Report from Dagstuhl Seminar 17351

Machine Learning and Formal Methods

Edited by

Sanjit A. Seshia[1], Xiaojin (Jerry) Zhu[2], Andreas Krause[3], and Susmit Jha[4]

1 **University of California, Berkeley** sseshia@eecs.berkeley.edu
2 **University of Wisconsin, Madison** jerryzhu@wisc.edu
3 **ETH Zürich** krausea@ethz.ch
3 **SRI International** susmit.jha@sri.com

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17351 "Machine Learning and Formal Methods". The seminar brought together practitioners and reseachers in machine learning and related areas (such as robotics) with those working in formal methods and related areas (such as programming languages and control theory). The meeting highlighted the connections between the two disciplines, and created new links between the two research communities.

Seminar August 28–September 01, 2017 – http://www.dagstuhl.de/17351
1998 ACM Subject Classification B.3.3 TBD Performance Analysis and Design Aids, B.5.1 TBD Design, C.1.2 TBD Multiple Data Stream Architectures (Multiprocessors), D.1.3 TBD Concurrent Programming
Keywords and phrases Formal Methods, Machine Learning
Digital Object Identifier 10.4230/DagRep.7.8.55

1 Executive Summary

Sanjit A. Seshia (University of California, Berkeley,
Xiaojin (Jerry) Zhu (University of Wisconsin, Madison)
Andreas Krause (ETH Zürich)
Susmit Jha (SRI International)

The seminar was successful in bringing the following two communities together:

- The community that works on machine learning (ML), both on theoretical topics and on applications to areas such as robotics and cyber-physical systems, and
- The community that works on formal methods (FM), both on computational proof techniques and on applications to formal verification and program/controller synthesis.

Both communities have long and vibrant histories, with associated conferences and journals. However, they have rarely intersected. The machine learning community has traditionally focused on *inductive* learning from data, with the data set considered as partial (potentially noisy) observations of some phenomenon. The formal methods community has traditionally emphasized automated *deduction*, e.g., using theorem proving or model checking, as a core reasoning method, with a heavy emphasis placed on formal models and proofs of correctness using those models. However, recent ideas and methods have appeared that demonstrate new connections between the two disciplines, which suggested that the time is ripe for a meeting

to promote cross-fertilization between the areas at a deep technical level. This seminar has been a significant step forward to bring the two communities together.

More concretely, the Seminar and the interaction it facilitated has brought three kinds of benefits. First, formal methods can benefit from a more effective use of machine learning techniques particularly in the context of automated synthesis. Similarly, the increasing use of machine learning in applications that require a high level of assurance points to the need for integration with formal methods. However, the potential synergies between the two areas go beyond a simple application of the techniques in one area to the other area. Importantly, there is new fundamental science to be explored in the intersection of machine learning and formal methods, related to the the confluence of inductive and deductive reasoning, and which can inform a range of new industrially-relevant applications as well.

The seminar had about 40 participants from both the FM and ML communities. The organizers took several steps to foster discussion and cross-pollination of ideas between the two communities, including the following:

- The seminar began with a day of tutorials: a half-day tutorial on Machine Learning for Formal Methods participants, and a half-day tutorial on Formal Methods for a Machine Learning audience. These tutorials helped to establish a common vocabulary to discuss ideas, problems and solutions.
- Sessions were organized based on themes that emerged in discussions before the seminar and during the first day. The list of session topics is as follows:
  1. Probabilistic Programming
  2. Teaching and Oracle-Guided Synthesis
  3. Safe Learning-Based Control
  4. Probabilistic Program Analysis
  5. Adversarial Analysis and Repair for Machine Learning
  6. Inductive Synthesis and Learning
  7. Machine Learning for Theorem Proving and Optimization
  8. Explainable and Interpretable Machine Learning
  9. Deep Learning and Verification/Synthesis

  In organizing these sessions, the organizers tried to combine speakers from both ML and FM areas to foster discussion and comparison of approaches.
- Seating arrangements at meals were organized so that (a) each table had an approximately equal number of participants from both communities, and (b) the seating was randomly changed from meal to meal.
- A joint session was organized with the concurrent seminar on analysis and synthesis of floating-point programs. This session had a panel discussion on floating-point issues in machine learning programs.

After the seminar, we have heard positive feedback from multiple participants. One told us that he started a new research project as a direct result of the seminar. A group of participants are planning to continue the interaction via joint workshops at major venues of both communities such as CAV, PLDI, ICML, NIPS, etc.

2 Table of Contents

3 Overview of Talks

3.1 Reachability and regularity for Markov chains

S Akshay (IIT, Bombay – Mumbai, India)

The dynamic behavior of a Markov chain – a basic model for probabilistic systems – can be described using sequences of probability distributions starting from an initial set of distributions. Our goal is to study the patterns and trajectories formed by such sequences during the evolution of a Markov chain over time.

A basic reachability problem which lies at the heart of this question is: does there exist an integer n such that the probability to reach in n steps from a given state to another in a Markov chain is exactly some pre-specified value r. Surprisingly, it turns out that this problem is already as hard as the Skolem Problem: a number-theoretic decision problem whose decidability has been open for decades. We look at some approximate variants as well as restrictions which allow us to reason about the trajectories in an automata-theoretic framework. If time permits, we will also draw links between these problems and classical problems such as program termination, as well as explore links to POMDP setting.

3.2 A Non-monotonic Theory of Oracle-guided Inductive Synthesis

Dalal Alrajeh (Imperial College London, UK, dalal.alrajeh@ic.ac.uk)

Specifications provide significant aid in the formal analysis of software supporting tasks such as consistency management, system verification, program synthesis, program repair and software maintenance. However writing such specifications is difficult and time-consuming. Several approaches have been proposed for automatically generating complete specifications from abstract descriptions (such as UML diagrams and user requirements) which mainly differ in their method of computation, e.g., refinement operators [7], patterns library [6]. Recent years have seen the emergence of techniques based on inductive learning, for instance [4]. The input to these techniques are samples classified as either positive or negative examples. The aim is to learn a specification that is consistent with all positive examples and inconsistent with all the negative ones. However, the quality of a learnt specification (and its proximity to the target specification) heavily depends on the quality of the samples provided to the learner. Oracle-guided inductive synthesis (OGIS) is a class of approaches that restrict the set of samples for learning to those directly relevant to some target specification [5]. The learner can query an oracle (e.g., a user or verifier) for examples. The oracle in return responds with positive or negative example that is intended to guide the learner's search for candidate specifications. The oracle is also tasked with determining whether the learner has found the correct specification. Each time a negative (resp. positive) example that is consistent (resp. inconsistent) with the current candidate specification is given, the learner proceeds in one of the following directions: (1) the candidate specification is discarded and a new one is synthesized from the set of examples accumulated thus far; or (2) an additional specification is synthesized from the new example and added to the previous ones; we say in

the second case a specification has been refined. Both directions may lead to learning a correct specification (consistent with all positive examples but none of the negatives). We argue that the former, from a practical perspective, requires users to abandon any development activities they may have started based on the earlier specification. From a conceptual view, it violates the principle of elaboration tolerance [2, 8]. The applicability of the latter, however, depends on the kind of inference allowed by the learner. Typical OGIS instances that follow this direction assume monotonicity of the synthesis procedure. By this, it is guaranteed that candidate specifications generated from new examples are consistent with those of previous iterations. However, this guarantee does not hold in the case of non-monotonic learners. In this work, we conduct a formal investigation into properties of oracleguided inductive synthesis for specification refinement by examining the impact of using two types of learners: monotonic and non-monotonic [3]. In monotonic learning, inferences cannot be invalidated simply by adding new expressions to a specification. Non-monotonic learning, on the other hand, allows inferences to be made provisionally which can be retracted as new information becomes available and thus specifications are extended. Our investigation seeks to answer the following questions: Does the quality of a specification improve with a non-monotonic learner? What are the termination guarantees with nonmonotonic learner? In cases where termination is guaranteed, does the use of a non-monotonic learner improve the speed of termination? In answering these, we assume an oracle that defines a fixed ordering over the examples generated and seek to understand the influence of the following factors: (i) the types of queries submitted to an oracle (e.g., correctness and whether they provide both positive and negative examples or positive witness only); and (ii) the resources available to the learner: finite versus infinite memory. We direct our analysis to a particular instance of OGIS for synthesizing target specifications in Linear Temporal Logic (LTL) [9]. The quality of a specification is measured in terms of: the size of the formulas, and the size of the language defined by the formulas. For monotonic learning, we have developed a monotonic learner that can compute properties in tight Signal Temporal Logic (a flavor of LTL over continuous signals) from positive examples only. As for non-monotonic learning, we consider the approach described in [1] which provides a transformation function from a subclass of LTL to a non-monotonic logic and vice-versa.

References

1 Alrajeh, D., Ray, O., Russo, A., Uchitel, S.: Using abduction and induction for operational requirements elaboration. J. Applied Logic 7(3), 275–288 (2009)

2 Baral, C., Zhao, J.: Non-monotonic temporal logics for goal specification. In: Proceedings of IJCAI07,. pp. 236–242 (2007)

3 Frankish, K.: Non-monotonic inference. In: Barber, A. (ed.) Encyclopedia of Language and Linguistics. Elsevier (2005)

4 Gehr, T., Dimitrov, D., Vechev, M.: Learning commutativity specifications. In: Proceedings of CAV15. pp. 307–323 (2015)

5 Jha, S., Seshia, S.: A theory of formal synthesis via inductive learning. Acta Informatica (2017)

6 van Lamsweerde, A.: Requirements Engineering – From System Goals to UML Models to Software Specifications. Wiley (2009)

7 Li, F.L., Horkoff, J., Borgida, A., Guizzardi, G., Liu, L., Mylopoulos, J.: From stakeholder requirements to formal specifications through refinement. In: Proceedings of REFSQ15 (2015)

8 McCarthy, J.: The artificial intelligence debate: False starts, real foundations. chap. Mathematical Logic in Artificial Intelligence, pp. 297–311. MIT Press (1988)

9 Pnueli, A.: The temporal logic of programs. In: Proceedings of SFCS77. pp. 46–57 (1977)

3.3 Semantic Mapping and Mission Planning in Robotics

Nikolay A. Atanasov (University of California at San Diego, US)

Recent years have seen impressive progress in robot perception leading to real-time super-human object recognition. Surprisingly, however, most existing approaches to simultaneous localization and mapping (SLAM) in robotics rely on low-level geometric features and do not take advantage of semantic information provided by object recognition methods. In this talk, I will address the semantic SLAM problem which utilizes object identity information for loop closure and construction of meaningful maps. A major contribution of our approach is in proving that the complexity of incorporating probabilistic data association is equivalent to computing the permanent of a suitable matrix. The resulting probabilistic semantic maps allow us to specify complex robot missions as temporal logic constraints over objects in the environment.

3.4 Safe learning of regions of attractions

Felix Berkenkamp (ETH Zürich, CH)

Reinforcement learning is a powerful paradigm for learning optimal policies from experimental data. However, to find optimal policies, most reinforcement learning algorithms explore all possible actions, which may be harmful for real-world systems. As a consequence, learning algorithms are rarely applied on safety-critical systems in the real world. In this paper, we present a learning algorithm that explicitly considers safety in terms of stability guarantees. Specifically, we extend control theoretic results on Lyapunov stability verification and show how to use statistical models of the dynamics to obtain high-performance control policies with provable stability certificates. Moreover, under additional regularity assumptions in terms of a Gaussian process prior, we prove that one can effectively and safely collect data in order to learn about the dynamics and thus both improve control performance and expand the safe region of the state space. In our experiments, we show how the resulting algorithm can safely optimize a neural network policy on a simulated inverted pendulum, without the pendulum ever falling down.

3.5 Polynomial Inference of Equational Theories

Ben Caulfield (University of California – Berkeley, US)

Equational logic is a formalism used to describe infinite sets of equations between terms (theories) using finite sets of equations (presentations). Both functional programs and logic programs can be naturally represented as equational logic presentations. Therefore, learning presentations in equational logic can be seen as a form of program synthesis. In general, it is

undecidable whether terms are equivalent via a finite presentation. So learning equational presentations is generally intractable.

We investigate the exact learning of a restricted class of theories known as non-collapsing shallow theories, which can be presented by equations where variables only appear at depth one. The learning algorithms use examples and queries of equations between ground terms, meaning there are no variables in the equations. It is shown that these theories cannot be learned in the limit from only positive examples. A polynomial time algorithm is given which creates a hypothesis presentation consistent with positive and negative examples which will learn in the limit a presentation for the target theory. Finally, an algorithm is given which learns a presentation in polynomial time from a minimally adequate teacher. It is shown that the learned presentations are canonical with respect to an ordering on terms and the presentation is at most polynomially larger than the minimal presentation of the same theory.

3.6 Constraint Learning and Dynamic Probabilistic Programming

Luc De Raedt (KU Leuven, BE)

The talk focussed on two issues. The first was the synthesis of a set of constraints that hold in tabular data (say a set of excel tables). The second was concerned with the use of hybrid probabilistic programs in dynamic domains and its applications in simple robotics and planning settings.

More details on our probabilistic programming language Problog can be found at https://dtai.cs.kuleuven.be/problog/.

3.7 Logical Clustering for Time-Series Data

Jyotirmoy Deshmukh (USC – Los Angeles, US)

Techniques for unsupervised learning for signals (time-series data) typically use distance metrics on the signal space to perform clustering. Clusters obtained in this fashion group qualitatively similar signal shapes, but may not respect logical properties of signals and may not be easy to interpret and explain. We propose a new paradigm of logical clustering, where we use parametric temporal logic formulas as a feature extraction mechanism, and then use off-the-shelf machine learning tools to automatically cluster similar traces with respect to the "projection" of the traces in the parameter-space. We demonstrate how this technique produces interpretable formulas that are amenable to analysis and understanding using a few representative examples.

3.8 Query Learning of Regular Languages and Richer Formalisms

Dana Fisman (Ben Gurion University – Beer Sheva, IL)

We review results on Angluin style query learning of automata. In particular for the concept classes of regular languages (via succinct representations such as alternating finite automata), regular omega-languages, regular omega-tree-languages, weighted languages, I/O languages, and languages over infinite alphabets.

3.9 Learning Invariants using Decision Trees and Implication Counterexamples

Pranav Garg (Amazon – Bangalore, IN)

We introduce ICE, a robust learning paradigm for synthesizing invariants, that learns using positive, negative and implication counter-examples, and show that it admits honest teachers and strongly convergent mechanisms for invariant synthesis. We propose the first learning algorithms in this model with implication counter-examples that are based on machine learning techniques. In particular, we extend classical decision-tree learning algorithms in machine learning to handle implication samples, building new scalable ways to construct small decision trees using statistical measures. We also develop a decision-tree learning algorithm in this model that is guaranteed to converge to the right concept (invariant) if one exists. We implement the learners and an appropriate teacher, and show that the resulting invariant synthesis is efficient and convergent for a large suite of programs.

3.10 Explaining AI Decisions Using Sparse Boolean Formula

Susmit Jha (SRI International – Menlo Park, CA, susmit.jha@sri.com)

This talk describes the problem of learning Boolean formulae from examples obtained by actively querying an oracle that can label examples as positive or negative. This problem has received attention in machine learning as well as formal methods, and it has been shown to have exponential worst-case complexity in the general case as well as for many restrictions. In this talk, we focus on learning sparse Boolean formulae which depend on only a small (but unknown) subset of the overall vocabulary of atomic propositions. We propose an efficient algorithm to learn these sparse Boolean formulae with a given confidence. This assumption of sparsity is motivated by the problem of mining explanations for decisions made by artificially intelligent algorithms, where the explanation of individual decisions may depend on a small but unknown subset of all the inputs to the algorithm. We demonstrate the use of proposed learning algorithm to automatically generate explanations of these decisions. These explanations will make intelligent systems more understandable and accountable to

human users, facilitate easier audits and provide diagnostic information in the case of failure. The proposed approach treats the AI algorithm as a black-box oracle, and is therefore, broadly applicable and agnostic to the specific AI algorithm. We illustrate the practical effectiveness of our approach on a set of diverse case-studies.

3.11 Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation

Matthias Hein (Universität des Saarlandes, DE)

Recent work has shown that state-of-the-art classifiers are quite brittle, in the sense that a small adversarial change of an originally with high confidence correctly classified input leads to a wrong classification again with high confidence. This raises concerns that such classifiers are vulnerable to attacks and calls into question their usage in safety-critical systems. We show in this talk formal guarantees on the robustness of a classifier by giving instance-specific lower bounds on the norm of the input manipulation required to change the classifier decision. Based on this analysis we propose the Cross-Lipschitz regularization functional. We show that using this form of regularization in kernel methods resp. neural networks improves the robustness of the classifier without any loss in prediction performance.

3.12 Syntax-Guided Synthesis

Rajeev Alur (University of Pennsylvania – Philadelphia, US)

The classical synthesis problem is to find a program or a system that meets a correctness specification given as a logical formula. Recent work on synthesis and optimization illustrates many potential benefits of allowing the user to supplement the logical specification with a syntactic template that constrains the space of allowed implementations. The formulation of the syntax-guided synthesis problem (SyGuS) is aimed at standardizing the core computational problem common to these proposals in a logical framework. The input to the SyGuS problem consists of a background theory, a semantic correctness specification for the desired program given by a logical formula, and a syntactic set of candidate implementations given by a grammar. The computational problem then is to find an implementation from the set of candidate expressions so that it satisfies the specification in the given theory.

In this tutorial we first describe how a wide range of problems such as automatic synthesis of loop invariants, program optimization, program repair to defend against timing-based attacks, and learning programs from examples can be formalized as SyGuS instances. We then describe the counterexample-guided-inductive-synthesis (CEGIS) strategy for solving the SyGuS problem. Finally we discuss our efforts over the past three years on defining the standardized interchange format built on top of SMT-LIB, repository of benchmarks from diverse applications, organization of the annual competition, SyGuS-COMP, of solvers, and experimental evaluation of solution strategies.

3.13 Formal Inductive Synthesis/Oracle-Guided Inductive Synthesis

Sanjit A. Seshia (UC Berkeley – Berkeley, CA, sseshia@berkeley.edu)

Joint work of Susmit Jha, Sanjit A. Seshia
Main reference Susmit Jha, Sanjit A. Seshia: "A theory of formal synthesis via inductive learning", Acta Inf., Vol. 54(7), pp. 693–726, 2017.
URL http://dx.doi.org/10.1007/s00236-017-0294-5

This talk presents a theoretical framework for synthesis from examples so as to satisfy a formal specification, termed as formal inductive synthesis. We discuss how formal inductive synthesis differs from traditional machine learning. We then describe oracle-guided inductive synthesis (OGIS), a framework that captures a family of synthesizers that operate by iteratively querying an oracle. An instance of OGIS that has had much practical impact is counterexample-guided inductive synthesis (CEGIS). We present a theoretical characterization of CEGIS for both finite and infinite program classes.

Details are given in [1, 2].

References

1 Susmit Jha, Sanjit A. Seshia. A Theory of Formal Synthesis via Inductive Learning. CoRR abs/1505.03953 (2015)
2 Susmit Jha, Sanjit A. Seshia. A theory of formal synthesis via inductive learning. Acta Inf. 54(7): 693-726 (2017)

3.14 Combining Logical and Probabilistic Reasoning in Program Analysis

Mayur Naik (University of Pennsylvania – Philadelphia, US)

Existing program analyses are expressed in the form of logical rules that are handcrafted by experts. While this logic-based approach has many benefits, however, it cannot handle uncertainty and lacks the ability to learn and adapt. This in turn hinders the accuracy, scalability, and usability of program analysis tools in practice.

We present a methodology and framework to incorporate probabilistic reasoning into existing program analyses that are based on logical reasoning. The framework comprises a front-end, which automatically integrates probabilities into a logical analysis by synthesizing a system of weighted constraints, and a back-end, which is a learning and inference engine for such constraints. We demonstrate how this approach advances three important applications of program analysis: automated verification, interactive verification, and bug detection. We will describe new algorithmic techniques to solve very large instances of weighted constraints that arise not only in our domain but also in other domains such as Big Data analytics and statistical AI.

3.15 Data-driven Program Synthesis From Examples

Nagarajan Natarajan (Microsoft Research India – Bangalore, IN)

A typical data analyst is often faced with tasks involving data formatting, and in fact studies say that more than 80% of the time is spent wrangling the data, before performing scientific analysis. Automatically synthesizing code pieces that can perform such tasks is a boon. There is a long line of work in the Programming Languages community on program synthesis from examples, and more recently, Machine Learning community has gained interest in the problem space. In this talk, I will outline some of the challenges faced when formulating a machine learning problem in this setting, and present our data-driven solution for program synthesis that builds on the successful PROSE framework. Results on benchmark datasets involving a fairly sophisticated text grammar show that, in most cases, with just one input-output example, the system can surface the "right" program at the top. I will also give a short demo of the deployed solution.

3.16 Leveraging computational models of human cognition for educational interventions

Anna Rafferty (Carleton College – Northfield, US)

Educational technologies offer the opportunity to provide personalized guidance or instruction to individual learners. Often, this is achieved by tracking what a learner knows based on her behavior in the system. This tracking can be challenging, however, especially in cases where learners' behaviors cannot easily be evaluated. For example, learners make a large number of choices in games or interactive simulations that should provide information about understanding, but it may not be possible to automatically mark each choice as correct or incorrect or easily relate the choices to knowledge in a domain. My approach is to leverage computational modeling and machine learning tools to develop general frameworks that can be applied to individual educational technologies. We have developed a Bayesian inverse planning framework to make fine-grained inferences about understanding based on learners' sequences of actions, demonstrating how a formal model of human behavior can lead to methods that are applicable across different educational domains. We have applied this framework to assess learners' understanding based on actions in game-based experiments in the lab, choices in a microbiology game played by middle schoolers, and step-by-step solutions to algebraic equations. Because the framework includes a generative model of human behavior, we have applied a variation of it to design more diagnostic assessments, both for algebra and for game-based experiments.

References

1 Rafferty, A. N., LaMar, M. M. and Griffiths, T. L. (2015), Inferring Learners' Knowledge From Their Actions. Cogn Sci, 39: 584–618. doi:10.1111/cogs.12157

3.17 Planning for Cars that Coordinate with People

Dorsa Sadigh (Stanford University, US)

As human-robot systems make their ways into our every day life, safety has become a core concern of the learning algorithms used by such systems. Examples include semi-autonomous vehicles such as automobiles and aircraft. The robustness of controllers in such systems relies on the accuracy of models of human behavior. In this talk, we propose a systematic methodology for analyzing the robustness of learning-based control of human-robot systems. We focus on the setting where human models are learned from data, with humans modeled as approximately rational agents optimizing their reward functions. In this setting, we provide a novel optimization-driven approach to find small deviations in learned human behavior that lead to violation of desired (safety) objectives.

3.18 Deduction and Induction – A Match Made in Heaven

Stephan Schulz (Duale Hochschule Baden-Württemberg – Stuttgart, DE)

First-order theorem provers search for proofs of a conjecture in an infinite and highly branching search space. This search critically depends on good heuristics. Unfortunately, designing good heuristics for the different choice points and classes of problems has proved to be very hard. Indeed, even classification of proof problems into classes with similar search behaviour is a largely open research question.

One way to address the difficulty of controlling search is to use inductive approaches, i.e. to try to find good heuristics by observing and generalizing examples of successful and failing proof searches. Here we discuss the three major choice points for superposition-based provers, and how good heuristics can be found via inductive processes.

We distinguish two different learning paradigms. In the first case, only the performance of different heuristics on a test set is used as input for the inductive process. Two examples for this are the automatic generation of automatic modes for theorem provers, and the improvement of clause evaluation heuristics via parameter optimization or genetic algorithms. The second paradigm not only considers the performance of a heuristic, but tries to find new heuristics by analyzing proofs, or, more generally, a graph of the proof search.

We give some results on established work, and discuss some preliminary progress and open questions from ongoing work.

3.19 Neural Program Synthesis

Rishabh Singh (Microsoft Research – Redmond, US)

The key to attaining general artificial intelligence is to develop architectures that are capable of learning complex algorithmic behaviors modeled as programs. The ability to learn programs allows these architectures to learn to compose high-level abstractions with complex control-flow, which can lead to many potential benefits: i) enable neural architectures to perform more complex tasks, ii) learn interpretable representations (programs which can be analyzed, debugged, or modified), and iii) better generalization to new inputs (like algorithms). In this talk, I will present some of our recent work in developing neural architectures for learning complex regular-expression based data transformation programs from input-output examples, and will also briefly discuss some other applications such as program repair and optimization that can benefit from learning neural program representations.

3.20 Learning Continuous Semantic Representations of Symbolic Expressions

Charles Sutton (University of Edinburgh, GB)

Combining abstract, symbolic reasoning with continuous neural reasoning is a grand challenge of representation learning. As a step in this direction, we propose a new architecture, called neural equivalence networks, for the problem of learning continuous semantic representations of algebraic and logical expressions. These networks are trained to represent semantic equivalence, even of expressions that are syntactically very different. The challenge is that semantic representations must be computed in a syntax-directed manner, because semantics is compositional, but at the same time, small changes in syntax can lead to very large changes in semantics, which can be difficult for continuous neural architectures. We perform an exhaustive evaluation on the task of checking equivalence on a highly diverse class of symbolic algebraic and boolean expression types, showing that our model significantly outperforms existing architectures.

3.21 Some ML Tasks in Theorem Proving

Josef Urban(Czech Technical University – Prague, CZ)

The talk will show several examples of how machine learning is today used in automated and interactive theorem proving tasks. I will also briefly mention the various theorem proving fields, recent developments in them, and the collaboration between interactive and automated provers.

3.22 PSI: Exact Inference for Probabilistic Programs

Martin Vechev (ETH Zürich, CH)

This talk introduces probabilistic programming and discusses exact inference with the PSI solver (psisolver.org) based on symbolic reasoning. PSI supports higher order functions, nested inference, discrete, continuous and mixed distributions and comes with its own symbolic integration engine. The talk also discusses various open problems in the area.

3.23 Learning from RNNs

Eran Yahav (Technion – Haifa, IL)

We address the problem of extracting an automaton from a given recurrent neural network (RNN). We present a novel algorithm that uses exact learning and abstract interpretation to perform efficient extraction of a minimal automaton describing the internal states of the given RNN. We use Angluin's L* algorithm as a learner and the given RNN as an oracle, employing abstract interpretation of the RNN for answering equivalence queries. Our technique allows automaton-extraction from the RNN while avoiding state explosion, even when very fine differentiation is required between RNN states.

3.24 Synthesis with Abstract Examples

Eran Yahav (Technion – Haifa, IL)

Interactive program synthesizers enable a user to communicate his/her intent via input-output examples. Unfortunately, such synthesizers only guarantee that the synthesized program is correct on the provided examples. A user that wishes to guarantee correctness for all possible inputs has to manually inspect the synthesized program, an error-prone and challenging task. We present a novel synthesis framework that communicates only through (abstract) examples and guarantees that the synthesized program is correct on all inputs. The main idea is to use abstract examples—a new form of examples that represent a potentially unbounded set of concrete examples. An abstract example captures how part of the input space is mapped to corresponding outputs by the synthesized program. Our framework uses a generalization algorithm to compute abstract examples which are then presented to the user. The user can accept an abstract example, or provide a counterexample in which case the synthesizer will explore a different program. When the user accepts a set of abstract examples that covers the entire input space, the synthesis process is completed. We have implemented our approach and we experimentally show that our synthesizer communicates with the user effectively by presenting on average 3 abstract examples until the user rejects false candidate programs. Further, we show that a synthesizer that prunes the program space based on the abstract examples reduces the overall number of required concrete examples in up to 96% of the cases.

3.25 Smooth Imitation Learning

Yisong Yue (California Institute of Technology – Pasadena, US)

We study the problem of smooth imitation learning for online sequence prediction, where the goal is to train a policy that can smoothly imitate demonstrated behavior in a dynamic and continuous environment in response to online, sequential context input. We present an approach that combines smooth model-based controllers with black-box machine learning approaches in order to obtain flexible function classes that are regularized to ensure smooth dynamics.

References

1 Learning Online Smooth Predictors for Real-time Camera Planning using Recurrent Decision Trees. Jianhui Chen, Hoang M. Le, Peter Carr, Yisong Yue, James J. Little. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2016

3.26 Preference-Based Teaching

Sandra Zilles (University of Regina, CA)

The classical teaching dimension model, used to describe the sample complexity of learning from helpful teachers, assumes that the learner simply produces any hypothesis consistent with the data provided by the teacher. Assuming instead that the learner expects data to be chosen in a helpful way improves the sample complexity. One model in this context is Preference-Based Teaching, in which the learner uses a preference relation over concepts and always hypothesizes a most preferred concept consistent with the data. In the case of finite concept classes, the corresponding sample complexity appears to be related to the VC-dimension. Some open problems and potential connections to Formal Methods are discussed.

3.27 Falsification of Cyber-Physical Systems with Machine Learning Components

Sanjit A. Seshia (UC Berkeley – Berkeley, CA, sseshia@berkeley.edu)

Joint work of Tommaso Dreossi, Alexandre Donze, Sanjit A. Seshia
Main reference Tommaso Dreossi, Alexandre Donzé, Sanjit A. Seshia: "Compositional Falsification of Cyber-Physical Systems with Machine Learning Components", in Proc. of the NASA Formal Methods - 9th International Symposium, NFM 2017, Moffett Field, CA, USA, May 16-18, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10227, pp. 357–372, 2017.
URL http://dx.doi.org/10.1007/978-3-319-57288-8_26

Cyber-physical systems (CPS), such as automotive systems, are starting to include sophisticated machine learning (ML) components. Their correctness, therefore, depends on properties of the inner ML modules. While learning algorithms aim to generalize from examples, they

are only as good as the examples provided, and recent efforts have shown that they can produce inconsistent output under small adversarial perturbations. This raises the question: can the output from learning components can lead to a failure of the entire CPS? In this work, we address this question by formulating it as a problem of falsifying signal temporal logic (STL) specifications for CPS with ML components. We propose a compositional falsification framework where a temporal logic falsifier and a machine learning analyzer cooperate with the aim of finding falsifying executions of the considered model. The efficacy of the proposed technique is shown on an automatic emergency braking system model with a perception component based on deep neural networks.

Details of this contribution are given in [2], and a broader perspective of applying formal methods to the goal of verified artificial intelligence is discussed in [1].

References

1 Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry: Towards Verified Artificial Intelligence. CoRR abs/1606.08514 (2016)
2 Tommaso Dreossi, Alexandre Donze, and Sanjit A. Seshia: Compositional Falsification of Cyber-Physical Systems with Machine Learning Components. NFM 2017, pages 357-372.

4 Breakout Groups

On the final two days of the seminar, multiple breakout sessions were organized on the following topics:

1. Machine Teaching and Oracle-Guided Synthesis
2. Solvers in Formal Methods and Machine Learning
3. Formal Guarantees for Machine Learning Systems

We present here a very brief summary of each of the breakout topics.

Machine Teaching and Oracle-Guided Synthesis

In principle, all of the work in machine learning and inductive synthesis (synthesis from examples) can be explained using an oracle-guided framework. In particular, there is a broad spectrum varying from passive learning on the one side to machine teaching on the other, with active learning, query-based learning, and counterexample-guided inductive synthesis in between. The breakout sessions on this topic discussed a common terminology to connect the formal methods and machine learning communities, and the role the framework of oracle-guided inductive synthesis can play in connecting the ideas in the two communities. The formal methods community can provide interesting sources of problems for theoretical work on machine teaching and query-based learning. Likewise, the machine learning community can provide theory and techniques to prove results about the power of oracle-guided synthesis.

Solvers in Formal Methods and Machine Learning

In Formal Methods, solvers based on SAT, SMT, BDDs, etc. play an important role. The discussions in these breakout sessions discussed how machine learning is already playing an important role in the development of these solvers as well as interactive theorem provers, such as, e.g., in portfolio SAT solving techniques. Further, the role of solvers to improve machine learning was also discussed. For example, verification tools could be used to generate more training data, such as the work on falsification of CPS with machine learning components

discussed earlier in the seminar. The importance of choosing good representations shows up in the work of both communities. On the frontier of solver research, the topic of model counting and volume estimation is of high relevance to both communities.

Formal Guarantees for Machine Learning

The different forms of guarantees for machine learning systems were discussed, as well as the role of formal specifications. Topics ranging from Boolean versus quantitative specifications, worst-case versus asymptotic versus probabilistic guarantees, and which guarantees permit compositional reasoning. Formal reasoning can be applied to learning algorithms, learned models, and training data. Interpretability is an important objective. The issue of specifying objectives using logical formulas or cost functions was also discussed.

Participants

- S. Akshay
Indian Institute of Technology – Mumbai, IN
- Dalal Alrajeh
Imperial College London, GB
- Rajeev Alur
University of Pennsylvania – Philadelphia, US
- Nikolay A. Atanasov
University of California at San Diego, US
- Ammar Ben Khadra
TU Kaiserslautern, DE
- Felix Berkenkamp
ETH Zürich, CH
- Ben Caulfield
University of California – Berkeley, US
- Swarat Chaudhuri
Rice University – Houston, US
- Luc De Raedt
KU Leuven, BE
- Jyotirmoy Deshmukh
USC – Los Angeles, US
- Dana Fisman
Ben Gurion University – Beer Sheva, IL
- Pranav Garg
Amazon – Bangalore, IN
- Matthias Hein
Universität des Saarlandes, DE
- Holger Hermanns
Universität des Saarlandes, DE
- Susmit Jha
SRI – Menlo Park, US
- Kristian Kersting
TU Darmstadt, DE
- Andreas Krause
ETH Zürich, CH
- Sasa Misailovic
University of Illinois – Urbana-Champaign, US
- Mayur Naik
University of Pennsylvania – Philadelphia, US
- Nagarajan Natarajan
Microsoft Research India – Bangalore, IN
- Anna Rafferty
Carleton College – Northfield, US
- Dorsa Sadigh
Stanford University, US
- Stephan Schulz
Duale Hochschule Baden-Württemberg – Stuttgart, DE
- Sanjit A. Seshia
University of California – Berkeley, US
- Rishabh Singh
Microsoft Research – Redmond, US
- Armando Solar-Lezama
MIT – Cambridge, US
- Charles Sutton
University of Edinburgh, GB
- Josef Urban
Czech Technical University – Prague, CZ
- Martin Vechev
ETH Zürich, CH
- Eran Yahav
Technion – Haifa, IL
- Yisong Yue
California Institute of Technology – Pasadena, US
- Xiaojin Zhu
University of Wisconsin – Madison, US
- Sandra Zilles
University of Regina, CA