Modular Duality in Deep Learning

Zeyu Bian
zebian@ucsd.edu

University of Science and Technology of China

May 11, 2025

Content

Content

Shall we directly subtract the gradient from the weights?

The author argues we shouldn't, due to the reason that

- The geometry of the loss function may be non-isotropic, so we need to adjust the **size** and **direction** of the gradient.

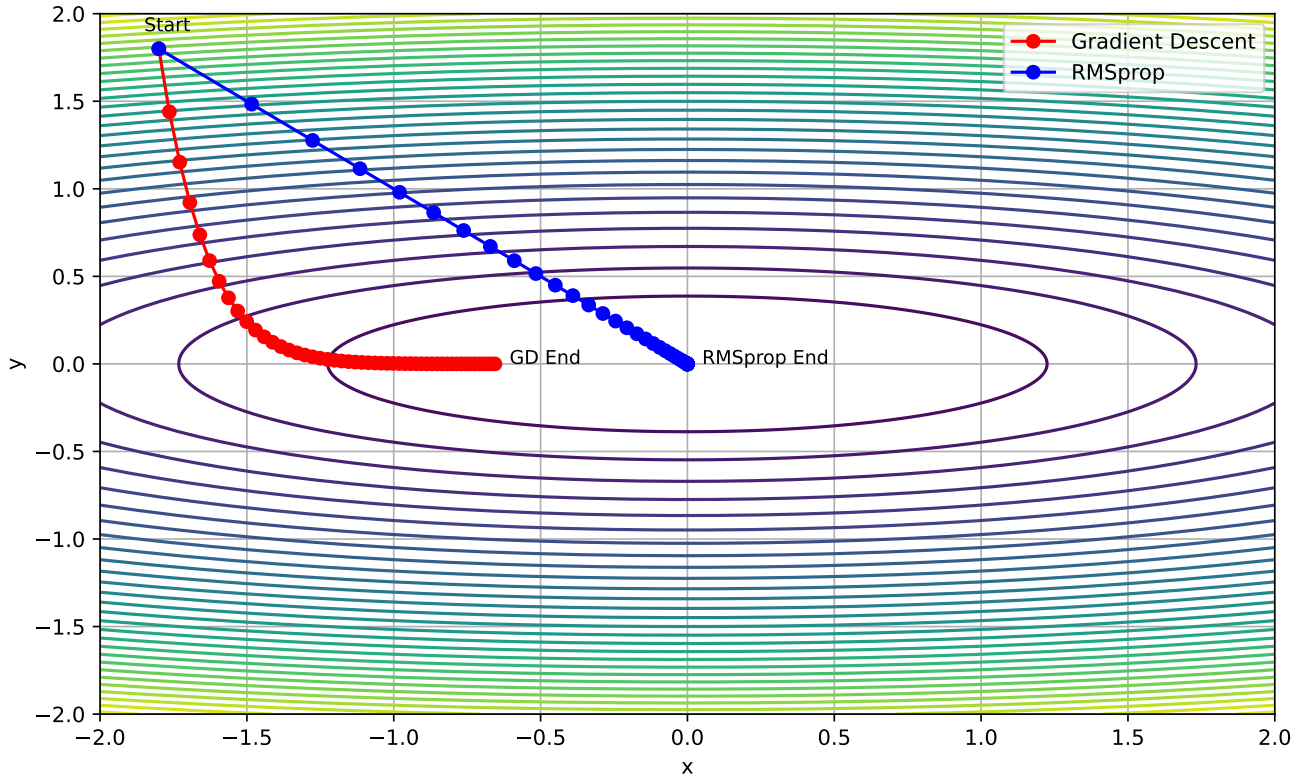


Figure 1: GD vs. RMSprop dynamic on $Loss(x, y) = 0.1x^2 + y^2$

Intuition From Dual Space Perspective

Definition (Dual Space)

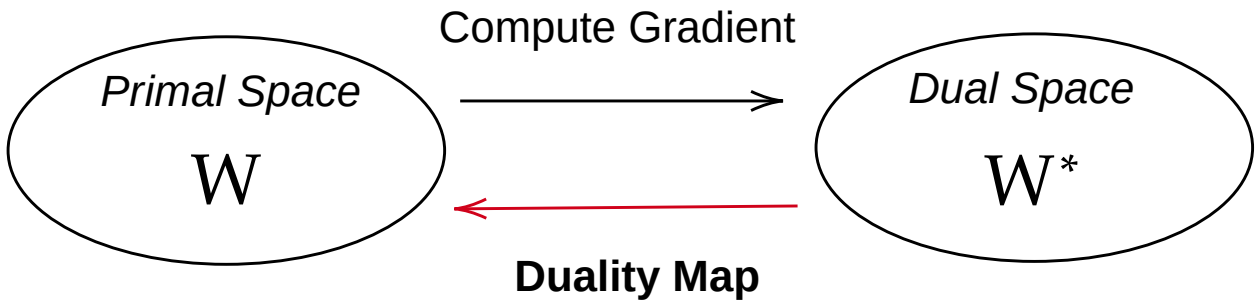
Given a vector space $\mathcal{V}$,

- Say $f : \mathcal{V} \to \mathbb{R}$ is a *linear functional* on $\mathcal{V}$ if f is linear.
- Define the dual space $\mathcal{V}^*$ to be the set of *linear functionals* on the vector space $\mathcal{V}$.

Consider the Taylor expansion of the loss function:

$$\mathcal{L}(\boldsymbol{w} + \Delta \boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})^\top \Delta \boldsymbol{w} + \text{ higher-order terms } . \quad (1)$$

The gradient $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$ is a linear functional paired with $\Delta \boldsymbol{w}$, so that $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \in \mathcal{W}^*$



Descent in Normed Spaces

We use the diamond operator $\diamond$ denotes tensor contraction. [1]
Suppose that we wish to use gradient descent to minimize a loss function $\mathcal{L} : \mathcal{W} \to \mathbb{R}$ over a weight space $\mathcal{W} = \mathbb{R}^N$. Then we may desire the following three properties :

1. The loss function is differentiable, meaning that the gradient map $\nabla_{\boldsymbol{w}} \mathcal{L} : \mathcal{W} \to \mathcal{W}$ exists;
2. The weight space $\mathcal{W}$ carries a norm $\|\cdot\| : \mathcal{W} \to \mathbb{R}$.
3. The loss is Lipschitz smooth in the norm $\|\cdot\|$, with sharpness constant $\lambda > 0$, which implies:

$$\mathcal{L}(\boldsymbol{w} + \Delta \boldsymbol{w}) \leq \mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \diamond \Delta \boldsymbol{w} + \frac{\lambda}{2} \|\Delta \boldsymbol{w}\|^2. \tag{2}$$

[1] e.g. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are compatible 4d tensors, then $\boldsymbol{x} \diamond \boldsymbol{y} = \sum x_{ijkl} y_{ijkl}$

Descent in Normed Spaces

Under the above three properties, the weight update given by

$$\Delta \boldsymbol{w} = \arg\min \left[\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \diamond \Delta \boldsymbol{w} + \frac{\lambda}{2} \|\Delta \boldsymbol{w}\|^2 \right]. \tag{3}$$

is guaranteed to reduce the loss (since the minimum is no greater than 0) We further give two definition to help discussions.

Definition (Dual Norm)

Given a norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$, the dual norm $\|\cdot\|^\dagger$ of a vector $\boldsymbol{g} \in \mathbb{R}^n$ is given by:

$$\|\boldsymbol{g}\|^\dagger := \max_{\boldsymbol{t} \in \mathbb{R}^n : \|\boldsymbol{t}\| = 1} \boldsymbol{g}^\top \boldsymbol{t}$$

Example

The dual norm of $\|\cdot\|_p$ is $\|\cdot\|_q$ where (p,q) is conjugate pair, $1 \leq p \leq \infty$

Descent in Normed Spaces

Definition (Duality map based on a norm)

Given a norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$, we consider the duality map:

$$\text{dualize}_{\|\cdot\|}\boldsymbol{g} := \underset{\boldsymbol{t}\in\mathbb{R}^n:\|\boldsymbol{t}\|=1}{\arg\max}\ \boldsymbol{g}^\top \boldsymbol{t},$$

where, if the arg max is not unique, dualize $_{\|\cdot\|}$ returns any maximizer.

We have the following expression for minimizing (3) in vector:

Proposition (Steepest descent under a norm)

For any $\boldsymbol{g} \in \mathbb{R}^n$, $\lambda \geqslant 0$, norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ with dual norm $\|\cdot\|^\dagger$ and duality map dualize $\|_{\cdot\|}$:

$$\underset{\Delta\boldsymbol{w}\in\mathbb{R}^n}{\arg\min}\left[\boldsymbol{g}^\top \Delta\boldsymbol{w} + \frac{\lambda}{2}\|\Delta\boldsymbol{w}\|^2\right] = -\frac{\|\boldsymbol{g}\|^\dagger}{\lambda} \times \textit{ dualize }_{\|\cdot\|}\boldsymbol{g}.$$

$\|\cdot\|$ influences the direction, while the sharpness λ influences the size.

Norm Choice

We would ideally like the optimal step-size to remain invariant as we scale the width and the depth of the network. Thus, a fundamental problem is to design a norm such that:

- The upper bound (2) actually holds (and is not hopelessly lax).
- The corresponding sharpness constant λ is **invariant** to the relevant architectural dimensions.

We shall see later that the **modular norm** meets these requirements.

Example (Duality Map)

- Euclidean norm: dualize ${}_{\|\cdot\|_2}\boldsymbol{g} = \boldsymbol{g}/\|\boldsymbol{g}\|_2$
- Infinity norm: dualize ${}_{\|\cdot\|_\infty}\boldsymbol{g} = \operatorname{sign}(\boldsymbol{g})$

Norm Examples

Definition (Induced operator norm)

Given a matrix $M \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and two normed vector spaces $\left(\mathbb{R}^{d_{\text{in}}}, \|\cdot\|_\alpha\right)$ and $\left(\mathbb{R}^{d_{\text{out}}}, \|\cdot\|_\beta\right)$, the "$\alpha$ to β" induced operator norm is given by:

$$\|\boldsymbol{M}\|_{\alpha \to \beta} = \max_{\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}} \frac{\|\boldsymbol{M}\boldsymbol{x}\|_\beta}{\|\boldsymbol{x}\|_\alpha}$$

For tensors, define the duality map via dualize $_{\|\cdot\|}\boldsymbol{G} := \arg\max_{\|\boldsymbol{T}\|=1} \boldsymbol{G} \diamond \boldsymbol{T}$

Definition (RMS norm)

For vector $\boldsymbol{v} \in R^d$, $\|v\|_{RMS} := \|\boldsymbol{v}\|_2 / \sqrt{d}$

Example (The RMS $\to$ RMS operator norm)

Given a matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we have $\|\boldsymbol{W}\|_{\text{RMS} \to \text{RMS}} = \sqrt{d_{\text{in}}/d_{\text{out}}} \times \|\boldsymbol{W}\|_*$, where $\|\cdot\|_*$ denotes the standard spectral norm.

Duality Map Examples

Linear layers:

Example (Duality map for the RMS $\to$ RMS operator norm)

For a matrix $\boldsymbol{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with reduced singular value decomposition $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, the corresponding duality map is given by $\text{dualize}_{\|\cdot\|_{\text{RMS}\to\text{RMS}}} \boldsymbol{G} = \sqrt{d_{\text{out}}/d_{\text{in}}} \times \boldsymbol{U}\boldsymbol{V}^\top$.

Embedding layers:

Example (Duality map for the $\ell_1 \to$ RMS operator norm)

Given a matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\|\boldsymbol{W}\|_{\ell_1\to\text{RMS}} = \max_i \|\text{col}_i(\boldsymbol{W})\|_{\text{RMS}}$.
For a matrix $\boldsymbol{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the corresponding duality map $\text{dualize}_{\|\cdot\|_{\ell_1\to\text{RMS}}} \boldsymbol{G}$ simply normalizes each column of $\boldsymbol{G}$ to have unit RMS norm: $\text{col}_i(\boldsymbol{G}) \mapsto \text{col}_i(\boldsymbol{G}) / \|\text{col}_i(\boldsymbol{G})\|_{\text{RMS}}$ for each $i = 1, \ldots, d_{\text{in}}$.

Content

Invariant Optimal Step Size

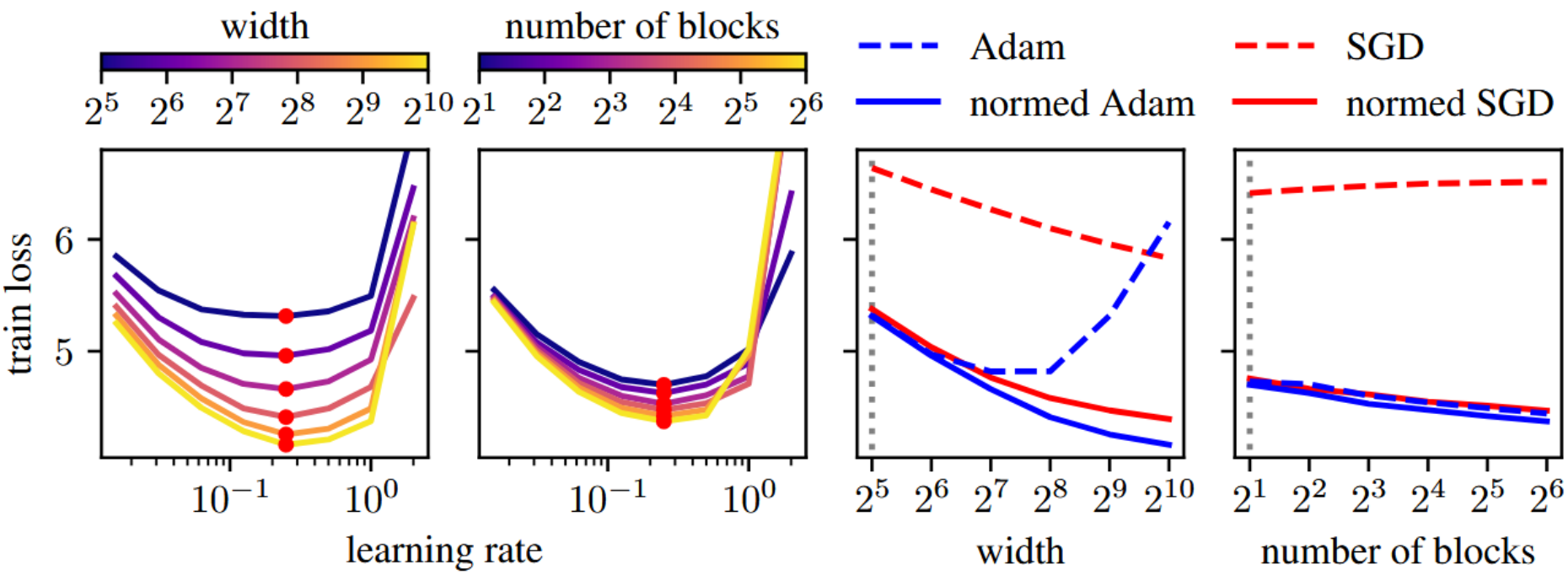


Figure 2: **Learning rate transfer in the modular norm**. We train GPT with context length 128 for 10k steps on OpenWebText.

Left: Learning rate sweeps for normed Adam across different width and number of blocks.

Right: For each method, the learning rate is tuned at the scale marked by the dotted line.

Full Weight Space Norm

The advantage of the non-standard $\|\cdot\|_{RMS\to RMS}$ matrix norm is that it allows one to estimate the amount of feature change induced by a gradient update:

$$\|\Delta \boldsymbol{W}\boldsymbol{x}\|_{\mathrm{RMS}} \leq \|\Delta \boldsymbol{W}\|_{\mathrm{RMS}-\mathrm{RMS}} \cdot \|\boldsymbol{x}\|_{\mathrm{RMS}}, \tag{4}$$

Now we know how to metrize individual layers, can we combine layer-wise norms to produce a norm on the full weight space $\mathcal{W} = \prod_k \mathcal{W}_k$ of the network?

The author propose to define the norm to be useful by the criteria 6. It ends up as a max (L^∞ combination) of scaled layer-wise norms $\|\cdot\|_{\mathcal{W}_k}$:

$$\|(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L)\|_{\mathcal{W}} := \max\left(s_1 \|\boldsymbol{w}_1\|_{\mathcal{W}_1}, \ldots, s_L \|\boldsymbol{w}_L\|_{\mathcal{W}_L}\right). \tag{5}$$

The positive scalar constants $s_1, \ldots, s_L$ are determined by both the architecture of the network and a set of user-specified "mass" parameters.

Normed optimization

We define the following operation on weight updates $\Delta \boldsymbol{w} = (\Delta \boldsymbol{w}_1, \ldots, \Delta \boldsymbol{w}_L) \in \mathcal{W}$:

$$\text{normalize}(\Delta \boldsymbol{w}) := \left(\frac{\Delta \boldsymbol{w}_1}{s_1 \left\| \Delta \boldsymbol{w}_1 \right\|_{\mathcal{W}_1}}, \ldots, \frac{\Delta \boldsymbol{w}_L}{s_L \left\| \Delta \boldsymbol{w}_L \right\|_{\mathcal{W}_L}} \right) \tag{6}$$

Use normalize as a wrapper, along with an explicit learning rate schedule for any base optimizer:

```
delta_w = optim(w.grad())          # get update from base optimizer
net.normalize(delta_w)             # normalize update in the modular norm
w -= eta(step) * delta_w           # apply update with learning rate eta
```

Modules

Definition (Module)

Given input vector space $\mathcal{X}$, output vector space $\mathcal{Y}$ and weight vector space $\mathcal{W}$, a module M is an object with the following four attributes:

- *a function*, M.forward : $\mathcal{W} \times \mathcal{X} \to \mathcal{Y}$, which maps an input and a weight vector to an output. we abbreviate this attribute to just $\mathrm{M} \equiv \mathrm{M}.\text{forward}$;
- *a number*, M.mass ≥ 0, which will turn out to set the **proportion of feature learning** that this module contributes to any supermodule;
- *a number*, M.sensitivity ≥ 0, which estimates the module's sensitivity to input perturbations;
- *a norm* over the weight space, M.norm : $\mathcal{W} \to \mathbb{R}_{\geq 0}$, sometimes abbreviated to just $\|\cdot\|_{\mathrm{M}}$.

Modules

We care about three kinds of module in practice:

- *atomic modules*, whose attributes are hand-declared, and have weights.(e.g. linear modules, embedding modules, and convolution modules.)
- *bond modules*, whose attributes are hand-declared, but have no weights. Formally, their weight space is the zero vector space $\mathcal{W} = 0$. (e.g. ReLu, Add, Abs)
- *compound modules*, built out of other modules, with automatically inferred attributes.

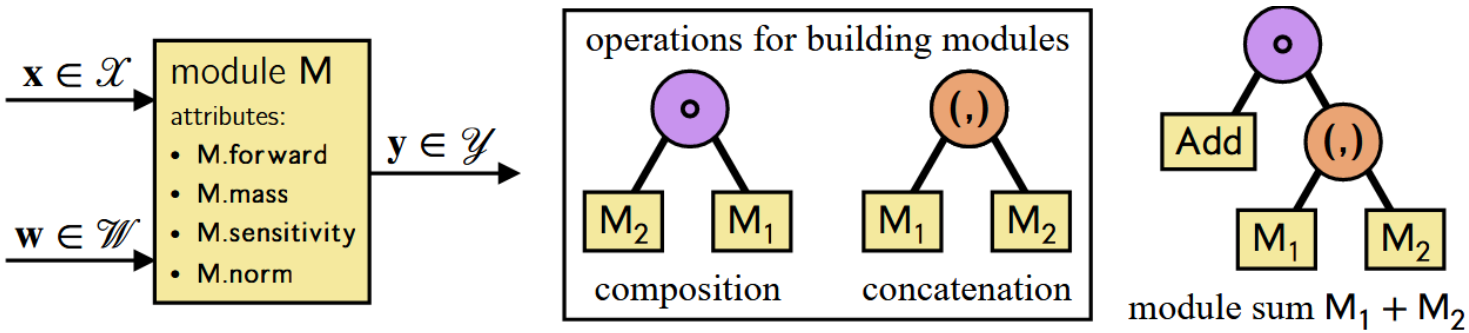


Figure 3: Modules and trees of modules

Well-normed Module

We what a module to be good when its attributes are predictive of its behaviour.

Definition (Well-normed)

Let M be a module on $(\mathcal{X}, \mathcal{Y}, \mathcal{W})$, where the input and output spaces have respective norms $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$. M is **well-normed** if for all inputs $\boldsymbol{x} \in \mathcal{X}$ and weights $\boldsymbol{w} \in \mathcal{W}$:

$$\|\nabla_{\boldsymbol{w}} \text{ M.forward } (\boldsymbol{w}, \boldsymbol{x}) \diamond \Delta \boldsymbol{w}\|_{\mathcal{Y}} \leq \text{ M.norm } (\Delta \boldsymbol{w}) \quad \text{for all } \Delta \boldsymbol{w} \in \mathcal{W}$$

$$\|\nabla_{\boldsymbol{x}} \mathrm{M}. \text{ forward } (\boldsymbol{w}, \boldsymbol{x}) \diamond \Delta \boldsymbol{x}\|_{\mathcal{Y}} \leq \mathrm{M}. \text{ sensitivity } * \|\Delta \boldsymbol{x}\|_{\mathcal{X}} \quad \text{for all } \Delta \boldsymbol{x} \in \mathcal{X}$$

Establish Lipschitz-continuity over input and weights space.

Building Upon Axioms

We could consider building new modules from old ones via the binary operations of **composition** and **concatenation**.

Definition (Module composition)

Consider module M_1 with input, output and weight space $(\mathcal{X}_1, \mathcal{Y}_1, \mathcal{W}_1)$ and module M_2 with input, output and weight space $(\mathcal{X}_2, \mathcal{Y}_2, \mathcal{W}_2)$ .M_1 and M_2 are composable if $\mathcal{X}_2 = \mathcal{Y}_1$. Their composite $\mathrm{M} = \mathrm{M}_2 \circ \mathrm{M}_1$ lives on $(\mathcal{X}_1, \mathcal{Y}_2, \mathcal{W}_1 \times \mathcal{W}_2)$ with attributes:

- M.forward $((\boldsymbol{w}_1, \boldsymbol{w}_2), \boldsymbol{x})) = \mathrm{M}_2$.forward $(\boldsymbol{w}_2, \mathrm{M}_1$.forward $(\boldsymbol{w}_1, \boldsymbol{x}))$;
- M.mass $= M_1.$ mass $+ M_2.$ mass;
- M.sensitivity $= \mathrm{M}_1$.sensitivity $* \mathrm{M}_2$.sensitivity;
- M.norm $((\boldsymbol{w}_1, \boldsymbol{w}_2))$ given by:

$$\max\left(M_2.\ \text{sensitivity}\ * \frac{M.\,\text{mass}}{M_1.\,\text{mass}} * M_1.\,\text{norm}\left(\boldsymbol{w}_1\right), \frac{M.\,\text{mass}}{M_2.\,\text{mass}} * M_2.\,\text{norm}\left(\boldsymbol{w}_2\right)\right)$$

Building Upon Axioms

Definition (Module concatenation)

Consider module M_1 with input, output and weight space $(\mathcal{X}_1, \mathcal{Y}_1, \mathcal{W}_1)$ and module M_2 with input, output and weight space $(\mathcal{X}_2, \mathcal{Y}_2, \mathcal{W}_2)$. We say that M_1 and M_2 are concatenatable if their input spaces match: $\mathcal{X}_1 = \mathcal{X}_2$. The tuple $\mathrm{M} = (\mathrm{M}_1, \mathrm{M}_2)$ has input, output and weight space $(\mathcal{X}_1, \mathcal{Y}_1 \times \mathcal{Y}_2, \mathcal{W}_1 \times \mathcal{W}_2)$ and attributes:

- M.forward $((\boldsymbol{w}_1, \boldsymbol{w}_2), \boldsymbol{x})) = (\mathrm{M}_1.\text{forward}\ (\boldsymbol{w}_1, \boldsymbol{x}), \mathrm{M}_2.\text{forward}\ (\boldsymbol{w}_2, \boldsymbol{x}))$;
- M.mass $= \mathrm{M}_1.\ \text{mass} + \mathrm{M}_2.\ \text{mass}$;
- M.sensitivity $= \mathrm{M}_1.\text{sensitivity} + \mathrm{M}_2.\text{sensitivity}$;
- M.norm $(\boldsymbol{w}_1, \boldsymbol{w}_2)$ given by:

$$\max\left(\frac{M \cdot \mathit{mass}}{M_1 \cdot \text{mass}} * M_1 \cdot \text{norm}\left(\boldsymbol{w}_1\right), \frac{M \cdot \text{mass}}{M_2 \cdot \text{mass}} * M_2 \cdot \text{norm}\left(\boldsymbol{w}_2\right)\right),$$

Content

Duality Maps Construction

We could first write down duality maps for atomic modules, then extend them to arbitrary compound modules by showing how these maps transfer. Typical procedure:

1. Select norms on the input and output spaces that respect the semantics[2] of M.forward.
2. Place a norm on the weight space to make M well-normed. Typically the induced operator norm.
3. Solve the duality maps for atomic modules, then extend to the whole network.

[2] Select norms that describe both how large we would like the inputs and outputs to be, and in what geometry we would like the outputs to evolve.

Duality Maps for Atomic Modules

Example (The Linear module)

The Linear module sends inputs from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to outputs in $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$. The weight space is given by the matrix space $\mathcal{W} = \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We endow the Linear module with attributes:

- Linear.forward $(\boldsymbol{W}, \boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$, the matrix-vector product;
- Linear.sensitivity $= 1$;
- Linear.mass $= \mu$, where $\mu \geqslant 0$ is a hyperparameter;
- Linear.norm $(\boldsymbol{W}) = \|\boldsymbol{W}\|_{\text{RMS} \to \text{RMS}}$, the RMS $\to$ RMS induced operator norm.

The duality map corresponding to Linear.norm is then given by: Linear.dualize $(\boldsymbol{G}) = \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \times \boldsymbol{U}\boldsymbol{V}^\top$, where the gradient $\boldsymbol{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ has reduced SVD $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$.

Well-normed if inputs and weights belong to the unit balls.

Duality Maps for Atomic Modules

Example (The Embed module)

The Embed module sends inputs from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to outputs in $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$. The weight space is given by the matrix space $\mathcal{W} = \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We endow the Embed module with attributes:

- Embed.forward $(\boldsymbol{W}, \boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$, the matrix-vector product;
- Embed.sensitivity $= 1$;
- Embed.mass $= \mu$, where $\mu \geqslant 0$ is a hyperparameter;
- Embed.norm $(\boldsymbol{W}) = \|\boldsymbol{W}\|_{\ell_1 \to \text{RMS}}$, the $\ell_1 \to$ RMS induced operator norm.

The duality map for Embed.norm is:
Embed.dualize $(\boldsymbol{G})$ performs the mapping $\text{col}_j(\boldsymbol{G}) \mapsto \dfrac{\text{col}_j(\boldsymbol{G})}{\|\text{col}_j(\boldsymbol{G})\|_{\text{RMS}}}$ for each column index $j = 1, \ldots, d_{\text{in}}$.

Well-normed if inputs and weights belong to the unit balls.

Duality Maps for Atomic Modules

Finally, we consider a Conv2D module with a $k \times k$ kernel. The calculations work by slicing up the weight tensor into a collection of k^2 matrices.

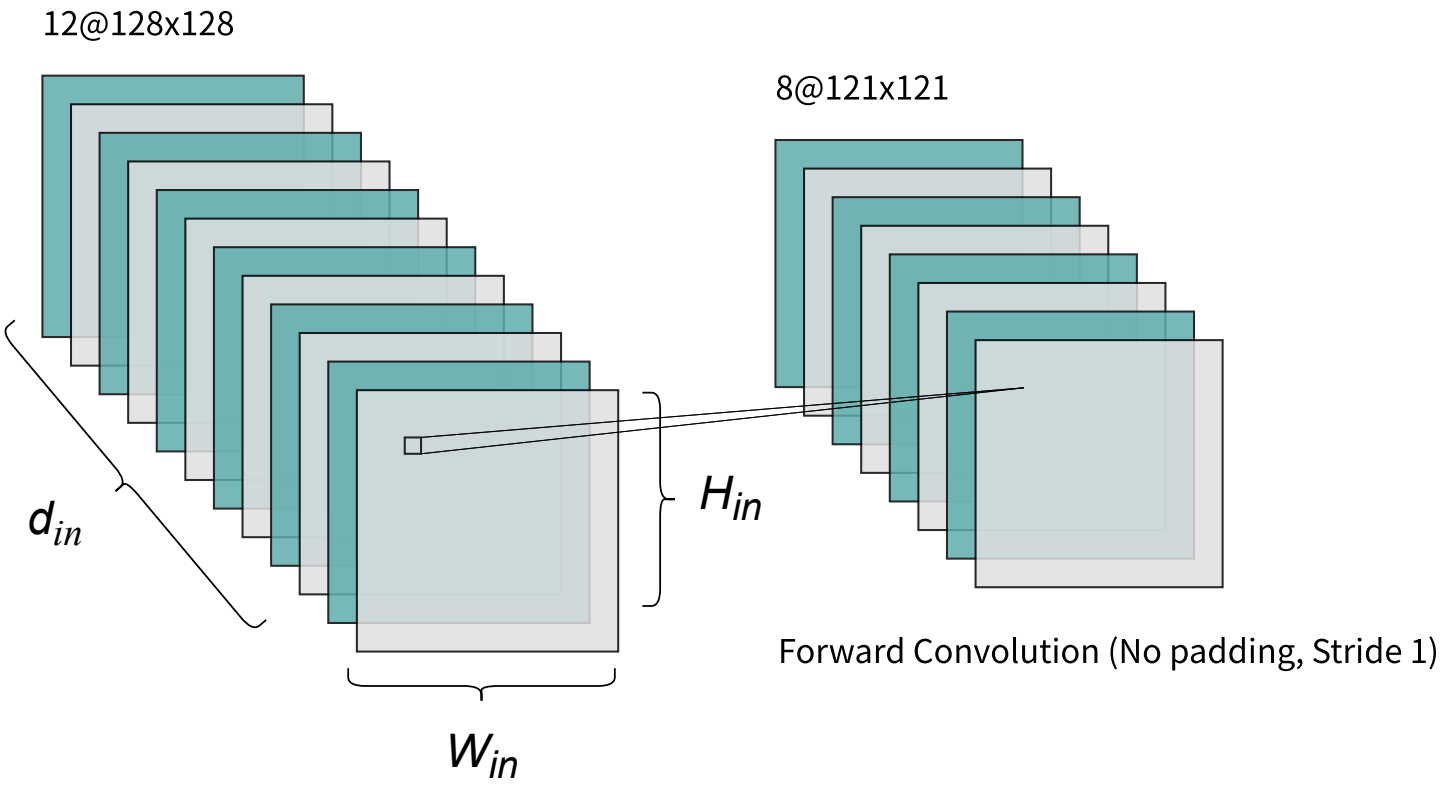


Figure 4: Illustration of forward passing in Conv2D

Duality Maps for Atomic Modules

Example (The Conv2D module)

The Conv2D module sends inputs from $\mathcal{X} = \mathbb{R}^{W_{\text{in}} \times H_{\text{in}} \times d_{\text{in}}}$ to outputs in $\mathcal{Y} = \mathbb{R}^{W_{\text{out}} \times H_{\text{out}} \times d_{\text{out}}}$. We think of this as mapping an input image of width W_{in}, height H_{in} and with d_{in} color channels to an output image of width W_{out}, height H_{out} and with d_{out} color channels. The weight space is given by the tensor space $\mathcal{W} = \mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times k \times k}$, where k is the kernel size. We endow Conv2D with attributes:

- Conv2D.forward $(\boldsymbol{W}, \boldsymbol{x}) = \boldsymbol{W} \circledast \boldsymbol{x}$, where $\circledast$ denotes $2D$ convolution;
- Conv2D.sensitivity $= 1$;
- Conv2D.mass $= \mu$, where $\mu \geqslant 0$ is a hyperparameter;
- Conv2D.norm $(\boldsymbol{W}) = k^2 \max_{i,j=1}^{k} \|\boldsymbol{W}_{..ij}\|_{\text{RMS} \to \text{RMS}}$, the max RMS $\to$ RMS norm over kernel indices.

Duality Maps for Atomic Modules

- We would like pixel intensities in the inputs and outputs to be order one and undergo order one change.
- We formalize this by taking the input and output norms to be the spatial maximum of the RMS norms of all the color channel vectors: $\|\boldsymbol{x}\|_{\mathcal{X}} = \max_{w=1}^{W_{\text{in}}} \max_{h=1}^{H_{\text{in}}} \|\boldsymbol{x}_{wh\cdot}\|_{\text{RMS}}$ and $\|\boldsymbol{y}\|_{\mathcal{Y}} = \max_{w=1}^{W_{\text{out}}} \max_{h=1}^{H_{\text{out}}} \|\boldsymbol{y}_{wh\cdot}\|_{\text{RMS}}$.
- Since the duality map for a max of norms decouples into one duality map per sub-norm, the duality map corresponding to Conv2D.norm is given by: $G \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times k \times k}$

  Conv2D.dualize $(\boldsymbol{G})$ does $\boldsymbol{G}_{..ij} \mapsto \frac{1}{k^2} \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \times \boldsymbol{U}_{ij} \boldsymbol{V}_{ij}^{\top}$, where $\boldsymbol{G}_{..ij}$ has reduced SVD $\boldsymbol{U}_{ij} \boldsymbol{\Sigma}_{ij} \boldsymbol{V}_{ij}^{\top}$.

Duality Maps for Bond Modules and Compound Modules

- Since the weight space of Bound Module B is $\{0\}$, then $B.norm = 0 \mapsto 0$, $B.dualize = 0 \mapsto 0$
- For compound modules,
  - Composition:

$$\text{M.dualize}\ (\boldsymbol{g}_1, \boldsymbol{g}_2) = \left(\frac{1}{\mathrm{M}_2.\text{sensitivity}} * \frac{\mathrm{M}_1.\text{mass}}{\text{M.mass}} * \mathrm{M}_1.\text{dualize}\,(\boldsymbol{g}_1)\,, \frac{\mathrm{M}_2.\text{mass}}{\text{M.mass}} * \mathrm{M}_2.\text{dualize}\ (\boldsymbol{g}_2) \right) \tag{7}$$

- - Concatenation:

$$\text{M.dualize}\,(\boldsymbol{g}_1, \boldsymbol{g}_2) = \left(\frac{\mathrm{M}_1.\text{mass}}{\text{M.mass}} * \mathrm{M}_1.\text{dualize}\,(\boldsymbol{g}_1)\,, \frac{\mathrm{M}_2.\text{mass}}{\text{M.mass}} * \mathrm{M}_2.\text{dualize}\,(\boldsymbol{g}_2) \right) \tag{8}$$

Rectangular Newton-Schulz Iteration

According to [Higham, 2008], the method works by first normalizing the matrix $\boldsymbol{G}$ according to $\boldsymbol{X}_0 = \boldsymbol{G}/\|\boldsymbol{G}\|_{\ell_2 \to \ell_2}$ and then iterating:

$$\boldsymbol{X}_{t+1} = \frac{3}{2} \cdot \boldsymbol{X}_t - \frac{1}{2} \cdot \boldsymbol{X}_t \boldsymbol{X}_t^\top \boldsymbol{X}_t,$$

then as $t \to \infty$, the sequence $\boldsymbol{X}_t \to \boldsymbol{U}\boldsymbol{V}^\top$.

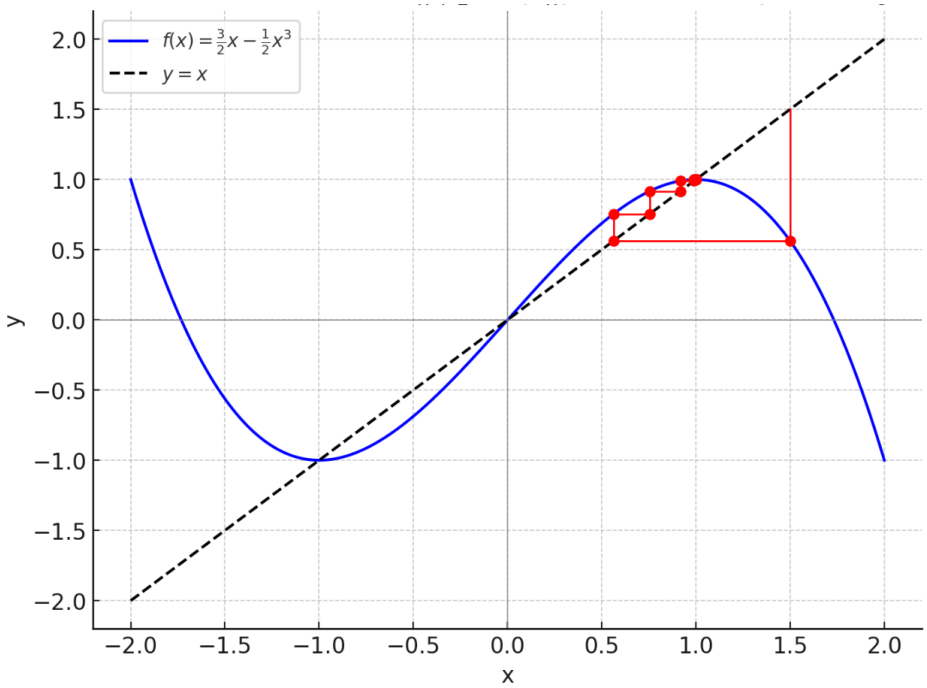


Figure 5: 1d illustration of iteration ($x_0 = 1.5$)

Deep Networks

Definition (Blocks and deep networks)

A deep neural network is a module M formed by a composition

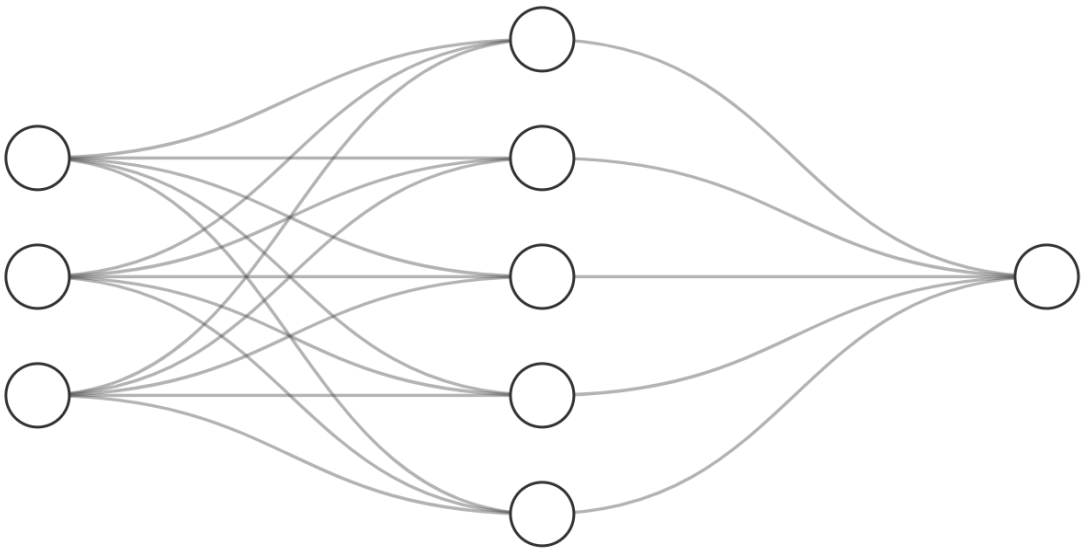
$$\mathrm{M} = \text{ OutputLayer } \circ \text{ Block }_{L} \circ \ldots \circ \text{ Block }_{1} \circ \text{ InputLayer}$$

where InputLayer, Block $_1, \ldots,$ Block $_L$, OutputLayer are modules. Typically, each of Block $_1, \ldots,$ Block $_L$ will be copies of the same module, so that the network can be written as an iterated composition

$$\mathrm{M} = \text{ OutputLayer } \circ \text{ Block }^{L} \circ \text{ InputLayer.}$$

Usually InputLayer is often some form of embedding module, and OutputLayer is usually a linear module.

Example: Two-layer network



$$M := \text{Linear}(1,5) \text{o} \Big(\text{ReLU o Linear } (5,3) \text{ o LayerNorm} \Big) \text{oEmbed}$$

We show *bond module* in cyan to mark no dualization.

Example: Multi-layer network

We denote $M(d) =$ ReLU o Linear (d, d) o LayerNorm for intermediate layers.
Then $M =$ OutputLayer o $M(d)^L$ o Embed

- The *bound module* still take effects in its sensitivity and specified mass. (e.g. ReLu layer has sensitivity $1/\sqrt{2}$)

Example: Multi-head Attention

Definition (Module broadcasting)

Suppose M is a module with inputs $\mathcal{X}$, outputs $\mathcal{Y}$ and weights $\mathcal{W}$. Then for any $h \geq 1$, the h-times-broadcast of M is the module $\mathrm{M}^{(h)}$ with the same weight space $\mathcal{W}$, mass, sensitivity and norm as M , but inputs the Cartesian power $\mathcal{X}^h = \mathcal{X} \times \ldots \times \mathcal{X}$ and outputs $\mathcal{Y}^h = \mathcal{Y} \times \ldots \times \mathcal{Y}$, and forward function

$$(\boldsymbol{w}, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_h)) \mapsto (\text{ M.forward } (\boldsymbol{w}, \boldsymbol{x}_1), \ldots, \text{ M.forward } (\boldsymbol{w}, \boldsymbol{x}_h)).$$

The attention module have both inputs and outputs.$\mathcal{X} = \mathbb{R}^{\ell \times d}$ where ℓ is the context length and d is the embedding dimension. The attention module itself will depend on three additional dimensional arguments:

- h, the number of heads;
- d_Q, the key/query dimension;
- d_V, the value dimension;

Example: Multi-head Attention

Also an $\ell \times \ell$ matrix mask, which we usually take to be either

$$\text{mask}_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases}$$

for causal attention, and mask $= 0$ for non-casual attention.

Definition (FuncAttention Module)

Inputs $\mathcal{X} = \mathbb{R}^{\ell \times d_Q} \times \mathbb{R}^{\ell \times d_Q} \times \mathbb{R}^{\ell \times d_V}$, outputs $\mathcal{Y} = \mathbb{R}^{\ell \times d_V}$, and forward function

$$\text{FuncAttention.forward}\ (\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = \text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{k}^\top}{d_Q} + \text{mask}\right)\boldsymbol{v}.$$

Exampke: Multi-head Attention

For positive integers ℓ, d, h, d_Q, d_V and a choice of mask, take four instances of the linear module:

- Query $=$ Linear $(h * d_Q, d)$
- Key $=$ Linear $(h * d_Q, d)$
- Value $=$ Linear $(h * d_V, d)$
- Exit $=$ Linear $(d, h * d_V)$

which by broadcasting we consider to have inputs of shape $\mathbb{R}^{\ell \times d}$. The multi-headed attention MultiHeadAttention module is then the composition:

$$\text{MultiHeadAttention} \;=\; \text{Exit} \circ \frac{1}{3} * \text{FuncAttention}^{(h)} \circ (\text{ Query, Key, Value })$$

Content

Summary and Takeaway

- The modular norm is proposed to let optimal learning rate transfer when scaling up width and depth using different "normed" base optimizers.
- We could build modules and duality maps for whole network through composition and concatenation of atomic and bond modules.
- We could compute the duality map (essentially the SVD) with Newton-Schulz Iteration.

Module	Weight Space $\mathcal{W}$	Module.norm	Module.dualize
Linear	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	$\boldsymbol{W} \mapsto \|\boldsymbol{W}\|_{\text{RMS}\to\text{RMS}}$	$\boldsymbol{G} \mapsto \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \times \boldsymbol{U}\boldsymbol{V}^\top$
Embed	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	$\boldsymbol{W} \mapsto \|\boldsymbol{W}\|_{\ell_1\to\text{RMS}}$	$\text{col}_j(\boldsymbol{G}) \mapsto \frac{\text{col}_j(\boldsymbol{G})}{\|\text{col}_j(\boldsymbol{G})\|_{\text{RMS}}}$
Conv2D	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times k \times k}$	$\boldsymbol{W} \mapsto k^2 \max_{i,j=1}^{k} \|\boldsymbol{W}_{\cdot\cdot ij}\|_{\text{RMS}\to\text{RMS}}$	$\boldsymbol{G}_{\cdot\cdot ij} \mapsto \frac{1}{k^2}\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \times \boldsymbol{U}_{ij}\boldsymbol{V}_{ij}^\top$

Figure 6: Summary of atomic duality maps

Content

Additional Modules

Example (Bond module ReLU)

In any dimension d, we define the "rectified linear unit" bond module ReLU to have input space $\mathcal{X} \subset \mathbb{R}^d$, output space $\mathcal{Y} = \mathbb{R}^d$, forward function
ReLU.forward : $(x_1, \ldots, x_d) \mapsto (\max(0, x_i))_{i=1,\ldots,d}$. and sensitivity $1/\sqrt{2}$.

Example (LayerNorm)

We define

$$\text{MeanSubtract.forward} : \ (x_1, \ldots, x_d) \mapsto (x_1 - \overline{\boldsymbol{x}}, \ldots, x_d - \overline{\boldsymbol{x}})$$

and

$$\text{RMSDivide.forward} : \ \boldsymbol{x} \mapsto \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_{\text{RMS}}} = \frac{\sqrt{d}\boldsymbol{x}}{\|\boldsymbol{x}\|_2}.$$

Such that: LayerNorm $=$ RMSDivide $\circ$ MeanSubtract.

Duality Map for Comosition and Concatenation

We prove for (8), the proof for (7) is similar by substituting the norm.
Denote $(\boldsymbol{q_1}, \boldsymbol{q_2}) = M.dualize(g_1, g_2)$.
Then $(q_1, q_2) = \underset{M.norm(q_1,q_2)=1}{argmax} \quad q_1 g_1 + q_2 g_2$
According to the norm constraints, we could assume WLOG that $\frac{M.mass}{M_1.mass} \times M_1.norm(q_1) = 1, \quad \frac{M.mass}{M_2.mass} \times M_2.norm(q_2) \leq 1$, then optimizing the $q_1 g_1$ term yields $q_1 \sim \alpha M_1.dualize(g_1)$, similarly for the second term $q_2 \sim \beta M_2.dualize(g_2)$
We then scale up the size of two terms to satisfy the norm constraints.

References I


Higham, N. J. (2008).
Functions of matrices: theory and computation.
SIAM.