Parallels in Automatic and Inductive Programming

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 communities 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 of automatic and inductive programming, a situation we hope to change.

1 Introduction

Neural information processing systems have benefited tremendously from the development of libraries and frameworks for automatic differentiation. Recent work has begun to develop similar frameworks for automating probabilistic inference and other domain specific languages for data processing.

Not only does machine learning benefit from tools and languages for programmable inference, learning can also be seen as a kind of programming language in its own right, based on its own code and data and which is now being used to generate self-similar code and data. Using techniques from programmable inference to generate programs, and using insights learned by developing those programs to drive innovation in AD and probabilistic programming is a virtuous cycle.

Many ideas are being reinvented and rediscovered... (AD was invented at least half a dozen times.)

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 operational and denotational semantics, which enables APIs to compose well and work correctly.

Similarly, programming language theory has long wrestled with issues of intensional and extensional representation, expressivity and performance, a distinction which has long since been reconciled by the statistical learning community under the umbrella of model-based learning and approximation theory. 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 communities have much to learn from each other. Exchanging ideas between these two communities could lead to other unrealized insights. Our workshop is designed to be as inclusive as possible. We include the following non-exhaustive list of topics:

- Differentiable programming / automatic differentiation
- Probabilistic programming / statistical inference
- Declarative programming / constraint programming

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

- 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.