Optimizing reverse-mode automatic differentiation with advanced activity-analysis

---

Petro Zarytskyi
Taras Shevchenko National University of Kyiv, Ukraine

Mentors:
Vassil Vassilev, David Lange
June, 2023

Introduction

Clad is an automatic differentiation clang plugin for C++. It automatically generates code that computes derivatives of functions given by the user. Automatic differentiation involves breaking up a function into elementary operations and computing the result by applying the chain rule to find the derivatives of all the intermediate variables. This process can be done both ways: from input variables to output variables and vice versa. These two methods are known as forward (tangent) mode and reverse (adjoint) mode respectively. Both modes have advantages and disadvantages and may be used in different scenarios. In particular, the reverse mode turns out to be more efficient when there are more dependent output variables than independent input variables (e.g. calculating a gradient).

Project Goals

The approach to blindly compute the derivatives of all the intermediate variables obviously produces code that does a lot of unnecessary calculations. With advanced activity analysis, the variables which are not used to compute the result can be found and removed, increasing the time- and memory- efficiency of the output code. The next stage of optimization would involve sorting out variables that despite being needed for the result aren’t used to compute its derivative.

Proposed Timeline

Over the course of the project, I will not have any major responsibilities.

Starting June 26

Week 1-2:
Reading papers and other materials on activity analysis in reverse-mode automatic differentiation. Setting up Clad and going through its codebase.

Week 3:
Planning precise algorithms' implementation for ClangAST. Developing the best optimizing strategy for the already existing Clad implementation. **Deliverable:** Making a report of the missing features and failing tests.

Weeks 4-6:
Implementing all the activity analysis algorithms that were previously planned, creating a working demo. **Deliverable:** Working tape-push branch which succeeds all tests. The implementation must be able to turn on and off the activity analysis.

Week 7-8:
Make the activity analysis default for clad. Test in major workflows such as ROOT's RooFit package. Doing tests of all possible use scenarios of Clad. Fixing bugs if found. **Deliverable:** Clad uses the developed activity analysis by default. Technical report of the problems in the adoption in RooFit if any.

Week 9:
Develop benchmarks in the clad benchmark system to compare running examples with activity analysis on and off.

Week 10:
Investigating possible ways to optimize existing activity analysis. Considering alternative methods of activity analysis that are yet to be discovered. Investigate the potential of the clang static analysis (SA) and data flow analysis infrastructure to capture advanced optimization opportunities. **Deliverable:** A report and a basic implementation based on clang SA.

Week 11:
Investigate if we can enable clad in the ADBench infrastructure. Upon success work on a comparative study between various tools using activity analysis.

Week 12:
Implementing new activity analysis methods if any relevant ones were found. Doing the final debugging. Writing documentation. **Deliverable:** A technical document describing the current status and a blog post.Creating representative tests to add to the already existing ones. Writing documentation to them.

Week 13:
Estimating the performance increase due to separate implemented activity analysis methods. Using this data in the final report/presentation.

About Myself

I am Petro Zarytskyi, a second-year bachelor's student at the Taras Shevchenko National University of Kyiv. Since my school years, I got interested in mathematics and physics. I participated in and won multiple Ukraine-wide and international olympiads in physics and mathematics. In university, I decided to pursue a degree in applied mathematics, which focuses on programming and areas of math that are tangent to computer science. Improving my skills in efficient code writing (C++, Python) allowed me to implement many projects (e.g. simulating thermal conduction, light traveling through a medium with a non-constant refractive index, physically accurate pixel game engine, etc.).