Introduction

Differential calculus was invented as a formal branch of mathematics in the middle of the 17th century independently by Isaac Newton and Gottfried Leibniz. Newton used it to solve the problem of the motion of the planets around the sun, and it's proven to be an essential tool for the sciences ever since. In the modern era, essentially all scientific calculations of interest are performed on computers. There are many scenarios where we know how to compute a function of interest and would like to efficiently evaluate its derivative(s). The canonical example is root finding. If we know a function's derivative, we can iteratively improve from a starting guess until we find a root using a simple procedure, Newton's Method. Many phenomena of interest in physics and other sciences can be described as differential equations (either ODEs or PDEs). The ability to efficiently evaluate derivatives is crucial in numerically solving these systems. In recent years, Machine Learning has been a hot research area. Solving ML problems often hinges on the ability to train a neural network using some form of a gradient descent algorithm. As the name suggests, this algorithm requires the ability to evaluate not just the function (here the output of the neural net) but also its first derivative, which in practice is typically the change in the error metric with respect to the parameter values. These neural networks are massively complex functions that can have millions of parameters. A procedure of numerically differentiating the network by shifting the value of each parameter has cost that scales linearly with the number of parameters. Some form of automatic differentiation is vital to the practical use of neural nets. One of the most successful machine learning libraries is TensorFlow by Google, which at its core is an enterprise class automatic differentiation library.

Background

Calculus gives us a few simple rules we can apply to compute the derivative of a function that is the result of a combination of two more simple functions. Calculus rules include the sum, product, and quotient of two functions. The most important calculus rule in Automatic Differentiation is the Chain Rule of calculus. This states that the derivative of a composition is the product of two derivatives. $f'(u(x)) = f'(u(x) \cdot u'(x)$ The chain rule works in multiple dimensions. If f is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, its derivative an m x n matric called the Jacobian. The chain rule in the multidimensional case tells us to take the matrix product of an m x r matrix and an r x n matrix to compute the derivative.

The essential idea of Automatic Differentiation is that any computation performed by a computer will consist of a set of basic steps, e.g. function evaluations. These may be strung together in a complex graph with multiple steps, but each step will be a basic operation. For the evaluation of a mathematical function, each step will consist of either a "primitive" or built-in function (e.g. +, -, x, /, sqrt, log, exp, pow, etc.) or a composition of these primitives. As long as we know how to evaluate both the function $f(x)$ and its derivative $f'(x)$ at each step, we can trace the calculation through the computation graph and also keep track of the first derivative. This is the idea at the heart of Automatic Differentiation. The rest as they say id bookkeeping...

How to Use Fluxions

Software Organization

Implementation