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

Anonymous Author(s)
Affiliation
Address
email

Abstract

The development of practical automatic differentiation has enabled much progress in gradient-based learning over the last decade. Other domain specific languages for automatic programming hold the promise of unleashing similar progress in nearby fields, such as probabilistic and classical logic. Concurrently, machines have made steady progress in representing and synthesizing programs. Other workshops have explored these themes 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 which humans program indirectly, and which can increasingly be used to reproduce human-readable procedures. Early examples of synthetic functions are starting to emerge thanks to recent progress in statistical langauge modeling and functional programming, resembing procedures a human programmer might plausibly write.

Using techniques from programmable inference to transform and generate programs, and by adapting insights gained developing those programs to drive innovation in AD and probabilistic programming is a virtuous cycle, with a growing stream of software and academic papers. We envision collaboration between automatic and synthetic programming will continue to unlock deeper insights as researchers become more accustomed to outsourcing low-level reasoning tasks to these systems.

Many ideas are being reinvented and rediscovered in this process. AD was invented over a half a dozen times over the last century and research continues to reveal interesting connections to implicit differentiation, bilevel optimization, stochastic processes and other fields. Semiring programming has existed in various forms for many decades and has deep connections to reinforcement learning, structured inference and probabilistic programming. Much work remains.

Many topics machine learners are just encountering have been well-studied in the programming language community. For example, functional and type-safe programming are well-studied in PL circles but relatively new to Python, the primary language used in machine learning. The duality between code and data for instance is well-known in PL under the guise of homoiconicity. Programming languages have thought deeply about higher-order functions, currying and rewriting, and denotational and operational semantics, which enables APIs to work smoothly and correctly.

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

Similarly, programming languages 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. PL could potentially benefit from structured inference and propagation algorithms as a medium for distributed computation... We believe many other examples exist.

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.