Perspective: Differentiable Programming in HEP

Matthew Feickert
(on behalf of LOI author team)

University of Illinois at Urbana-Champaign

Snowmass Community Planning Meeting

Collider Data Analysis Strategies

October 6th, 2020

Introduction

- Optimization in HEP traditionally done with finite difference approximation of gradient (e.g. MINUIT)
- Machine learning libraries exploit automatic differentiation (autodiff) to give **full gradient**
- **Differentiable programming** offers paradigm to exploit deep learning advances for (systematic aware) end-to-end optimization using efficient gradient-based optimization algorithms!
- Requires differentiable versions of non-differentiable operations (e.g. binning)

Differentiable Programming in High-Energy Physics

Atılım Güneş Baydin (Oxford), Kyle Cranmer (NYU), Matthew Feickert (UIUC), Lindsey Gray (FermiLab), Lukas Heinrich (CERN), Alexander Held (NYU) Andrew Melo (Vanderbilt) Mark Neubauer (UIUC), Jannicke Pearkes (Stanford), Nathan Simpson (Lund), Nick Smith (FermiLab), Giordon Stark (UCSC), Savannah Thais (Princeton), Vassil Vassilev (Princeton), Gordon Watts (U. Washington)

August 31, 2020

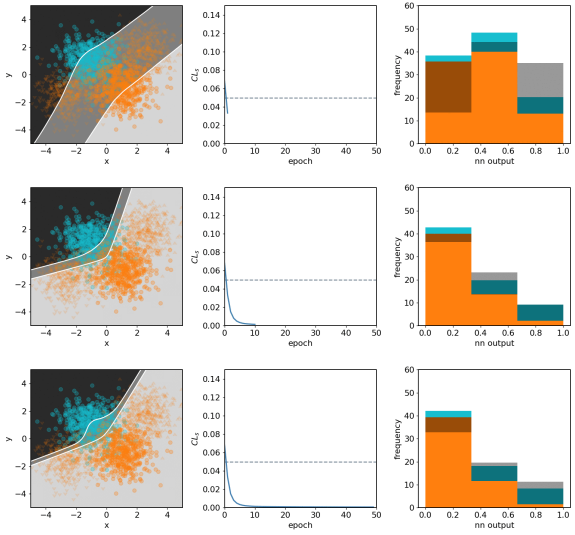
Abstract

A key component to the success of deep learning is the use of gradient-based optimization. Deep learning practitioners compose a variety of modules together to build a complex computational pipeline that may depend on millions or billions of parameters. Differentiating such functions is enabled through a computational technique known as automatic differentiation. The success of deep learning has led to an abstraction known as **differentiable programming**, which is being promoted to a first-class citizen in many programming languages and data analysis frameworks. This often involves replacing some common non-differentiable operations (eg. binning, sorting) with relaxed, differentiable analogues. The result is a system that can be optimized from end-to-end using efficient gradient-based optimization algorithms. A *differentiable analysis* could be optimized in this way — basic cuts to final fits all taking into account full systematic errors and automatically analyzed. This Snowmass LOI outlines the potential advantages and challenges of adopting a differentiable programming paradigm in high-energy physics.

Snowmass Computational Frontier LOI

Ongoing Work

- **Differentiable Programming in Analysis Code**
  - Simultaneously optimize free parameters with respect to the desired physics objective (e.g. optimize a cut value for significance)
  - Projects: INFERNO, neos
  - c.f. Lukas and Nathan's talks at PyHEP 2020 on automatic differentiation and neos
- **Differentiable Programming in Simulation Code**
  - Compute gradients for simulated samples with respect to simulation parameters
  - Reduce the number of simulated events required for simulation-based (aka likelihood-free) inference



neos 3 bin observable optimized with systematics w.r.t. CL_s

Outlook

- Exciting possibilities for many applications!
  - Medium-term goal: Develop a toolkit of differentiable versions of common operations in analysis
  - Long-term goal: Optimize realistic physics analyses by incorporating autodiff capabilities into analysis software
- Interesting challenges around implementation (e.g. distributed sharing of gradients)
- **gradHEP**: Informal interexperimental group formed under HSF to start research in this area
  - *The intention of this working group is to continue to investigate these issues, and provide input to the Snowmass process through ad-hoc contributions, with results and the state of the field* ***summarized in a white-paper****.* – Snowmass LOI on Differentiable Programming
  - c.f. HEP Software Foundation activity page for more info and contact