Backprop as Functor: A compositional perspective on supervised learning (Fong et al., [2017])

Overview.

The authors of this paper construct a category theoretic framework for reasoning about neural networks and gradient algorithms. Their core insight is that we can construct gradient descent as a monoidal functor. Backpropagation is therefore functor composition.

In order to build this construct, the authors construct a Learn category in which morphisms are learning algorithms defined by a parameter space, inference function (forward pass), and update function (backward pass). Since update functions cannot compose without additional information, the authors embellish these morphisms with an input space "request function". In the context of neural networks, the request function is equivalent to the gradient of the input parameters, which is an important quantity in sensitivity analysis and generative modeling.

The authors cast gradient descent as a map from the category of differentiable parameterized functions into Learn and show that the chain rule implies that this map forms a symmetric monoidal functor. Since we construct many machine learning algorithms from finite dimensional vector spaces (which form a compact closed category), it's not too surprising that symmetric monoidal structures repeatably pop up in papers that apply categorical constructions to ML.

The authors also show we can use the bimonoidal structure of Learn to combine algorithms serially (pass the output of one algorithm into another) and in parallel (sum the outputs of the algorithms). We can even form more complex combinations, such as the "splitting and combining" in modern neural network constructions. For example, they demonstrate that we can use co-multiplication to express the weight-sharing approach in convolutional and recurrent networks.

Comments.

Although the authors don't discuss the gradient descent variants that power contemporary deep learning systems (like Adam), it seems simple to incorporate them into their construction. These variants mainly differ from vanilla SGD in that they maintain running totals for each model parameter, so we could easily represent them in Learn by modifying the authors' definition of the update and request functions to include these totals.

The authors also don't discuss the more general category of not-necessarily-differentiable supervised learning algorithms (such as random forests). However, we can compose these algorithms both serially and in parallel with ensemble algorithms like voting and stacking, so we may be able to build a similar construction for reasoning about their update rules.

REFERENCES

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