PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Hypothesizing an algorithm from one example: the role of specificity

S. H. Muggleton FREng

Department of Computing, Imperial College London, London, UK

Research

Check for updates

Cite this article: Muggleton FREng SH. 2023 Hypothesizing an algorithm from one example: the role of specificity. *Phil. Trans. R. Soc. A* **381**: 20220046.
https://doi.org/10.1098/rsta.2022.0046

Received: 28 October 2022
Accepted: 13 February 2023

One contribution of 11 to a discussion meeting issue 'Cognitive artificial intelligence'.

Subject Areas:
artificial intelligence

Keywords:
program synthesis, Bayesian machine learning, logic learning

Author for correspondence:
S. H. Muggleton FREng
e-mail: s.muggleton@imperial.ac.uk

Statistical machine learning usually achieves high-accuracy models by employing tens of thousands of examples. By contrast, both children and adult humans typically learn new concepts from either one or a small number of instances. The high data efficiency of human learning is not easily explained in terms of standard formal frameworks for machine learning, including Gold's learning-in-the-limit framework and Valiant's probably approximately correct (PAC) model. This paper explores ways in which this apparent disparity between human and machine learning can be reconciled by considering algorithms involving a preference for specificity combined with program minimality. It is shown how this can be efficiently enacted using hierarchical search based on identification of certificates and push-down automata to support hypothesizing compactly expressed maximal efficiency algorithms. Early results of a new system called DeepLog indicate that such approaches can support efficient top-down construction of relatively complex logic programs from a single example.

This article is part of a discussion meeting issue 'Cognitive artificial intelligence'.

1. Introduction

Consider a typical IQ-test question, based on textual analogy (see figure 1), which says that 'alice' is to 'ECILA' as 'bert' is to '?'. In this case assume the word indicated by '?' is 'TREB' (figure 2) since the relation R between 'alice' and 'ECILA' is 'the reverse of the uppercase of the given letter sequence'. If there are no limits on the length of the letter sequences, an infinity of

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.6607461.

THE ROYAL SOCIETY
PUBLISHING

alice	ECILA
bert	?

Figure 1. Analogy problem.

alice	ECILA
bert	**TREB**

Figure 2. Expected response.

alternative answers might have been possible, including 'ECILA'. But for such an IQ-test question to be effective, the answer 'TREB' must have high consensus among typical participants. In this paper, it will be shown that standard computational learning theory (CoLT) [1,2] approaches do not account for humans' ability to hypothesize such a relation, R, from a single example.

Question: How can we model human abilities to hypothesize R from a single example?

An answer to this question has potential to provide insights into human perception and reasoning, and could help to identify new methods for efficient human–machine interaction. One relevant point worth noting is that human subjects presented with the problem are likely to already know the relevant commonly understood sub-concepts of *uppercase* and *reverse sequence*. However, it is unclear how a person could identify these two relations as the most relevant out of a large number of candidate relations which they already know.

To address the question above a modified version of an existing CoLT approach [3] is developed. In the process, it is shown that for high expected-accuracy (EA) to be achieved on unseen cases, the Bayes' optimal hypothesis selected needs to provide a trade-off between the description length and generality of the algorithm representing R. Additionally, in the case of learning from a single example it is shown that the algorithm being learned needs to have low generality, meaning the chances are low for it correctly predicting the truth value of a randomly selected instance. This model of highly data efficient concept learning has ramifications concerning the lifetime learning of concepts, such as R, by humans having access to an accumulated mental library of previously learned low-generality relations.

The paper is organized as follows. In §2, we review relevant work related to learning from one example, or *One-shot Learning*, in both Cognitive Science and Artificial Intelligence. Following this, §3 introduces a mathematical framework consisting of a learning protocol, a Bayes' optimal solution and an associated expected-error (EE) bound for one-shot learning. Section 4 describes a new meta-interpretive learning (MIT) [4] system, called DeepLog, which supports both one-shot and few-shot learning of efficient algorithms from relational examples. DeepLog achieves this by identifying algorithms which combine low description length with low generality. DeepLog is used in the experiments described in §5, which indicate that high-accuracy algorithms can be efficiently learned from one example, not only for simple memory-free automata, but also for cases such as the textual analogy (figure 1), in which a push-down stack, of unbounded size, is required. In §6, we conclude and discuss possible areas for future research on this topic.

2. Related work

(a) Cognitive science

Over the last two decades there have been an increasing number of papers (e.g. [5–9]) demonstrating machine learning algorithms which learn concepts from a single example. This is referred to as 'one-shot learning'. Such algorithms are motivated by the observation that human learning often involves generalizations from a single example. However, such work has, to date, lacked mathematical analysis of the error associated with this form of learning. According to one of these papers published in the Cognitive Science literature [5] 'People can learn ... concepts from just one example, but it remains a mystery how this is accomplished.' Clarifying the elements of this mystery is the central motivation of the present paper.

(b) Linguistics

As already pointed out in the example shown in figure 1, it is clear that some of the learning bias which enables one-shot learning comes from concepts already known to the human learner, such as *uppercase* and *reverse sequence*. However, it is not immediately obvious how many such background concepts humans have which might play a part in learning a new concept. One bound on available concepts comes from studies in linguistics on the typical size of human vocabulary. According to one such study [10] the average adult knows somewhere in the range of 10 000–42 000 words, including, in our 'alice' example case, 'uppercase' and 'reverse'. However, any algorithm employing tens of thousands of background concepts would be not only overwhelmed by the available combinations of these terms, but also have to deal with the danger of over-fitting the example provided (e.g. always predicting 'ECILA'). Despite this, humans are able to rapidly and reliably identify the relevance of 'uppercase' and 'reverse' and use these to hypothesize a new concept in the 'alice/ECILA' example.

(c) Identification in the limit

CoLT [11,12] is the study of algorithms which learn hypotheses from examples. The earliest such theoretical framework was introduced in 1967 by Gold [13], and is known as *Identification in the Limit*, in which learning algorithms which are provided with a finite enumeration of both hypotheses and examples of a target language, for positing consistent hypotheses. Learning is effective in the case there exists a finite prefix of the example sequence after which the learning algorithm chooses the target language, and does not subsequently alter its hypothesis. Gold proved that, in general, for infinite formal languages (such as regular and context-free languages in the Chomsky hierarchy) such identification is not possible when learning from positive examples alone. The reason, as shown in figure 3, is that for such language classes, given a finite number of examples, at no point can an algorithm discriminate between the most general language, consisting of all possible sequences from a given alphabet, or alternatively the most specific language, containing only the examples provided so far. Gold [13] pointed out the apparent disparity between this formal result and existing psycholinguistic studies by McNeill [14], which had shown that children learn language largely from positive examples, and tend to ignore corrections to their use of grammar.

(d) Probably approximately correct learning

In 1984, a new framework for computational learning was introduced by Leslie Valiant [2]. By contrast to Gold's requirement of exact identification, probably approximately correct (PAC) addresses learning algorithms which converge efficiently, with high probably, on an arbitrarily close approximation of the target theory, as increasing numbers of randomly selected examples are provided. Among those hypothesis classes considered by Valiant [2] only k-CNF propositional logic formulae, has a PAC convergence proof based on positive examples alone. However, for a given propositional signature a k-CNF formula has a finite domain, as opposed to the grammar classes with infinite domains studied by Gold. In general, positive learnability results for the PAC framework tend to be restricted to highly constrained hypothesis classes, and the upperbounds on out-of-sample error tend to be weak when compared to the actual error in experimental trials.

$E = \{\text{ab,aab,aaab}\}$

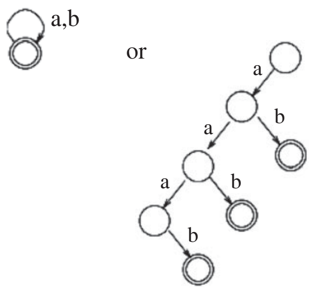

Figure 3. Identification in the limit. Complexity versus generality.

(e) Bayesian positive-only learning

In 1996 [3], the author introduced a framework to analyse the learning of logic programs from positive-only examples.[1] A Bayesian prior over the hypothesis space is assumed. The next section shows how a Bayesian approach allows the identification of maximal aposterior probably (MAP) hypotheses which provide a trade-off between the complexity of the hypothesis and its generality. It is shown that polynomial time logic programs can be learned with high accuracy from a randomly selected positive example sequence. This result goes beyond the positive-only results of Gold and Valiant, and supports efficient learning of infinite languages and programs. However, Muggleton [3] falls short of providing effective error bounds for one-shot learning. We will address this issue in the next section.

3. Theoretical framework

(a) Bayesian learning protocol

For the purposes of this paper, we introduce a specialization of the teacher-learner positive-only protocol introduced in [3]. In this variant, relational instances are atoms $r(a, b)$, where r is a relation (i.e. predicate of arity 2) and a and b are ground terms.[2] A relational program is a logic program in which all predicates have arity 2.

— X is a countable[3] set of relational instances.
— D_X is the teacher's probability distribution over X.
— $\mathcal{H} \subset 2^X$ is a countable hypothesis set for which each $H \in \mathcal{H}$ represents the least Herbrand model of a relational program.
— $D_{\mathcal{H}}$ is the teacher's probability distribution over $\mathcal{H}$.
— The teacher randomly chooses target theory $T \in \mathcal{H}$ from $D_{\mathcal{H}}$ and then chooses $E = x_1 \ldots x_m$ randomly and independently from $D_{X|T}$, with probability $x \in E$ such that
$$D_{X|T}(x) = D_X(x|T) = D_X(x \cap T)/D_X(T) = \begin{cases} 0 & \text{if } x \notin T \\ \frac{D_X(x)}{D_X(T)} & \text{otherwise} \end{cases}$$
— The learner now selects $H \in \mathcal{H}$ for which $E \subseteq H$.

[1]Later Tenenbaum [8] investigated positive-only learning in a Bayesian framework and referenced [3] but concentrated on an approach based on finite extensions. Note that infinite extensions, such as those considered in [3] and §3 of this paper, are required for learning algorithms such as *reverse*.

[2]For instance, $X = \{\text{rvu}(\langle\langle \text{A,l,i,c,e}\rangle, \langle \text{E,C,I,L,A}\rangle\rangle), \text{rvu}(\langle\langle \text{B,e,r,t}\rangle, \langle \text{T,R,E,B}\rangle\rangle), \ldots\}$ are relational instances of the relation *rvu*. Note that an algorithm like reverse uppercase is usually defined over a countably infinite set, and is both a relation and a one-to-one function.

[3]Either finite or countably infinite.

$$\boxed{\begin{aligned} p(H|E) &= \frac{p(H)p(E|H)}{p(E)} \ \textbf{[Bayes theorem]} \\ &= \frac{p(H)\prod_{i=1}^{m}\frac{D_X(x_i)}{g(H)}}{p(E)} \\ &= p(H)\left(\frac{1}{g(H)}\right)^m c_m \\ -\ln p(H|E) &= sz(H) + m\,(\ln g(H)) + d_m \\ \Longrightarrow \ \textbf{Minimize} & \ -\ln p(H|E) \text{ over } H \in \mathcal{H} \end{aligned}}$$

Figure 4. Positive-only MAP selection. $m = |E|, x_i \in E, c_m = \prod_{i=1}^{m} D_X(x_i)$ and $d_m = \ln\ c_m$.

$$\boxed{-\ln p(H|\,E) = sz(H) + \ln g(H) + d_1}$$

Figure 5. Bayesian one-shot learning, $m = 1$ case.

— For $H \in \mathcal{H}$, $D_X(H) = \sum_{x \in H} D_X(x)$.
— The teacher then assesses $\text{Error}(H, T)$ as $D_X(H \setminus T) + D_X(T \setminus H)$.
— The EE of hypothesis H is then

$$\text{EE}(m) = \sum_{T \in \mathcal{H}} D_{\mathcal{H}}(T) \sum_{x \in E \in T^m} D_X(x|T)\text{Error}(H, T).$$

where the set of all cardinality m training sets T^m is defined recursively as follows. $T^1 = T$ and $T^m = T \times T^{m-1}$.
— $sz(H) = -\ln D_{\mathcal{H}}(H)$ is referred to as the size of H.
— $g(H) = \sum_{x \in H} D_X(x)$ is referred to as the generality of H.

(b) Positive-only MAP selection

Using the Bayesian Learning Protocol the learner's positive-only MAP selection method introduced in [3] is shown in figure 4. The complexity versus generality problem highlighted by Gold's result (figure 3) is resolved by the learner selecting an hypothesis H which trades-off $sz(H)$ and $g(H)$.[4] As the number of training examples m increases, the need to minimize $g(H)$ progressively dominates minimizing $sz(H)$, which contrasts with the fixed hypothesis ordering assumption of Gold [13]. Let us now consider Bayesian one-shot learning.

(c) Bayesian one-shot learning

One-shot learning is simply the special case of Bayesian positive-only learning for which $m = 1$ (see figure 5). However, this case was not considered in [3], since the EE bound derived in that paper gives $\text{EE}(1) \leq 2.33$, which is trivially true since error is in the $[0, 1]$ interval.

(d) Expected-error bounds

EE results are shown in figure 6. The positive-only result was based on the assumption that $g(T) \leq 1/2$. Under this assumption, it was noted [3] that the bounds indicate (see figure 6)

[4]While [3] assumes $p(E|H) = (1/g(H))^m$, Tenenbaum's *size principle* in [8] is that $p(E|H) = (1/\text{size}(H))^m$. The former corresponds to the probability of H, with potentially infinite extension, being true of m instances randomly selected from D_X, while the latter is the non-normalized probability of concept H, with *finite extension* size (H), being true of m instances randomly selected from a uniform distribution over X.

type	expected-error
positive-only [3]	$\mathrm{EE}(m) \le \frac{2.33 + 2\ln m}{m}$
positive + negative [3]	$\mathrm{EE}(m) \le \frac{1.51 + 2\ln m}{m}$
one-shot	$\mathrm{EE}\,(m = 1 \vert g(T)) \le 4.66 g(T)$

Figure 6. Previous and new expected-error bounds.

$$\boxed{\mathrm{EE}(m = 1 | g(T)) \le 0.05 \text{ when } g(T) \le 0.01}$$

Figure 7. Low-generality/high-accuracy region.

monkey	banana
sparrow	**worm**

Figure 8. Analogy involving relation *eats*.

that it does not take many more examples to learn from positive-only than from a mixture of positive and negative examples. This was found to be consistent with out-of-sample results for an implementation of these two approaches [3]. The third is a new result of the present paper. This can be derived by generalizing the $g(T) \le 1/2$ assumption to $g(T) \le \alpha$ where $0 \le \alpha \le 1$, in which case $\mathrm{EE}(m) \le \alpha(4.66 + 4\ln m)/m$. In the one-shot learning case, $m = 1$, we get $\mathrm{EE}(m = 1) \le \alpha(4.66 + 4\ln 1)/1$ and therefore $\mathrm{EE}(m = 1|g(T)) \le 4.66g(T)$.

(e) Low-generality targets

The new one-shot EE bound in figure 6 allows us to predict high-accuracy hypotheses are possible from one example if and only if the generality of the target hypothesis is low (see figure 7).

(f) Choice of representation

The motivation for this work is to analyse and demonstrate a general approach to learning from a single example in a way that models human abilities. One-shot learning of an algorithm is particularly challenging, since it involves a countably infinite domain (e.g. reversing and uppercasing a sequence such as $\langle a, l, i, c, e \rangle$). However, humans also have a capacity to identify relations from a single example, as part of a broad range of analogy problems. For instance, the answer 'Insect' to the analogy problem in figure 8 would be equally acceptable for a human to that given. For this reason, rather than select a functional programming language, the base representation is that of relational logic programs rather than functional programs, though the results in figure 6 would apply in either case since functions can be considered as many-to-one relations. In the following two sections, we will investigate the learning of relational logic programs from a single example, in order to test the one-shot result of figure 6 in practice.[5]

4. DeepLog implementation

[5]Since this result applies to very low-generality hypotheses, finding a single satisfying solution is considered to be of more importance than the order in which multiple satisfying solutions should be presented.

$$\boxed{\textbf{chain: } R(x, y) \leftarrow S(x, z), T(z, y)}$$

Figure 9. Chain metarule defines relation R as composition of S and T. Only clauses of this form are included in programs hypothesized by DeepLog.

(a) DeepLog system

DeepLog[6][7] is a new inductive logic programming (ILP) [15–17] system, implemented in SWI-Prolog, based on MIL [4,18–21]. ILP systems hypothesize a relational logic program H from background knowledge B, primitive relations library $B_0 \subseteq B$, positive examples E^+ and negative examples E^-. MIL systems additionally require the user to provide a set of metarules M which indicate the structure of rules to be learned. By contrast, DeepLog minimizes the requirements on the user by providing background knowledge from an existing library of definitions and uses a fixed metarule set M consisting of only one of the metarules from the Metagol system [19], namely *Chain*, shown in figure 9. The chain rule allows the introduction of a relation R based on relational composition of S and T.

(b) DeepLog architecture

The overall structure of DeepLog is shown in figure 10.

(c) Meta-compilation

The positive examples E^+ are used to find a set of minimal length input–output certificates,[8] where C is a length n certificate of an example $r(a, b) \in E+$ if $r(a, b)$ is a relational instance, a and b are the logical terms and there exists a sequence of ground instances $C = \langle p_0(a, x_0), \ldots, p_n(x_n, b)\rangle$ (simplified below to $C = \langle p_0, \ldots, p_n\rangle$). $B_0 \models \alpha$ for each ground atom α in C. C has minimal length n in case there is no certificate C' for $r(a, b)$ of length n' where $n' < n$. This step is used to identify both a minimal signature of primitive relations from the library B_0 and a minimal universe of logical terms x_i to be considered as intermediate states in the certificates considered by the meta-interpretation stage. Minimal (*Min*[9]) [3] and maximal (*Max*[10]) bounds on the number of clauses in any hypothesized program are identified. Additionally, repeated subsequences of the minimal certificates are identified as potentially useful auxiliary primitives. Finally, a set of binary square matrices are computed to provide a look-up oracle for the meta-interpreter to rapidly identify minimal certificates and optimal choice points in the meta-interpreter's derivation of hypotheses.

(d) Meta-interpretation

A binary search procedure (figure 11) is used to progressively reduce the $\langle Min, \ Max\rangle$ boundaries in order to find the minimum number of clauses in any consistent hypothesis returned by the meta-interpreter. For any specific $\langle Min, \ Max\rangle$ pair the meta-interpreter's search order considers more specific hypotheses before more general ones, and returns the first consistent hypothesis. This strategy is aimed at finding low complexity hypotheses which have low generality, in accordance with the positive-only MAP selection strategy described in §3.

[6]'Deep' refers to the initial deep dive to find a consistent certificate.

[7]DeepLog code and datasets available at https://github.com/StephenMuggleton/DeepLog/.

[8]In Computability theory these are called *certificates*, in Graph theory they are called *edge-labelled paths* and in the AI planning literature they are sequential plans.

[9]*Min* is the length of the minimal number of relations in any minimal certificate identified.

[10]*Max is one less than the length of a minimal certificate.*

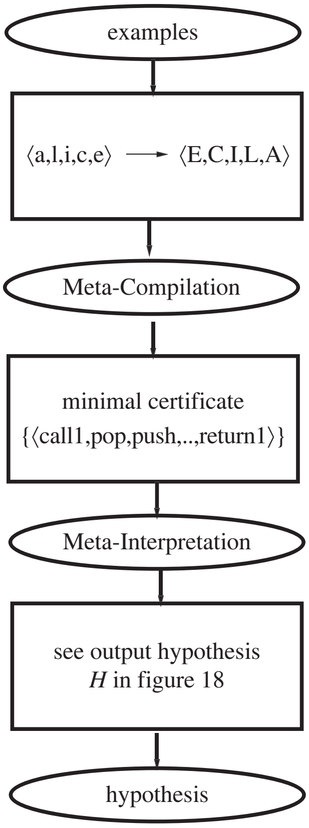

Figure 10. DeepLog stages.

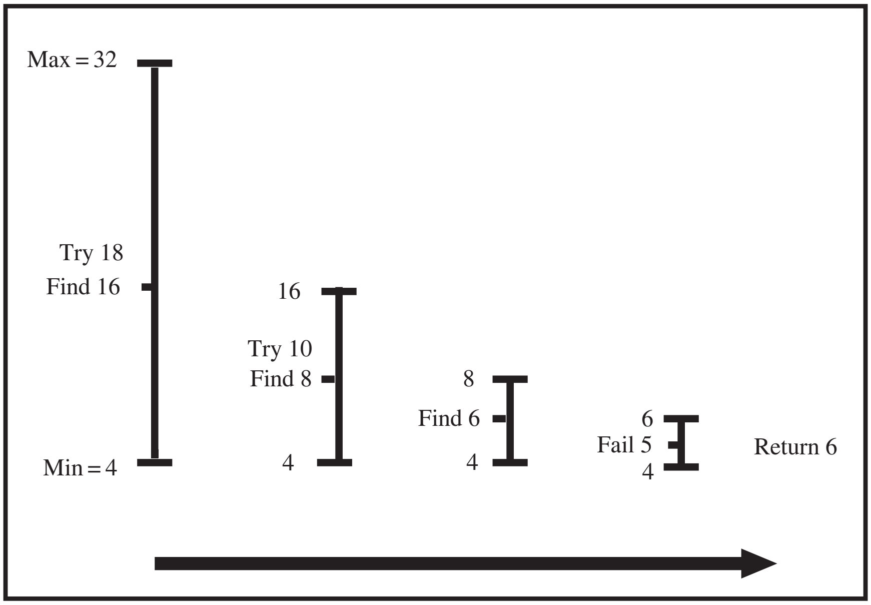

Figure 11. Binary search exponentially reduces hypothesis space to return consistent program with minimal number of clauses.

target	**example** e^+	**primitives** P
abc4	$\langle$a,b,c,d,e,f,c,d,e,f,g,h$\rangle \rightarrow \langle\rangle$	library 62 primitives

minimal certificates	$\{\langle a, b, c, d, e, f, c, d, e, f, g, h\rangle\}$

hypothesized program H **(7 clauses)**	**primitives used (8 of 62)**	**training**
abc4(X,Y) :- a(X,Z), abc4_1(Z,Y).	a([a | T],T) :- type(T,cl).	*time* = 0.15 s / 0.56 s
abc4_1(X,Y) :- b(X,Z), abc4_1_1(Z,Y).	b([b | T],T) :- type(T,cl).	
abc4_1_1(X,Y) :- g(X,Z), h(Z,Y).	c([c | T],T) :- type(T,cl).	$g(H) = \frac{1}{1080} \approx 0.0009$
abc4_1_1(X,Y) :- cdef(X,Z), abc4_1_1(Z,Y).	d([d | T],T) :- type(T,cl).	
auxiliary primitives	e([e | T],T) :- type(T,cl).	$-\ln p\,(H\|E) = 17.97$
cdef(X,Y) :- cd(X,Z), ef(Z,Y).	f([f | T],T) :- type(T,cl).	
cd(X,Y) :- c(X,Z), d(Z,Y).	g([g | T],T) :- type(T,cl).	EA $(1, T) > 99.57\%$
ef(X,Y) :- e(X,Z), f(Z,Y).	h([h | T],T) :- type(T,cl).	

Figure 12. Problem1: Example, minimal certificate, hypothesis, primitives used and training. EA is defined as $\mathrm{EA}(1,T) = 100(1-\mathrm{EE}(H))$. Primitives $a,b,\ldots,h$ remove a single character from the input to produce the output. The input and output type *cl* is a character list. The hypothesized program represents the regular grammar $ab(cdef)^*gh$

5. Experiments

In this section, the EE predictions from §3 are compared against the behaviour of DeepLog in the learning of three specific target programs. For each target program the same background library of 62 primitive relations is used. The library was developed progressively from investigating 57 different problems, and contains a set of type identifiers and basic relations over various types of entities, including characters, lists, stacks, numbers, three-dimensional positions, chess and family relations. In the problem descriptions below one-shot percentage EA is defined as $\mathrm{EA}(1,T) = 100(1 - \mathrm{EE}(m = 1|g(T)))$.

(a) Problem1: regular grammar

Consider a formal language which involves repeated letter sequences. For instance, the positive example sequence $e^+ =$'abcdefcdefgh' might be used to exemplify the target language $G =$ ab(cdef)*gh. G corresponds to letter sequences, such as e^+, which have a prefix $\langle a, b\rangle$, a suffix $\langle g, h\rangle$ and zero or more repetitions of $\langle c, d, e, f\rangle$ inbetween. Figure 12 shows e^+ as an input/output pair and DeepLog's hypothesized Logic Program, corresponding to G. This program is constructed from primitives which reduce the sequence by one letter (such as $\mathrm{a}, \mathrm{b}, \ldots$), auxiliary relations (such as $\mathrm{cdef}, \mathrm{cd}, \ldots$) composed from the primitives introduced during Meta-Compilation (figure 10), and invented relations (such as $\mathrm{abc4\_1}, \mathrm{abc4\_1\_1}, \ldots$) introduced during Meta-Interpretation (figure 10).

Training time taken on a laptop (Intel-i7/2.80 GHz) is 0.15 s for meta-compilation and 0.56 s for meta-interpretation. The generality of the hypothesized program is $g(H) \approx 0.0009$, leading to an EA of $\mathrm{EA}(1,T) > 99.57\%$ and negative log posterior of 17.97. This corresponds to the low-generality criterion for one-shot learning, shown in figure 7.

Figure 13 illustrates the way in which the value of $g(H)$ is derived from equations using SWI-Prolog's CLPR rational solver [22]. CLPR solves rational number equational constraints based on a polynomial-time solver. The equations used in figure 13 are derived by Deeplog from the definitions in the hypothesis. For instance, $w = (1/6)^2 + (w/6)$ is the sum of the generalities of the first and second clause of *abc*4_1_1.[11] The generality of the first clause is $1/6^2$ since $g(g) =$

[11]The recursive definition of *abc*4_1_1 is reflected in the infinite sum $w = ((1/6)^2 + (w/6)) = ((1/6)^2 + (1/6)^3 + (1/6)^4 + \cdots)$. CLPR efficiently solves sets of such interrelated equations to provide an exact rational solution.

abc4(X,Y) :- a(X,Z), abc4_1(Z,Y).
abc4_1(X,Y) :- b(X,Z), abc4_1_1(Z,Y).
abc4_1_1(X,Y) :- g(X,Z), h(Z,Y).
abc4_1_1(X,Y) :- cdef(X,Z), abc4_1_1(Z,Y).
	u = g(abc4)
	v = g(abc4_1)
	w = g(abc4_1_1)

$u = \frac{v}{6}, v = \frac{w}{6}, w = \frac{1}{6}^2 + \frac{w}{6}$ $\Longrightarrow u = \frac{w}{36}, w = \frac{1}{36} + \frac{w}{6}$ $\Longrightarrow \frac{5w}{6} = \frac{1}{36}$ $\Longrightarrow w = \frac{6}{180} = \frac{1}{30}$ $\Longrightarrow u = \frac{1}{1080}$	**equations for CLPR solver**

Figure 13. Calculation of $g(H)$.

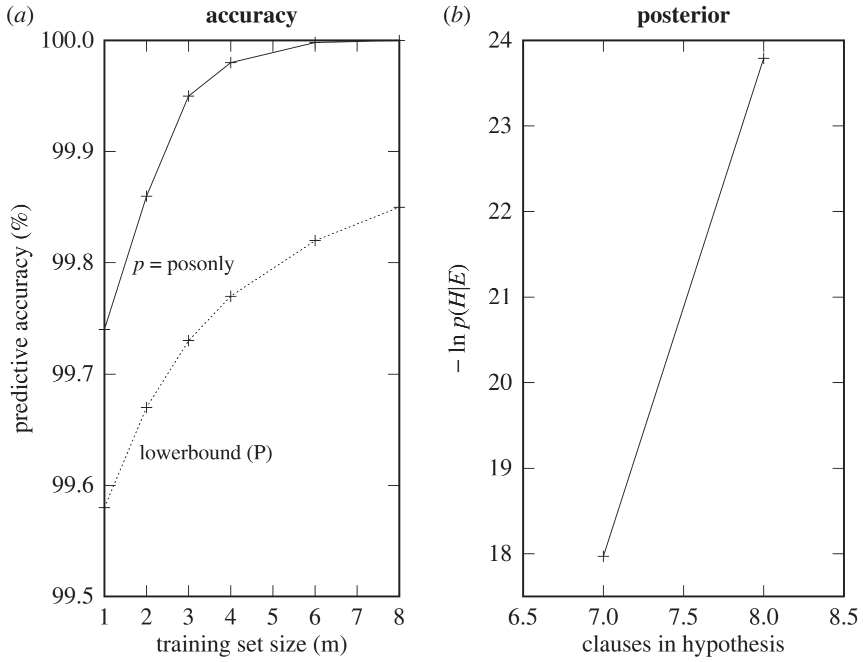

Figure 14. Problem1: (*a*) Accuracy increase and (*b*) negative log probability decrease.

$g(h) = 1/6$ and g, h are selected from the six non-invented relations in the identified program. The generality of the second clause is $w/6$ since it is the product of $g(abc\_1\_1)$ and $g(cdef)$.

Figure 14 shows (*a*) a comparison of Actual predictive accuracy for DeepLog versus the positive-only EE bound of figure 6 and (*b*) the variation of negative log posterior with decreasing clause bounds.

(b) Problem2: Fire escape plan

This problem involves a general fire escape plan for leaving a 16 storey building. The floorplan is shown in figure 15. All floors have the same layout as Floor 16, except Floor 1, which contains the EXIT. Figures 16 and 17 show the problem description and results for the fire escape problem. Figure 17*a* shows once more that the learned program has low generality leading to high predictive accuracy from the first example provided, in accordance with the predictions of

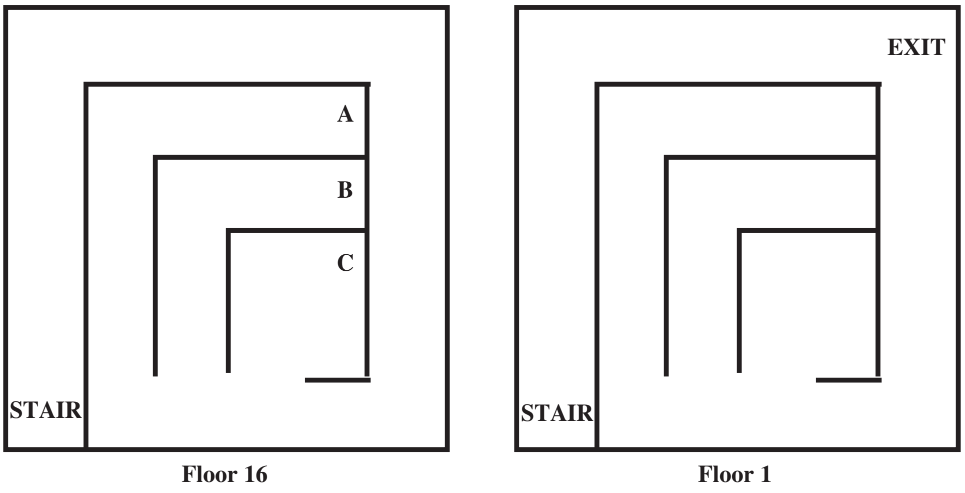

Figure 15. Problem2: Floorplan.

target	**example** e^+	**primitives** P
fire16	at(8,8,16) $\rightarrow$ at(10,10,1)	library 62 primitives

minimal certificates	$\{\langle ws, ss, es, ns, ws, ss, d, d, d, d, d, d, d, d, d, d, d, d, d, d, ns, es\rangle\}$

hypothesized program H **(7 clauses)**	**primitives (5 of 62)**	**training**
fire16(X,Y) :- ws(X,Z), fire16_1(Z,Y).	ws (X,Y) :- . . .	*time* = 0.2 s / 4.14 s
fire16_1(X,Y) :- ss(X,Z), fire16_1_1(Z,Y).	ss (X,Y) :- . . .	$g(H) = \frac{1}{1079} \approx 0.0009$
fire16_1_1(X,Y) :- ns(X,Z), es(Z,Y).	ns (X,Y) :- . . .	
fire16_1_1(X,Y) :- d(X,Z), fire16_1_1(Z,Y).	d (X,Y) :- . . .	$-\ln p(H\|E) = 32.57$
fire16_1_1(X,Y) :- es(X,Z), fire16_1_1_1(Z,Y).	es (X,Y) :- . . .	
		EA(1, T) > 99.57%
fire16_1_1_1(X,Y) :- ns(X,Z), fire16(Z,Y).		

Figure 16. Problem2: Example, hypothesis and training. The example indicates that the agent starts at coordinate $\langle 8, 8, 16\rangle$ position **A** (see figure 15) on floor 16 and needs to reach coordinate $\langle 10, 10, 1\rangle$, position **Exit** on Floor 1. Primitives *ws, ss, ns* and *es* repeatedly move the agent in a given direction until obstructed, while primitive *d* moves the agent down one floor. While the Minimal Certificate represents an optimal plan for getting to the **Exit** from **A** on floor 16, the hypothesized program will get the agent to the **Exit** from **A**, **B** or **C** from any floor of the building.

the positive-only EE bound in figure 6. Additionally, figure 17*b* indicates how two cycles of the Deeplog's binary search process (see figure 11) successively reduce the number of clauses and negative log posterior of the hypothesis.

(c) Problem3: Reverse uppercase

We now consider the 'alice' to 'ECILA' textual analogy problem introduced in §1. This is shown in figure 18, which involves the construction of a function for reversing and making uppercase a letter sequence. Efficient Prolog programs normally require the introduction of an arity 3 auxiliary predicate to reverse a list. The same end is achieved with arity 2 predicates for DeepLog by dynamically creating a stack in the program state. Since Problem3 requires such a stack to be of unbounded size, the resulting program can be thought of as a push-down automaton

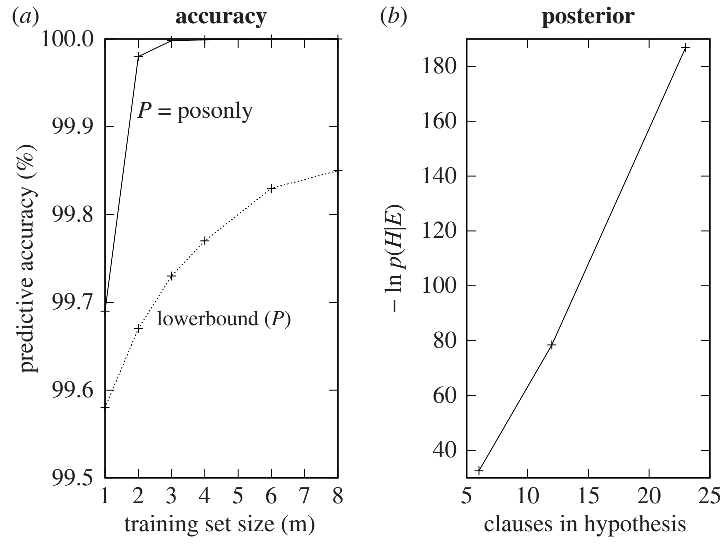

Figure 17. Problem2: (*a*) Accuracy increase and (*b*) negative log probability decrease.

(PDA) as opposed to the finite state automata (FSA) solutions in Problem1 and Problem2. PDAs represent a higher expressivity function class than FSAs since the stack can be treated as an unbounded Turing-machine tape. This additional functionality is enabled by the availability of the background library primitives *call1* (see figure 18), which introduces a new 'empty' object on the top of the calling stack. The type of the new object, in this case a list, is assigned by DeepLog using type-recognition predicates applied to the example provided. On exit this object is passed back using *ret1*. It should be noted that the resulting program would be relatively complex to encode manually, since it involves introduction of three subsidiary relations with mutual recursion and efficient interleaving of pushing, popping and uppercasing of letters. Training is achieved with EA from one example in under 1s on a laptop, and the generality and negative log probability are lower than in Problem1 and Problem2. This is reflected by more rapid accuracy and posterior convergence in figure 19*a*. Figure 19*b* shows that posterior converges is achieved in three binary search iterations (see figure 11) which reduce the hypothesis from 16 clauses down to the final 5 clause hypothesis returned.

6. Conclusion and further work

(a) One-shot learning

The paper introduces a form of textual analogy problem as an example of human ability to hypothesize a concept effectively from a single example. This phenomenon, known as one-shot learning, has been extensively studied over the last decade within both Cognitive Science and Machine Learning. However, one-shot learning has not previously been explained within CoLT, and no theoretical framework exists for analysing its EE.

(b) Bayes' model of one-shot learning

A framework for analysing EE when learning from one example is introduced in this paper based on an adaptation of the author's existing Bayesian analysis of positive-only learning. The approach enables, for the first time, an upper-bound error analysis of the human phenomenon of one-shot learning. EE bounds for one-shot learning are introduced as a special case of

target	**example e^+**	**primitives P**
rvu	$\langle$a,l,i,c,e$\rangle \rightarrow \langle$E,C,I,L,A$\rangle$	library 62 primitives

minimal certificates	$\{\langle call1, pop, up, psh, pop, up, psh, pop, up, psh, pop, up, psh, pop, up, psh, ret1\rangle\}$

output hypothesis *H* (5 clauses)	**primitives (4 of 62)**	**training**
rvu(X,Y) :- call1(X,Z), rvu_1(Z,Y).	call1(X | Z,A | (X | Z)) :- ..	*time* = 0.21 s / 0.65s
rvu_1(X,Y) :- pop(X,Z), rvu_1_1(Z,Y).	pop(Z | [X | Y],X | Z | Y) :- ..	$g(H) = \frac{1}{7740} \approx 0.0001$
rvu_1_1(X,Y) :- up(X,Z), rvu_1_1_1(Z,Y).	up(C | S,Up | S) :- ..	
rvu_1_1_1(X,Y) :- psh(X,Z), ret1(Z,Y).	psh(X | Z | Y,[X | Z] | Y) :- ..	$-\ln p(H\|E) = 24.01$
rvu_1_1_1(X,Y) :- psh(X,Z), rvu_1(Z,Y).	ret1(Y | Ident,Y) :- ..	EA(1, *T*) > 99.94%

Figure 18. Problem3: Example, hypothesis and training. The primitive *call1* pushes an empty list *A* onto the program stack as an accumulator. Primitive *pop* removes the head of the input list and pushes it onto the program stack. Primitive *psh* replaces the top two elements X,Z of the program stack by a list with head *X* and tail *Y*. Primitive *up* replaces the letter at the top of the program stack by its uppercase version. Lastly, *ret1* replaces the program stack *Y*—*Ident* by the output list *Y*.

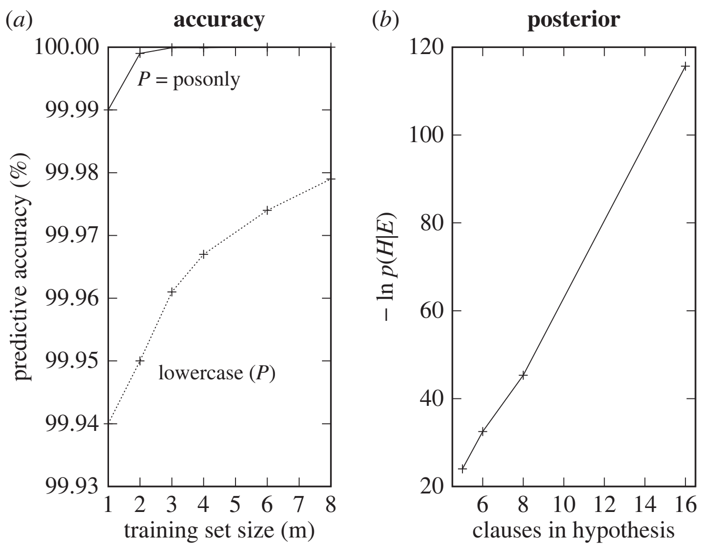

Figure 19. Problem3: (*a*) Accuracy increase and (*b*) negative log probability decrease.

Bayesian positive-only learning. The analysis indicates that the effectiveness of one-shot learning is dependent on the target theory having generality below 0.01.

(c) DeepLog experiments

A new system called DeepLog is introduced and its convergence properties are compared against the Bayesian EE bounds for one-shot and positive-only learning on three problems involving the conjecture of algorithms from one positive example. The error results for Deeplog are found to be consistent with the Bayes' model error bounds.

(d) One-shot learning in the context of science

Within the context of scientific discovery we suggest that one-shot hypotheses play the part of an initial conjecture, where subsequent examples provide either further confirmation, resulting in increased EA, or alternatively a refutation of the initial conjecture. Such initial conjectures, concerning observed phenomena, have played a vital role historically in the initial development of novel scientific theories, whether in positing Newtonian gravity or the theory of Mendelian genes. However, it seems reasonable to assume that such an ability is reliant on innate properties and abilities of human perception and reasoning.

(e) Further work

The result in figure 7 indicates high-accuracy concepts can be learned with high data efficiency when the target has low generality. If learning is used to progressively accumulate low-generality background concepts, this should lead to a reduction in search combinations when learning new concepts for which the new concepts are relevant. Future work is needed to explore this effect.

For the sake of simplicity it has been assumed that examples are (i) noise-free and (ii) background primitives are correct and complete. Situations in which these assumptions do not hold should be explored in future work.

Finally, the EE bounds given in this paper and [3] fit empirical results more tightly than worst case bounds found in PAC learning. It might be possible to find even tighter bounds, though this is likely to come from making stronger assumptions than those found in [3].

Data accessibility. The data are provided in electronic supplementary material [23].

Authors' contributions. S.H.M.F.: conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, project administration, resources, software, supervision, validation, visualization, writing—original draft, writing—review and editing.

Conflict of interest declaration. I declare I have no competing interests.

Funding. This work was supported by Engineering and Physical Sciences Research Council (grant no. EP/R022291/1).

Acknowledgements. The author acknowledges support from the UK's EPSRC Human-Like Computing Network (EP/R022291/1), for which he acts as director. The author would also like to thank his wife, Thirza, for her unfailing encouragement and willingness to always listen with care during the development of the work described.

References

1. Gold EM. 1978 Complexity of automaton identification from given data. *Inf. Control* **37**, 302–320. (doi:10.1016/S0019-9958(78)90562-4)
2. Valiant LG. 1984 A theory of the learnable. *Commun. ACM* **27**, 1134–1142. (doi:10.1145/1968.1972)
3. Muggleton SH. 1996 Learning from positive data. In *Proc. of the Sixth Int. Workshop on Inductive Logic Programming* (*Workshop-96*) (ed. SH Muggleton), *LNAI 1314, Stockholm, Sweden, 26–28 August 1996*, pp. 358–376. Berlin: Springer-Verlag.
4. Muggleton SH, Lin D, Tamaddoni-Nezhad A. 2015 Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Mach. Learn.* **100**, 49–73. (doi:10.1007/s10994-014-5471-y)
5. Lake BM, Salakhutdinov R, Gross J, Tenenbaum JB. 2011 One shot learning of simple visual concepts. In *Proc. of the 33rd Annual Conf. of the Cognitive Science Society, 20–23 July 2011, Boston, MA, USA*, pp. 2568–2573.
6. R Fergus F-FL, Perona P. 2006 One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 594–611.
7. Shaban A, Bansal S, Liu Z, Essa I, Boots B. 2017 One-shot learning for semantic segmentation. (http://arxiv.org/abs/1709.03410)
8. Tenenbaum JB. 1999 *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology.

9. Vinyals O, Blundell C, Lillicrap T, Wierstra D. 2016 Matching networks for one shot learning. *Advances in neural information processing systems, Barcelona, Spain, 5–10 December 2016*, 29. Red Hook, NY: Curran Associates.
10. Goulden R, Nation P, Read J. 1990 How large can a receptive vocabulary be?. *Appl. Linguist.* **11**, 341–363. (doi:10.1093/applin/11.4.341)
11. Anthony MHG, Biggs N. 1997 *Computational learning theory*. Cambridge, UK: Cambridge University Press.
12. Kearns MJ, Vazirani UV. 1995 Computational learning theory. *ACM SIGACT News* **26**, 43–45. (doi:10.1145/203610.606411)
13. Gold EM. 1967 Language identification in the limit. *Inf. Control* **10**, 447–474. (doi:10.1016/S0019-9958(67)91165-5)
14. McNeill D. 1971 *Acquisition of Language: Study of Developmental Psycholinguistics*. Joanna Cotler Books.
15. Cropper A, Dumančić S, Evans R, Muggleton SH. 2021 Inductive logic programming at 30. *Mach. Learn.* **111**, 147–172. (doi:10.1007/s10994-021-06089-1)
16. Muggleton SH. 1991 Inductive logic programming. *New Gener. Comput.* **8**, 295–318. (doi:10.1007/BF03037089)
17. Muggleton SH, De Raedt L. 1994 Inductive logic programming. Theory and methods. *J. Log. Programm.* **19**, 629–679. (doi:10.1016/0743-1066(94)90035-3)
18. Dai W-Z, Muggleton SH. 2021 Abductive knowledge induction from raw data. In *Proc. of the 30th Conf. on Artificial Intelligence (IJCAI 2021), Montreal, Canada, 19–27 August 2021*, pp. 1845–1851. IJCAI.
19. Muggleton SH, Lin D, Pahlavi N, Tamaddoni-Nezhad A. 2014 Meta-interpretive learning: application to grammatical inference. *Mach. Learn.* **94**, 25–49. (doi:10.1007/s10994-013-5358-3)
20. Patsantzis S, Muggleton SH. 2022 Meta-interpretive learning as metarule specialisation. *Mach. Learn.* **111**, 3703–3731. (doi:10.1007/s10994-022-06156-1)
21. Zhou Z-H. 2019 Abductive learning: towards bridging machine learning and logical reasoning. *Sci. China Inf. Sci.* **62**, 1–3.
22. Harvey W, Stuckey PJ. 1995 *A unit two variable per inequality integer constraint solver for constraint logic programming*. Melbourne, Australia: University of Melbourne.
23. Muggleton FREng SH. 2023 Hypothesizing an algorithm from one example: the role of specificity. Figshare. (doi:10.6084/m9.figshare.c.6607461)