Combinatory adjoints and differentiation

Martin Elsman (DIKU), Fritz Henglein (DIKU), Robin Kaarsgaard (U. Edinburgh), Mikkel Kragh Mathiesen (DIKU), Robert Schenck (DIKU)

DIKU

MSFP 2022
Munich, 2022-04-02

What is a derivative? The SD view

- Leibniz: $f : \mathbb{R} \to \mathbb{R}$, $f'(x) = a \in \mathbb{R}$ if

$$\lim_{|dx| \to 0} \frac{|f(x + dx) - (f(x) + a \cdot dx)|}{|dx|} = 0$$

- Jacobi: $f : \mathbb{R}^n \to \mathbb{R}^m$, $f'(v) = M \in \mathbb{R}^{m \times n}$ if

$$\lim_{||dv|| \to 0} \frac{||f(v + dv) - (f(v) + M \star dv)||}{||dv||} = 0$$

- Fréchet (total): $f : V \to W$ (Banach), $f'(v) = h \in V \multimap W$ if

$$\lim_{||dv||_V \to 0} \frac{||f(v + dv) - (f(v) + A(dv))||_W}{||dv||_V} = 0$$

Observe: Input to derivative is a single value. Output is a *function* from input to output differentials.

What is a derivative? The AD view

- Gateaux (directional) differential: $f : V \to W$ (Banach), $f'(v, dv) = dy \in W$ if

$$dy = \lim_{t \to 0} \frac{||f(v + t \cdot dv) - f(v)||_W}{||t||_K}$$

- Define $f^{[fad]}(v, dv) = (f(v), f'(v, dv))$. Then

$$(g \circ f)^{[fad]} = g^{[fad]} \circ f^{[fad]}.$$

- Idea: Interpret function f over *dual tensors* $(v, dv) \in \mathbb{R}^{n_1 \times \ldots \times n_k} \times \mathbb{R}^{n_1 \times \ldots \times n_k}$ instead of $v \in \mathbb{R}^{n_1 \times \ldots \times n_k}$.
- Easy to implement for sequential source code:
  - Use your existing compiler or interpreter for the program that defines f.
  - Just replace standard abstract data type implementations for numbers, vectors, tensors by *dual numbers, vectors, tensors*.
  - But now we need *two* inputs: a primal value v *and* an input differential dv.

Hilbert spaces

- Hilbert space: Vector space $(V, +, \cdot, 0)$ + inner product $\odot$ + limits; e.g. Euclidean space.
- Constructions:

$$U, V, W ::= 0 \mid K \mid \bigoplus_{x \in X} V_x \mid V \otimes W$$

where X is a set.

  - Here: Finite sets $X ::= \mathbf{n} | X_1 \times X_2 \mid \ldots$ and $K = \mathbb{R}$.
- Direct sums $\bigoplus_{x \in X} V_x$, including $V_1 \times \ldots \times V_n$ and copowers V^X (e.g. $\mathbb{R}^{\mathbf{n}}$)
- Tensor products, $V \otimes W$ = the *terms*

$$w ::= 0 \mid k \cdot w \mid w_1 + w_2 \mid u \otimes v,$$

treated modulo vector space axioms and

$$(k \cdot v) \otimes w = k \cdot (v \otimes w) = v \otimes (k \cdot w)$$
$$(v_1 + v_2) \otimes w = (v_1 \otimes w) + (v_2 \otimes w)$$
$$v \otimes (w_1 + w_2) = (v \otimes w_1) + (v \otimes w_3).$$

Fréchet differentiation calculus

Theorem

$$
\begin{array}{rcll}
(g \circ f)'(v) & = & g'(f(v)) \bullet f'(v) & \\
{K_w}'(v) & = & 0 & \\
h'(v) & = & h & \textit{if } h : V \multimap W \\
\diamond'(u, v) & = & (u\diamond) \bullet \pi_2 + (\diamond v) \bullet \pi_1 & \textit{if } \diamond : U \times V \to_2 W \\
(\Pi_{x \in X} f_x)'(v) & = & \Delta((\Pi_{x \in X} {f_x}')(v)) & \textit{if } f_x : V_x \to W_x
\end{array}
$$

where $\bullet$ linear composition, $K_w(v) = w$, $\diamond$ bilinear, $(u\diamond)(v) = u \diamond v$, $(\diamond v)(u) = u \diamond v$, $\Delta(f_1, \ldots, f_n)(v_1, \ldots, v_n) = (f_1(v_1), \ldots, f_n(v_n))$.

Note:

- 5 rules: 3 for multilinear functions (constant, linear, bilinear), plus sequential (chain rule) and parallel composition.
- Special cases of parallel composition:

$$
\begin{array}{rcl}
(f_1 \times f_2)'(v_1, v_2) & = & {f_1}'(v_1) \times {f_2}'(v_2) \\
(\mathit{map}\, f)'(v) & = & \Delta(\mathit{map}\, f'(v))
\end{array}
$$

Adjoints

- $f^* : W \multimap V$ is *adjoint* of $f : V \multimap W$ if

$$f(v) \odot w = v \odot f^*(w)$$

  for all $v \in V, w \in W$.
    - Example: $+^* = \mathrm{dup}$.
- Adjoint of linear map expressed as matrix is its *transpose*:

$$(M\star)^* = (M^T\star).$$

- Standard approach: Represent linear maps as matrices. Perform transposition to implement adjoint.
    - Bad idea for high-dimensional vector spaces!
    - Adjoint of id is id, but matrix representation may be huge. (Worse for other linear maps whose matrices have no zero entries.)
- Better idea: Symbolic representation of linear functions; calculus for computing adjoints symbolically.

Adjoint calculus

Theorem

Let X, Y be finite sets, $R \subseteq X \times Y$, and $R^T = \{(y,x) \mid (x,y) \in R\}$.

$$
\begin{aligned}
\mathrm{id}^* &= \mathrm{id} \\
(g \bullet f)^* &= f^* \bullet g^* \\
0^* &= 0 \\
(v*)^* &= (v^T *) \\
(*w)^* &= (*w^T) \\
(\iota_x^X)^* &= \pi_x^X \\
(\Pi_{x \in X} f_x)^* &= \Pi_{x \in X} f_x{}^* \\
\mathrm{red}_R{}^* &= \mathrm{red}_{R^T}
\end{aligned}
$$

Furthermore, the inverses of unitary operators are also their adjoints.

where $(u \otimes v) * (v' \otimes w) = (v \odot v') \cdot (u \otimes w)$ is *tensor contraction*;
$\mathrm{red}_R(v) = \oplus_{y \in Y} \sum_{(x,y) \in R} v_x$ is *relational reduction*.

Adjoint affine interpretation

Compute value and symbolic (not matrix) adjoint derivative of function at given input x. (No additional input required.)

$$
\begin{array}{rcl}
(g \circ f)^{[1r]}(x) & = & \textbf{let } (fx, f'xa) = f^{[1r]}(x) \textbf{ in} \\
 & & \textbf{let } (gfx, g'fxa) = g^{[1r]}(fx) \textbf{ in} \\
 & & \quad (gfx, f'xa \bullet g'fxa) \\
{K_w}^{[1r]}(x) & = & (w, 0) \\
h^{[1r]}(x) & = & (h(x), h^*) \\
\diamond^{[1r]}(x) & = & \underline{\textbf{let}} \ (u, v) = x \textbf{ in} \\
 & & \quad (u \diamond v, \iota_2^2 \bullet (u\diamond)^* + \iota_1^2 \bullet (\diamond v)^*) \\
(\Pi_{y \in Y} f_y)^{[1r]}(x) & = & \textbf{let } (w, d) = unzip((\Pi_{y \in Y}(\lambda x. {f_y}^{[1r]}(x)))(x)) \textbf{ in} \\
 & & \quad (w, \Delta(d))
\end{array}
$$

Theorem

If $f^{[1r]}(x) = (y, h)$ then $y = f(x)$ and $h = f'(x)^*$.

where $f'(x)$ is Fréchet derivative.

Why adjoints?

- "Cheap gradients": Efficiently Computing gradient ∇f of scalar function $f : \mathbb{R}^n \to \mathbb{R}$ for $n \gg 0$.
  - Using derivative to compute gradient requires application of derivative to each basis vector of $\mathbb{R}^n$:

    $$\nabla f(x) = (f'(x)(e_1), \ldots, f'(x)(e_n))$$

    where $e_i = (0, \ldots, 0, 1, 0, \ldots 0)$ with 1 in i-th position.
  - Using adjoint to compute gradient requires only one application to (single) basis vector 1 of $\mathbb{R}$:

    $$\nabla f(x) = f'(x)^*(1).$$

- Applications independent of differentiation:
  - Computational science (PDEs, error estimation, inverse problems, etc)
  - Physics (quantum field theory)

More information and future work

- More details in the paper:
  - Relation of derivatives to each other
  - Expression swell myth debunked
  - Tensor decomposition/tensor products as efficient data structures for low-rank matrices
  - Examples (including neural networks)
- Future work:
  - DSL for binary relations (to avoid enumeration)
  - Fréchet: Functional DSL with affine (adjoint) interpretation, embedded in Haskell
  - Caddy: DSL embedded in Standard ML
  - Second-order/higher-order differentiation based on quadratic/polynomial (adjoint) interpretation
  - Generating Futhark code for high-performance (parallel, GPU) execution of (adjoint) derivatives.

Thank you!