Learning and Exploration in Automated Theorem Proving

Moa Johansson
Chalmers University of Technology

Abstract

This note describes a new project with the purpose of combining the advantages of statistical machine learning with those of symbolic methods for lemma discovery, such as theory exploration. We can thus better exploit the resources in large electronic proof libraries to gain heuristic knowledge to, for example, create more efficient lemma discovery techniques or provide hints as to which induction scheme to try in an automated prover.

1 Introduction and Motivation

Machine learning and pattern recognition techniques are statistical methods which can be used for discovering similarities in data. Statistical methods are well suited for fast processing of big libraries and are generally tolerant to noise. While such statistical methods focus on extracting information from large volumes of data, they lack in capacity for conceptualisation. They can, for example, tell that a correlation between two proof patterns exists, but not why this happens nor formulate any conceptual proof hints for similar situations. Theory exploration techniques on the other hand, are concerned with exactly that: the discovery of new lemmas in new theories [5, 11, 1, 6]. However, theory exploration techniques may not cope well with very large theories when the search space becomes too big. We thus want to combine these two techniques exploiting the strengths of both.

For example, suppose we try to prove a new conjecture. We can use pattern recognition and data mining to detect if some similar theorem has been proved before, and if so, extract information about which proof technique and which lemmas were used. This information can then be fed into a theory exploration system to restrict its search space and produce suggestion of, for example, analogous lemmas applicable to the new conjecture. We have done some preliminary work in this area for the ACL2 proof assistant [4]. We used *clustering methods* to decide which theorems and conjectures were similar to each other based on their term structure, which functions were involved, and how similar the definitions of those functions were. In the context of inductive theorem proving, information about proofs of similar conjectures can help the prover to for example make a choice as to which induction scheme to use.

2 Machine Learning in Automated Reasoning

The main application of machine learning methods in the context of theorem proving and automated reasoning has been *premise selection* for first-order provers in large theories with many background facts. If presenting such a first-order prover with too many premises, its speed of operation will deteriorate. If given too few, the problem at hand might become unsolvable. Premise selection methods using machine learning techniques can help by using information from previous proofs to find correlations between theorem statements and the facts used in their proofs. When presented with a new conjecture, a suitably sized subset of premises most likely to be useful are passed to the prover. This has been implemented in for instance the provers E and SPASS [3, 9, 13]. The interactive proof assistant Isabelle allows calling external provers through the Sledghammer tool [12]. As Isabelles's proof library is very large, Sledgehammer uses this kind of relevance filtering to select facts estimated to be most relevant from the entire Isabelle library. The relevance filter is implemented with a naive Bayes learning algorithm [8]. The most commonly used learning algorithms for automated theorem proving integration seem to be fast but simple algorithms like naive Bayes and k-nearest neighbours. In a theorem prover, particularly interactive ones like Isabelle, the learning of new facts should interfere as little as possible with the user experience, which is presumably why these fast algorithms have been preferred.

The success and efficiency of machine learning algorithms is dependent on among other things *feature selection*, in our context: what characteristics of a conjecture or theorem we should focus on when trying

to assess their similarity. These features are typically translated into numerical values, and the resulting vector or matrix is used for learning. Different machine learning algorithms may work better or worse with different kinds of features, different numbers of features and so on. The most common features used in the context of theorem proving is simply which symbols occur the statements. However, many other features has been explored in various works , such as types, subterms, information about theories in which the statement occurs, just to name a few (see [7] for an overview). Recently, Kalinszyk et al. proposed to also use features based on matching and unification [7], thus basically relating statements through their generalisations. If such a generalisation is not present in the corpus, they propose to heuristically introduce such terms. They conclude that statements that have a common generalisations often have similar proofs, and show that this can improve premise selection.

3 Learning for Lemma Discovery and Inductive Proofs

We are initially interested in two areas where we believe that machine learning techniques further can assist automated theorem provers: selection of induction schemes and restricting the search space for theory exploration.

3.1 Selecting Induction Schemes

We recently added support for *recursion induction*[1] in our theory exploration system Hipster for Isabelle/HOL [10]. We found that certain properties that Hipster could not previously prove with structural induction were possible to automate with a simple switch to recursion induction. The prover now has a choice as to which induction to apply, structural- or recursion induction. While is power is increased, the search space becomes larger and the prover is sometimes slower. The choice of induction scheme should depend on which functions are present in the conjecture, and how they are defined. We expect that machine learning can help us choosing which kind of induction to try first:

1. A new conjecture that involves functions which often has occurred in other theorems whose proof required e.g. recursion induction are likely to also require recursion induction.
2. Given a new function f, and suppose its definition is judged similar to an existing function, g. Suppose further that many theorems involving the function g use a particular induction scheme. It may then make sense to try the same induction scheme also for conjectures about f.

The features required for realising (1) should simply be the standard symbol occurrence features used in for instance Sledgehammer. For (2) we need features also capturing the similarities in the structure of a function definition, in particular the functions recursive structure.

3.2 Theory Exploration

The problem we are interested in is different from premise selection for first-order provers. We want to be able to construct missing interesting lemmas, rather than just selecting from a given corpus, using *theory exploration*. A theory exploration system is typically given a set of functions, datatypes and constants and generates candidate terms using random testing, followed by automated proofs. Only theorems with non-trivial proofs are deemed interesting and presented to the user. Like automated first order provers, the performance of theory exploration systems can degrade if given too many functions as input at once (as a rough estimate, more than 20-25 can be problematic). Some areas where we think machine learning can help theory exploration include:

1. Discovering lemmas by analogy from similar proofs.
2. Speculating new properties by analogy to properties about similar functions.
3. Discovering conjectures bridging theories between similar datatypes.

For (1), we expect to follow a similar approach taken in our previous work [4], where we cluster new conjectures with existing theorems and their proofs, using features based on term structure and the similarity of functions appearing in each. The hypothesis is that lemmas used in the proof of a similar theorem can be used to extract an *analogous lemma* for our new conjecture. An option here is to abstract over the structure of an existing lemma, creating one or several schemas, which can then be given to a theory exploration system. Search will be required at this stage, but the search space is reduced compared to searching through all possible terms. Of interest here are also techniques similar to those suggested

[1] Induction following the termination order of a function in the conjecture, rather than structural induction over datatypes.

by Kalinszyk et al. [7], if we detect several theorems in the training data have a common generalisation, such lemmas could be speculated and proofs attempted.

For (2), we expect to cluster similar function definitions together, with the hypothesis that if a function f is similar to a function g, and we know some properties about f, then it is worth exploring if g has some similar properties. Of course, this will not always be the case, but at least gives some guidance as to which properties *might* hold about a new function, thus reducing the search space.

For (3), we suppose that we have two theories about two datatypes and functions on these, and furthermore, that some properties have been proved over the two datatypes. If many of these properties are similar, it is worth speculating some *bridging lemmas* to connect the two theories, for instance converting one datatype into the other.

3.3 Inductive Proofs to Learn From

Good training data is crucial for machine learning. In addition to the existing libraries for interactive provers such as Isabelle, we have started to collect a set of benchmarks specifically for inductive provers [2]. The TIP (Tons of Inductive Problems) benchmark suite is expressed in a syntax which is an extension of SMT-LIB and currently contains a few hundred problems (we invite anyone interested to submit additional benchmarks to us). We plan to further complement this format with support for expressing at least some high-level description of proof-plans, e.g. what induction scheme and what lemmas the prover has used. This would be very useful for future machine learning applications.

References

[1] K. Claessen, M. Johansson, D. Rosén, and N. Smallbone. Automating inductive proofs using theory exploration. In *Proceedings of the Conference on Auomtated Deduction (CADE)*, 2013.

[2] K. Claessen, M. Johansson, D. Rosén, and N. Smallbone. TIP: Tons of inductive problems. In *Proceedings of the Conference on Intelligent Computer Mathematics (CICM)*, 2015.

[3] J. Denzinger and S. Schulz. Automatic Acquisition of Search Control Knowledge from Multiple Proof Attempts. *Inform. and Comput.*, 162(1-2):59–79, 2000.

[4] J. Heras, E. Komendantskaya, M. Johansson, and E. Maclean. Proof-pattern recognition and lemma discovery in ACL2. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 8312 of *LNCS*, pages 389–406. Springer, 2013.

[5] M. Johansson, L. Dixon, and A. Bundy. Conjecture synthesis for inductive theories. *Journal of Automated Reasoning*, 47(3):251–289, 2011.

[6] M. Johansson, D. Rosén, N. Smallbone, and K. Claessen. Hipster: Integrating theory exploration in a proof assistant. In *Proceedings of the Conference on Intelligent Computer Mathematics (CICM)*, 2014.

[7] C. Kaliszyk, J. Urban, and J. Vyskocil. Efficient semantic features for automated reasoning over large theories. In *Proceedings of IJCAI 2015*, pages 3084 – 3090. AAAI, 2015.

[8] D. Kühlwein, J. C. Blanchette, C. Kaliszyk, and J. Urban. MaSh: Machine Learning for Sledgehammer. In *Proceedings of ITP'13*, LNCS, 2013.

[9] D. Kühlwein, T. van Laarhoven, and T. H. E. Tsivtsivadze, J. Urban. Overview and evaluation of premise selection techniques for large theory mathematics. In *IJCAR'12*, volume 7364 of *LNCS*, pages 378–392, 2012.

[10] I. Lobo-Valbuena and M. Johansson. Conditional lemma discovery and recursion induction in Hipster. In *15th International Workshop on Automated Verification of Critical Systems (AVoCS)*, 2015.

[11] O. Montano-Rivas, R. McCasland, L. Dixon, and A. Bundy. Scheme-based theorem discovery and concept invention. *Expert Systems with Applications*, 39(2):1637–1646, Feb. 2012.

[12] L. Paulson and J. Blanchette. Three years of experience with Sledgehammer, a practical link between automation and interactive theorem provers. In *Proceedings of IWIL-2010*, 2010.

[13] J. Urban, G. Sutcliffe, P. Pudlak, and J. Vyskocil. MaLARea SG1 - Machine learner for automated reasoning with semantic guidance. In *IJCAR'08*, volume 5195 of *LNCS*, pages 441–456, 2008.