NLP Reading Group

Gradient Descent Finds Global Minima for Generalizable Deep Neural Networks of Practical Sizes

by Kenji Kawaguchi and Jiaoyang Huang

Preliminaries

- Feed forward neural network with $H \geq 1$ layers, the parameter vector Θ and the input vector $x \in \mathbb{R}^{m_x}$
- We can define such a network as :

$$\begin{cases} x^{(0)} = x \\ x^{(l)} = \dfrac{1}{\sqrt{m_l}} \times \sigma(W^{(l)} x^{(l-1)} + b^{(l)}), \forall l \in [\![1\ ;\ H]\!] \end{cases} \text{where} \begin{cases} W^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}} \\ b^{(l)} \in \mathbb{R}^{m_l} \\ \sigma \text{ activation unit} \end{cases}$$

- The output of such a network is : $f(x, \Theta) = W^{(H+1)} x^{(H)} + b^{(H+1)} \in \mathbb{R}^{m_y}$ where :

$$\begin{cases} W^{(H+1)} \in \mathbb{R}^{m_y \times m_H} \\ b^{(H+1)} \in \mathbb{R}^{m_y} \end{cases}$$

- The parameter vector Θ which contains all the trainable parameters is such as :

$$\Theta = (vec(\bar{W}^{(1)})^T, \ldots, vec(\bar{W}^{(H)})^T)^T \text{ where } \begin{cases} \bar{W}^{(l)} = [W^{(l)}, b^{(l)}] \\ vec(M) \text{ is the standard vectorization of M.} \end{cases}$$

- By defining the number of neuron of the l-th layer as m_l, we can thus calculate the total number of parameters as :

$$\boxed{d = \sum_{l=0}^{H} (m_l m_{l+1} + m_{l+1})}$$

Reminders and Assumptions

- **k-lipschitz :** Let (E, d_E) and (F, d_F) be two metric spaces where d_X denotes the metric on the set X, a function, $f : E \to F$ is called Lipschitz continous if there exists a real constant $K \geq 0$ such that, for all x1 and x2 in E : $d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$
- **Assumption 1 : Use of common loss criteria** For any $i \in \{1, \dots, n\}$, the function $\ell_i(q) = \ell(q, y_i) \in \mathbb{R}_{\geq 0}$ is differentiable and convex, and $\nabla \ell_i$ is ζ-Lipschitz (with the metric induced by the Euclidian norm $\| \cdot \|_2$).
    - Satisfied with common loss criterion such as the squared loss or cross-entropy loss.
- **Assumption 2 : Use of common activation units** The activation function $\sigma(x)$ is real analytic, monotonically increasing, 1-Lipschitz, and the limits exist as: $\lim_{x \to -\infty} \sigma(x) = \sigma_- > -\infty$ and $\lim_{x \to +\infty} \sigma(x) = \sigma_+ \leq +\infty$.
    - This Assumption is satisfied by using common activation units such as sigmoid and hyperbolic tangents.

Problem Formalization : Presentation

- Analysis of the trainability with the of empirical risk minimization:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i,\theta), y_i), \text{ where } \begin{cases} \{(x_i, y_i)\}_{i=1}^{n} \text{is a training dataset,} \\ y_i \text{ is the i-th target,} \\ \ell(\cdot, y_i) \text{represents a loss criterion} \end{cases}$$

- Neural networks initialization : initial parameter vector θ^0 is randomly drawn as

$$\begin{cases} (W_{ij}^{(l)})^0 \sim \mathcal{N}(0, c_w) \\ (b^{(l)})^0 \sim \mathcal{N}(0, c_b) \end{cases}$$

$$\text{where } \begin{cases} c_w \text{ and } c_b \text{ constants} \\ \theta^0 = (vec((\bar{W}^{(1)})^0)^T, \ldots, vec((\bar{W}^{(H)})^0)^T)^T with (\bar{W}^{(l)})^0 = [(W^{(l)})^0, (b^{(l)})^0] \end{cases}$$

- *Intuitive definition of the Probable trainability $P_{n,H,\delta}$:*

Given dataset size n, depth H, and any $\delta > 0$, we can define :

$$\begin{cases} P_{n,H,\delta}(d) = true \text{ if trainability with d parameters } \forall \text{ datasets with probability at least } 1-\delta \\ P_{n,H,\delta}(d) = false \text{ otherwise.} \end{cases}$$

Problem Formalization : Notations

- σ satisfy Assumption 2.
- $\mathcal{F}_d^H$ the set of all neural network architectures $f(\cdot,\cdot)$ with H hidden layers and at most d parameters.
- $\mathcal{S}_n$ the set of all training datasets $S = \{(x_i, y_i)\}_{i=1}^n$ of size n with data points as : $\|x_i\|_2^2 = 1$
  $y_i \in [-1, 1]^{m_y}$
  for all $i \in \{1, \dots, n\}$.
- $\mathcal{L}_S^\zeta$ the set of all loss functionals L such that for any $L \in \mathcal{L}_S^\zeta$, we have
  $$\begin{cases} L(g) = \frac{1}{n}\sum_{i=1}^n \ell(g(x_i), y_i) \\ argmin_{g:\mathbb{R}^{m_x}\to\mathbb{R}^{m_y}} L(g) \neq \emptyset \end{cases}, \text{where} \begin{cases} g : \mathbb{R}^{m_x} \to \mathbb{R}^{m_y} \text{ function} \\ S \in \mathcal{S}_n \text{ a training dataset} \\ q \mapsto \ell(q, y_i) \text{ loss satisfying Assumption 1.} \end{cases}$$
- $\forall(\theta, \bar{W}), \psi_l(\theta, \bar{W}) \in \mathbb{R}^d$ is the parameter vector θ with the corresponding $\bar{W}^{(l)}$ entries replaced by $\bar{W}$.

  $$example : \psi_{H+1}(\theta, \bar{W}) = (vect(\bar{W}^{(1)})^T, \dots vect(\bar{W}^{(H)})^T, vect(\bar{W})^T)T.$$
- $\odot$ represents the entrywise product (i.e., Hadamard product).

Problem Formalization : Definition

$\mathcal{P}_{n,H,\delta} : \mathbb{N} \to \{true, false\}$ is a function such that $\mathcal{P}_{n,H,\delta}(d) = true$ if and only if :

- $\forall \zeta > 0, \exists f \in \mathcal{F}_d^H, \exists \eta \in \mathbb{R}^d, \forall S \in \mathcal{S}_n, \forall L \in \mathcal{L}_S^{\zeta}, \exists c_\theta \in \mathbb{R}$, and $\forall \epsilon > 0$, with probability at least $1 - \delta$ (over randomly drawn initial weights θ^0), $\exists\, t = O(c_r \zeta / \epsilon)$ such that

$$\boxed{J(\theta^t) = L(f(\cdot, \theta^t)) \leq L(f^*) + \epsilon}$$

and $\|\theta^t\|_2^2 \leq c_\theta$,

- where

$$\begin{cases} f^* \in argmin_{g: \mathcal{R}^{m_x} \to \mathcal{R}^{m_y}} L(g) \text{ is a global minimum of the functional } L \\ (\theta^k)_{k \in \mathcal{N}} \text{is defined by :} \theta^{k+1} = \theta^k - \eta \odot \nabla J(\theta^k) \\ c_r = \max_{l \in \{1, \ldots, H+1\}} \inf_{\bar{W}^* \in \mathcal{W}_l^*} \|\bar{W}^* - (\bar{W}^{(l)})^0\|_F^2 \ , \ \mathcal{W}_l^* = argmin_{\bar{W}} L(f(\cdot, \psi_l(\theta^0, \bar{W}))) \end{cases}$$

- Let $\mathcal{P'}_{n,H,\delta}$ be equivalent to $\mathcal{P}_{n,H,\delta}$, replacing the inequality on J by :

$$L(f(\cdot, \theta^t)) \leq L(f(\cdot, \theta^*)) + \epsilon,$$

where $\theta^* \in \mathbb{R}^d$ is a global minimum of $J(\theta) = L(f(\cdot, \theta))$.

As $L(f^*) \leq L(f(\cdot, \theta^*))$, $\mathcal{P}_{n,H,\delta}(d) = true$ implies that $\mathcal{P'}_{n,H,\delta}(d) = true$

Major Results

- ***Theorem 1 :***

  For any $n \in \mathbb{N}^+$, $H \geq 2$, and $\delta > 0$, it holds that $\mathcal{P}_{n,H,\delta}(d) = true$ for any

  $$d \geq \mathfrak{c}\left(\left(n + m_x H^2 + H^5 \log\left(\frac{Hn^2}{\delta}\right)\right) \log\left(\frac{Hn^2}{\delta}\right) + nm_y\right),$$

  where $\mathfrak{c} > 0$ is a universal constant.

- ***Theorem 2 :***

  There exists a universal constant $\mathfrak{c} > 0$ such that the following holds: for any large $\beta > 0$, $\frac{nm_y}{d} - 1 \geq \frac{\mathfrak{c}\beta H \log n}{\log(1/\epsilon)}$, and deep neural network architecture $f \in \mathcal{F}_d^H$, there exists a dataset $S \in \mathcal{S}$ such that if $\sum_{i=1}^n \|f(x_i, \theta) - y_i\|_2^2 \leq \epsilon$, then $\|\theta\|_2^2 \geq n^\beta$.

- ***Corollary :***

  For any $n \in \mathbb{N}^+$, $H \geq 1$, and $\delta > 0$, it holds that

  $\mathcal{P}_{n,H,\delta}(d) = false$ for any $d < nm_y$.

Remarks and Summary

- In Theorem 1 : restriction to the case of $H \geq 2$.
  - If $H = 1$: by setting $m_{H-1} = m_x$ and $x_i^{(H-1)} = x_i$, $\mathcal{P}_{n,1,\delta}(d) = true$ for any $d \geq \mathfrak{c}n(m_x + m_y)$.
  - The lower bound $\tilde{\Omega}(nm_x)$ for the case of $H = 1$ is worse than the lower bound $\tilde{\Omega}(nm_y)$.
- Trainable networks of size $\tilde{\Omega}(nm_y + m_x H^2 + H^5)$ if $H \geq 2$, and size $\tilde{\Omega}(n(m_x + m_y))$ if $H = 1$.
- Previously : The state of the art gave
  - $\tilde{\Omega}(2^{O(H)} n^8 + n^4(m_x + m_y))$ for deep neural Networks
  - $\tilde{\Omega}(n^2(m_x + m_y))$ for shallow ones

Previous Results

TABLE I: Number of parameters required to ensure the trainability, in terms of n, where n is the number of samples in a training dataset and H is the number of hidden layers.

Reference	# Parameters	Depth H	Trainability
[3], [4], [5]	$\tilde{\Omega}(n)$	1,2	No (expressivity only)
[8], [9], [7]	$\tilde{\Omega}(n)$	any H	No (expressivity only)
[13]	$\tilde{\Omega}(\text{poly}(n))$	1	Yes
[14]	$\tilde{\Omega}(n^6)$	1	Yes
[15]	$\tilde{\Omega}(n^2)$	1	Yes
[16], [18]	$\tilde{\Omega}(\text{poly}(n, H))$	any H	Yes
[17]	$\tilde{\Omega}(2^{O(H)} n^8)$	any H	Yes
[19]	$\tilde{\Omega}(H^{12} n^8)$	any H	Yes
this paper	$\tilde{\Omega}(n)$	any H	Yes

Study of the Generalization

- Let's consider multiclass classification with the one-hot vector $y \in \{0,1\}^{m_y}$:
  - Let $j(y) \in \{1, \ldots, m_y\}$ be the index of the one-hot vector y having entry one as $y_{j(y)} = 1$.
  - Let ℓ_{01} represent the 0--1 loss as $\ell(f(x,\theta), y) = \mathbb{1}\{argmax_j f(x,\theta)_j \neq j(y)\}$, with which we can write the expected test error $\mathbb{E}_{(x,y)}[\ell_{01}(f(x,\theta), y)]$.
  - Let ℓ_ρ be a standard multiclass margin loss defined by $\ell_\rho(f(x,\theta), y) = \min(\max(1 - (f(x,\theta)_{j(y)} - \max_{j' \neq j(y)} f(x,\theta)_{j'})/\rho, 0), 1)$.
  - In the proof of Theorem 1, constructing f as following ensures the trainability :

$$\begin{cases} m_1, m_2, \ldots, m_{H-2} = O(H^2 \log(Hn^2/\delta)) \\ m_{H-1} = O(\log(Hn^2/\delta)) \\ m_H = O(n) \end{cases}$$

Study of the Generalization

- ***Proposition for Generalization :***

Fix $\rho > 0$ and $\varsigma \geq 1$. Then, for any $\delta' > 0$, with probability at least $1 - \delta - \delta'$ over θ^0 and i.i.d. $((x_i, y_i))_{i=1}^n$, the following holds for any θ^t generated by the gradient descent (as $\theta^t = \theta^{t-1} - \eta \odot \nabla J(\theta^{t-1})$):

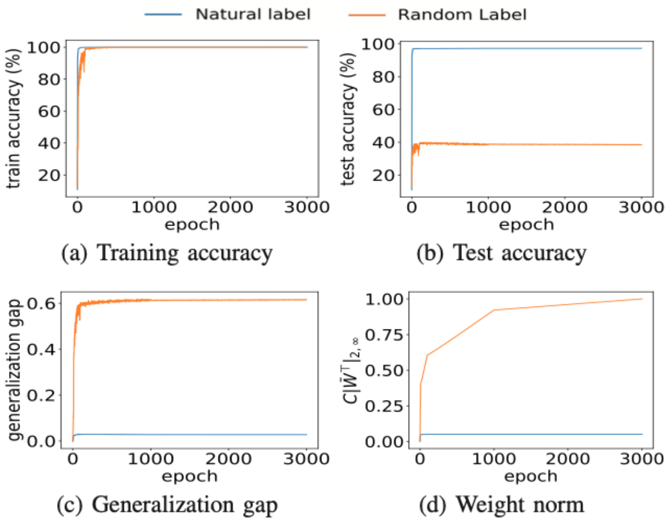
$$\mathbb{E}_{(x,y)}[\ell_{01}(f(x,\theta^t), y)] - \frac{1}{n}\sum_{i=1}^n \ell_\rho(f(x_i,\theta^t), y_i)$$

$$\leq \frac{cm_y^2 \lceil \varsigma \|(\bar{W}^t)^T\|_{2,\infty} \rceil}{\rho\varsigma\sqrt{n}} \cdot + \sqrt{\frac{\ln \frac{\pi^2 \lceil \varsigma \|(\bar{W}^t)^T\|_{2,\infty} \rceil^2}{\delta'}}{2n}}.$$

for some constant $c = O(1)$.

Study of the Generalization : Results :

Training accuracy, test accuracy, generalization gap, and weight norm for a neural network of practical size with the trainability guarantee, which is constructed in the proof of Theorem 1



(a) Training accuracy (b) Test accuracy

(c) Generalization gap (d) Weight norm

Thank you for your attention

Any questions?