Optimization for Machine Learning

Derivatives and differentiation

Clément Royer

CIMPA School “Control, Optimization and Model Reduction in Machine Learning”

February 27, 2025

Previously on...

Optimization problems in ML

$$\operatorname*{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w})$$

- $\boldsymbol{w}$: Model parameters
- f_i: Loss (w/o regularization) on ith data point.

Previously on...

Optimization problems in ML

$$\operatorname*{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w})$$

- $\boldsymbol{w}$: Model parameters
- f_i: Loss (w/o regularization) on ith data point.

Algorithm: (Batch) stochastic gradient

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{1}{|\mathcal{S}_k|}\sum_{i\in\mathcal{S}_k} \nabla f_i(\boldsymbol{w}_k)$$

- $\mathcal{S}_k = \{i_k\}$: Vanilla stochastic gradient.
- $\mathcal{S}_k = \{1,\dots,n\}$: Gradient descent ("Full batch").

A neural network problem

- Fully connected, three-layer network:

$$\boldsymbol{x} \in \mathbb{R}^{d_0} \mapsto \boldsymbol{W}_3 \ \mathrm{ReLU}\left(\boldsymbol{W}_2 \, \mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_3.$$

where $\mathrm{ReLU}(\boldsymbol{v}) = \max(\boldsymbol{v}, 0)$, $\boldsymbol{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$, $\boldsymbol{b}_j \in \mathbb{R}^{d_j}$.

A neural network problem

- Fully connected, three-layer network:

$$\boldsymbol{x} \in \mathbb{R}^{d_0} \mapsto \boldsymbol{W}_3 \ \mathrm{ReLU}\left(\boldsymbol{W}_2 \, \mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_3.$$

  where $\mathrm{ReLU}(\boldsymbol{v}) = \max(\boldsymbol{v}, 0)$, $\boldsymbol{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$, $\boldsymbol{b}_j \in \mathbb{R}^{d_j}$.
- Parameters: $(\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \boldsymbol{W}_3, \boldsymbol{b}_3) \rightarrow \boldsymbol{w} \in \mathbb{R}^d$,
  $d = d_1 d_0 + d_1 + d_2 d_1 + d_2 + d_3 d_2 + d_3$.

A neural network problem

- Fully connected, three-layer network:

$$\boldsymbol{x} \in \mathbb{R}^{d_0} \mapsto \boldsymbol{W}_3 \operatorname{ReLU}\left(\boldsymbol{W}_2 \operatorname{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_3.$$

  where $\operatorname{ReLU}(\boldsymbol{v}) = \max(\boldsymbol{v}, 0)$, $\boldsymbol{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$, $\boldsymbol{b}_j \in \mathbb{R}^{d_j}$.
- Parameters: $(\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \boldsymbol{W}_3, \boldsymbol{b}_3) \rightarrow \boldsymbol{w} \in \mathbb{R}^d$, $d = d_1 d_0 + d_1 + d_2 d_1 + d_2 + d_3 d_2 + d_3$.

Regression task (with squared loss)

- Model: $\boldsymbol{x} \mapsto \mathrm{NN}(\boldsymbol{x}; \boldsymbol{w})$.
- Data: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$, $\boldsymbol{y}_i \in \mathbb{R}^{d_3}$.

$$\operatorname*{minimize}_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \|\mathrm{NN}(\boldsymbol{x}_i; \boldsymbol{w}) - \boldsymbol{y}_i\|_2^2$$

A neural network problem ('ed)

$$\operatorname*{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\|\mathrm{NN}(\boldsymbol{x}_i;\boldsymbol{w}) - \boldsymbol{y}_i\|_2^2}_{f_i(\boldsymbol{w})}$$

A neural network problem ('ed)

$$\mathop{\text{minimize}}_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\|\mathrm{NN}(\mathbf{x}_i;\mathbf{w}) - \mathbf{y}_i\|_2^2}_{f_i(\mathbf{w})}$$

Applying stochastic gradient

- Requires derivatives of $f_i(\mathbf{w})$, hence derivatives of

$$\|\mathbf{W}_3 \,\mathrm{ReLU}\left(\mathbf{W}_2 \,\mathrm{ReLU}\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2\right) + \mathbf{b}_3\|_2^2$$

with respect to $\mathbf{W}_j, \mathbf{b}_j$.

A neural network problem ('ed)

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\|\mathrm{NN}(\boldsymbol{x}_i;\boldsymbol{w}) - \boldsymbol{y}_i\|_2^2}_{f_i(\boldsymbol{w})}$$

Applying stochastic gradient

- Requires derivatives of $f_i(\boldsymbol{w})$, hence derivatives of

$$\|\boldsymbol{W}_3\,\mathrm{ReLU}\,(\boldsymbol{W}_2\,\mathrm{ReLU}\,(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3\|_2^2$$

  with respect to $\boldsymbol{W}_j, \boldsymbol{b}_j$.
- **Issues:**
  - Hard to do/code by hand.
  - The gradient does not always exist!

Outline

1 Subgradients

2 Computing (sub)gradients

Outline

1. Subgradients
2. Computing (sub)gradients

Nonsmooth functions

Definition

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called **nonsmooth** if the gradient is not defined at every point.

Nonsmooth functions

Definition

A function $f : \mathbb{R}^d \to \mathbb{R}$ is called **nonsmooth** if the gradient is not defined at every point.

Examples of nonsmooth functions

- $w \mapsto |w|$ from $\mathbb{R}$ to $\mathbb{R}$;
- $\mathbf{w} \mapsto \|\mathbf{w}\|_1$ from $\mathbb{R}^d$ to $\mathbb{R}$;
- ReLU: $w \mapsto \max\{w, 0\}$ from $\mathbb{R}^d$ to $\mathbb{R}$.
- $w \mapsto 1(w \geq 0)$ from $\mathbb{R}$ to $\mathbb{R}$.

NB: Nonsmooth $\neq$ Discontinuous.

Subgradients for nonsmooth convex problems

Focus: Nonsmooth **convex** (hence continuous) functions.

Subgradients for nonsmooth convex problems

Focus: Nonsmooth **convex** (hence continuous) functions.

Definition

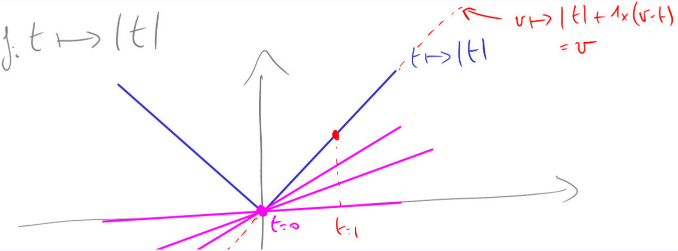
Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function. A vector $\boldsymbol{g} \in \mathbb{R}^d$ is called a *subgradient* of f at $\boldsymbol{w} \in \mathbb{R}^d$ if

$$\forall \boldsymbol{z} \in \mathbb{R}^n, \qquad f(\boldsymbol{z}) \;\geq\; f(\boldsymbol{w}) + \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{w}).$$

The set of all subgradients of f at $\boldsymbol{w}$ is called the *subdifferential* of f at $\boldsymbol{w}$, and denoted by $\partial f(\boldsymbol{w})$.

Subdifferentials and optimization

- If f differentiable at $\boldsymbol{w}$, $\partial f(\boldsymbol{w}) = \{\nabla f(\boldsymbol{w})\}$;
- $0 \in \partial f(\boldsymbol{w}) \Leftrightarrow \boldsymbol{w}$ minimum of f!



$$\partial(|\cdot|)(t) = \begin{cases} -1 & \text{if } t < 0 \\ 1 & \text{if } t > 0 \\ [-1,1] & \text{if } t = 0. \end{cases}$$

Subgradient method

Iteration for nonsmooth convex f

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{g}_k, \qquad \mathbf{g}_k \in \partial f(\mathbf{w}_k).$$

- Depends on the subgradient: a subgradient can be a direction of increase!
- Depends on α_k: typically chosen constant or decreasing.

Subgradient method

Iteration for nonsmooth convex f

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{g}_k, \qquad \mathbf{g}_k \in \partial f(\mathbf{w}_k).$$

- Depends on the subgradient: a subgradient can be a direction of increase!
- Depends on α_k: typically chosen constant or decreasing.

Guarantees

Let $\overline{\mathbf{w}}_K = \frac{1}{\sum_{k=0}^{K-1} \alpha_k} \sum_{k=0}^{K-1} \alpha_k \mathbf{w}_k$. Then,

$$f(\overline{\mathbf{w}}_K) - f^* \le \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

Worst rate than gradient descent ($\frac{1}{K}$) *but a lot more general!*

Subgradient algorithms

- Can define stochastic subgradient algorithms!
- Allows to use nonsmooth losses/regularizers.
- Guarantees even in the nonconvex setting (Davis, Drusvyatskiy '19).

What's next?

How can I compute a subgradient of

$$\| \boldsymbol{W}_3 \,\mathrm{ReLU}\,(\boldsymbol{W}_2 \,\mathrm{ReLU}\,(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3\|_2^2$$

w.r.t. $\boldsymbol{b}_j$ or $\boldsymbol{W}_j$?

Outline

1 Subgradients

2 Computing (sub)gradients

Derivatives in Deep Learning

What you do in PyTorch, JAX, etc

- Encode a neural network using blocks$\Rightarrow$Defines the parameters $\boldsymbol{w}$!
- Define a forward pass $\boldsymbol{x} \mapsto \mathrm{NN}(\boldsymbol{x}; \boldsymbol{w})$.

Derivatives in Deep Learning

What you do in PyTorch, JAX, etc

- Encode a neural network using blocks$\Rightarrow$Defines the parameters $\boldsymbol{w}$!
- Define a forward pass $\boldsymbol{x} \mapsto \mathrm{NN}(\boldsymbol{x}; \boldsymbol{w})$.

What happens next: Automatic differentiation

- A computational graph is created.
- Gradients w.r.t. any parameters can be computed through a backward pass in the graph.

Key mathematical tool: The chain rule!

The mathematical theorem

Let $f = g \circ h$, $h : \mathbb{R}^n \times \mathbb{R}^\ell$, $g : \mathbb{R}^\ell \times \mathbb{R}^m$ be smooth functions. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$,

$$\underbrace{\mathrm{J}_{\boldsymbol{x}} f(\boldsymbol{x})}_{m \times n} = \underbrace{\mathrm{J}_{\boldsymbol{y}} g(h(\boldsymbol{x}))}_{m \times \ell} \times \underbrace{\mathrm{J}_{\boldsymbol{x}} h(\boldsymbol{x})}_{\ell \times n}$$

where $\mathrm{J}_{\boldsymbol{z}} \phi(\boldsymbol{z})$ is the Jacobian of ϕ w.r.t. $\boldsymbol{z}$.

The chain rule

The mathematical theorem

Let $f = g \circ h$, $h : \mathbb{R}^n \times \mathbb{R}^\ell$, $g : \mathbb{R}^\ell \times \mathbb{R}^m$ be smooth functions. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$,

$$\underbrace{\mathrm{J}_{\boldsymbol{x}} f(\boldsymbol{x})}_{m \times n} = \underbrace{\mathrm{J}_{\boldsymbol{y}} g(h(\boldsymbol{x}))}_{m \times \ell} \times \underbrace{\mathrm{J}_{\boldsymbol{x}} h(\boldsymbol{x})}_{\ell \times n}$$

where $\mathrm{J}_{\boldsymbol{z}} \phi(\boldsymbol{z})$ is the Jacobian of ϕ w.r.t. $\boldsymbol{z}$.

The practice

- Functions from tensors to tensors: $\boldsymbol{z} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$, $f(\boldsymbol{z}) \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_q}$.
- Get $\mathrm{D}_{\boldsymbol{z}} \phi(\boldsymbol{z}) \in \mathbb{R}^{\mathrm{size}(\boldsymbol{z})}$ from $\mathrm{J}_{\boldsymbol{z}} \phi(\boldsymbol{z}) \in \mathbb{R}^{\mathrm{size}(f(\boldsymbol{z})) \times \mathrm{size}(\boldsymbol{z})}$.
- Nonsmooth calculus rules (Bolte & Pauwels '20).

Example: My 3-layer network (no bias for simplicity)

Let $\phi = \|\boldsymbol{W}_3 \, \mathrm{ReLU}\left(\boldsymbol{W}_2 \, \mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x}\right)\right)\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{x}} \phi$.**

Example: My 3-layer network (no bias for simplicity)

Let $\phi = \|\boldsymbol{W}_3 \, \mathrm{ReLU}\,(\boldsymbol{W}_2 \, \mathrm{ReLU}\,(\boldsymbol{W}_1 \boldsymbol{x}))\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{x}}\phi$.**

Decompose:

$$
\begin{aligned}
\phi &= \|\boldsymbol{z}_5\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \mathrm{ReLU}(\boldsymbol{z}_3) \\
\boldsymbol{z}_3 &= \boldsymbol{W}_2 \boldsymbol{z}_2 \\
\boldsymbol{z}_2 &= \mathrm{ReLU}(\boldsymbol{z}_1) \\
\boldsymbol{z}_1 &= \boldsymbol{W}_1 \boldsymbol{z}_0 \\
\boldsymbol{z}_0 &= \boldsymbol{x}.
\end{aligned}
$$

Example: My 3-layer network (no bias for simplicity)

Let $\phi = \| \boldsymbol{W}_3 \ \mathrm{ReLU}\left(\boldsymbol{W}_2 \, \mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x}\right)\right)\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{x}}\phi$.**

Decompose:

$$
\begin{aligned}
\phi &= \|\boldsymbol{z}_5\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \mathrm{ReLU}(\boldsymbol{z}_3) \\
\boldsymbol{z}_3 &= \boldsymbol{W}_2 \boldsymbol{z}_2 \\
\boldsymbol{z}_2 &= \mathrm{ReLU}(\boldsymbol{z}_1) \\
\boldsymbol{z}_1 &= \boldsymbol{W}_1 \boldsymbol{z}_0 \\
\boldsymbol{z}_0 &= \boldsymbol{x}.
\end{aligned}
$$

Compute Jacobians:

$$
\begin{aligned}
\mathrm{J}_{\boldsymbol{z}_5}\phi &= 2\boldsymbol{z}_5^{\mathrm{T}} \\
\mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5 &= \boldsymbol{W}_3 \\
\mathrm{J}_{\boldsymbol{z}_3}\boldsymbol{z}_4 &= \boldsymbol{\Lambda}(\boldsymbol{z}_3), \quad \boldsymbol{\Lambda}(\boldsymbol{u}) = \mathrm{diag}\left(\max\left(\frac{\boldsymbol{u}_i}{|\boldsymbol{u}_i|}, 0\right)\right) \\
\mathrm{J}_{\boldsymbol{z}_2}\boldsymbol{z}_3 &= \boldsymbol{W}_2 \\
\mathrm{J}_{\boldsymbol{z}_1}\boldsymbol{z}_2 &= \boldsymbol{\Lambda}(\boldsymbol{z}_1) \\
\mathrm{J}_{\boldsymbol{z}_0}\boldsymbol{z}_1 &= \boldsymbol{W}_1 \\
\mathrm{J}_{\boldsymbol{x}}\boldsymbol{z}_0 &= \boldsymbol{I}.
\end{aligned}
$$

Example: My 3-layer network (no bias for simplicity)

Let $\phi = \|\boldsymbol{W}_3 \, \mathrm{ReLU}\,(\boldsymbol{W}_2 \, \mathrm{ReLU}\,(\boldsymbol{W}_1 \boldsymbol{x}))\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{x}}\phi$.**

Decompose:

$$
\begin{aligned}
\phi &= \|\boldsymbol{z}_5\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \mathrm{ReLU}(\boldsymbol{z}_3) \\
\boldsymbol{z}_3 &= \boldsymbol{W}_2 \boldsymbol{z}_2 \\
\boldsymbol{z}_2 &= \mathrm{ReLU}(\boldsymbol{z}_1) \\
\boldsymbol{z}_1 &= \boldsymbol{W}_1 \boldsymbol{z}_0 \\
\boldsymbol{z}_0 &= \boldsymbol{x}.
\end{aligned}
$$

Compute Jacobians:

$$
\begin{aligned}
\mathrm{J}_{\boldsymbol{z}_5}\phi &= 2\boldsymbol{z}_5^{\mathrm{T}} \\
\mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5 &= \boldsymbol{W}_3 \\
\mathrm{J}_{\boldsymbol{z}_3}\boldsymbol{z}_4 &= \boldsymbol{\Lambda}(\boldsymbol{z}_3), \quad \boldsymbol{\Lambda}(\boldsymbol{u}) = \mathrm{diag}(\max(\tfrac{\boldsymbol{u}_i}{|\boldsymbol{u}_i|}, 0)) \\
\mathrm{J}_{\boldsymbol{z}_2}\boldsymbol{z}_3 &= \boldsymbol{W}_2 \\
\mathrm{J}_{\boldsymbol{z}_1}\boldsymbol{z}_2 &= \boldsymbol{\Lambda}(\boldsymbol{z}_1) \\
\mathrm{J}_{\boldsymbol{z}_0}\boldsymbol{z}_1 &= \boldsymbol{W}_1 \\
\mathrm{J}_{\boldsymbol{x}}\boldsymbol{z}_0 &= \boldsymbol{I}.
\end{aligned}
$$

Chain rule:

$$
\begin{aligned}
\mathrm{J}_{\boldsymbol{x}}\phi &= \mathrm{J}_{\boldsymbol{z}_5}\phi \, \mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5 \cdots \mathrm{J}_{\boldsymbol{z}_1}\boldsymbol{z}_2 \, \mathrm{J}_{\boldsymbol{z}_0}\boldsymbol{z}_1 \, \mathrm{J}_{\boldsymbol{x}}\boldsymbol{z}_0 \\
&= 2\boldsymbol{z}_5^{\mathrm{T}} \boldsymbol{W}_3 \boldsymbol{\Lambda}(\boldsymbol{z}_3) \boldsymbol{W}_2 \boldsymbol{\Lambda}(\boldsymbol{z}_1) \boldsymbol{W}_1 \in \mathbb{R}^{1 \times \mathrm{len}(\boldsymbol{x})}.
\end{aligned}
$$

My 3 layer network ('ed)

Let $\phi = \| \boldsymbol{W}_3 \, \mathrm{ReLU}\left(\boldsymbol{W}_2 \, \mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x}\right)\right) \|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{W}_2}\phi$.**

My 3 layer network ('ed)

Let $\phi = \|\boldsymbol{W}_3 \ \mathrm{ReLU}\left(\boldsymbol{W}_2 \,\mathrm{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{x}\right)\right)\|_2^2$. **Compute** $\mathrm{J}_{\boldsymbol{W}_2}\phi$.

Decompose:

$$
\begin{aligned}
\phi &= \|\boldsymbol{W}_3 \,\mathrm{ReLU}(\boldsymbol{v}_2)\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \mathrm{ReLU}(\boldsymbol{v}_2) \\
\boldsymbol{v}_2 &= \boldsymbol{W}_2 \boldsymbol{v}_1 \\
\boldsymbol{v}_1 &= \mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}).
\end{aligned}
$$

Let $\phi = \|\boldsymbol{W}_3 \text{ ReLU}(\boldsymbol{W}_2 \text{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}))\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{W}_2}\phi$.**

Decompose:

$$\begin{aligned}
\phi &= \|\boldsymbol{W}_3 \text{ReLU}(\boldsymbol{v}_2)\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \text{ReLU}(\boldsymbol{v}_2) \\
\boldsymbol{v}_2 &= \boldsymbol{W}_2 \boldsymbol{v}_1 \\
\boldsymbol{v}_1 &= \text{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}).
\end{aligned}$$

Compute Jacobians:

$$\begin{aligned}
\mathrm{J}_{\boldsymbol{z}_5}\phi &= 2\boldsymbol{z}_5^{\mathrm{T}} \\
\mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5 &= \boldsymbol{W}_3 \\
\mathrm{J}_{\boldsymbol{v}_2}\boldsymbol{z}_4 &= \boldsymbol{\Lambda}(\boldsymbol{v}_2) \\
\mathrm{J}_{\boldsymbol{W}_2}\boldsymbol{v}_2 &= \mathcal{T} \in \mathbb{R}^{\mathrm{len}(\boldsymbol{v}_1)\times \mathrm{size}(\boldsymbol{W}_2)} \\
[\mathcal{T}]_{ijk} &= [\boldsymbol{v}_1]_i \delta_{jk}.
\end{aligned}$$

Let $\phi = \|\boldsymbol{W}_3 \ \mathrm{ReLU}\,(\boldsymbol{W}_2\, \mathrm{ReLU}\,(\boldsymbol{W}_1 \boldsymbol{x}))\|_2^2$. **Compute $\mathrm{J}_{\boldsymbol{W}_2}\phi$.**

Decompose:

$$
\begin{aligned}
\phi &= \|\boldsymbol{W}_3\, \mathrm{ReLU}(\boldsymbol{v}_2)\|_2^2 \\
\boldsymbol{z}_5 &= \boldsymbol{W}_3 \boldsymbol{z}_4 \\
\boldsymbol{z}_4 &= \mathrm{ReLU}(\boldsymbol{v}_2) \\
\boldsymbol{v}_2 &= \boldsymbol{W}_2 \boldsymbol{v}_1 \\
\boldsymbol{v}_1 &= \mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}).
\end{aligned}
$$

Compute Jacobians:

$$
\begin{aligned}
\mathrm{J}_{\boldsymbol{z}_5}\phi &= 2\boldsymbol{z}_5^{\mathrm{T}} \\
\mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5 &= \boldsymbol{W}_3 \\
\mathrm{J}_{\boldsymbol{v}_2}\boldsymbol{z}_4 &= \boldsymbol{\Lambda}(\boldsymbol{v}_2) \\
\mathrm{J}_{\boldsymbol{W}_2}\boldsymbol{v}_2 &= \mathcal{T} \in \mathbb{R}^{\mathrm{len}(\boldsymbol{v}_1)\times \mathrm{size}(\boldsymbol{W}_2)} \\
[\mathcal{T}]_{ijk} &= [\boldsymbol{v}_1]_i \delta_{jk}.
\end{aligned}
$$

Chain rule:

$$
\begin{aligned}
\mathrm{J}_{\boldsymbol{W}_2}\phi &= \mathrm{J}_{\boldsymbol{z}_5}\phi\, \mathrm{J}_{\boldsymbol{z}_4}\boldsymbol{z}_5\, \mathrm{J}_{\boldsymbol{v}_2}\boldsymbol{z}_4\, \mathrm{J}_{\boldsymbol{W}_2}\boldsymbol{v}_2 \\
&= 2\boldsymbol{z}_5^{\mathrm{T}} \boldsymbol{W}_3 \boldsymbol{\Lambda}(\boldsymbol{v}_2)\mathcal{T} \in \mathbb{R}^{1\times \mathrm{size}(\boldsymbol{W}_2)}.
\end{aligned}
$$

Summary

Gradients/Subgradients

- Gradients needed for optimization!
- Can be replaced by subgradients.

Computing derivatives

- All you need is a code for the function!
- Get (sub)gradients through automatic differentiation!
- Efficient implementation in deep learning packages.

References

- J. C. Duchi, *Introductory lectures on stochastic optimization*. In *The Mathematics of Data*, AMS, 2018.
  ⇒ Lecture notes on stochastic subgradient methods.
- M. Hardt & B. Recht, *Patterns, predictions and actions*, Princeton University Press, 2022.
  ⇒ Chapter 7: Presentation of automatic differentiation.

For (even) more math:

- D. Davis & D. Drusvyatskiy, *Subgradient methods under weak convexity and tame geometry*, SIAG/OPT Views and News, 2020.
  ⇒ Theory of subgradient methods for a broad audience.
- J. Bolte & E. Pauwels, *Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning*, Mathematical Programming, 2021.
  ⇒ A rigorous subdifferential theory for neural networks.