The Simple Essence of Automatic Differentiation (Elliott, [2018])

Overview.

The author proposes a functional programming framework for building differentiable programs. His major contribution is a Haskell plugin that defines a category of differentiable primitive types with an associated automatic differentiation operation.

The author chooses to embellish the derivative operation in order to make it composable, which allows him to represent the derivative operation as a functor from functions to embellished derivatives. This is reminiscent of the request function strategy from Fong et al. [2].

One of the author's key ideas is to replace the explicit matrix representation of linear maps with general monoidal functions. He does this by taking advantage of the isomorphism between "generalized matrices" and the hom-functor. This approach allows him to incorporate type-safety, native parallelism, compiler optimizations and functional memoization into his system. It's unclear how this approach compares with contemporary GPU implementations of matrix multiplication.

Comments.

Because the author's construction allows us to build models from Haskell primitives, we can benefit from Haskell's rich type system. On the surface this could make it simpler to prevent runtime errors from dimension mismatch, but we may be able to extend this so that it becomes simpler to construct fair and regulation-compliant models by limiting interactions between "controlled" parameters or inputs.

Beyond this additional type safety, it is unclear to me how the author's construction is sufficiently different from the graph constructions in frameworks like TensorFlow. The major implementation difference seems to be that the author's framework constructs the computation graph implicitly. Although the author makes the claim that his construction is a generalized form of automatic differentiation, it's unclear what kinds of functions it can differentiate that a graph-based approach cannot.

REFERENCES

[1] Elliott, C. (2018). The simple essence of automatic differentiation (differentiable functional programming made easy). *CoRR*, abs/1804.00746.

[2] Fong, B., Spivak, D. I., and Tuyéras, R. (2017). Backprop as functor: A compositional perspective on supervised learning. *arXiv preprint arXiv:1711.10455*.