Designing and implementing a tensor calculus

Farin Lippmann

August 2022

1 Abstract

2 Table of contents

3 Introduction

With the rise of machine learning applications, interest in optimization problems has been growing. In these problems, one seeks to find values for a number of parameters that maximize or minimize a given scalar objective function. While some optimization problems have a closed-form solution which can be computed directly, many can only be solved approximately. When computing these approximate solutions, derivatives play a crucial role. In particular, the first derivatives of the objective function are used in the ubiquitous gradient descent procedure. In every even slightly advanced optimization problem, multiple parameters need to be optimized simultaneously. This also means that derivatives need to be computed with respect to multiple variables, organized either in vectors, matrices or higher-order tensors. Commonly used machine learning frameworks (like Tensorflow, PyTorch, Theano) focus heavily on the first derivatives of the objective function.

Higher-order derivatives can, however, also be of interest, for example in the case of convexity checks. The convexity of objective functions and their associated optimization problems play a major role in the field of optimization. If a problem is strictly convex or concave, then it has a unique global optimum and the gradient descent algorithm can approximate it arbitrarily well. To check for convexity, the objective function's Hessian matrix, the matrix of second-order partial derivatives, needs to be computed and checked for positive semidefiniteness. If the function's inputs are naturally organized in a matrix or higher-order tensor, then the resulting Hessian will also be a higher-order tensor. The prevalent machine learning frameworks have no way to directly compute these derivatives, and classical computer algebra systems struggle with efficiency, as they work on the level of individual tensor entries.

To bridge this gap, we design and implement a tensor calculus that allows for automatic symbolic differentiation of tensor expressions of any order. In addition, this calculus will use Einstein summation notation to represent tensor products, rather than the more complex ricci notation.

This thesis builds on the work by Laue, Mitterreiter and Giesen, who have built a similar calculus for matrix derivatives [Matrix Calculus] and developed the theoretical foundation for an Einstein notation based tensor calculus [Tensor Paper].

The structure of the thesis is as follows. First, the input syntax of the calculus is developed in form of a grammar. Then, the parsing and representation of tensor expression is discussed. ...

3.1 (Main text)

3.2 Literature

3.3 Appendices

3.4 Declaration of autonomy