

# Chebyshev KAN Derivations

da1sypetals.iota@gmail.com

## 1. Notations

- Let  $a$  be the cheby tensor in the code, and  $a_d(b, i)$  be `cheby[b, i, d]` where  $b$  for batch,  $i$  for the index of input features, and  $d$  for the current operating degree.
- Let  $x$  be the input tensor `x`, and  $x(b, i)$  be `x[b, i]`, where  $b, i$  are explained above.
- Let  $b_d$  be the derivative of cheby tensor w.r.t. input tensor `x`.
  - Specifically,

$$b_d(b, i) = \frac{\partial a_d(b, i)}{\partial x(b, i)} \quad (1)$$

- If not specified, we denote  $a_d = a_d(b, i)$ ,  $b_d = b_d(b, i)$  and  $x = x(b, i)$  for any  $(b, i)$  pair.

## 2. Forward

- The formula is recursive.
- $\forall b, i$ , we have

$$\begin{aligned} a_0 &= 0 \\ a_1 &= 1 \\ a_d &= 2x \cdot a_{d-1} - a_{d-2} \end{aligned} \quad (2)$$

## 3. Backward

- Taking derivative on both sides of equation (2) w.r.t.  $x$ , we get:

$$\frac{\partial a_d}{\partial x} = 2 \left( a_{d-1} + x \frac{\partial a_{d-1}}{\partial x} \right) - \frac{\partial a_{d-2}}{\partial x} \quad (3)$$

$$b_d = 2a_{d-1} + 2x \cdot b_{d-1} - b_{d-2}$$

Specifically,

$$a_0 = 0, a_1 = 1 \quad (4)$$

- Therefore, given loss function  $\mathcal{L}$ , we can compute the gradient:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x} &= \sum_d \frac{\partial L}{\partial a_d} \cdot \frac{\partial a_d}{\partial x} \\ &= \sum_d \frac{\partial L}{\partial a_d} \cdot b_d\end{aligned}\tag{5}$$

where the first term flows in from the previous layers, and the second term is computed recursively inside CUDA kernel where each thread computes for a unique  $(b, i)$  pair.