RETROFUNCTIONS

This is an attempt to present back-propagation in an especially simple way.

GRAPH APPROACH

We can build a computation graph for a complicated loss function from very simple parts. These parts require a forward function, a **retro function**, and —in most cases — local storage.

We'll call these parts **chips.** Each chip includes a forward function $\overrightarrow{f}$ and a retro function $\overleftarrow{f}$. Because these chips also have local memory and update their own parameters, they aren't a pair of pure mathematical functions.

Many chips are of the form $f : \mathbb{Q} \to \mathbb{Q}$.[1] But some, which we cannot do without, are of the form $f : \mathbb{Q}^n \to \mathbb{Q}$ or $f : \mathbb{Q} \to \mathbb{Q}^n$.

NOTATION

We use $\mathbf{e}_i$ for basis vectors, so the vector $(1, 2, 3)$ is written $1\mathbf{e}_1 + 2\mathbf{e}_2 + 3\mathbf{e}_2$.

AMPLIFIER

We want to be able to scale or amplify a rational number.

Let $\overrightarrow{f}(x) = px$. Then $\overleftarrow{f}(y) = py$. This example emphasizes that $\overrightarrow{f}$ and $\overleftarrow{f}$ are not functional inverses, though their relation is analogous to that between inverses.

Here p is the local parameter of the chip. It must be updated with every calculation of $\overleftarrow{f}(y) = py$ by $p \leftarrow p - \alpha yp$.

Note that α is a fixed parameter of the amplifier chip which should be shared by *all* chips. While p evolves, α (the learning rate) stays fixed.

1 CLONE

We need to split or clone a single number in order to send it to more than one

[1] Note that I use $\mathbb{Q}$ rather than $\mathbb{R}$ for philosophical and technical reasons. The most direct reason is that digital computers use only a finite subset of the rational numbers.