Optimization for Deep Learning

Lecture 10-2: Gradient Clipping

Kun Yuan

Peking University

Main contents in this lecture

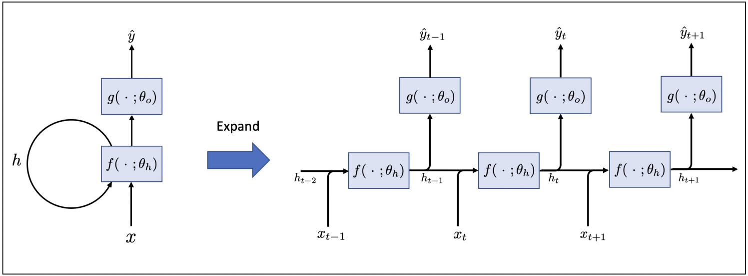
- Gradient exploding
- (L_0, L_1)-smoothness
- Gradient clipping

Recurrent neural network (RNN)

- RNN has the following recursion:

$$h_t = f(x_t, h_{t-1}; \theta_h)$$
$$\hat{y}_t = g(h_t; \theta_o)$$

where θ_h and θ_o are the parameters of $f(\cdot)$ and $g(\cdot)$, respectively, and h_0 can be initialized to arbitrary values.



Backpropagation in RNN

- Given a sequence of training data $\{x_t, y_t\}_{t=1}^{T}$, we consider the loss function

$$F(\theta_h, \theta_o) = \frac{1}{T}\sum_{t=1}^{T} L(\hat{y}_t, y_t)$$

  where $L(\hat{y}_t, y_t)$ measures the discrepancy between $\hat{y}_t$ and y_t.

- We next calculate $\nabla_{\theta_h} F(\theta_h, \theta_o)$. To this end, we have

$$\begin{aligned}\frac{\partial F(\theta_h, \theta_o)}{\partial \theta_h} &= \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L(\hat{y}_t, y_t)}{\partial \theta_h} \\ &= \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L(\hat{y}_t, y_t)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_t} \cdot \frac{\partial h_t}{\partial \theta_h}\end{aligned}$$

- The third term $\partial h_t / \partial \theta_h$ is tricky to handle.

Backpropagation in RNN

- Since $h_t = f(x_t, h_{t-1}; \theta_h)$, we have

$$\frac{\partial h_t}{\partial \theta_h} = \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial \theta_h} + \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial \theta_h} \tag{1}$$

  which is a recursion in terms of $\partial h_t / \partial \theta_h$.

- By letting

$$a_t = \frac{\partial h_t}{\partial \theta_h}$$
$$b_t = \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial \theta_h}$$
$$c_t = \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial h_{t-1}}$$

  Recursion (1) becomes

$$a_t = b_t + c_t a_{t-1}$$

Backpropagation in RNN

- By iterating the above recursion, we have

$$a_t = b_t + \sum_{i=1}^{t-1} \Big(\prod_{j=i+1}^{t} c_j \Big) b_i.$$

- Substituting a, b, and c, we have

$$\frac{\partial h_t}{\partial \theta_h} = \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial \theta_h} + \sum_{i=1}^{t-1} \Big(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}; \theta_h)}{\partial h_{j-1}} \Big) \frac{\partial f(x_i, h_{i-1}; \theta_h)}{\partial \theta_h},$$

where the chain $\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}; \theta_h)}{\partial h_{j-1}}$ can be very long for large t.

Backpropagation in RNN

- In summary, the back-propagation in RNN is derived as

$$\frac{\partial F(\theta_h, \theta_o)}{\partial \theta_h} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial L(\hat{y}_t, y_t)}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial \theta_h},$$

$$\frac{\partial h_t}{\partial \theta_h} = \frac{\partial f(x_t, h_{t-1}; \theta_h)}{\partial \theta_h} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}; \theta_h)}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}; \theta_h)}{\partial \theta_h}.$$

- The term $\partial F(\theta_h, \theta_o)/\partial \theta_o$ can be calculated in a similar manner

- We next consider a concrete example

Backpropagation in RNN

- Consider the following RNN formulation

$$h_t = W_x x_t + W_h h_{t-1}$$
$$\hat{y}_t = W_o h_t$$

where $W_x \in \mathbb{R}^{n \times d}, W_h \in \mathbb{R}^{n \times n}$, and $W_o \in \mathbb{R}^{m \times n}$ are parameters to learn, $x \in \mathbb{R}^d$ is the input data, $h \in \mathbb{R}^n$ is the hidden state, and $\hat{y} \in \mathbb{R}^m$ is the output label. We omit nonlinear activation for simplicity

- According to the above derivations for RNN backpropagation, we have

$$\frac{\partial F}{\partial W_x} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{t} (W_h^\intercal)^{t-i} W_o^\intercal \frac{\partial L(\hat{y}_t, y_t)}{\partial \hat{y}_t} x_i^\intercal \in \mathbb{R}^{n \times d}.$$

$\partial F / \partial W_h$ and $\partial F / \partial W_o$ can be derived similarly. We leave it as an exercise.

Vanishing gradient and exploding gradient

- Recall the gradient in linear RNN:

$$\frac{\partial F}{\partial W_x} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{t}(W_h^{\intercal})^{t-i}W_o^{\intercal}\frac{\partial L(\hat{y}_t, y_t)}{\partial \hat{y}_t}x_i^{\intercal} \in \mathbb{R}^{n\times d}.$$

- $(W_h^{\intercal})^t$ will cause a significant numerical issue in $\partial F/\partial W_x$
- If the largest magnitude of the eigenvalue is less than 1, i.e., $|\lambda(W_h^{\intercal})| < 1$, it holds that $(W_h^{\intercal})^{t-i} \to 0$ as t (or T) gets large; Gradient vanishing!
- If the largest magnitude of the eigenvalue is greater than 1, i.e., $|\lambda(W_h^{\intercal})| > 1$, it holds that $(W_h^{\intercal})^t \to +\infty$ as t (or T) gets large; Gradient exploding!
- Activation functions may also amplify gradient vanishing and exploding

References on gradient vanishing and exploding

RNN backpropagation:

- `https://zhuanlan.zhihu.com/p/273729929`
- M. Li et.al, *Dive into Deep Learning, Sec. 9.7*

Gradient vanishing and exploding

- R Pascanu et. al., *On the Difficulty of Training Recurrent Neural Networks*, ICML 2013.

How to overcome gradient vanishing and exploding?

- Residual deep neural network
- Batch normalization
- Proper initialization
- Gradient clipping (to be discussed in detail)

Gradient clipping

- Consider the following non-convex optimization problem

$$\min_{x \in \mathbb{R}^d} \quad f(x)$$

- The gradient clipping algorithm iterates as follows

$$x_t = x_t - \gamma g_t \quad \text{where} \quad g_t = \mathrm{clip}(\nabla f(x_t), c)$$

  for some positive constant $c > 0$.

- The clipping operator is defined as

$$\begin{aligned} \mathrm{clip}(u, c) &= \min\{1, \frac{c}{\|u\|}\}u \quad \forall u \in \mathbb{R}^d \\ &= \begin{cases} u & \text{if } \|u\| \leq c \\ \frac{c}{\|u\|}u & \text{if } \|u\| > c \end{cases} \end{aligned}$$

  where $\| \cdot \|$ is an ℓ_2-norm.

Gradient clipping helps preventing gradient exploding

- Clipping operator does not change the gradient direction (in deterministic scenario); just scales gradient
- Clipping operator squeezes large gradient when $\|\nabla f(x)\| > c$, but does nothing to small gradient
- After clipping, it is guaranteed that $\|u\| \leq c$ for any $u \in \mathbb{R}^d$
- It is intuitive that gradient clipping can prevent gradient exploding

L-smooth condition

- Recall the traditional Lipschitz smoothness condition

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d. \tag{2}$$

- GD works well under the above Lipschitz smoothness condition

- But this condition cannot capture the gradient exploding phenomenon

- Recall the gradient in linear RNN:

$$\frac{\partial F}{\partial W_x} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{t}(W_h^\intercal)^{t-i}W_o^\intercal\frac{\partial L(\hat{y}_t, y_t)}{\partial \hat{y}_t}x_i^\intercal \in \mathbb{R}^{n\times d}.$$

where the term $(W_h^\intercal)^t$ breaks condition (2) when t is large

L-smooth condition

- A toy example $f(x) = x^3$ breaks condition (2). Similarly, $(W_h^\intercal)^t$ will also break condition (2) when t is large.

- GD (or SGD) cannot work without assumption (2). To fix it, it is common to assume the iterate x_t to be within a compact set $\mathcal{C}$ for any t. In this scenario, it is enough to assume

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathcal{C} \subset \mathbb{R}^d.$$

- The constant L is proportional to the diameter of $\mathcal{C}$

- For example, if $\mathcal{C} = \{x \mid \|x\| \leq 100\}$, we have $L = 1200$.

$$\begin{aligned}\|\nabla f(x) - \nabla f(y)\| &= 3\|x^2 - y^2\| \leq 3\|x + y\|\|x - y\| \\ &\leq 6(\|x\| + \|y\|)\|x - y\| \leq 1200\|x - y\|\end{aligned}$$

(L_0, L_1)-smooth condition

Assumption 1 (Zhang et al. (2020); Koloskova et al. (2023))

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ *is said to be* (L_0, L_1)*-smooth if it verifies for all* $x, y \in \mathbb{R}^d$ *with* $\|x - y\| \leq \frac{1}{L_1}$ *that*

$$\|\nabla f(x) - \nabla f(y)\| \leq \left(L_0 + L_1\|\nabla f(x)\|\right)\|x - y\|.$$

- Can be interpreted as

  $$\|\nabla^2 f(x)\| \leq L_0 + L_1\|\nabla f(x)\|$$

  when $f(x)$ is twice-differentiable (Zhang et al., 2019).

- Recover the L-smoothness when $L_1 = 0$

(L_0, L_1)-smooth condition

- Recall the toy example $f(x) = x^3$
- When constraining x into the compact set $\mathcal{C} = \{x \mid \|x\| \leq 100\}$, it is L-smooth with $L = 1200$
- Now we evaluate its Hessian

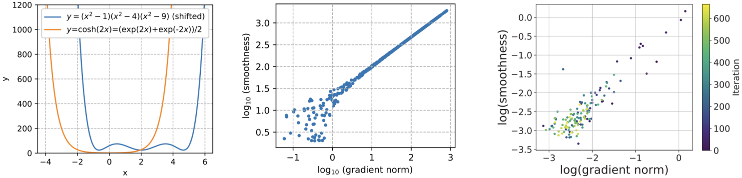
$$\|\nabla^2 f(x)\| = 6\|x\| \leq 6 + 2\|3x^2\| = 6 + 3\|\nabla f(x)\|$$

  which is (L_0, L_1)-smooth with $L_0 = 6$ and $L_1 = 2$

- It is important to note that $L_0 \ll L$ and $L_1 \ll L$
- In fact, any polynomial function satisfies the (L_0, L_1)-smooth condition

(L_0, L_1)-smooth condition

- The (L_0, L_1)-smooth condition well captures the RNN curvature
- An illustration on how smoothness varies with gradient norm[1]



[1] Figure is from (Zhang et al., 2019, 2020)

Convergence: non-convex scenario

Theorem 1 (Koloskova et al. (2023))

Under Assumption 1, if the learning rate $\gamma \leq [9(L_0 + cL_1)]^{-1}$*, the clipped gradient descent will converge at the following rate*

$$\frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\| \leq \mathcal{O}\Big(\sqrt{\frac{\Delta_0}{\gamma T}} + \frac{\Delta_0}{\gamma T c}\Big)$$

where $\Delta_0 = f(x_0) - f^\star$*.*

- Please note that the metric is $\|\cdot\|$ not $\|\cdot\|^2$
- Gradient clipping is typically not used in later phase due to small gradient
- The clip threshold c only affects the higher-order term (i.e., the initial phrase)
- Smaller c results in slower convergence

Convergence: non-convex scenario

- Recall the convergence rate of GD with standard L-smooth condition:

$$\Big(\frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\|\Big)^2 \le \frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\|^2 = \mathcal{O}\Big(\frac{\Delta_0}{\gamma T}\Big)$$

when $\gamma \le 1/L$. The above inequality implies that

$$\frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\| = \mathcal{O}\Big(\sqrt{\frac{\Delta_0}{\gamma T}}\Big)$$

- Comparing it with clipped GD

$$\text{GD:}\quad \frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\| = \mathcal{O}\Big(\sqrt{\frac{\Delta_0}{\gamma_{\text{gd}} T}}\Big)$$

$$\text{Clipped GD:}\quad \frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\| = \mathcal{O}\Big(\sqrt{\frac{\Delta_0}{\gamma_{\text{clip}} T}} + \frac{\Delta_0}{\gamma_{\text{clip}} T c}\Big)$$

Convergence: non-convex scenario

- Recall the learning rate in different algorithms:

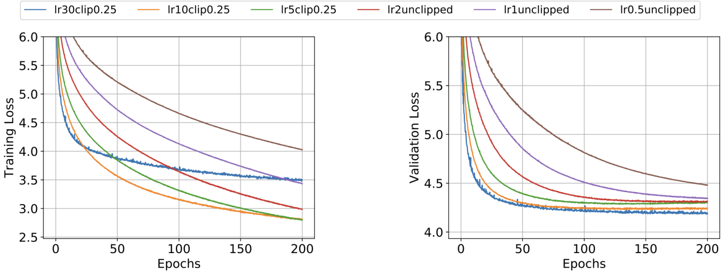
$$\gamma_{\text{gd}} = \mathcal{O}(\frac{1}{L}) \qquad \gamma_{\text{clip}} = \mathcal{O}(\frac{1}{L_0 + cL_1})$$

- Since $L_0 \ll L$ and $L_1 \ll L$, we can find Clipped GD is faster than GD

- **Clipping not only stabilizes, but also accelerates the training process!**

Convergence: non-convex scenario

Language modeling (Zhang et al., 2019)

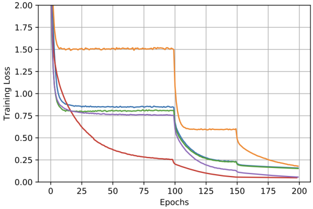


(a) Training loss of LSTM with different optimization parameters.

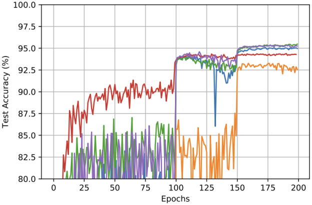
(b) Validation loss of LSTM with different optimization parameters.

Convergence: non-convex scenario

Image classification (Zhang et al., 2019)



(c) Training loss of ResNet20 with different optimization parameters.



(d) Test accuracy of ResNet20 with different optimization parameters.

References I

J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," *arXiv:1905.11881*, 2019.

B. Zhang, J. Jin, C. Fang, and L. Wang, "Improved analysis of clipping algorithms for non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 511–15 521, 2020.

A. Koloskova, H. Hendrikx, and S. U. Stich, "Revisiting gradient clipping: Stochastic bias and tight convergence guarantees," in *ICML 2023-40th International Conference on Machine Learning*, 2023.