Learning and derivatives, categorically

Geoff Cruttwell
Mount Allison University

CogSci 2022
Workshop on Category Theory for Cognitive Science
July 27th, 2022

Overview

Question: how can we represent learning categorically?

We'll discuss:

- "Action-takers" vs. "learners"
- A category of learners
- The (reverse) derivative as a means to construct a learner
- Conclusion and pointers to research

Claim: at the core of many machine learning/deep learning algorithms is a functor.

Action-takers

We can think of an entity that takes some action as a function

$$f : A \longrightarrow B$$

where A are the inputs and B are the outputs.

- For example, A might be the set of all pictures of a certain size, B might be the unit interval of real numbers $[0, 1]$, and f a function which, given a picture, returns the probability we (or a machine) believes that the picture contains a cat.

Learning

But we want an action-taker, represented by a function $f : A \longrightarrow B$, to *learn*. How can we represent this?

- Imagine that for each set A we have some associated set U_A which represents "ways to update the points of A". (We'll keep this quite general for now).
- One way to think about learning: when we perform an action at some point a, we get an input $f(a) \in B$. We are then told what we would need to do to update $f(a)$ to get the true value b': that is, for each $a \in A$ we are given some update action $u \in U_B$.
- A *learner* should be able to take this update action and know how to use it to update their own state, represented by giving a point of U_A.

Learning continued

That is:

- A learner would have not only a way to take actions, represented by a function
$$f : A \longrightarrow B$$
- But can also receive feedback when they take such an action and update their own internal state accordingly, represented by a function
$$U_f : A \times U_B \longrightarrow U_A.$$

This represents the “bidirectional” nature of learning.

Category of learners

Even better: we can build a category with learners as the morphisms!

- **Objects**: an object in this category is a pair of sets (A, U_A), representing a set of inputs/ouputs and ways to update those inputs/outputs
- **Arrows**: an arrow from (A, U_A) to (B, U_B) is a learner as on the previous slide: a pair (f, U_f) with

$$f : A \longrightarrow B \text{ (an “action”)}$$

and

$$U_f : A \times U_B \longrightarrow U_A \text{ (an “update”)}$$

Category of learners continued

What are identities and composites?

- **Identity**: the identity on (A, U_A) is the identity function on A, together with the map $U_1 : A \times U_A \longrightarrow U_A$ sending (a, u) to u.
- **Composition**: the composite of

  $$(f, U_f) : (A, U_A) \longrightarrow (B, U_B) \text{ and } (g, U_g) : (B, U_B) \longrightarrow (C, U_C)$$

  is given by the composite function $g \circ f : A \longrightarrow C$ and the composite update

  $$U_{g \circ f}(a, w) := U_f(a, U_g(f(a), w))$$

  (where $a \in A, w \in U_C$).

Note that the composite update goes backwards: *first* it uses the update for g, then the update for f.

Learners go by many names...

This category of learners (and variants of it) have been discovered and rediscovered many times[1] and given many different names:

- They are used in database theory, where they are known as *lenses* or *bidirectional transformations*
- Lenses and variants of them (such as *optics*) are used frequently in functional programming
- Jules Hedges has used them to study *open games*
- Valeria de Paiva used them to study Gödel's Dialectica interpretation of logic from a categorical viewpoint

Part of the advantange of category theory is to help make these types of connections!

[1] For more on this, see https://julesh.com/2018/08/16/lenses-for-philosophers/

Action takers to learners?

Big question: if we have some "action-taker", represented by a function

$$f : A \longrightarrow B$$

can we canonically turn it into a "learner"

$$(f, U_f) : (A, U_A) \longrightarrow (B, U_B)?$$

More precisely: is there a functor from the category of action-takers to the category of learners?

Key idea: the (reverse) derivative provides a means to go from action-takers to learners!

Setting up the category

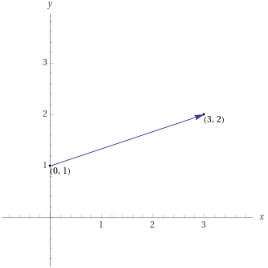
In more detail, suppose we can represent our input/output sets as subsets of Cartesian spaces $\mathbb{R}^n$, so that our functions are of the form

$$f : (A \subseteq \mathbb{R}^n) \longrightarrow (B \subseteq \mathbb{R}^m)$$

For $A \subseteq \mathbb{R}^n$, we will let the set of all vectors

$$U_A = \mathbb{R}^n$$

represent our possible update actions we can make to A.



The (transpose of) the Jacobian

Now suppose our functions/action-takers

$$f : (A \subseteq \mathbb{R}^n) \longrightarrow (B \subseteq \mathbb{R}^m)$$

are differentiable. Then we can build a matrix of all partial derivatives of f. For example, if

$$f(x, y, z) = (x^2 + y^2, 3x\sin(z)),$$

we can build the *transpose* of the Jacobian of f, which gives the matrix

$$\begin{pmatrix} 2x & 3\sin(z) \\ 2y & 0 \\ 0 & 3x\cos(z) \end{pmatrix}$$

(We will see in a minute why we use the *transpose* of the Jacobian).

Evaluating the transpose of the Jacobian

Evaluated at a point, say $a = (x, y, z) = (2, 1, 0)$, we get a matrix of numbers

$$\begin{pmatrix} 2(2) & 3\sin(0) \\ 2(1) & 0 \\ 0 & 3(2)\cos(0) \end{pmatrix} = \begin{pmatrix} 4 & 0 \\ 2 & 0 \\ 0 & 6 \end{pmatrix}$$

Which we could then apply to an vector in $\mathbb{R}^2$, say $u = (-1, 2)$, to get a vector in $\mathbb{R}^3$:

$$\begin{pmatrix} 4 & 0 \\ 2 & 0 \\ 0 & 6 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \end{pmatrix} = \begin{pmatrix} -4 \\ -2 \\ 12 \end{pmatrix}$$

But this is exactly what we need for a learner: given a point of the domain a and an update action (=vector) u in the *codomain*, we have an update action (= vector) in the *domain*.

Transpose of the Jacobian gives a learner

That is, altogether, we started with a function

$$f : A \subseteq \mathbb{R}^3 \longrightarrow B \subseteq \mathbb{R}^2$$

and the transpose of the Jacobian, when evaluated at a point of A and a vector in $\mathbb{R}^2$, gives a vector in $\mathbb{R}^3$, giving us a map which we will write as $R(f)$:

$$R(f) : A \times \mathbb{R}^2 \longrightarrow \mathbb{R}^3,$$

ie.,

$$R(f) : A \times U_B \longrightarrow U_A$$

in other words, we have a learner!

Reverse derivative as a functor

The claim is now that we have a functor

$$(\text{Differentiable functions}) \longrightarrow (\text{Learners})$$

given by sending

$$(A \subseteq \mathbb{R}^n) \text{ to } (A, \mathbb{R}^n)$$

and

$$f : (A \subseteq \mathbb{R}^n) \longrightarrow (B \subseteq \mathbb{R}^m)$$

to

$$(f, R(f))$$

But wait: does this preserve composition?

Reverse derivative preserves composition

In the simplest case, suppose we have differentiable functions

$$f : \mathbb{R} \longrightarrow \mathbb{R}, g : \mathbb{R} \longrightarrow \mathbb{R}$$

If we apply the process in the previous slide to both of f and g individually, then compose them (in the category of learners) we get the update function we get sends

$$(a, w) \mapsto R_f(a, R_g(f(a), w)) = R_f(a, g'(f(a) \cdot w) = f'(a) \cdot g'(f(a)) \cdot w$$

Reverse derivative preserves composition

In the simplest case, suppose we have differentiable functions

$$f : \mathbb{R} \longrightarrow \mathbb{R}, g : \mathbb{R} \longrightarrow \mathbb{R}$$

If we apply the process in the previous slide to both of f and g individually, then compose them (in the category of learners) we get the update function we get sends

$$(a, w) \mapsto R_f(a, R_g(f(a), w)) = R_f(a, g'(f(a) \cdot w) = f'(a) \cdot g'(f(a)) \cdot w$$

whereas if we first compose f and g as functions, then apply the above process, the update function we get sends

$$(a, w) \mapsto R_{g \circ f}(a, w) = (g \circ f)'(a) \cdot w = g'(f(a)) \cdot f'(a) \cdot w$$

by the chain rule. So they are in fact equal!

In other words, **functoriality of this operation holds precisely because of the chain rule!**

More on the reverse derivative and its properties

Why is knowing that it is a functor useful? It tells you that if the way you calculate an action is based on composing many small parts (as in a neural network), then you can make an update by making updates to each of its constitutent parts. This is the essence of backpropagation. Slogan:

$$\text{Backpropagation} = \text{Chain rule} = \text{Functoriality}$$

More on the reverse derivative and its properties

Why is knowing that it is a functor useful? It tells you that if the way you calculate an action is based on composing many small parts (as in a neural network), then you can make an update by making updates to each of its constitutent parts. This is the essence of backpropagation. Slogan:

$$\text{Backpropagation} = \text{Chain rule} = \text{Functoriality}$$

The reverse derivative also has other nice properties:

- If the input update is 0, then the output update will also be 0 ("if you got the right answer, you don't need to make any changes").
- More generally, the (reverse) derivative is linear, so that eg., if the an update $u = u_1 + u_2$, then you can update by summing the result of updating for u_1 and for u_2.

Categorical machine learning setup

This functor is the core of any gradient-based learning process. The full process has additional components to it:

- You might vary how you measure the “degree of wrongness’ of an answer (use a different loss function)
- You might vary how large of an update you make at each step (eg., initially making larger updates then later making smaller updates)
- You might wait to make changes after seeing the results of a number of different feedbacks and/or use the results of previous feedbacks

In the paper “Categorical foundations of Gradient-based learning”, myself and my co-authors describe how to fit these various parts into a categorical setup as well. But at the core of it is always this reverse derivative functor!

Conclusions

In conclusion:

- Very generally, one can represent a "learner" as an arrow in a particular category.
- The (reverse) derivative provides a well-defined (ie., functorial) process to go from certain types of action-takers to certain types of learners.
- This process is at the core of many machine-learning algorithms.
- **Question**: are there other well-defined ways to build learners?

References

At the end of this session we'll provide links to a variety of categorical papers, but here are a few specific to this talk:

- (2022) G. Cruttwell, B. Gavranovic, N. Ghani, P. Wilson, and F. Zanasi. **Categorical foundations of Gradient-based learning**. To be published in the proceedings of ESOP 2022.
- (2019) B. Fong, D. Spivak, and R. Tuyeras. **Backprop as Functor: A compositional perspective on supervised learning**. LICS 2019.
- (2021) D. Shiebler, B. Gavranovic, and P. Wilson. **Category theory in machine learning**. ACT 2021.