Placeholder title: Formal methods in constraint-based learning

December 21, 2023

Abstract

1 Introduction

Using formal methods, such as using proof assistants or abstract interpretation, in combination with machine learning (ML) is a relatively new approach that has been gaining popularity in recent years [11, 7]. This can be connected primarily to the rising importance of machine learning verification and an increased awareness of the fragility of machine learning models, both of which value the guarantees that formal methods can provide CITATION. Traditionally confined to the realm of software and hardware verification, formal methods are increasingly being repurposed to scrutinize the inherently probabilistic nature of ML models.

The integration of formal methods with machine learning poses many challenges, stemming from the inherent differences between the two domains, from the black-box nature of many ML models, through scalability issues especially in the case of neural networks (NNs) to the lack of well-defined specifications required in formal methods. It is the last of those that is of most interest to this work. While there has been a move toward increasingly rigorous approaches in specification, both for verification and training, such specification are often mathematically complex.

This complexity is inherent especially to neural networks but is widely known to pose a challenge to formalisation tools. The most prominent reasons behind this issue come from the necessity to reason about continuous real space, vectors and vector operations as well as the ability to express mathematical concepts such as partial differentiation. This is unavoidable when reasoning about certain methods, as due to common usage of gradient descent algorithm in neural network training CITE.

However we believe that there exist current state of the art formalisation tools which are expressive enough to reason about mathematical concepts needed for expressing properties and methods for machine learning.

Mathematical Components (MathComp) [10] in Coq CITE stands as a testament to the expressive power of formal proof systems in addressing advanced mathematical concepts. At its core, MathComp leverages the Calculus of Inductive Constructions (CIC), the underlying type theory of Coq, to encode

mathematical abstractions in a way that allows bridging the gap between mathematics and the formalization of machine learning concepts.

One of such concepts is Logic of Differentiable Logics (LDL), a rigorously defined meta-language used for property based training of neural networks. ADD A MORE IN DEPTH EXPLANATION OF LDL ITSELF

The goal of this paper is two-fold. Firstly, we would like to offer it as demonstration that current state of the art already allows for application of formal approaches to well-defined problems in machine learning and is expressive enough to handle the needed underlying mathematical theory. To demonstrate this we formalise the meta-language LDL, a method used to train neural networks, together with its properties and proofs, with emphasis on properties desirable in machine learning.

Furthermore we believe that using formal methods approaches can be beneficial not only in verification, but also in helping to deepen the understanding of machine learning as well as enforcing a more formal approach to more mathematical areas of machine learning. For future programming languages created for AI verification proofs may, in the future, become compulsory, enforcing an additional level of rigour [2, 9].

2 Related work

To the best of our knowledge there is very little work that is closely related to our approach and involves applying formal methods to the training approaches itself instead of an already trained machine learning model or its properties.

The primary area where formal methods and machine learning combine is verification. This field has been rapidly gaining traction in the last few years and there are many different approaches present in neural network verification[11, 7]. One of them is abstract interpretation, used by many state of the art verifiers CITE PARTICULAR VERIFIER . These methods are partial formal verification techniques and are sound but not complete [4]. There are however also approaches that are both sound and complete based on different techniques including use of SMTs (Satisfiability Modulo Theory) [6] solvers or MILP (Mixed Integer Linear Programming). There has also been similar work done for other machine learning approaches than neural networks such as for support vector machines (SVMs) [8] or Decision Tree Ensembles [5] though we do not focus on such models in our work.

However majority of approaches involving formal methods and machine learning apply to already trained machine learning models, and not approaches such as constraint-based training itself and thus are not very closely related to this work. This area of research is much less explored as of yet, primarily due the difficulty of expressing the mathematical theory underlying neural networks. There have however been attempts at it such as [1] who formalise an embedding of linear temporal logic and an associated evaluation function in Isabelle to prove its soundness - and furthermore are able to produce OCaml versions of the aforementioned functions that can be integrated with Python and PyTorch eliminating many risks associated with ad-hoc implementations. While another related example are formalisation of other types of algorithms [3] or formalisation of neural networks using Haskell [12]. They, while not directly related to the subject matter, utilise similar approaches and also have to contend with

similar issues related to mathematical complexity.

POSSIBLY AN ADDITIONAL PARAGRAPH ON NON-ML MATH FORMALISATION THAT IS TANGENTIALLY SIMILIAR.

3 Background - notes

Notes on what probably needs to make its way into background (non-exhaustive)

- A good explanation of DL and similar methods in machine learning. Including syntax, semantics in tables. Also explanation of loss functions and their role in training.
- Possibly properties here, not later.
- Other work on similar/related topics (pointing out most formalise finalised trained models, not methods of training itself so can't find a good comparison)

Some of other work (more or less related depending on the specific case:

4 Temporary Name: Formalisation - notes

General notes to self on formalisation.

Different functions, lemmas, and things that peraphs stand out.

4.1 Syntax and semantics

Implementation of the syntax: *expr* with it's custom dependant type *simple_type*. Since the type system prevents non-termination of the lambda expressions they, together with let statements, have been omitted in the Coq formalisation (argumentation why it's okay goes here).

Implementation of the semantics:

- *type_translation* semantics of types
- a word on translation of indices and vectors using tuples
- mention Real library from Mathcomp to handle reals
- *translation* - a semantics for all fuzzy DLs. Here there needs to be explanation maybe of how DL2 and STL are separate from the fuzzy logics as they are different enough to warrant a separate definition and lemmas, not enough overlap translation itself is dependently typed, handles 4 different fuzzy logics through use of global variable (for now it's global variable, up for discussion)

No quantifiers in the semantics. That does not impact the formalisation of most properties aside from soundness as the other properties do not concern quantifiers.

4.2 Soundness (and associated lemmas)

- $expr\_ind'$ custom induction principle was needed due to use of sequences. Explain sequences in MathComp (hidden tuples)? Explain that this is needed due to possibility of non-associative binary operations in the language.
- $[nameofDL]\_translate\_Bool\_T\_01$ - lemmas that prove that the fuzzy DL semantics fall within the range [0,1]

Inversion lemmas also mentioned here possibly.

4.3 Logical properties

On associativity and commutativity for the DLs that have them (and that associativity is much more tricky then commutativity to prove). Not entirely sure if antyhing more interesting is here specifically so likely not its own subsection.

4.4 Shadow-lifting

Using limit definition of partial differentiation.

It does work out of the box using MathComp)in that limits exist but does require more lemmas as it is a new library. Just a lot more detail on this proof when finished.

5 Other notes

Complexity increase depending on DL is. While "complexity" is not easy to measure for a DL on paper it is obvious in its impact in formalisation. Associativity of conjunction for Łukawsiewic, Godel and product all have around 10 lines (get exact after cleanup) - but Yager takes 38 (again, correct after cleanup). If I prove STL mention that (if case split in analogous ways to other proofs we get 38 cases)

A lot of difficulty is added due to using nary operations for the sake of generality (bigops, etc.). Will need to have good reasoning for that (primarily htat it's following the very general idea behind LDL).

References

[1] Mark Chevallier, Matthew Whyte, and Jacques D Fleuriot. Constrained training of neural networks via theorem proving. *arXiv preprint arXiv:2207.03880*, 2022.

[2] Matthew Daggitt, Wen Kokke, Ekaterina Komendantskaya, Robert Atkey, Luca Arnaboldi, Natalia Slusarz, Marco Casadio, Ben Coke, and Jeonghyeon Lee. The vehicle tutorial: Neural network verification with vehicle. In Nina Narodytska, Guy Amir, Guy Katz, and Omri Isac, editors, *Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems*, volume 16 of *Kalpa Publications in Computing*, pages 1–5. EasyChair, 2023. doi: 10.29007/5s2x. URL `https://easychair.org/publications/paper/Rkrv`.

[3] Ieva Daukantas, Alessandro Bruni, and Carsten Schürmann. Trimming data sets: a verified algorithm for robust mean estimation. In *23rd International Symposium on Principles and Practice of Declarative Programming*, pages 1–9, 2021.

[4] Steve Easterbrook and John Callahan. Formal methods for verification and validation of partial specifications: A case study. *Journal of Systems and Software*, 40(3):199–210, 1998.

[5] Gil Einziger, Maayan Goldstein, Yaniv Sa'ar, and Itai Segall. Verifying robustness of gradient boosted models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2446–2453, 2019.

[6] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*, pages 443–452. Springer, 2019.

[7] Moez Krichen, Alaeddine Mihoub, Mohammed Y. Alzahrani, Wilfried Yves Hamilton Adoni, and Tarik Nahhal. Are formal methods applicable to machine learning and artificial intelligence? In *2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 48–53, 2022. doi: 10.1109/SMARTTECH54121.2022.00025.

[8] Francesco Ranzato and Marco Zanella. Robustness verification of support vector machines. In *Static Analysis: 26th International Symposium, SAS 2019, Porto, Portugal, October 8–11, 2019, Proceedings 26*, pages 271–295. Springer, 2019.

[9] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. Toward verified artificial intelligence. *Communications of the ACM*, 65(7):46–55, 2022.

[10] Mathematical Components Team. Mathematical components library, 2007. URL `https://github.com/math-comp/math-comp`. Last stable version: 2.1 (2023).

[11] Caterina Urban and Antoine Miné. A review of formal methods applied to machine learning. *arXiv preprint arXiv:2104.02466*, 2021.

[12] Ningning Xie. Haskell for choice-based learning (keynote). In *Proceedings of the 16th ACM SIGPLAN International Haskell Symposium*, pages 1–1, 2023.