increases (Tables 1 and 2). This is because the value (in terms of improved accuracy) of an extra example becomes less and less as the training set size grows.
2. The QM on its own (i.e. with zero training examples) does not contribute to an accuracy improvement. This is because our method does not use the QM directly for prediction, but solely as a filter for inductive hypotheses. (In fact it cannot, as our QMs do not specify parameters' distinguished or 'landmark' values. It is precisely the inductive learner's task to identify these values).
3. It is not clear how far the curves in Figure 6 can be meaningfully extrapolated. For larger training

set sizes, the no-QM curves in Figures 4 and 5 may eventually touch the QM curves (resulting in an example-equivalence of zero for the QM), or even cross them (resulting in a negative-example equivalence). Thus we qualify our 'examples per training example' values as only being valid within a certain range of training set sizes.

5 DISCUSSION AND CONCLUSION

We have presented and evaluated a technique for using qualitative models to guide inductive learning. The learning algorithm produces rules which not only have improved performance but which are explainable by this background knowledge, reflecting the normally manual knowledge engineering which accompanies application of machine learning algorithms. This is significant as qualitative knowledge is a ubiquitous component of common-sense knowledge; being able to harness it to positive effect offers a means for both improved learning performance and for better integration of learning and reasoning systems in the future.

We have also defined the notion of an example-equivalence metric for qualitative models, by which a model's value for learning can be quantified and hence different models compared. In both the applications investigated, the models had a positive example-equivalence, i.e. produced an overall improvement in learning behaviour.

Our method assumes the existence of a reasonable qualitative model of the domain under investigation, and thus imposes a cost as well as saving in knowledge engineering. It can perhaps best be viewed as providing a solid framework for incorporating domain knowledge in induction, which otherwise has to be incorporated by rather ad hoc means (Section 1.2). In addition to explainability, it offers a practical way in which domain knowledge can reduce required search, helping to solve the ubiquitous tractability problems faced by knowledge-poor learning systems in non-trivial domains.

We note also that the benefits of our method depend on the quality of the QM used; a QM poorly approximating the physical system may harm rather than improve accuracy (i.e. have a negative example-equivalence). This fact, combined with the ability to quantify the QM's value, suggests the exciting possibility of including the reverse process reported by knowledge engineers, in which the QM itself could be revised based on strong correlations in the data. The example-equivalence suggests one way in which this could be done, based on heuristically searching the space of perturbations to the original qualitative model using example-equivalence as a search heuristic.

Acknowledgements: We are greatly indebted to Rob Holte, Peter Turney, Donald Michie and Claude Sammut for valuable contributions and comments.

References

Bergadano, F. and Giordana, A. (1988). A knowledge-intensive approach to concept induction. In Laird, J., editor, *ML-88 (Proc. Fifth Int. Machine Learning Conference)*, pages 305–317, Ca. Kaufmann.

Bratko, I., Mozetic, I., and Lavrač, N. (1989). *Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press, Cambridge, Ma.

Bratko, I., Muggleton, S., and Varsek, A. (1991). Learning qualitative models of dynamic systems. In *ML91 (Proc. Eigth Int. Machine Learning Workshop)*, pages 385–388, Ca. Kaufmann.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In Kodratoff, Y., editor, *Machine Learning – EWSL-91*, pages 151–163, Berlin. Springer-Verlag.

Clark, P. and Matwin, S. (1993). Learning domain theories using abstract background knowledge. In Brazdil, P., editor, *Proceedings of the Sixth European Conference in Machine Learning (ECML-93)*. Springer-Verlag. (in press).

Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning Journal*, 3(4):261–283.

Cohen, W. W. (1992). Compiling prior knowledge into an explicit bias. In Sleeman, D. and Edwards, P., editors, *Proc. Ninth Int. Machine Learning Conference (ML-92)*, pages 102–110, CA. Kaufmann.

DeJong, G. (1989). Explanation-based learning with plausible inferencing. In *Proc. 4th European Machine Learning Conference (EWSL-89)*, pages 1–10, London. Pitman.

Flann, N. S. and Dietterich, T. G. (1990). A study of explanation-based methods for inductive learning. In Shavlik, J. W. and Dietterich, T. G., editors, *Readings in Machine Learning*, pages 808–827. Kaufmann, CA.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

JKTech Ltd. (1991). *The JKSimMet Steady State Mineral Processing Simulator*. JKTech, QLD, Australia.

Kietz, J. U. and Wrobel, S. (1992). Controlling the complexity of learning in logic through syntactic and task-oriented models. In Muggleton, S., editor, *Inductive Logic Programming*, pages 335–359. Academic Press, London.

Mowforth, G. (1992). Personal communication (taped interview).

Mozetic, I. (1987). The role of abstractions in learning qualitative models. In Langley, P., editor, *Proc. 4th International Workshop on Machine Learning*, CA. Kaufmann.

Muggleton, S. and Feng, C. (1990). Efficient induction of logic programs. In *First International Conference on Algorithmic Learning Theory*, pages 369–381, Tokyo, Japan. Japanese Society for Artificial Intelligence.

Pazzani, M. and Kibler, D. (1992). The utility of knowledge in inductive learning. *Machine Learning Journal*, 9(1):57–94.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5(3):239–266.