1 Einops

Reference: https://github.com/arogozhnikov/einops

1.1 Features

- self-documenting notation for layouts of input and output arrays
- low number of backend functions to implement
- focus on data rearrangements and simple transformations (axes reordering, decomposition, composition, reduction, repeats)
- focus on 1 tensor/array transformations
- notation uses strings
- supported notations: named axis, anonymous axis, unitary axis, ellipsis, (de)compose parenthesis
- supports a list of arrays as input with implied additional outer dimension corresponding to the list
- inferrable dimension sizes, given partial info as parameters
- hide backend framework inconsistency of notations for common array rearrangement operations
- use of proxy classes for specific backends
- caching of tensor type map to backend type for performance
- caching of patterns and axes
- caching of patterns, axes, and input shape: compute unknown axis sizes and shape verification on first time, otherwise reuse sequence of commands previously generated
- inverse transformations are easy to read off by switching patterns for input and output

1.2 Approaches

- evidence based for API design, via real world use cases
- explicit separation of a few functions over 1 function, for better error messages
- consideration of adoption friction and ease of use

1.3 Known Issues

- does not enforce axes alignment between operations
- no means of integrated analysis/tracking of shapes

2 Tensor Derivatives

Index notation: $*(s_1, s_2, s_3)$, where
s_1: input index set
s_2: input index set
s_3: output index set

2.1 Properties

- associative

  let
  $s_3 \subseteq s_1 \cup s_2$
  $s_4 \cap (s_1 \cup s_2) = \emptyset$
  then,
  $*(s_3 s_4, s_4, s_3)(*(s_1, s_2 s_4, s_3 s_4)(A, B), C)$
  $= *(s_1, s_2, s_3)(A, *(s_2 s_4, s_4, s_2)(B, C))$

  order of evaluations:
  $(s_1 \to s_2 s_4) \to s_4 \to s_3$
  vs
  $s_1 \to (s_2 s_4 \to s_4) \to s_3$

- commutative

  $(s_1, s_2, s_3)(A, B) = *(s_2, s_1, s_3)(B, A)$

- distributive

  $*(s_1, s_2, s_3)(A, B) + *(s_1, s_2, s_3)(A, C)$
  $= *(s_1, s_2, s_3)(A, B + C)$
  where $s_3 \subseteq s_1 \cup s_2$

2.2 Derivative Definition

$f : \mathbb{R}^{n_1 \times .. \times .. n_k} \to \mathbb{R}^{m_1 \times .. \times m_l}$
$D \in \mathbb{R}^{m_1 \times .. \times m_l \times n_1 \times .. \times n_k}$

$\lim_{h \to 0} \frac{\|f(x+h) - f(x) - D \circ h\|}{\|h\|} = 0$
$\iff D$ is a derivative of f at x

where inner tensor product is:
$D \circ h = *(s_1 s_2, s_2, s_1)(D, h)$

3 Forward Mode

$\sum_i \frac{\partial v_i}{\partial x_j} \frac{\partial f}{\partial x_j} = \frac{\partial f}{\partial x_j}$
where x_j are leaf input variables and
where pushforwards of predecessor nodes v_i are computed and cached by the time $\frac{\partial f}{\partial x_j}$ is computed

notation: let $\bar{v} = \frac{\partial v}{\partial x_j}$ be the pushforward of v

Generalized cases of local node connections:

- unary function
- element-wise unary function
- binary addition
- binary multiplication

We seek to compute pushforwards for the above types of ops.

3.1 Pushforward of Unary Function

todo

3.2 Pushforward of Elementwise Unary Function

todo

3.3 Binary Addition

todo

3.4 Binary Multiplication

todo

4 Reverse Mode

todo