On the Correctness of Automatic Differentiation for Neural Networks with Machine-Representable Parameters

Wonyeol Lee[1], Sejun Park[2], Alex Aiken[1]

[1]Stanford University, [2]Korea University

TL;DR. We theoretically study the correctness of AD in a practical setting, namely when parameters of neural networks take only machine-representable numbers.

Summary

- It has been shown that automatic differentiation (AD) is "almost always" correct over the reals in a mathematically precise sense.
- However, actual programs work with machine-representable numbers (e.g., floats), not real numbers.
- We study the correctness of AD when the parameter space of a neural network consists only of machine-representable numbers.

Preliminary: AD

- Let $P : \mathbb{R}^n \to \mathbb{R}^m$ be a program and $x \in \mathbb{R}^n$ be an input to P.
- AD refers to a family of algorithms based on the chain rule that aim to compute the derivative of P at x:

$$\mathcal{D}P(x) \in \mathbb{R}^{m\times n} \quad \text{(when exists)}.$$

- We focus on two popular modes of AD: forward mode and reverse mode, which include the well-known backpropagation algorithm.

Preliminary: Correctness of AD

- The correctness of AD has been actively studied.
- If P uses differentiable functions (e.g., fully-connected, convolution, and softmax layers), then for all $x \in \mathbb{R}^n$,

$$\mathcal{D}P(x) \text{ exists} \quad \text{and} \quad \mathcal{D}^{\mathsf{AD}}P(x) = \mathcal{D}P(x)$$

  where $\mathcal{D}^{\mathsf{AD}}P(x)$ is the output of AD when applied to P and x.
- If P starts to use non-differentiable functions (e.g., ReLU, max, and abs), then for some $x \in \mathbb{R}^n$,

$$\mathcal{D}P(x) \text{ might not exist} \quad \text{or} \quad \mathcal{D}^{\mathsf{AD}}P(x) \neq \mathcal{D}P(x).$$

  - E.g., for $P(x) = \mathrm{ReLU}(x)$, $\mathcal{D}P(0)$ does not exist.
  - E.g., for $P(x) = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x)$ and $\mathcal{D}^{\mathsf{AD}}\mathrm{ReLU}(0) = 0$, $\mathcal{D}^{\mathsf{AD}}P(0) = 0 \neq 1 = \mathcal{D}P(0)$.

Theorem 1 ([Bolte+20; Lee+20; Huot+23]). If P uses "piecewise-analytic" functions (which include ReLU, max, and abs), then

$$\mathcal{D}P(x) \text{ might not exist} \quad \text{or} \quad \mathcal{D}^{\mathsf{AD}}P(x) \neq \mathcal{D}P(x)$$

only for measure-zero (i.e., negligible) $x \in \mathbb{R}^n$.

Limitations of Prior Work

- In practice, inputs are not reals, but machine-representable numbers (e.g., floats). Since the set $\mathbb{M}$ of all machine-representable numbers is countable, it has measure zero in $\mathbb{R}$.
- Hence, AD can be incorrect for all $x \in \mathbb{M}$ and this is possible.
  - E.g., for $P = \frac{1}{|\mathbb{M}|}\sum_{c\in\mathbb{M}}\big(\mathrm{ReLU}(x-c) - \mathrm{ReLU}(-x+c)\big)$ and $\mathcal{D}^{\mathsf{AD}}\mathrm{ReLU}(0) = 0$, $\mathcal{D}^{\mathsf{AD}}P(0) = 0 \neq 1 = \mathcal{D}P(0)$ for all $x \in \mathbb{M}$.
- That is, prior work fails to say anything meaningful about the correctness of AD in a practical setting, i.e., when inputs are in $\mathbb{M}^n$.

Problem Setup

Goal. Study the correctness of AD when inputs to a program are machine-representable numbers.

Definition 1. We focus on programs $P : \mathbb{R}^n \to \mathbb{R}^m$ that express *neural networks*:

$$P(w) = \sigma_L\big(\tau_L(\cdots\sigma_1(\tau_1(c, w_1))\cdots, w_L)\big)$$

where $w = (w_1, \ldots, w_L)$ is parameters, τ_l is an analytic *pre-activation function* (e.g., fully-connected layer), and σ_l is a pointwise piecewise-analytic *activation function* (e.g., ReLU).

Definition 2. We study two sets of machine-representable parameters on which AD can be incorrect:

$$\mathsf{inc}(P) = \{w \in \mathbb{M}^n : \mathcal{D}P(w) \text{ exists but } \mathcal{D}^{\mathsf{AD}}P(w) \neq \mathcal{D}P(w)\},$$
$$\mathsf{ndf}(P) = \{w \in \mathbb{M}^n : \mathcal{D}P(w) \text{ does not exist}\}.$$

We call them the *incorrect set* and the *non-differentiable set*.

Bias Parameters

- We consider particular pre-activation functions defined as follows.

Definition 3. τ_l has *bias parameters* if there exists f_l such that

$$\tau_l(u, w_l) = f_l(u, w_l') + w_l'' \qquad \text{where } w_l = (w_l', w_l'').$$

- Many popular pre-activation functions are implemented typically with bias parameters. These include fully-connected layers, attention layers, and some normalization layers (e.g., LayerNorm).

AD on Networks with Bias Parameters

- Consider a program P that expresses a neural network where all pre-activation functions have bias parameters. We prove the following results for such programs.

Theorem 2. The incorrect set is always empty:

$$|\mathsf{inc}(P)| = 0.$$

Theorem 3. The density of the non-differentiable set over $\mathbb{M}^n$ is bounded by:

$$\frac{|\mathsf{ndf}(P)|}{|\mathbb{M}^n|} \leq \frac{(\#\text{ non-differentiabilities in } P)}{|\mathbb{M}|}.$$

This bound is tight up to a constant multiplicative factor.

Theorem 4. $w \in \mathsf{ndf}(P)$ if and only if the following hold at w for some activation neuron ν and its activation function f:

(i) The input to f touches a non-differentiable point of f.
(ii) The derivative of P with respect ν is not zero.

Theorem 5. On the non-differentiable set, AD computes a generalized derivative called a Clarke subderivative:

$$\mathcal{D}^{\mathsf{AD}}P(w) = \lim_{n\to\infty}\mathcal{D}P(v_n) \quad \text{for some } v_n \to w.$$

- Theorem 2 is somewhat surprising, given that there has been no such type of results before.
- Theorem 3 implies that the density of the non-differentiable set is often small for practical neural networks if we use high-precision (e.g., 32-bit) parameters.
- Theorem 4 is somewhat surprising, given that deciding the non-differentiability of a neural network is in general NP-hard.

AD on Networks without Bias Parameters

- We extend the above results to neural networks possibly without bias parameters. Some notable changes are as follows.
  - The incorrect might not be empty.
  - We prove a tight bound on the density of $|\mathsf{inc}(P) \cup \mathsf{ndf}(P)|$, which is in general larger than the above bounds.
  - AD might not compute a generalized derivative.