On the Smoothing of Deep Networks

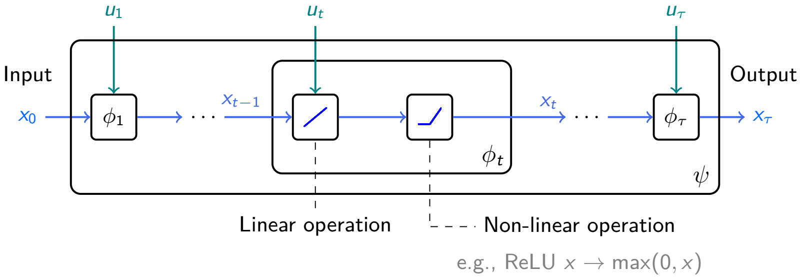
Vincent Roulet, Zaid Harchaoui

CISS 2021

Deep Networks

- A deep network
  - transforms an input x_0 into an output x_τ
  - through τ layers ϕ_t
  - parameterized by $u_1, \ldots u_\tau$
- Given n input-output pairs $(x^{(i)}, y^{(i)})_{i=1,\ldots,n}$, supervised learning is

$$\min_{u=(u_1,\ldots,u_\tau)} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^{(i)}, \psi(x^{(i)}, u)) + \lambda \|u\|_2^2$$

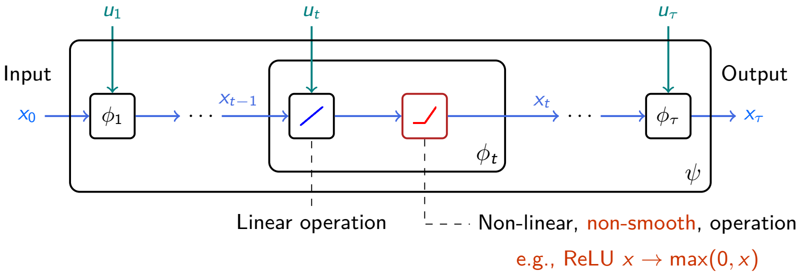


Non-smooth Deep Networks

Problem

Many deep networks use non-smooth layers,
i.e., functions that are *not everywhere differentiable* e.g. the ReLU

- Resulting problem is non-convex and non-smooth, theoretical guarantees are only asymptotic
- Classical automatic differentiation theory requires smooth functions



Analysis of Non-smooth Deep Networks

Previous work

- Analyze convergence of non-smooth, non-convex functions (Davis et al. 2020)
- Develop a theory for non-smooth automatic differentiation (Kakade & Lee 2018, Bolte & Pauwels 2020)

This talk: Approximate non-smooth layers by smooth counterparts

Questions

- How to build an ε-accurate smooth approximation of non-smooth networks for any fixed ε?
- How does this smoothing impact the performance of the deep networks?
- How does this smoothing impact the optimization path of e.g. SGD?

Smoothable Functions

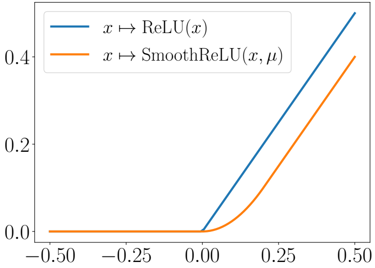
Definition

A function f is smoothable on a set C if, for any $\mu > 0$, we have access to an approximation f_μ of f on C such that

1. $\|f_\mu(x) - f(x)\|_2 \leq \mu \quad \forall x \in C$,
2. f_μ is differentiable with Lipschitz-continuous gradients, i.e., f_μ is smooth

Examples

- ReLU: $f(x) = \max(x, 0)$
  - SmoothReLU f_μ
  - $|f(x) - f_\mu(x)| \leq \mu$
  - f_μ is $1/(2\mu)$-smooth
- Piecewise affine functions



Smoothing Compositions

Smoothing Compositions

Let f, g be ℓ_f, ℓ_g Lipschitz-continuous resp. and smoothable.
Let $\mu > 0$, then $f_{\mu_f} \circ g_{\mu_g}$ for

$$\mu_f = \mu/2 \quad \text{and} \quad \mu_g = \mu/(2\ell_f)$$

is a smooth μ-accurate approximation of $f \circ g$, i.e.,

$$\forall x, \quad \|f \circ g(x) - f_{\mu_f} \circ f_{\mu_g}(x)\|_2 \leq \mu$$

Take-away:

- Compositions of Lip. continuous, smoothable functions are smoothable

Deep network case

- Layers of deep networks are not Lipschitz continuous w.r.t. both input and parameter
- We focus on the smoothing of deep networks on bounded sets

$$B_R = \{u = (u_1, \dots, u_\tau) : \|u_t\|_2 \leq R_t, \text{ for } t \in \{1, \dots, \tau\}\}$$

for $R = (R_1, \dots, R_\tau)$

Automatic Smoothing

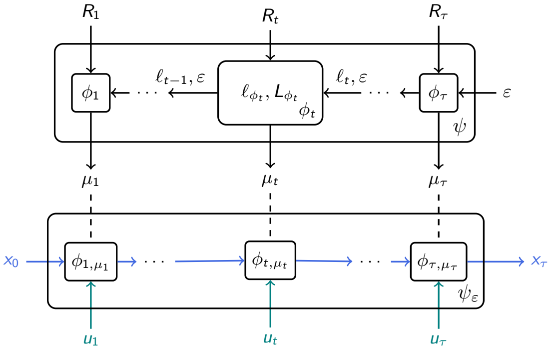
Inputs: deep network ψ composed of layers ϕ_t, accuracy ε,
bounds $R_1, \ldots, R_\tau$ on the parameters

Output: smooth deep network ψ_ε s.t.

$$\|\psi(x, u) - \psi_\varepsilon(x, u)\|_2 \leq \varepsilon \quad \forall x \text{ and } \forall u \in B_R$$

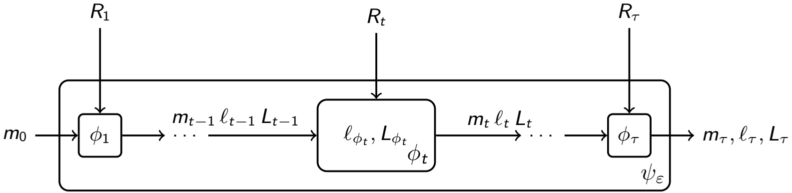
Overall scheme:

1. Forward pass to collect the smoothness properties of the layers
2. Backward pass to compute a ε-accurate smooth approx.



Smoothness estimation

- For f smoothable, smoothness of f_μ generally takes the form $L + K/\mu$
- Given the Lip. continuity and the smoothness constant of the layers, an estimate of the smoothness of ψ_ε can be computed in a forward pass
- We get
  - a bound m_τ on the output of ψ_ε
  - a Lip. continuity constant ℓ_τ of ψ_ε
  - a smoothness constant L_τ of ψ_ε



Optimization Consequences

Is it a local minimum indeed?

- Denote

$$F(u) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^{(i)}, \psi(x^{(i)}, u)) \quad \text{and} \quad F_{\varepsilon}(u) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^{(i)}, \psi_{\varepsilon}(x^{(i)}, u))$$

  such that

$$|F(u) - F_{\varepsilon}(u)| \leq \varepsilon \quad \forall u \in B_R$$

- If $\hat{u}$ is a ε-minimum of F_{ε} on its neighborhood, i.e.,

$$F_{\varepsilon}(\hat{u}) - \min_{u \in B_{\eta}(\hat{u})} F_{\varepsilon}(u) \leq \varepsilon$$

  where e.g. $B_{\eta}(\hat{u}) = \{u : \|\hat{u} - u\|_2 \leq \eta\} \subset B_R$

- Then $\hat{u}$ is 3ε-minimal for F on this neighborhood, i.e.

$$F(\hat{u}) - \min_{u \in B_{\eta}(\hat{u})} F(u) \leq 3\varepsilon$$

Performance Comparison

Setup

- MLP on MNIST, 3 hidden layers with ReLU
- ConvNet on CIFAR10, 3 hidden layers with ReLU
- Logistic loss, regularization $\lambda = 10^{-6}$

Comparison procedure

- Run SGD on original network ψ with stepsize found by grid-search
- Compute ε-accurate smooth approx. of ψ, denoted ψ_ε
- Run SGD with the same stepsize on ψ_ε

Smoothing used for each layer for an $\varepsilon = 1$ accurate approx.

Layer	1	2	3	4
MLP	$5 \cdot 10^{-6}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-2}$	0
ConvNet	$2 \cdot 10^{-6}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-2}$	0

Performance Comparison

	MLP	ConvNet
Train Loss	$1.64 \cdot 10^{-2} \pm 4.39 \cdot 10^{-6}$	$0.57 \pm 1.85 \cdot 10^{-2}$
Test loss	$1.54 \cdot 10^{-2} \pm 4.52 \cdot 10^{-5}$	$2.04 \cdot 10^{-2} \pm 6.21 \cdot 10^{-4}$
Train Acc.	100.0 ± 0	76.31 ± 0.79
Test Acc.	$98.33 \pm 4.20 \cdot 10^{-2}$	72.77 ± 0.76

Average performance of the non-smooth network and the smoothed counterparts for a range of accuracies $\varepsilon \in \{10^{-6}, \ldots, 10^{1}\}$

Take away:
No noticeable changes in performance between non-smooth and smoothed networks

Optimization Path

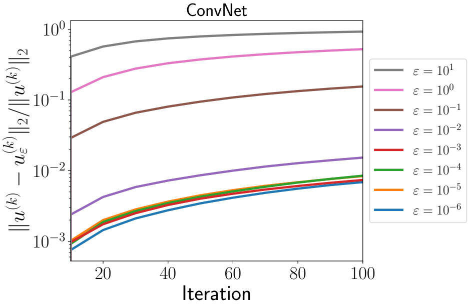
Plots

- Relative difference between
  - $u^{(k)}$ the iterate of SGD on ψ
  - $u_\varepsilon^{(k)}$ the iterate of SGD on ψ_ε
- Same seeds are used for ψ and ψ_ε
- Results are averaged over 10 seeds

Interpretation

- As ε decreases the relative difference does not tend to 0
- The non-smoothness has an impact on the optimization path

Thanks!



Bolte, J. & Pauwels, E. (2020), 'A mathematical model for automatic differentiation in machine learning', *NeurIPS* .

Davis, D., Drusvyatskiy, D., Kakade, S. & Lee, J. D. (2020), 'Stochastic subgradient method converges on tame functions', *Foundations of computational mathematics* **20**(1), 119–154.

Kakade, S. M. & Lee, J. D. (2018), 'Provably correct automatic sub-differentiation for qualified programs', *NeurIPS* .