Placeholder title: Formal methods in constraint-based learning

December 13, 2023

Abstract

1 Introduction

- formal methods in combination with machine learning as approach
- normally for verification
- use it to deepen understanding/inform ML research itself

2 Background

- A good explanation of DL and similiar methods in machine learning. Including syntax, semantics in tables.
- Possibly properties here, not later.
- Other work on similiar/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:

Formalization of an embedding of linear temporal logic and an associated evaluation function to prove its soundness in Isabelle and producing OCaml versions of the aforementioned functions that can be integrated with Python and PyTorch [Chevallier et al.(2022)Chevallier, Whyte, and Fleuriot]. Formalisation of other types of algorithms [Daukantas et al.(2021)Daukantas, Bruni, and Schürmann] or formalisation of neural networks using Haskell [Xie(2023)]

Reviews of works in formal methods in machine learning (not only neural networks) [Urban and Miné(2021), Krichen et al.(2022)Krichen, Mihoub, Alzahrani, Adoni, and Nahhal]. These are however mostly verification work.

Generally most work is done on trained models - not on the method itself - with very few exceptions. Need to find more of the exceptions, put in related work with caveat that they are not always very directly related to this.

3 Temporary Name: Formalisation

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

3.1 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.

3.2 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.

3.3 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.

3.4 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.

4 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

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

[Daukantas et al.(2021)Daukantas, Bruni, and Schürmann] 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.

[Krichen et al.(2022)Krichen, Mihoub, Alzahrani, Adoni, and Nahhal] 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.

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

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