Automatic Differentiation Friendly Complexity Guarantees

Vincent Roulet, Zaid Harchaoui
University of Washington

Workshop Beyond First order methods in ML ICML2020

17 Jul. 2019

Structure of Deep Neural Networks

Training of a deep neural network of k layers reads

$$\min_{v_1,\ldots,v_k} \quad \sum_{i=1}^{n} f_i(z_k^{(i)}) + \sum_{l=1}^{k} r_l(v_l)$$

$$\text{subject to} \quad z_l^{(i)} = \phi_l(v_l, z_{l-1}^{(i)}) \quad \text{for } l = 1, \ldots, k, \qquad z_0^{(i)} = x^{(i)}$$

- $v_1, \ldots, v_k$ are the weights of each layer l
- ϕ_l denotes the l^{th} layer with input z_{l-1} and output z_l
- $f^{(i)}(\hat{y}) = \mathcal{L}(\hat{y}, y^{(i)})$ are losses on the data $x^{(i)}$
- r_l are regularizations

Definition of a chain of layers

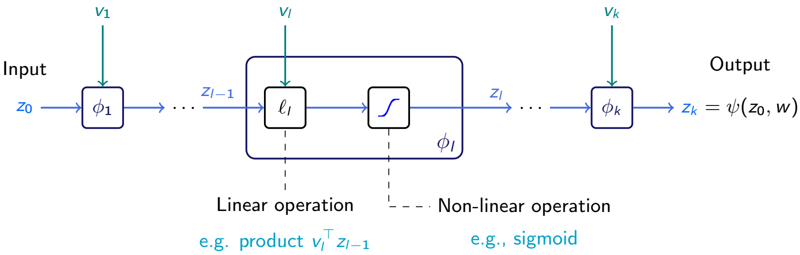
Definition

A function $\psi : \mathbb{R}^p \to \mathbb{R}^q$ is a *chain of k layers*, if it is defined for $w = (v_1; \dots; v_k) \in \mathbb{R}^p$ with $v_l \in \mathbb{R}^{\pi_l}$ by

$$\psi(w) = z_k,$$

with $$z_l = \phi_l(v_l, z_{l-1}) \quad \text{for } l = 1, \dots, k, \qquad z_0 = x,$$

where $x \in \mathbb{R}^{\delta_0}$ and $\phi_l : \mathbb{R}^{\pi_l} \times \mathbb{R}^{\delta_{l-1}} \to \mathbb{R}^{\delta_l}$.



Generic formulation

The objective reads then

$$\min_{w} \quad f(\psi(w)) + r(w)$$

where $f = \sum_i f^{(i)}$, $r = \sum_l r_l$, $\psi = (\psi_{x^{(1)}}; \ldots; \psi_{x^{(n)}})$.

Questions:

1. What smoothness properties can be stated for ψ?
2. How this applies to specific layers used in deep learning?
3. (How the structure of ψ is exploited to compute optim. oracles?)

Generic recursive smoothness bounds

Proposition

Given a chain ψ of k layers by layers ϕ_l, that are ℓ_{ϕ_l} Lipschitz-continuous and L_{ϕ_l} smooth,

(i) *An estimate of the Lipschitz-continuity of the chain ψ is given by $\ell_\psi = \ell_k$, where for $l \in \{1, \ldots, k\}$,*

$$\ell_l = \ell_{\phi_l} + \ell_{l-1}\ell_{\phi_l}, \qquad \ell_0 = 0.$$

(ii) *An estimate of the smoothness of the chain ψ is given by $L_\psi = L_k$, where for $l \in \{1, \ldots, k\}$,*

$$L_l = L_{l-1}\ell_{\phi_l} + L_{\phi_l}(1 + \ell_{l-1})^2, \qquad L_0 = 0.$$

Generic recursive smoothness bounds

Proposition

Given a chain ψ of k layers by layers ϕ_l, that are ℓ_{ϕ_l} Lipschitz-continuous and L_{ϕ_l} smooth,

(i) *An estimate of the Lipschitz-continuity of the chain ψ is given by $\ell_\psi = \ell_k$, where for $l \in \{1, \ldots, k\}$,*

$$\ell_l = \ell_{\phi_l} + \ell_{l-1}\ell_{\phi_l}, \qquad \ell_0 = 0.$$

(ii) *An estimate of the smoothness of the chain ψ is given by $L_\psi = L_k$, where for $l \in \{1, \ldots, k\}$,*

$$L_l = L_{l-1}\ell_{\phi_l} + L_{\phi_l}(1 + \ell_{l-1})^2, \qquad L_0 = 0.$$

Problem: Layers of deep neural networks are neither Lipschitz continuous nor smooth, needs to dwell into specific structure.

Smoothness details

Layers of deep neural network read

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, z_{l-1})\big)$$

where

- b_l is linear in v_l, affine in z_{l-1},
- a_l is non-linear, defined by an element-wise application of an *activation* function, potentially followed by a *pooling* operation

Smoothness details

Layers of deep neural network read

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, z_{l-1})\big)$$

where

- b_l is linear in v_l, affine in z_{l-1},
- a_l is non-linear, defined by an element-wise application of an *activation* function, potentially followed by a *pooling* operation

Examples:

- *Fully connected layer*

$$Z_l = V_l^\top Z_{l-1} + \nu_l \mathbf{1}_m^\top$$

  - $z_l = \mathrm{Vect}(Z_l)$, $v_l = \mathrm{Vect}((V_l^\top, \nu_l)^\top)$,
  - $b_l(v_l, z_{l-1}) = \mathrm{Vect}(V_l^\top Z_{l-1}) + \mathrm{Vect}(\nu_l \mathbf{1}_m^\top)$

Smoothness details

Layers of deep neural network read

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, z_{l-1})\big)$$

where

- b_l is linear in v_l, affine in z_{l-1},
- a_l is non-linear, defined by an element-wise application of an *activation* function, potentially followed by a *pooling* operation

Examples:

- *Fully connected layer*

$$Z_l = V_l^\top Z_{l-1} + \nu_l \mathbf{1}_m^\top$$

  - $z_l = \mathrm{Vect}(Z_l)$, $v_l = \mathrm{Vect}((V_l^\top, \nu_l)^\top)$,
  - $b_l(v_l, z_{l-1}) = \mathrm{Vect}(V_l^\top Z_{l-1}) + \mathrm{Vect}(\nu_l \mathbf{1}_m^\top)$
- Applies also to convolutional layers with vectorized images

Recursive smoothness bound for deep networks

Proposition

For a chain of layers ψ defined by layers of the form

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, z_{l-1})\big)$$

the boundedness, Lipschitz continuity and smoothness of ψ on a bounded set can be estimated by a forward pass on the network, given smoothness properties of each layer.

Recursive smoothness bound for deep networks

Proposition

For a chain of layers ψ defined by layers of the form

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, z_{l-1})\big)$$

the boundedness, Lipschitz continuity and smoothness of ψ on a bounded set can be estimated by a forward pass on the network, given smoothness properties of each layer.

Implementation

- We provide a list of smoothness constants for *supervised*, *unsupervised* objectives and various layers.
- This can be automatically plugged in an automatic differentiation package as PyTorch or tensor Flow.

VGG Network

Architecture

Benchmark architecture for image classification in 1000 classes, composed of 16 layers:

0 $x_i \in \mathbb{R}^{224\times224\times3}$,

1 $\phi_1(v, z) = a_{\text{ReLu}}(b_{\text{conv}}(v, z))$

2 $\phi_2(v, z) = p_{\max}(a_{\text{ReLu}}(b_{\text{conv}}(v, z)))$

$\vdots$

16 $\phi_{16}(v, z) = a_{\text{softmax}}(b_{\text{full}}(v, z) + \tilde{b}_{\text{full}}(v))$

17 $f(\hat{y}) = \sum_{i=1}^{n} \mathcal{L}_{\log}(\hat{y}_i, y_i)/n$

Batch-normalization effect

Introduce batch-normalization as modified layer

$$\phi_l(v_l, z_{l-1}) = a_l\big(b_l(v_l, c_l(z_{l-1}))\big)$$

where for $z = \mathrm{Vect}(Z)$ with $Z \in \mathbb{R}^{d\times n}$, $c(z) = \tilde{Z}$ defined as

$$(\tilde{Z})_{ij} = \frac{Z_{ij} - \mu_i}{\epsilon + \sigma_i},$$

$$\text{with} \qquad \mu_i = \frac{1}{m}\sum_{j=1}^{m} Z_{ij}, \quad \sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(Z_{ij} - \mu_i)^2.$$

Batch-normalization effect

Compare Lipschitz and smoothness bounds obtained with or without batch-norm on the smoothed VGG architecture.

$$
\text{for} \quad \epsilon = 10^{-2}, \qquad \begin{aligned} \ell_{\text{VGG}} &\leq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\leq L_{\text{VGG-batch}} \end{aligned}
$$

$$
\text{for} \quad \epsilon = 10^{2}, \qquad \begin{aligned} \ell_{\text{VGG}} &\geq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\geq L_{\text{VGG-batch}} \end{aligned}
$$

Batch-normalization effect

Compare Lipschitz and smoothness bounds obtained with or without batch-norm on the smoothed VGG architecture.

$$
\begin{aligned}
\text{for} \quad \epsilon = 10^{-2}, \qquad & \begin{aligned} \ell_{\text{VGG}} &\leq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\leq L_{\text{VGG-batch}} \end{aligned} \\
\text{for} \quad \epsilon = 10^{2}, \qquad & \begin{aligned} \ell_{\text{VGG}} &\geq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\geq L_{\text{VGG-batch}} \end{aligned}
\end{aligned}
$$

- Corrects "How does batch normalization help optimization?" of [Santurkar et al, 2018] that studies non-Lipschitz-continuous batch-norm ($\epsilon = 0$)

Batch-normalization effect

Compare Lipschitz and smoothness bounds obtained with or without batch-norm on the smoothed VGG architecture.

$$\text{for} \quad \epsilon = 10^{-2}, \qquad \begin{aligned} \ell_{\text{VGG}} &\leq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\leq L_{\text{VGG-batch}} \end{aligned}$$

$$\text{for} \quad \epsilon = 10^{2}, \qquad \begin{aligned} \ell_{\text{VGG}} &\geq \ell_{\text{VGG-batch}} \\ L_{\text{VGG}} &\geq L_{\text{VGG-batch}} \end{aligned}$$

- Corrects "How does batch normalization help optimization?" of [Santurkar et al, 2018] that studies non-Lipschitz-continuous batch-norm ($\epsilon = 0$)
- Our framework can be used to quickly compare architectures given their components in terms of smoothness

Conclusion

Smoothness properties

- Automatic framework to compute estimates of the smoothness properties
- Can be used to design architectures in a principled way