Inductive Learning from Good Examples

Xiaofeng (Charles) Ling
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
Email: ling@csd.uwoxa

Abstract

We study what kind of data may ease the computational complexity of learning of Horn clause theories (in Gold's paradigm) and Boolean functions (in PAC-learning paradigm). We give several definitions of good data (basic and generative representative sets), and develop data-driven algorithms that learn faster from good examples, and degenerate to learn in the limit from the "worst" possible examples. We show that Horn clause theories, k-term DNF and general DNF Boolean functions are polynomially learnable from generative representative presentations.

1 Introduction

In any inductive learning model, how data of the target theory are supplied to the learning programs is a crucial assumption. Identification in the limit [Gold, 1967] assumes that the series of examples is an admissible enumeration of all (positive and/or negative) examples of the target concept, and requires the learning algorithm to produce a correct hypothesis in some *finite* time. However, the computational time and the number of examples needed for convergence depend on the example series, and can not be specified *a priori.* Although there is an enumeration algorithm that identifies any h-easy [Blum and Blum, 1975] model of first order theories in the limit, such an algorithm is extremely inefficient in practice. Shapiro in his seminal work [Shapiro, 1981] presented an *incremental* method MIS which searches the hypothesis space from general to specific. However, MIS has two major shortcomings. First, the refinement of the refuted clauses may introduce a large number of faulty clauses, which need to be removed using a large number of negative examples. Second, the algorithm is still very inefficient (exponential).

Approximate identification, or PAC-learning (Probably Approximately Correct learning) [Valiant, 1984, Blumer *et a*/., 1989] assumes that the examples of the target concept are drawn randomly according to some fixed distribution for learning *and* for testing the conjecture. It requires the learner within feasible amount of time (quickly), to produce, with a high probability (confidently), a hypothesis with a small error (accurately). However, few things are shown PAC-learnable from random examples: many classes of Boolean concepts which seem simple to human learners (for example, k-term-DNF) are proved not PAC-learnable (by k-term-DNF) unless $RP = NP$. It is still open if general DNF is PAC-learnable.

Human learning, on the other hand, is efficient, interactive, and data-dependent. The study of the interaction between learners and teachers may reveal the essence of efficient human learning. Learning equipped with teachers answering various types of *queries* [Angluin, 1988] is one of such studies. We believe that the current learning model does not reflects the quality of the examples in the role of efficient learning. Here we present some results on what kind of data may ease the computational complexity of learning[1].

We start with an attempt to improve the efficiency of MIS. We first discover a heuristic very useful to improve the efficiency of Shapiro's MIS, while preserving the property of identification in the limit. Further study of the heuristic leads us to discover a *constructive* (or *data-driven)* algorithm SIM[2] for model inference. SIM opens a way for lis to study what kinds of data are necessary for successful learning. We give precise definition of "good" examples, and design constructive algorithms that learn faster from the good data, learn in longer time when examples become worse, and degenerate to learn in the limit from the "worst" possible data presentations. However, we believe SIM is not a polynomial algorithm. This leads us to study the problem in a more rigorous model of PAC-learning. We provide a stronger definition of "good" examples (generative representative sets), and show that k-term DNF and general DNF are PAC-learning from generative representative sets. Then we return to model inference problem, and give a similar stronger definition of "good" examples for polynomial inference of logic programs (provided enough negative examples are supplied).

*It may be argued that "good** examples may make learning arbitrarily fast. But as we will see, the nature of good data is *implicit.* That is, what is important is that a initial series of examples *contains* a set of good examples (to be defined). The order of examples in the series is irrelevant, and other examples in the series can be arbitrary. This eliminates the possibility of coding the target concept by examples.

[2]SIM is the inverse process of Shapiro's MIS.

Many data driven algorithms [Banerji, 1988, Ling, 1989, Muggleton and Buntine, 1988, Ishizaka, 1988] have been studied. However, most of them are heuristic. They learn faster from some good data, but fail to learn (in the limit) from "bad" data; little is done to characterize the "good" data for efficient learning. Some theoretical studies on learning from "good" examples have been explored. For example, Angluin [Angluin, 1981] defines "representatives" for live states as good strings for identifying a DFA polynomially. Rivest and Sloan [Rivest and Sloan, 1988] show how to learn an arbitrary concept by first learning its relevant subconcepts. Freivalds *et al* [Freivalds *et* a/., 1989] study good examples in recursive theoretic inductive inference.

2 Shapiro's Model Inference System

Shapiro's Model Inference System (MIS) [Shapiro, 198l] starts with the most general conjecture (a theory with an empty clause). When the conjecture T proves a negative example (T is too strong) the backtracing algorithm is invoked to remove faulty clauses from T. Removing a clause may overly specilize T, in which case the refinements of faulty clauses are added into T. However, the possible refinements of any clause are infinite. So which refinements of which clauses should be added into T? As Shapiro pointed out, [Shapiro, 1981], MIS identifies the theory in the limit "...as long as it [the refinement] is 'exhaustive' in some natural way". Therefore, unless using "good heuristics [unspecified in the paper] for ordering the addition of refinements" an indefinite number of faulty clauses can be inserted into the conjecture, which need to be removed using a large number of negative examples.

For any faulty clause a removed from the conjecture T it suffices to add a set of *most general specialization* of a; in this case, the set of immediate refinements (a finite number) of α. We define $mgs(\alpha) - \{p(\alpha)\}$ where p is a refinement operator complete for the theories [Shapiro, 1981]. However, a good rule of thumb (heuristics) is to add those clauses in $mgs(\alpha)$ that can be used to prove more (or most: a greedy algorithm) positive examples! Such a heuristic not only provides a reasonable guideline on theory revision, but also guarantees to find a solution if it exists. The following algorithm is the improved version of MIS with the heuristic.

```
$T = \{\Box\}; T' = \{\}$
repeat
examine the next example
while $T$ proves a negative example
        do backtracing, remove a faulty $\alpha \in T$
            $T' = T' \cup mgs(\alpha)$
while $T$ fails to prove a positive example $b$
        do find a set $S \subseteq T'$ such that $T \cup S \vdash b$
            $T = T \cup S; T' = T' - S$
until $T$ is neither too strong nor too weak
output T
loop {* repeat *}
```

Now the question is: What examples are necessary for convergence, and what examples are crucial for the conjecture to be updated appropriately? It turned out that the top-down algorithm MIS (driven by negative examples) is not a good place to study this problem: the overgeneralized clauses are not sensitive to the positive examples! Since our data-driven algorithm SIM constructs hypotheses from specific to general, we may characterize more acutely the properties of "good examples".

3 Intuitions of Good Data

The philosophy of what good examples are necessary for successful learning is quite simple. Even though the set of data is infinite (but recursively enumerable), there must exist some finite characterizations (such as logic programs) of the data[3]. If is these finite characterizations that are sought in inductive learning. To be successful in learning, the learner must receive data that *go through (use* or *exercise)* every "live" component of *one of* the finite characterizations[4]. For example, data that go through all "live" transactions in a DFA (see [Angluin, 1981]), all "live" production rules in a grammar for a language, or all "live" Horn clauses in a logic program[5]. Clearly such a set of data provides a *minimum* amount of information about the theory to be identified, and should be supplied in the *early stage* of learning. We call this set a *basic representative set.* Later we will provide a stronger definition of "good" examples (called generative *representative set)* which go through every component of a finite characterization several times, and which can be used to construct the component.

We give appropriate definitions of "good" examples and study the learnability results using good examples. The good examples supplied in the early stage of learning appear in the example series only implicitly. We assume that the first $p(n)$ [6] positive data should contain good examples (to be defined). In the Gold's model, the learning algorithm is allowed to ask membership queries; while in PAC-leaming, more random examples may be drawn according to the fixed distribution. In both models, the total sample size and computational complexity for successful learning is measured by n and the size of the target theory (and $1/\epsilon, 1/\delta$ in PAC). If the total sample size and computational complexity are polynomial, we say that the class of theories is polynomially learnable from good examples. If a class of Boolean functions not PAC-learnable becomes PAC-learnable with good examples, it implies that drawing a polynomial sample randomly will not have a high probability of containing good examples.

Obviously, p measures the quality of the data (or the teacher): as p increases, the learning algorithm gradually degrades to be identification "in the limit". Thus,

[3]In model inference of Horn clause theories, if such a finite characterization does not exist, theoretical terms may be needed. This is outside the scope of the paper.

[4]We assume that the components are of disjunctive nature in this paper.

[5]Something is "live" means it has to be *used* in deriving some data.

[6]p is a polynomial function and n is the maximum of the problem sizes.

our learnability model unifies efficient learning and identification in the limit based on the quality of the data presentation.

4 Basic Representative Sets

Given a language L, a logic program *LP* for *L* is any finite set of Horn clauses of *L*. The model of a logic program is its least Herbrand model, i.e., the set of all ground atoms of *L* derivable from *LP. A complete data presentation of a model* is an admissible enumeration of all positive and negative examples from the Herbrand base. The *class of models of logic programs of L* is the set of least Herbrand models of all logic programs of *L*. The *model inference problem* is: given language *L* and a data presentation of a unknown model *M* from the class of models of logic programs of L, find a logic program whose least Herbrand model is M.

Definitions. A basic representative *set* of a model *M* is a basic representative set of any logic program *LPM* with least Herbrand model *M*. A *basic representative* set* of a logic program *LP is* a set *S* of ground atoms obtained by taking, for each clause $p = P$:- $Q_1, \ldots, Q_n$ *of LP,* all ground atoms in a true instantiation of *p* (all atoms are true in the model).

Basic representative sets for a model can be very small; the size of a basic representative set is no more than the number of occurrences of atoms used in the corresponding *LP*. For example, a model about list reverse has a logic program:

$$rev([],[]).$$
$$rev([X|Y], Z) \text{ :- } rev(Y, Y_1), conc(Y_1, X, Z).$$

A basic representative set for the model may contain: $rev([],[]), rev([a,b],[b,a]), rev([b],[b]), conc([b], a, [b,a])$.

Notice that for any model to be inferred, there may exist more than one logic program whose least model is the target model. For example, there are several sorting programs to sort a list of integers. Depending on which logic program whose basic representative set is given, that logic program will be inferred.

Definition. A basic representative *presentation* of a model *M* is an admissible data presentation whose initial portion (size *p(n)* for a polynomial function *p)* contains a basic representative set of *M.*

5 Abstraction Operators and SIM

Our data-driven algorithm SIM is the inverse process of MIS. SIM starts with the most specific conjecture and makes it more general by adding generalizations of clauses in the current conjecture obtained by applying abstraction operators, guided by good examples. On the other hand, we prove completeness for the abstraction operators, showing that SIM identifies *h-easy* models in the limit.

Let *size* be a total recursive function from clauses to positive integers that gives a measurement of the complexity of clauses. The only requirement on *size* is that for any integer n, the set $\{p \mid p$ is a Horn clause with $size(p) < n\}$ is finite.

Definition. An *abstraction operator A* is a mapping from clauses to sets of clauses, such that

$$A(p) = \{p\} \cup A'$$

where $A' \subseteq \{q \mid q \vdash p \text{ and } size(p) > size(q)\}$.

Any such operator *A* induces an operator (also denoted *A)* which takes sets of clauses to sets of clauses; for a set *S* of clauses,

$$A(S) = S \cup \bigcup_{p \in S} A(p).$$

Note that this operator is *monotonic,* i.e. $S \subseteq S'$ implies $A(S) \subseteq A(S')$. As usual, we define the operator A^n by induction: $A^1(S) = A(S)$, and $A^{n+1}(S) = A(A^n(S))$. One important difference between refinement and abstraction operators is that for any abstraction operator *A* and any finite set *S of* clauses, there is a finite "least fixed point", the *closure of S* under *A.*

Lemma 5.1 *For any finite set S of Horn clauses, there is an integer n with* $A^{n+r}(S) = A^n(S)$ *for any* $r > 0$[7].

For convenience we will often use a (finite) set *A* of abstraction operators. The closure of *S* under *A* will be denoted .4(5).

5.1 Completeness and Convergence

Definition. A set *A* of abstraction operators is *complete* if for every logic program *LP* and every basic representative set *S* of LP, *A(S)* contains every clause of *LP* (up to possible renaming of variables in clauses).

Since abstraction operators are monotonic, we observe that if *A* is complete, and *S contains* a basic representative set of *LP,* then *A(S)* contains every clause of *LP.*

A simple version of the constructive algorithm SIM works as follows: from any finite set of positive data 5, *A(S)* can be calculated. If *A(S)* cannot prove some positive data, they are added to *S* and *A(S)* is recalculated. Eventually the basic representative set of the model will be contained in *S,* and with a complete set of abstraction operators, *A(S)* will contain all clauses in the logic program of the model. *A(S)* may contain faulty clauses, which are removed by Shapiro's contradictory backtracing algorithm when a negative example is proved to be true. Unlike Shapiro's MIS, the removal of a faulty clause does not introduce any more clauses into the conjecture. Clearly, this simple algorithm identifies any h-easy models in the limit.

Theorem 5.1 *If a set of abstraction operators A for the class of theories is complete, then the constructive algorithm above identifies any h-easy model*[8] *of that class in the limit given an enumeration of positive and negative examples of the model.*

[7]Most proofs of the paper are omitted and can be supplied from [Ling and Dawes, 1990, Ling, 1991].

[8]The attempted derivation of an atom from T may not halt. A total recursive function *h* is used to bound the resources in proving. For more details see [Shapiro, 1981, Blum an d Blum, 1975].

```
Read a set $S$ of positive data
$T = initial(S)$
1: read more examples.
   Let $F+$ be positive examples, $F-$ be negative
2: repeat
   if $f \in F+$ is not provable in $T$ then
      if $\exists f$ is provable in $\mathcal{A}(T)$ but not in $T$
         then $T = T \cup \{p \mid p \in \mathcal{A}(T) - T\}$
            where $p$ is used in the proof of $f$
         else $T = \mathcal{A}(T)$
   while $a \in F-$ is proved true in $T$
         do call backtracing algorithm
   until $T$ proves all in $F+$ or $\mathcal{A}(T) = T$
   if $T \not\vdash f \in F+$ then $T = T \cup \{f\}$, goto 2
   output $T$
   goto 1     {* loop forever*}
```

FIGURE 1: THE SIM ALGORITHM

However, even if S is small, $A(S)$ can be huge (exponential). We use the same heuristic for MIS to improve the efficiency of the algorithm while retaining the property of identification in the limit.

5.2 Using Good Data

In general, from the current conjecture set T, only a small number of clauses in $A(T)$ will be useful in further generalizations (via abstraction operators). These clauses may be identified by some "simple" positive data that can be proved using these clauses. If $F+$ in the following algorithm contains positive examples ranging from "simple" to "complex" with respect to applications of abstraction operators, SIM will be much more efficient and need much fewer negative data than Shapiro's MIS.

In practice what good examples should be supplied to SIM? First of all, a small set of basic representative set should be supplied. Then, simple examples are those positive examples similar to the examples in the representative set, and more complex examples are those less similar ones. For example, $rev([a],[a])$, $rev([c],[c])$ are close to $rev([b],[b])$; $rev([a,c],[c,a])$, $rev([c,b],[b,c])$ are similar to $rev([a,b],[b,a])$; while $rev([c,d],[d,c])$ and $rev([a,b,c],[c,b,a])$ are less similar.

Also we notice that for a given set T containing a representative set, $A(T)$ will be calculated. The size of $A(T)$ may be exponentially larger than $size(a)$ for $a \in T$. Thus, examples in S of SIM must be small (i.e., $size(a)$ must be small). This verifies our intuition that in learning list reverse for instance, the examples that reverse very long lists are not good examples (while in PAC-learning, examples drawn randomly may have an indefinite size).

Figure 1 gives an improved version of SIM. The conjecture T starts with the set *initial(S)*, which is the most specific conjecture from S in the representational language of the theory (See Section 6 for details). The algorithm identifies any A-easy models in the limit, and learns faster given a good set of data.

6 Various Abstraction Operators

We design abstraction operators for various sub-classes of clauses, similar to Shapiro's refinement operators for sub-classes of clauses [Shapiro, 1981]. Thus, the algorithm is more efficient if the class of models to be inferred is known.

6.1 Abstraction Operators for Atoms

We first discuss the simplest class of models, which have logic programs containing only unit clauses. Here initia/(S) in the algorithm SIM is just the set of all positive examples in 5. The set of abstraction operators A_{at} for the atom p contains two operators A_1 and A_2 defined as follows:

- $A_1(p)$ is the set of atoms obtained from p by replacing some or all occurrences of a constant in p by a variable not in p.
- $A2(p)$ is the set obtained by replacing some or all occurrences of a compound ground term in p by a variable not in p.

Theorem 6.1 *$\mathcal{A}_{at}$ is a set of abstraction operators completc for atomic Horn clauses.*

6.2 Abstraction Operators for Horn Clauses

Skipping other subclasses, we now study abstraction operators for general Horn clause theories. The most specific conjecture *(initial(S))* contains n most specific clauses, each in the form of Q_k :- $Q_1, \ldots, Q_{k-1}, Q_{k+1}, \ldots, Q_n$ where $Q_1, Q_2, \ldots, Q_n$ are ground atoms in S. The set A_{cl} of abstraction operators for general Horn clauses contains three operators A_1, A_2, and A_3.

- A_1 and A_2 are the same as in A_{at} except that they now apply to occurrences of a constant or term in a whole clause rather than in one atom.
- if p is any Horn clause with at least one atom in its body, $A_3(p)$ is the set of clauses formed by removing any one atom from the body of p.

Theorem 6.2 *$\mathcal{A}_{cl}t$ is a set of abstraction operators complete for general Horn clauses.*

If the class of models is a sub-class of models of general Horn clauses, then more efficient abstraction operators are possible. For example, we might consider only Horn clauses with no variable occurring only once in a clause, or with the length of body bounded by a small constant. It is easy to modify the abstraction operators for these and other syntactically restricted classes.

7 Implementation of SIM

We have developed SIM in Quintus Prolog. SIM learns faster with a small set of good examples containing a representative set. However, in learning various logic programs, SIM does not have a consistent good performance. The major reason is that there may not exist any new positive examples (besides those in S of SIM) that can be proved in A(7) but not in T (see algorithm SIM in figure 1). In this case abstraction operators have to be applied to the current conjecture ($T = A(T)$ in the algorithm). This makes the conjecture to grow too fast to be manageable. In general, we believe SIM is not a polynomial algorithm. In fact, there is no polynomial algorithm that identifies any Horn clause theory from basic representative presentations if the conjecture must contain the same number of clauses as the target theory (see next section).

8 PAC-Learning Boolean Functions from Good Examples

Definition. A class of concepts is *PAC-learnable from good examples* if for any / in the class, there exists a learning algorithm *A* such that given any set *S* of positive examples of / that contains the good examples, and $|S| = p(n, size(f))$ where *p* is a polynomial function and n is the number of variables, *A* PAC-identifies /. Notice that *A* may draw more random examples from the fixed distribution.

Similar to the good examples in the model inference discussed in the previous sections, we may define good examples in PAC-learning as a set of examples that go through every term of the DNF function once (basic representative set). However, a little more thoughts reveal that such definition is insufficient for polynomial learning of k-term DNF. Assume that such a polynomial algorithm (denoted as *A)* existed, then k-term DNF would be PAC-learnable (by k-term DNF) without basic representative sets as follows: First a sample of a polynomial size is drawn such that with a high probability each term in the target function is sampled. Choose at most A: data from the sample as basic representative sets in *A*. Apply *A* on all sets (since there is a polynomial number of the sets, the sample size *A* would draw would be polynomially larger). Clearly this polynomial algorithm PAC-identifies any k-term DNF. This is contradictory to k-teYm DNF is not PAC-learnable by k-term DNF (unless *RP = NP).* Essentially it is NP hard to find a k-term DNF that is consistent with a set of examples even when a set of basic representative set is provided. Since any k-term DNF can be transferred to a propositional Horn clause theory with k clauses, therefore, even with a basic representative set, there is no polynomial algorithm that identifies the theory if the conjecture must contain the same number of clauses as in the target theory. It is unknown to us if basic representative set is sufficient for polynomial inference of general DNF formulas (by producing a DNF with a size at most polynomially larger).

8.1 Polynomial Learning of DNF from Generative Examples

We provide a stronger definition of good examples where k-term DNF and genera) DNF are polynomially learnable.

A Boolean function in DNF consists of terms. We first define generative sets for a term. Let a be an instance $\{0,1\}^n$. Let < be a term, and *t(a)* be the truth value of *t* with a as the assignment of the variables in *t.*

Definitions. For a set *S* of instances, *lgg(S)* (least general generalization) is a term *t* such that for all $a \in S$, *t(a)* is true; and for any other term t', if for all $a \in S$, *t'(a)* is true, then $t \vdash t'$. *S* is a generative set of *t* if for all $a \in S, t(a) = 1$, and *lgg(S)* = t.

lgg(S) is easy to calculate: check i^{th} bits of all instances in *S.* If they are all 1, then x_i is in t; if all 0, then x_i- is in t; otherwise neither x_i- nor x_i is in *t.* In another word, if *t* does not contain a variable x nor x (i.e. x is an *irrelevant* variable in t), there are at least two instances in a generative set of *t* whose bits on x are different. Intuitively the generative set of *t* contains positive examples showing contrast information. Notice however, these examples may not constitute a "near-miss" [Winston, 1984].

For any term t, there exists a smallest generative set *S* of *t* with $|S| \le r$, where $r \ge 2$ is a constant throughout the paper. This is the set of positive examples *with* a *constant size* showing all contrast information of a term.

Definition. *S* is a generative *representative set* for a term *t* if *S* is a generative set oft with $|S| \le r$. A *generative representative set for a Boolean formula f in DNF* is the union of generative representative sets for all terms in an equivalent DNF formula with a smaller or equal size. A *generative representative presentation R* for / is a set of positive examples that contains a generative representative set of / with $|R| < p(n, size(f))$, where *p* is a polynomial function and *n* is the number of variables.

Using generative representative presentations, we show that many classes of DNF formulas are PAC-learnable. Basically, from a generative representative presentation, we can form all possible terms by applying *lgg* on at most r examples, and form all possible Boolean formulas in the hypothesis space (i.e., form all k-term DNF formulas in learning k-term DNF and form one disjunction of all terms in learning general DNF). Draw enough random examples according to a simple and important theorem due to [Valiant, 1984, Blumer et a/., 1989]: *If C contains a finite number of Boolean functions, then any polynomial algorithm that requests sample size* at ieast $1/\varepsilon \ln(|C|/\delta)$ and *outputs any consistent function in C PAC-identifies C.* For proofs of the following theorems, see [Ling, 199l].

Theorem 8.1 *k-term-DNF is PAC-learnable (by k-term DNF) from generative representative presentations with a sample size* $O(\varepsilon^{-1}kr\ln(p(n)/\delta))$; *and time complexity* $O(\varepsilon^{-1}k^2rn\ p(n)^{(r+1)(k+1)}\ \ln(p(n)/\delta))$.

Theorem 8.2 *Any DNF f is PAC-learnable (by DNF with at most p(n, size*$(f))^{(r+1)}$ *terms) from generative representative presentations with sample size* $O(\varepsilon^{-1}r\ln(p(n, size(f))/\delta))$; *and time complexity* $O(\varepsilon^{-1}rn\ p(n, size(f))^{(r+1)}\ln(p(n, size(f))/\delta))$.

Is the definition of generative representative set too strong? In some sense it is not. In the worst cases it is necessary to enumerate all terms formed by applying *lgg* on subsets of data from initial portion of p(n) data. For $\binom{p(n)}{r(n)}$ to be polynomial, $r(n)$ (when $r(n) < p(n)/2$) can only be a constant. Of course, generative representative set is not a *necessary* condition for polynomial learning of DNF formulas: there are other spacial cases where DNF is PAC-learnable.

9 Model Inference from Generative Examples

The definitions of generative examples for the model inference can be given in a similar way as for PAC-learning DNF. The least general generalization *(lgg)* of a set of clauses is taken from Protkin's work [Plotkin, 1970]. We define a set of ground atoms as a *generative representative set* for a logic program *LP* if it is the union of generative representative sets for all clauses in *LP.* A set *S* of ground atoms is a generative *representative set* for a clause A :- $B_1, \ldots, B_n$ if there are at most r true instances of the clause in the form of A_i :- $B_{1,i}, \ldots, B_{n,i}$ $(1 \leq i \leq r)$ such that all $A_i, B_{1,i}, \ldots, B_{n,i}$ are in *S,* and $lgg_{1 \leq i \leq r}(A_i$:- $B_{1,i}, \ldots, B_{n,i})$ contains the clause A :- B_1 •••, *Bn-* The resulting clauses can be very long. Some techniques of logical reduction of clauses [Buntine, 1988, Muggleton and Feng, 1990] and syntactic restriction (such as ij-determination [Muggleton and Feng, 1990]) may be applied. Clearly the total number of possible clauses from the *lgg* of at most r most specific clauses is polynomial, so is the maximum number of faulty clauses in the conjecture. Thus the model inference problem can be solved in polynomial time from a generative representative presentation if a proper set (with a polynomial size) of negative examples is provided. This is, as we believe, the condition for other constructive algorithms based on least general generalization to learn in polynomial time.

10 Conclusions and Further Research

A new learning model that learns faster if the initial portion of the examples contains a set of good data is developed. The model captures the quality of the data (or teacher) in the role of the speed of the learning. Thus, the model provides a new approach to study human learning, which is efficient, interactive and data-driven.

Currently we are studying the use of "negative representative sets", and the appropriate definitions of good examples for theories with non-disjunctive components (such as M-formulas). We hope to investigate language learning from good sentences in the near future.

Acknowledgement
The author gratefully thanks referees for their thoughtful and valuable comments, and Mike Dawes, Bob Webber for helpful discussions on various aspects of the topics. The author also acknowledges the support from NSERC Operating Grant OGP0046392, the Internal NSERC grants, and the New Faculty Start-up grant.

References

[Angluin, 1981] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control,* 51, 1981.

[Angluin, 1988] D. Angluin. Queries and concept learning. *Machine Learning,* 2(4), 1988.

[Banerji, 1988] R.B. Banerji. Learning theories in a subset of a polyadic logic. In *Proceedings of First Workshop on Computational Learning Theory,* 1988.

[Blum and Blum, 1975] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control,* 28:125-155, 1975.

[Blumer *et ai,* 1989] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learn ability and the vapnik-chervonenkis dimension. *J ACM,* 36(4):929 965, 1989.

[Buntine, 1988] W.L. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence,* 36(2):149-176, 1988.

[Freivalds *et ai,* 1989] R. Freivalds, E.B. Kinber, and R. Wiehagen. Inductive inference from good examples. In *Proceedings of International Workshop of Analogical and Inductive Inference,* 1989.

[Gold, 1967] E. Gold. Language identification in the limit. *Information and Control,* 10:47-474, 1967.

[Ishizaka, 1988] H. Ishizaka. Model inference incorporating generalization. *Journal of Information Processing,* 11(3), 1988.

[Ling and Dawes, 1990] X.C. Ling and M. Dawes. Learning with representative presentations the inverse of shapiro's mis. Technical Report 263, Department of Computer Science, University of Western Ontario, 1990.

[Ling, 1989] X.C. Ling. Learning and inventing of horn clause theories - a constructive method. In Z.W. Ras, editor, *Methodologies for Intelligent Systems, 4,* pages 323-331. North-Holland, 1989. *

[Ling, 1991] X.C. Ling. Inductive learning from good examples. Technical report, Department of Computer Science, University of Western Ontario, 1991. Forthcoming.

[Muggleton and Buntine, 1988] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the fifth international conference on Machine Learning,* 1988.

[Muggleton and Feng, 1990] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of first International Conference on Algorithmic Learning Theory.* OHMSUA Tokyo, 1990.

[Plotkin, 1970] G.D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5,* pages 153-163. Elsevier North-Holland, 1970.

[Rivest and Sloan, 1988] R.L. Rivest and R. Sloan. Learning complicated concepts reliably and usefully. In *Proceedings of the 1988 Workshop on Computational Learning Theory (COLT-88).* Morgan Kaufmann, 1988.

[Shapiro, 1981] E. Shapiro. Inductive inference of theories from facts. Technical Report TR 192, Computer Science Department, Yale University, 1981.

[Valiant, 1984] L.G. Valiant. A theory of the learnable. *Comm. ACM,* 27(11):1134-1142, 1984.

[Winston, 1984] P.H. Winston. *Artificial Intelligence.* MA: Addison-Wisley, 2 edition, 1984.