

Assessing Exploratory Theory Formation Programs

Simon Colton

Mathematical Reasoning Group
Division of Informatics
University of Edinburgh
Edinburgh EH1 1HN
United Kingdom

Theory Formation Programs in Pure Mathematics

Broadly speaking, machine learning programs are asked to identify a single concept given a set of examples and some background knowledge. Mathematical theory formation programs, such as the AM program, (Davis & Lenat 1982) and the HR program, (Colton, Bundy, & Walsh 1999), are also given a set of examples and some background knowledge. However, they are not asked to find a single concept, but rather to explore the domain and attempt to gain some understanding of it. Because the domain is mathematics, there are a range of ways by which the program can gain an understanding of the domain, including inventing concepts, performing calculations, making conjectures, proving theorems and finding counterexamples. The HR system is able to perform all of these activities in domains such as group theory, where it employs the Otter theorem prover, (McCune 1990), and MACE counterexample finder, (McCune 1994), to prove and disprove theorems.

Whereas machine learning programs are in general goal directed, theory formation programs perform forward chaining. To control this, measures of interestingness for concepts are derived and employed so that an effective best first search through the space of concepts can be achieved. For example, measures based on the clarity of the definition of a concept are common in theory formation programs and help increase the yield of understandable concepts. Theory formation programs are not limited to just mathematics. We have recently enabled HR to work in “train theory”, in which it invents concepts which involve properties of trains as described in the original Michalski problem, (Michalski & Larson 1977). For example, it invents the concept of trains where each carriage has a different shape.

In general, the kinds of concepts which machine learning programs and theory formation programs can invent in pure mathematics are very similar, but theory formation programs have to avoid many uninteresting concepts. A good example of this is that the Progol ILP program, (Muggleton 1995),

will easily learn the integer sequence:

17, 17, 17, 17, 17, . . .

This concept is present in HR’s search space, but HR will not invent the concept because the search strategy prohibits it (because the sequence is not interesting).

Research Problem Statement

We wish to define some concrete qualitative and quantitative measures for the performance of exploratory theory formation programs.

In the next section we list three of the many subjective ways to assess a scientific theory formation program. Following this, we suggest two quantitative measures for the performance of a theory formation program, based on applying the program to machine learning tasks. Finally, we summarise with a brief assessment of the problem.

Subjective Measures

By saying that the sequence 17, 17, 17, 17, . . . is not interesting, we have been subjective. Ideas about the interestingness of individual concepts can often be turned into general rules and even concrete calculations which assess the interestingness of a concept. For example, HR chooses to develop concepts with a variety of examples, which means it avoids the above sequence because it only contains one number. Each theory formation program has its own measures for interestingness and we have surveyed five such programs in (Colton, Bundy, & Walsh 2000c), and attempted to derive some common approaches to the assessment of interestingness.

Heuristic searches based on these measures of interestingness are intended to increase the quality of the theories produced. Assessment of the theories themselves is also usually subjective, and likewise assessments of the programs which produce the theories. We list here some of the subjective ways in which theory formation programs have been assessed historically, and highlight some of the drawbacks of these assessments.

- The number of ‘classically interesting’ concepts or conjectures it re-invents in a theory. For example, AM re-invented square numbers and prime numbers, found Goldbach’s conjecture and the prime factorisation theorem, and gained great credit for doing so. Certainly, a program which re-invents no classically interesting concepts or conjectures should be assessed poorly. However, what constitutes a classically interesting concept is highly subjective, and in areas such as train theory, this assessment is not applicable. Moreover, it can encourage the authors of programs to continually fine tune their programs so that they eventually reach particular concepts. Certainly, AM was criticised for, amongst other things, the fact that some of its heuristics were very specialised and often only used once in a session, (Ritchie & Hanna 1984).
- The number of ‘genuinely interesting’ new concepts or conjectures the program invents. Unfortunately, it is usually difficult to know in advance even whether a concept is new, never mind interesting. Valdés-Pérez classifies machine discovery programs as either those developed to model theory formation or those developed to assist experts, (Valdés-Pérez 1999). An example of the latter is the Graffiti program, (Fajtlowicz 1988), which has produced many very interesting conjectures in graph theory which some of the best minds in the field have tackled. With programs which assist experts, the user can often quickly identify the importance of a concept. Programs which model theory formation play an important role in artificial intelligence and cognitive science. However, their authors are often not experts in the domain of interest and the programs are not designed to actually add to the domain. This makes assessment of the novelty and importance of the concepts produced difficult. Lenat originally believed AM had invented a new type of number: those with more divisors than any smaller number. However, these turned out to be highly composite numbers defined much earlier by Ramanujan. We had a similar experience with refactorable numbers, (Colton 1999), but have since redeemed the situation, (Colton, Bundy, & Walsh 2000b).
- The overall quality of the theory. This approach involves examining an example theory produced and grading the interestingness of the concepts in the theory. An overall score is then assigned to the theory, based on the average interestingness of the concepts. This is again subjective, and it is difficult to tell which concepts are genuinely interesting. This is a good measure to use while developing a theory formation program: examine the output and alter the program so that the yield of interesting concepts increases (based on the subjective notions of interesting being used). If this can be done in a general way, the program will benefit greatly from such an analysis. For example, if a program was consistently producing concepts with very complicated definitions, we could define a clarity measure and encourage the production of concepts with greater clarity. If we develop a program in this way, we will require different notions of interestingness to assess the program, as it will obviously score well using the criteria employed to assess the theory while developing it.

Machine learning programs perform in a reactive way. They are given a set of examples, some background knowledge and a categorisation of the examples. They are then asked to use the background information to define a concept which achieves the categorisation. For example, we could give the ILP program Progol, (Muggleton 1995), the set of integers up to 10 and background information which includes the concept of divisors of integers. If we then supply this categorisation of the integers: [1, 3, 5, 7, 9], [2, 4, 6, 8, 10], Progol will learn the concept of even numbers (ie. divisible by 2) because when we use this concept to categorise the integers 1 to 10 into even numbers and non-even numbers, the categorisation is above, as required.

If we now imagine a more pro-active approach to machine learning, where the program is given a set of examples and some background knowledge, and we know that after some given time limit, it is going to be given a categorisation of the examples which it must learn a concept for. In the interim, the program can do anything which will enable it to provide an answer quickly when the categorisation is finally presented. For example, we could give the program a set of trains and background knowledge in terms of describing the trains, eg. carriages, objects in carriages, wheels, etc. We then give it five minutes to prepare, and return to ask it why a particular set of trains are going East. If the program has used its time effectively, it would reply immediately, for example: the trains going East have exactly four carriages. If the program has been completely effective with its preparation time, we will be able to give it any categorisation of the trains, for example into trains going North, South, East and West, and it will be able to supply a concept immediately which explains their behaviour.

This imaginary pro-active experiment describes the function of a theory formation program. This enables us to define a measure for the success of the program. We could simply count the number of concepts the program comes up with, but an inferior program might produce thousands of concepts which categorise the trains in only a few ways. Therefore, we are actually interested in the number of different categorisations which are achieved by the set of concepts a theory formation program comes up within a given time limit. If a program comes up with 100 categorisations of, say 8 example trains, after a given time limit, it is more equipped to carry out machine learning tasks than a program which only manages to come up with 50 categorisations.

Note that the number of different ways to categorise a set of n objects is defined as the n th Bell number, (Bell 1934). The Bell numbers are: 1, 2, 5, 15, 52, 203, 877, ... For example, there are 15 ways to categorise the numbers 1 to 4:

[1, 2, 3, 4]	[1, 2, 3], [4]	[1, 2, 4], [3]
[1, 2], [3, 4]	[1, 2], [3], [4]	[1, 3, 4], [2]
[1, 3], [2, 4]	[1, 3], [2], [4]	[1, 4], [2, 3]
[1, 4], [2], [3]	[1], [2, 3, 4]	[1], [2, 3], [4]
[1], [2, 4], [3]	[1], [2], [3, 4]	[1], [2], [3], [4]

We have performed some experiments with HR in number theory and found that it can produce concepts categorising the integers 1 to 5 in all 52 ways. Similarly, we have achieved around 90% of the 203 categorisations of the integers 1 to 6. Note that we report an increase in the number of categorisations HR finds in a given time limit when a cooperative agency, rather than a single copy of HR is employed, (Colton, Bundy, & Walsh 2000a).

A measure of the interestingness of concepts which we can project to measure the entire theory, is the clarity of the concepts. If a program has a clarity measure for individual concepts, then the average clarity of a concept in the theory can be used as a measure for the theory itself. Even if a clarity measure is not used in the heuristic search, it should still be possible to derive some measure of the complexity of the definitions of the concepts produced.

Assessment of the Research Problem

We have so far employed a variety of measures to assess the HR program in the hope that the combination will prove sufficient [the ‘shotgun’ approach, (Bundy 1998)]. Some of these measures are subjective and some are more concrete. The research problem we state here is to derive some additional qualitative and quantitative measures for theory formation programs, specifically those which work in scientific domains. In particular, we do not yet have an objective way to measure the average quality of the concepts produced in a session. We hope to adapt some of the ideas developed in (Sommer 1996) and similar papers to help assess the theories produced by exploratory theory formation.

Such concrete measures will not only allow us to assess the performance of a program, but to determine whether particular heuristics improve its performance, which was an initial motivation for this problem. Further, we hope to use such measures to compare and contrast theory formation programs. Investigation of why programs perform badly with respect to these measures will allow future developers of theory formation programs to implement better systems.

There are many ways to exhibit an understanding of a domain. We have presented one based on machine learning tasks: an ability to immediately explain categorisations presented. For example, HR has completely categorised the numbers 1 to 5, so we can ask it to provide a property shared by any two of them which the others do not share, and it will always supply an answer immediately. Another way to exhibit an understanding of the domain is to present an intelligent question. In mathematics, this entails proposing a conjecture which is simple to state, difficult (but not seemingly impossible) to prove, and certainly not obvious. Perhaps the first step towards defining measures is therefore to determine the ways in which an understanding of the domain can be communicated by a theory formation program. From this, we may be able to measure how much it really understands the domain it has explored.

Acknowledgements

This work is supported by EPSRC grant GR/M98012.

References

- Bell, E. 1934. Exponential numbers. *American Mathematics Monthly* 41:411–419.
- Bundy, A. 1998. *Personal communication*.
- Colton, S.; Bundy, A.; and Walsh, T. 1999. HR: Automatic concept formation in pure mathematics. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.
- Colton, S.; Bundy, A.; and Walsh, T. 2000a. Agent based cooperative theory formation in pure mathematics. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*.
- Colton, S.; Bundy, A.; and Walsh, T. 2000b. Automatic invention of integer sequences. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.
- Colton, S.; Bundy, A.; and Walsh, T. 2000c. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies* Forthcoming.
- Colton, S. 1999. Refactorable numbers - a machine invention. *Journal of Integer Sequences* 2.
- Davis, R., and Lenat, D. 1982. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill Advanced Computer Science Series.
- Fajtlowicz, S. 1988. On conjectures of Graffiti. *Discrete Mathematics* 72 23:113–118.
- McCune, W. 1990. The OTTER user’s guide. Technical Report ANL/90/9, Argonne National Laboratories.
- McCune, W. 1994. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories.
- Michalski, R., and Larson, J. 1977. Inductive inference of VL decision rules. In *Proceedings of the Workshop in Pattern-Directed Inference Systems (Published in SIGART Newsletter ACM, No. 63)*.
- Muggleton, S. 1995. Inverse entailment and Progol. *New Generation Computing* 13:245–286.
- Ritchie, G., and Hanna, F. 1984. AM: A case study in methodology. *Artificial Intelligence* 23.
- Sommer, E. 1996. An approach to measuring theory quality. In *Proceedings of the 9th European Knowledge Acquisition Workshop*.
- Valdés-Pérez, R. 1999. Principles of human computer collaboration for knowledge discovery in science. *Artificial Intelligence* 107(2).