A categorical framework for gradient-based learning (part two)

Geoff Cruttwell
Mount Allison University

Joint work with Bruno Gavranović, Neil Ghani,
Paul Wilson, and Fabio Zanasi

March 16th, 2021

Introduction

- General goal: representing some of the structures and ideas in gradient-based learning categorically.
- Note: nothing we do here (yet) solves any problems in machine learning.
- At this point, we're simply giving a new perspective on the subject so that one can ask different questions about it, see how it might be related to other areas, etc.

Today:

- Given certain data, see how to build two endofunctors on our category of "learners".
- Compose these endofunctors with our reverse derivative functor from last time to build a very general gradient descent algorithm.
- See how specific examples of such data give rise to known gradient descent algorithms.
- Discuss future work.

From last time...

How we're modelling gradient-based learning categorically:

- Start with a Cartesian reverse differential category (CRDC) $\mathcal{C}$
- Build the category **Para**($\mathcal{C}$): the category of parameterized maps (a generalization of neural networks). In this category a map from A to B in this category is a pair (P, f) where $f : P \times A \longrightarrow B$.
- Build the category **Lens**($\mathcal{C}$), in which a map from A to B consists of a pair of maps (f, f^*), with

$$f : A \longrightarrow B, \ f^* : A \times B \longrightarrow A.$$

- The category **Para**(**Lens**($\mathcal{C}$)) is our category of "learners"; in this category a map from A to B is a triple (P, f, f^*) where

$$f : P \times A \longrightarrow B$$

and

$$f^* : P \times A \times B \longrightarrow P \times A.$$

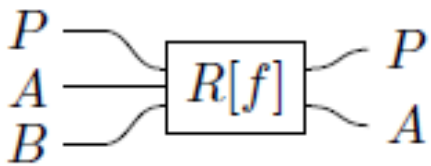
(Which is the type of map that appears in gradient descent algorithms)

Functors to learners

- The R operation in the CRDC $\mathcal{C}$ gives rise to a (2-)functor

$$\mathbf{Para}(R^*) : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathbf{Lens}(\mathcal{C})).$$

Which sends $(P, f) : A \longrightarrow B$ to the lens which has backwards part

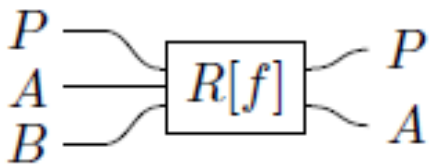


Functors to learners

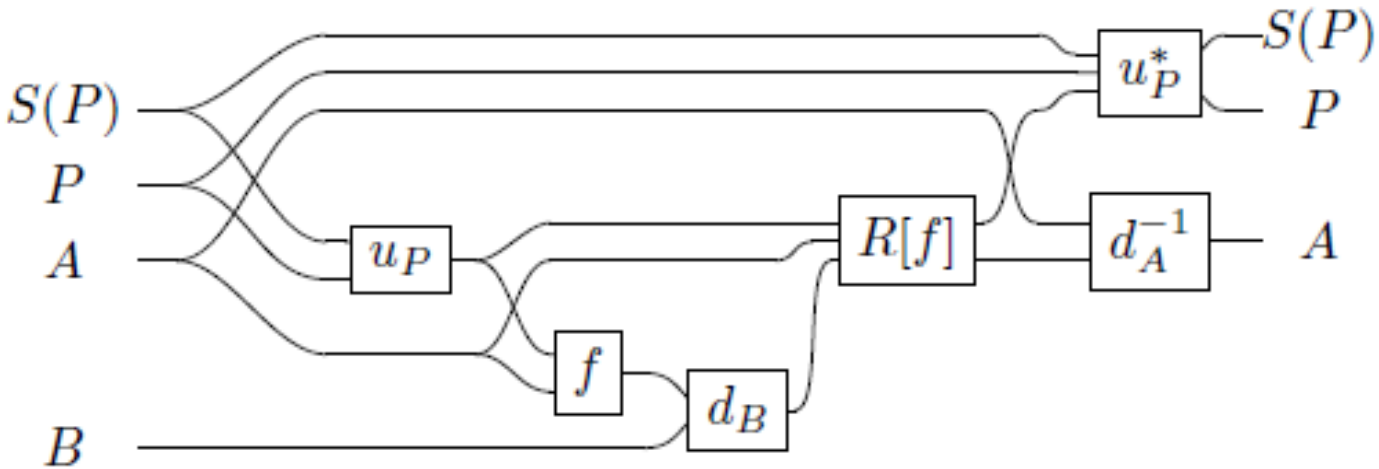
- The R operation in the CRDC $\mathcal{C}$ gives rise to a (2-)functor

$$\mathbf{Para}(R^*) : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathbf{Lens}(\mathcal{C})).$$

  Which sends $(P, f) : A \longrightarrow B$ to the lens which has backwards part



- We are going to build a more general functor out of data S, u, d, which sends $(P, f) : A \longrightarrow B$ to a lens which has backwards part



- For example, in basic gradient descent, $S(P)$ is the terminal object, $u_P = \pi_2$, $u_p^*(p, q) = p - \epsilon q$, $d_B(b, b') = b - b'$.

The more general functor to learners

- It's not obvious that this construction is functorial (and of course it depends on what properties S, u and d have).
- Our plan is to see it as built up out of smaller pieces, each piece of which is relatively easy to verify functoriality.
- The functoriality of the smaller pieces has its origin in a very simple endofunctor one can define on any category.

A silly endofunctor on any category

Lemma

If $\mathcal{C}$ is any category, and we have, for each object A, an isomorphism $i_A : A \longrightarrow A$, then there is an endofunctor

$$I : \mathcal{C} \longrightarrow \mathcal{C}$$

defined as the identity on objects, and sends $f : A \longrightarrow B$ to

$$A \xrightarrow{i_A^{-1}} A \xrightarrow{f} B \xrightarrow{i_B} B$$

While not particularly interesting in their own right (such functors are naturally isomorphic to the identity functor), we can usefully generalize this idea in two different ways to categories of the form **Para**($\mathcal{C}$).

Endofunctors on **Para**($\mathcal{C}$) I

Lemma

Suppose $\mathcal{C}$ is a Cartesian category, and we have, for each object A, an isomorphism $d_A : A \longrightarrow A$, and this collection is product-preserving ($d_{A\times B} \cong d_A \times d_B$), then there is an endo-2-functor

$$D : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathcal{C})$$

defined as the identity on objects, sends $(P, f) : A \longrightarrow B$ to P with the map

$$P \times A \xrightarrow{1 \times d_A^{-1}} P \times A \xrightarrow{f} B \xrightarrow{d_B} B,$$

and sends a 2-cell $r : (P, f) \longrightarrow (P', f')$ to itself.

(Note this is **not** just **Para** applied to the functor from the previous slide - that would also use the isomorphisms on the parameter part, which this does not.)

For example...

We'll be applying this ideas to the category **Lens**($\mathcal{C}$), in which case each d_A would be an invertible map in **Lens**($\mathcal{C}$), ie., a pair of maps

$$d_A : A \longrightarrow A, d_A^* : A \times A \longrightarrow A$$

forming an invertible lens. The basic example of this in $\mathcal{C}$ the smooth category has

$$d_A = 1_A, d_A^*(p, p') = p - p'$$

(calculating how far the current output is from the desired output). One can check that this is indeed an invertible map in **Lens**($\mathcal{C}$).

Endofunctors on **Para**($\mathcal{C}$) II

We can also make an endofunctor which changes the parameters:

Lemma

Suppose $\mathcal{C}$ is a Cartesian category, and we have, for each object P, an isomorphism $u_A : A \longrightarrow A$, and this collection is product-preserving ($u_{A\times B} \cong u_A \times u_B$), then there is an endo-2-functor

$$U : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathcal{C})$$

defined as the identity on objects, sends $(P, f) : A \longrightarrow B$ to P with the map

$$P \times A \xrightarrow{u_P \times 1} P \times A \xrightarrow{f} B,$$

and sends a 2-cell $r : (P, f) \longrightarrow (P', f')$ to

$$P' \xrightarrow{u_{P'}} P' \xrightarrow{r} P \xrightarrow{u_P^{-1}} P$$

For example...

Applying these ideas in **Lens**($\mathcal{C}$), u_P would have to be an invertible map in **Lens**($\mathcal{C}$), ie., a pair of maps

$$u_P : P \longrightarrow P, u_P^* : P \times P \longrightarrow P$$

In basic gradient descent, with $\mathcal{C}$ the smooth category, we will take

$$u_P = 1_P, \;\; u_P^*(p, p') = p - \epsilon p'$$

(for any $\epsilon \neq 0$).

More general update data

However, we'll also want to consider updates which hold on to some "state" and use that in future updates. For this, we need a more general version of the second endofunctor.

Definition

Suppose $\mathcal{C}$ is a Cartesian category. **Update data** on $\mathcal{C}$ consists of:

- a product-preserving endofunctor $S : \mathcal{C} \longrightarrow \mathcal{C}$;
- for each object P of $\mathcal{C}$, a map

$$u_P : S(P) \times P \longrightarrow P$$

  which is invertible in its second variable

such that the u_P's respect products in an appropriate way.

(We'll see examples of this soon.)

Endofunctors on **Para**($\mathcal{C}$) III

Lemma

Suppose $\mathcal{C}$ is a Cartesian category, and we have update data $(S, \{u_P\})$ on $\mathcal{C}$. Then there is an endo-2-functor

$$U : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathcal{C})$$

defined as the identity on objects, sends $(P, f) : A \longrightarrow B$ to $S(P) \times P$ (note the change in parameter object!) with the map

$$S(P) \times P \times A \xrightarrow{u_P \times 1} P \times A \xrightarrow{f} B$$

and sends a 2-cell $r : (P, f) \longrightarrow (P, f')$ to the composite

$$S(P') \times P' \xrightarrow{\langle \pi_1, u_{P'} \rangle} S(P') \times P' \xrightarrow{S(r) \times r} S(P) \times P \xrightarrow{\langle \pi_1, u_P^{-1} \rangle} S(P) \times P.$$

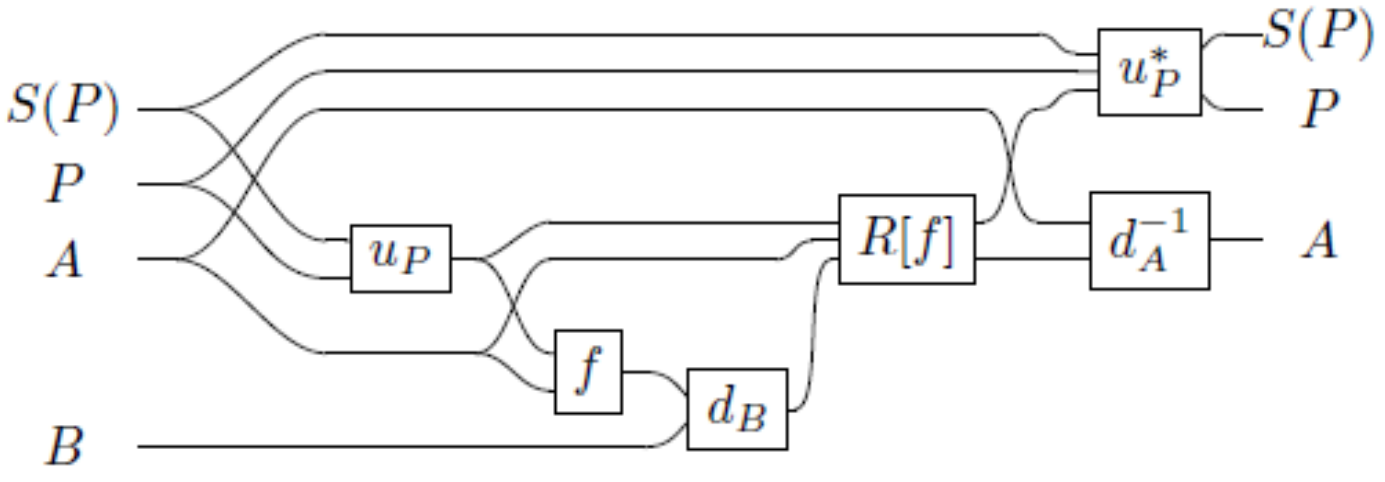
Putting it all together

Lemma

If $\mathcal{C}$ is a CRDC, and we have both a family of isomorphisms $\{d_A\}$ and update data $(S, \{u_P\})$ in **Lens**($\mathcal{C}$)*, then we get a 2-functor*

$$U \circ D \circ \mathbf{Para}(R^*) : \mathbf{Para}(\mathcal{C}) \longrightarrow \mathbf{Para}(\mathbf{Lens}(\mathcal{C}))$$

which sends a parametrized map $(P, f) : A \longrightarrow B$ to the map with object $S(P) \times P$, forward part of the lens f, and backward part



(Actually, this is the case when the forward part of each d_A is the identity...all our examples will have this.)

Example: basic gradient descent

With d_A as before, basic gradient is the case when

$$S(P) = 1$$

for all P,

$$u_P : 1 \times P \longrightarrow P$$

is the second projection, and

$$u_P^* : 1 \times P \times P \longrightarrow 1 \times P$$

maps $(*, p, p')$ to

$$(*, p - \epsilon p')$$

(for some non-zero constant ϵ).

Example: gradient descent with momentum

An important variant of gradient descent is gradient descent with *momentum*: it stores the previous change and uses a fraction of that to also update the next change. In our setup we get this by taking

$$S(P) = P$$

for all P,

$$u_P : P \times P \longrightarrow P$$

is the second projection, and

$$u_P^* : S(P) \times P \times P \longrightarrow S(P) \times P$$

maps (v, p, p') to

$$(v', p - v')$$

where $v' = \gamma v + \epsilon p'$ (for some non-zero constants γ, ϵ).

Example: gradient descent with Nesterov momentum

A further important variation of this is Nesterov momentum, which "looks ahead" from the current point by applying the previous update to the current point and calculating the required change at that point instead. This is achieved in our setup by using a lens with non-trivial forward part: we take

$$S(P) = P$$

for all P,

$$u_P : P \times P \longrightarrow P$$

sends $(v, p) \mapsto p - \gamma v$ (note the difference from the previous example!) and

$$u_P^* : S(P) \times P \times P \longrightarrow S(P) \times P$$

is as before: it maps (v, p, p') to

$$(v', p - v')$$

where $v' = \gamma v + \epsilon p'$ (for some non-zero constants γ, ϵ).

More examples

I think it's really neat how lens and their composition exactly give us what we need to implement some of these variants of gradient descent!

One can show that other variants of gradient descent also fit into this setup:

- Adagrad
- Adam
- Nadam
- etc.

Some store several different pieces of information from previous updates, and so use, for example, $S(P) = P \times P$.

A different error map

A variant of the basic mean-squared error is cross-entropy loss, which results in a lens d_A whose forward part is the identity, and whose backwards part is the map

$$d_A^* : A \times A \longrightarrow A$$

given by

$$(a, a') \mapsto \frac{\exp(a_i)}{\sum_j \exp(a_j)} - a'$$

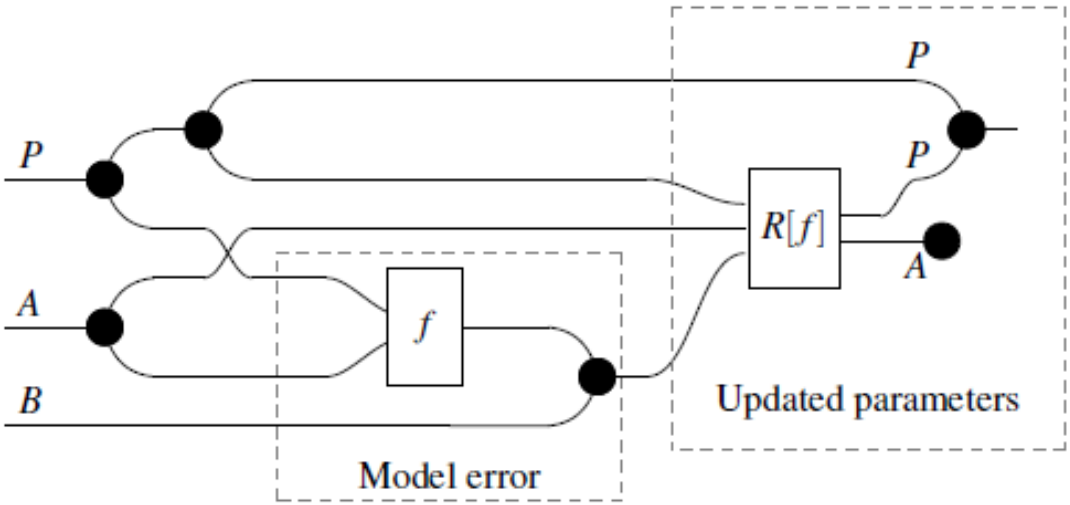
This can be combined with any of the previous update maps (instead of mean-squared error).

Polynomial and boolean circuit example

- A different example comes from considering the CRDC

$$\mathcal{C} = \text{polynomials over } \mathbb{Z}_2.$$

- In "Reverse derivative ascent", the authors define a learning algorithm in this category which they draw graphically as



- This is also an example of our framework, with $S(P) = 1, d_A^* = u_A* = +$.
- In the paper, the authors show how to extend this idea to boolean circuits.

Python code

- One advantage of our approach is that it makes building a learning algorithm very "plug and play": you specify an update map and an error map, and the framework gives you an explicit recipe for how to build a learning algorithm from that data.
- We've built some Python code to do this: see https://bit.ly/3rNkyNX.
- We've used this code to try some standard machine learning problems (such as MNIST), and achieved good accuracy and speed.

Future work I

We have lots of avenues we'd like to explore as follow-ups to these ideas. These include:

- **Partiality**: generalizing to categories of *partial* maps (represented axiomatically by restriction categories). This should allow us to handle some of the other neural networks and error maps which occur in practice but aren't differentiable everywhere.
- **Meta-learning**: Some advanced machine-learning techniques involve changing the learning algorithm itself as time passes. We believe we can model this by using the category **Para**(**Para**(**Lens**($\mathcal{C}$))).
- **Dreaming**: Understanding how the A output that our model builds relates to "dreaming" in machine learning.

Future work II

- **Learning on manifolds**: by generalizing the lens construction to dependent lenses (= dual fibration of the codomain fibration) and CRDCS to "cotangent categories" (the reverse analogue of tangent categories), we should be able to model learning in categories of smooth manifolds.
- **Non gradient-based learning**: Everything we've said above applies to any section of the dual of the simple fibration (that is, instead of considering the functor $R^* : \mathcal{C} \longrightarrow \mathbf{Lens}(\mathcal{C})$ coming from a CRDC, instead start with an arbitrary functor $L : \mathcal{C} \longrightarrow \mathbf{Lens}(\mathcal{C})$), and this should help model non gradient-based learning.
- **Use of 2-categorical structure**: Everything we've done comes with 2-categorical structure; we need to investigate further how this can be used to compare and contrast different learning algorithms.

References

The paper the talk is based on is

- Cruttwell, G., Gavranović, B., Ghani, N., Wilson, P., and Zanasi, F. **Categorical foundations of gradient-based learning**, arXiv:2103.01931, 2021.

The most closely related categorical work is

- Fong, B., Spivak, D., and Tuyeras, R. **Backprop as functor**, LICS 2019.

Other references include

- Cockett, R., Cruttwell, G., Gallagher, J., Lemay, J-S., MacAdam, B., Plotkin, G., and Pronk, D. **Reverse derivative categories**. CSL 2020.
- Gavranović, B. **Compositional deep learning**, arXiv:1907.08292, 2019.
- Hedges, J. *Lenses for philosophers* (blog post at https://julesh.com/2018/08/16/lenses-for-philosophers/, contains many other references and history about lenses)
- Wilson, P. and Zanasi, F. **Reverse derivative ascent: a categorical approach to learning boolean circuits**. Proceedings of ACT 2020.