Workshop Proposal: Advances in Programming Languages and Neurosymbolic Systems (AIPLANS)

Anonymous Author(s)
Affiliation
Address
email

Abstract

The development of practical libraries for automatic differentiation has enabled rapid progress in gradient-based learning over the last decade. Other forms of automatic programming now emerging in the statistical learning and programming language community hold the promise of unleashing similar progress in nearby fields, from probabilistic to classical logic. Concurrently, machine learners have made steady progress in representing and synthesizing programs. Other workshops have explored these topics separately, yet few have highlighted the interplay between automatic and synthetic programming, a situation we hope to remedy.

1 Introduction

Neural information processing systems have benefited tremendously from the development of languages and abstractions for automatic differentiation. Similar domain-specific languages have begun to automate inference in other logical disciplines, such as probabilistic inference, classical logic, and message passing schemes on tree- and graph-structured data.

Not only does machine learning itself benefit from tools and languages for programmable inference, learning can also be seen as a kind of programming language with a unique intermediate representation, and which is now being used to produce human-readable procedures. Early synthetic functions are now possible thanks to recent progress in statistical langauge modeling and functional programming, resembing procedures a human might plausibly write.

Using techniques from programmable inference to generate programs, and using insights learned developing those programs to drive innovation in AD and probabilistic programming is a now virtuous cycle, with a steady and growing stream of software and academic papers.

However many ideas are being reinvented in this process. Automatic differentiation was invented over a half a dozen times in various disciplines over the last century. Functional and type-safe programming are only now being added to Python, the primary language used in machine learning. AD designers continue rediscover connections to implicit differentiation, bilevel optimization, stochastic processes and other fields. Much work remains.

The duality between code and data for instance is well-known in PL under the guise of homoiconicity. Many other topics that machine learners are just encountering have been well-studied in the programming language community. Programming language designers have thought deeply about higher-order functions, currying and rewriting, and denotational and operational semantics, which enables APIs to compose well and work correctly.

Similarly, programming language theory has wrestled with issues of expressivity and tractability, and intensional and extensional representation, a distinction which has long since been reconciled by the statistical learning community under the umbrella of model-based learning and approximation theory.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Other areas where the interaction could be fruitful are tools for equivalence, proof search and metrics. New language models could enable natural langauge and assitive programming.

As outlined above, we believe that recent advances in statistical learning and programming languages have been largely siloed, and these two communities have much to learn from each other. Exchanging ideas in a joint workshop could help reveal unrealized connections. Our workshop is designed to be as inclusive as possible. For illustration, we include the following non-exhaustive list of topics:

- Differentiable programming / automatic differentiation
- Probabilistic programming / statistical inference
- Declarative programming / constraint programming
- Dynamic programming / reinforcement learning
- Functional programming / λ-calculus
- Array programming / linear algebra
- Semiring programming / message passing
- Metaprogramming / reflection
- Logic programming / proof search
- Domain-specific languages

We also encourage developers of libraries and frameworks to submit their work for evaluation.