Provable convergence of NN training in the overparametrized setting

Banach Center – Oberwolfach Graduate Seminar: Mathematics of Deep Learning

Julius Berner

November 21, 2019

University of Vienna

Gradient descent finds global minima of deep neural networks [PDF] arxiv.org
SS Du, JD Lee, H Li, L Wang, X Zhai - arXiv preprint arXiv:1811.03804, 2018 - arxiv.org
Gradient descent finds a global minimum in training deep neural networks despite the objective function being non-convex. The current paper proves gradient descent achieves zero training loss in polynomial time for a deep over-parameterized neural network with …
Cited by 137 Related articles All 6 versions

A convergence theory for deep learning via over-parameterization [PDF] arxiv.org
Z Allen-Zhu, Y Li, Z Song - arXiv preprint arXiv:1811.03962, 2018 - arxiv.org
Deep neural networks (DNNs) have demonstrated dominating performance in many fields, eg, computer vision, natural language progressing, and robotics. Since AlexNet, the neural networks used in practice are going wider and deeper. On the theoretical side, a long line of …
Cited by 141 Related articles All 3 versions

Stochastic gradient descent optimizes over-parameterized deep relu networks [PDF] arxiv.org
D Zou, Y Cao, D Zhou, Q Gu - arXiv preprint arXiv:1811.08888, 2018 - arxiv.org
We study the problem of training deep neural networks with Rectified Linear Unit (ReLU) activiation function using gradient descent and stochastic gradient descent. In particular, we study the binary classification problem and show that for a broad family of loss functions …
Cited by 87 Related articles All 2 versions

Gradient descent provably optimizes over-parameterized neural networks [PDF] arxiv.org
SS Du, X Zhai, B Poczos, A Singh - arXiv preprint arXiv:1810.02054, 2018 - arxiv.org
One of the mystery in the success of neural networks is randomly initialized first order methods like gradient descent can achieve zero training loss even though the objective function is non-convex and non-smooth. This paper demystifies this surprising phenomenon …
Cited by 153 Related articles All 3 versions

Learning overparameterized neural networks via stochastic gradient descent on structured data [PDF] nips.cc
Y Li, Y Liang - Advances in Neural Information Processing Systems, 2018 - papers.nips.cc
Neural networks have many successful applications, while much less theoretical understanding has been gained. Towards bridging this gap, we study the problem of learning a two-layer overparameterized ReLU neural network for multi-class classification …
Cited by 115 Related articles All 5 versions

Learning and generalization in overparameterized neural networks, going beyond two layers [PDF] nips.cc
Z Allen-Zhu, Y Li, Y Liang - Advances in Neural Information …, 2019 - papers.nips.cc

Introduction and Overview

Introduction

- overparametrized setting: #neurons >> #samples
- convergence of SGD with high probability (over the initialization)
- zero training loss: globally optimal solution
- related literature: [1, 2, 4, 5, 6, 8]

Overview

- training data $((x_i, y_i))_{i=1}^m \subseteq (\mathbb{R}^d \times \mathbb{R})^m$ and label vector $y := (y_i)_{i=1}^m$.
- prediction map $\mathcal{R}\colon \mathbb{R}^P \mapsto \mathbb{R}^m$ of, e.g. a neural network, mapping parameters $\Phi \in \mathbb{R}^P$ to the corresponding prediction, e.g.

$$\mathcal{R}\Phi = \big((W_L \circ \varrho \dots \varrho \circ W_1)(x_i)\big)_{i=1}^m \in \mathbb{R}^m.$$

- consider optimization via gradient flow using squared loss $\mathcal{L}(\hat{y}) := \frac{1}{2}\|\hat{y} - y\|^2$ (for simplicity)

gradient flow

$$\Phi'(t) := -\nabla_\Phi\big[\mathcal{L}(\mathcal{R}\Phi(t))\big] = -\nabla\mathcal{R}\Phi(t)^T(\mathcal{R}\Phi(t) - y)$$

with $\Phi(0) = \Phi_0$ initialized (randomly).

Overview

- training data $((x_i, y_i))_{i=1}^m \subseteq (\mathbb{R}^d \times \mathbb{R})^m$ and label vector $y := (y_i)_{i=1}^m$.
- prediction map $\mathcal{R}\colon \mathbb{R}^P \mapsto \mathbb{R}^m$ of, e.g. a neural network, mapping parameters $\Phi \in \mathbb{R}^P$ to the corresponding prediction, e.g.

$$\mathcal{R}\Phi = \big((W_L \circ \varrho \ldots \varrho \circ W_1)(x_i)\big)_{i=1}^m \in \mathbb{R}^m.$$

- consider optimization via gradient flow using squared loss $\mathcal{L}(\hat{y}) := \frac{1}{2}\|\hat{y} - y\|^2$ (for simplicity)

gradient flow

$$\Phi'(t) := -\nabla_\Phi\big[\mathcal{L}(\mathcal{R}\Phi(t))\big] = -\nabla\mathcal{R}\Phi(t)^T(\mathcal{R}\Phi(t) - y)$$

with $\Phi(0) = \Phi_0$ initialized (randomly).

Dynamics

- linearized prediction model $\bar{\mathcal{R}}\colon \mathbb{R}^P \to \mathbb{R}^m$ at initialization

$$\bar{\mathcal{R}}\Phi := \mathcal{R}(\Phi_0) + \nabla\mathcal{R}(\Phi_0)(\Phi - \Phi_0)$$

- evolve parameter $\bar{\Phi}(t)$ via gradient flow using the same initialization $\bar{\Phi}(0) = \Phi_0$

dynamics of the predictions

1. exact model : $\frac{d\mathcal{R}\Phi(t)}{dt} := -H(t)(\mathcal{R}\Phi(t) - y)$
2. linearized model : $\frac{d\bar{\mathcal{R}}\bar{\Phi}(t)}{dt} := -H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y)$

where $H(t) := \nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T \in \mathbb{R}^{m\times m}$

- Proof:

1. $\frac{d\mathcal{R}\Phi(t)}{dt} = \nabla\mathcal{R}\Phi(t)\Phi'(t) = -\underbrace{\nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T}_{:=H(t)}(\mathcal{R}\Phi(t) - y)$

2. $\nabla\bar{\mathcal{R}}\bar{\Phi}(t) = \nabla\mathcal{R}(\Phi_0)$

Dynamics

- linearized prediction model $\bar{\mathcal{R}}\colon \mathbb{R}^P \to \mathbb{R}^m$ at initialization

$$\bar{\mathcal{R}}\Phi := \mathcal{R}(\Phi_0) + \nabla\mathcal{R}(\Phi_0)(\Phi - \Phi_0)$$

- evolve parameter $\bar{\Phi}(t)$ via gradient flow using the same initialization $\bar{\Phi}(0) = \Phi_0$

dynamics of the predictions

1. exact model : $\frac{d\mathcal{R}\Phi(t)}{dt} := -H(t)(\mathcal{R}\Phi(t) - y)$
2. linearized model : $\frac{d\bar{\mathcal{R}}\bar{\Phi}(t)}{dt} := -H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y)$

where $H(t) := \nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T \in \mathbb{R}^{m\times m}$

- Proof:

1. $\frac{d\mathcal{R}\Phi(t)}{dt} = \nabla\mathcal{R}\Phi(t)\Phi'(t) = -\underbrace{\nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T}_{:=H(t)}(\mathcal{R}\Phi(t) - y)$
2. $\nabla\bar{\mathcal{R}}\bar{\Phi}(t) = \nabla\mathcal{R}(\Phi_0)$

Dynamics

- linearized prediction model $\bar{\mathcal{R}}\colon \mathbb{R}^P \to \mathbb{R}^m$ at initialization

$$\bar{\mathcal{R}}\Phi := \mathcal{R}(\Phi_0) + \nabla\mathcal{R}(\Phi_0)(\Phi - \Phi_0)$$

- evolve parameter $\bar{\Phi}(t)$ via gradient flow using the same initialization $\bar{\Phi}(0) = \Phi_0$

dynamics of the predictions

1. exact model : $\frac{d\mathcal{R}\Phi(t)}{dt} := -H(t)(\mathcal{R}\Phi(t) - y)$
2. linearized model : $\frac{d\bar{\mathcal{R}}\bar{\Phi}(t)}{dt} := -H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y)$

where $H(t) := \nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T \in \mathbb{R}^{m\times m}$

- Proof:

1. $\frac{d\mathcal{R}\Phi(t)}{dt} = \nabla\mathcal{R}\Phi(t)\Phi'(t) = -\underbrace{\nabla\mathcal{R}\Phi(t)\nabla\mathcal{R}\Phi(t)^T}_{:=H(t)}(\mathcal{R}\Phi(t) - y)$
2. $\nabla\bar{\mathcal{R}}\bar{\Phi}(t) = \nabla\mathcal{R}(\Phi_0)$

Lazy Training

- $\mathcal{L}(\bar{\mathcal{R}}\bar{\Phi}(t)) \approx \mathcal{L}(\mathcal{R}\Phi(t))$ for sufficiently small t
- "lazy training regime": this holds until algorithm stops
- linear model well understood: linear convergence to global minimizer with speed depending on smallest EV $\lambda = \lambda_{min}(H(0))$
- Proof idea:

$$\frac{d\|\bar{\mathcal{R}}\bar{\Phi}(t)-y\|^2}{dt} = -2(\bar{\mathcal{R}}\bar{\Phi}(t) - y)^T H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y) \leq -2\lambda\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2$$

$\rightarrow$ Grönwall: $\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2 \leq e^{-2\lambda t}\|\bar{\mathcal{R}}\Phi_0 - y\|^2$

- under suitable conditions this regime can be achieved by scaling the prediction function [2]

Lazy Training

- $\mathcal{L}(\bar{\mathcal{R}}\bar{\Phi}(t)) \approx \mathcal{L}(\mathcal{R}\Phi(t))$ for sufficiently small t
- ”lazy training regime”: this holds until algorithm stops
- linear model well understood: linear convergence to global minimizer with speed depending on smallest EV $\lambda = \lambda_{min}(H(0))$
- Proof idea:

  $\frac{d\|\bar{\mathcal{R}}\bar{\Phi}(t)-y\|^2}{dt} = -2(\bar{\mathcal{R}}\bar{\Phi}(t) - y)^T H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y) \leq -2\lambda\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2$

  $\rightarrow$ Grönwall: $\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2 \leq e^{-2\lambda t}\|\bar{\mathcal{R}}\Phi_0 - y\|^2$

- under suitable conditions this regime can be achieved by scaling the prediction function [2]

Lazy Training

- $\mathcal{L}(\bar{\mathcal{R}}\bar{\Phi}(t)) \approx \mathcal{L}(\mathcal{R}\Phi(t))$ for sufficiently small t
- "lazy training regime": this holds until algorithm stops
- linear model well understood: linear convergence to global minimizer with speed depending on smallest EV $\lambda = \lambda_{min}(H(0))$
- Proof idea:

$$\frac{d\|\bar{\mathcal{R}}\bar{\Phi}(t)-y\|^2}{dt} = -2(\bar{\mathcal{R}}\bar{\Phi}(t) - y)^T H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y) \leq -2\lambda\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2$$

$\rightarrow$ Grönwall: $\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2 \leq e^{-2\lambda t}\|\bar{\mathcal{R}}\Phi_0 - y\|^2$

- under suitable conditions this regime can be achieved by scaling the prediction function [2]

Lazy Training

- $\mathcal{L}(\bar{\mathcal{R}}\bar{\Phi}(t)) \approx \mathcal{L}(\mathcal{R}\Phi(t))$ for sufficiently small t
- "lazy training regime": this holds until algorithm stops
- linear model well understood: linear convergence to global minimizer with speed depending on smallest EV $\lambda = \lambda_{min}(H(0))$
- Proof idea:

  $\frac{d\|\bar{\mathcal{R}}\bar{\Phi}(t)-y\|^2}{dt} = -2(\bar{\mathcal{R}}\bar{\Phi}(t) - y)^T H(0)(\bar{\mathcal{R}}\bar{\Phi}(t) - y) \leq -2\lambda\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2$

  $\rightarrow$ Grönwall: $\|\bar{\mathcal{R}}\bar{\Phi}(t) - y\|^2 \leq e^{-2\lambda t}\|\bar{\mathcal{R}}\Phi_0 - y\|^2$

- under suitable conditions this regime can be achieved by scaling the prediction function [2]

Example

Definitions

- for illustration: Du et al. - Gradient Descent Provably Optimizes Over-parameterized Neural Networks (ICLR '19) [5]

2-layer biasless ReLU predictions

Let ϱ be the ReLU activation function and fix weights $a \in \mathbb{R}^{1\times N}$ (specified later). For weights $W = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \in \mathbb{R}^{N\times d}$ we identify $W \in \mathbb{R}^{N\times d} \sim \mathbb{R}^P$ and define the predictions of the corresponding ReLU network

$$\mathcal{R}W := (a\varrho(Wx_i))_{i=1}^m \in \mathbb{R}^m.$$

- for the ease of presentation we do not train $a \in \mathbb{R}^{1\times N}$ (convex optimization problem w.r.t. the weights in the last layer)

Definitions

- for illustration: Du et al. - Gradient Descent Provably Optimizes Over-parameterized Neural Networks (ICLR '19) [5]

2-layer biasless ReLU predictions

Let ϱ be the ReLU activation function and fix weights $a \in \mathbb{R}^{1\times N}$ (specified later). For weights $W = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \in \mathbb{R}^{N\times d}$ we identify $W \in \mathbb{R}^{N\times d} \sim \mathbb{R}^P$ and define the predictions of the corresponding ReLU network

$$\mathcal{R}W := (a\varrho(Wx_i))_{i=1}^m \in \mathbb{R}^m.$$

- for the ease of presentation we do not train $a \in \mathbb{R}^{1\times N}$ (convex optimization problem w.r.t. the weights in the last layer)

Omitting Bias

- omission of bias weights is standard in NN optimization literature [3, 4, 7, 8]
- severely limits the functions that can be realized with a given architecture
- BUT: augmenting the input data $x_i := (\tilde{x}_i, 1)$ and defining

$$W^{biasless} := \begin{bmatrix} W & b \\ 0 & 1 \end{bmatrix}$$

$\rightarrow$ recover the full range of functions

Omitting Bias

- omission of bias weights is standard in NN optimization literature [3, 4, 7, 8]
- severely limits the functions that can be realized with a given architecture
- BUT: augmenting the input data $x_i := (\tilde{x}_i, 1)$ and defining

$$W^{biasless} := \begin{bmatrix} W & b \\ 0 & 1 \end{bmatrix}$$

$\rightarrow$ recover the full range of functions

Gram/Kernel Matrix H and its Expectation

- $\nabla_{w_n} \mathcal{R}W(t)_i = a_n \varrho'(w_n x_i) x_i^T$

$$\to H_{ij}(t) = \nabla \mathcal{R}W(t)_i \nabla \mathcal{R}W(t)_j^T = x_i^T x_j \sum_{n=1}^{N} a_n^2 \varrho'(w_n(t)x_i)\varrho'(w_n(t)x_j)$$

$$= \sum_{n=1}^{N} a_n^2 g_i^T(w_n(t)) g_j(w_n(t))$$

where $g_i(w) := x_i \varrho'(w x_i)$

Assumptions

(1) assume $\|x_i\| = 1$ (normalized input data)
(2) independent $w_n(0) = w_0 \sim \mathcal{N}(0, I)$, $n = 1, \dots, N$
(3) assume $a \sim \mathcal{U}\left(\{-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\}^N\right)$ independent of W_0

$\to$ $H_{ij}(0)$ is Monte-Carlo approximation of

$$H_{ij}^{\infty} := \mathbb{E}[H_{ij}(0)] = \mathbb{E}_{w \sim \mathcal{N}(0,I)}\left[g_i^T(w) g_j(w)\right]$$

Gram/Kernel Matrix H and its Expectation

- $\nabla_{w_n} \mathcal{R} W(t)_i = a_n \varrho'(w_n x_i) x_i^T$

$$\to H_{ij}(t) = \nabla \mathcal{R} W(t)_i \nabla \mathcal{R} W(t)_j^T = x_i^T x_j \sum_{n=1}^{N} a_n^2 \varrho'(w_n(t) x_i) \varrho'(w_n(t) x_j)$$

$$= \sum_{n=1}^{N} a_n^2 g_i^T(w_n(t)) g_j(w_n(t))$$

where $g_i(w) := x_i \varrho'(w x_i)$

Assumptions

(1) assume $\|x_i\| = 1$ (normalized input data)
(2) independent $w_n(0) = w_0 \sim \mathcal{N}(0, I)$, $n = 1, \dots, N$
(3) assume $a \sim \mathcal{U}\left(\{-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\}^N\right)$ independent of W_0

$\to$ $H_{ij}(0)$ is Monte-Carlo approximation of

$$H_{ij}^{\infty} := \mathbb{E}[H_{ij}(0)] = \mathbb{E}_{w \sim \mathcal{N}(0,I)}\left[g_i^T(w) g_j(w)\right]$$

Gram/Kernel Matrix H and its Expectation

- $\nabla_{w_n} \mathcal{R}W(t)_i = a_n \varrho'(w_n x_i) x_i^T$

$$\rightarrow H_{ij}(t) = \nabla \mathcal{R}W(t)_i \nabla \mathcal{R}W(t)_j^T = x_i^T x_j \sum_{n=1}^{N} a_n^2 \varrho'(w_n(t)x_i)\varrho'(w_n(t)x_j)$$

$$= \sum_{n=1}^{N} \underbrace{a_n^2}_{=1/N} g_i^T(w_n(t)) g_j(w_n(t))$$

where $g_i(w) := x_i \varrho'(w x_i)$

Assumptions

(1) assume $\|x_i\| = 1$ (normalized input data)
(2) independent $w_n(0) = w_0 \sim \mathcal{N}(0, I)$, $n = 1, \ldots, N$
(3) assume $a \sim \mathcal{U}\big(\{-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\}^N\big)$ independent of W_0

$\rightarrow$ $H_{ij}(0)$ is Monte-Carlo approximation of

$$H_{ij}^{\infty} := \mathbb{E}[H_{ij}(0)] = \mathbb{E}_{w \sim \mathcal{N}(0,I)}\left[g_i^T(w) g_j(w)\right]$$

High-Level Overview

Assumptions

(4) assume $x_i \not\parallel x_j$ for $i \neq j$

Steps: | **Proof Idea:**

(A) $\lambda = \lambda_{min}(H^\infty) > 0$ (smallest EV) — $(g_i)_i$ lin. indep.

(B) $\|H^\infty - H(0)\|_2 \leq \frac{\lambda}{4}$ — Concentration ineq.

(C) W close to $W(0) \implies \|H^W - H(0)\|_2 \leq \frac{\lambda}{4}$ — scaled ridge func.

(D) $\lambda_{min}(H(t)) \geq \frac{1}{2}\lambda \quad (0 \leq s \leq t)$ — Grönwall's ineq.

$$\implies \begin{cases} W(t) \text{ close to } W(0) \\ \|\mathcal{R}W(t) - y\|^2 \leq e^{-\lambda t}\|\mathcal{R}W_0 - y\|^2 \end{cases}$$

....for suff. large N w.h.p., where $H^W := \frac{1}{N}\sum_{n=1}^{N} g_i^T(w_n)g_j(w_n)$.

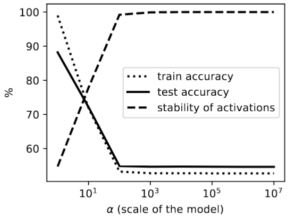
Result

Theorem: Zero NN Training Loss with Linear Convergence Rate

For every $\delta \in (0,1)$ and $N \gtrsim \frac{m^6}{\lambda^4 \delta^3}$ it holds that

$$\mathbb{P}\Big[\forall t \geq 0 : \|\mathcal{R}W(t) - y\|^2 \leq e^{-\lambda t}\|\mathcal{R}W_0 - y\|^2\Big] \geq 1 - \delta.$$

Performance in the Lazy Regime [2]



(a)

Model	Train acc.	Test acc.
ResNet wide, linearized	55.0	56.7
VGG-11 wide, linearized	61.0	61.7
Prior features [31]	-	82.3
Random features [36]	-	84.2
VGG-11 wide, standard	99.9	89.7
ResNet wide, standard	99.4	91.0

(b)

Figure 3: (a) Accuracies on CIFAR10 as a function of the scaling α. The stability of activations suggest a linearized regime when high. (b) Accuracies on CIFAR10 obtained for $\alpha = 1$ (standard, non-linear) and $\alpha = 10^7$ (linearized) compared to those reported for some linear methods without data augmentation: random features and prior features based on the scattering transform.

References

[1] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang. “Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks”. In: *arXiv preprint arXiv:1901.08584* (2019).

[2] L. Chizat and F. Bach. “A note on lazy training in supervised differentiable programming”. In: *arXiv preprint arXiv:1812.07956* (2018).

[3] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. “The loss surfaces of multilayer networks”. In: *Artificial Intelligence and Statistics*. 2015, pp. 192–204.

[4] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. “Gradient Descent Finds Global Minima of Deep Neural Networks”. In: *arXiv:1811.03804* (2018).

[5] S. S. Du, X. Zhai, B. Poczos, and A. Singh. “Gradient descent provably optimizes over-parameterized neural networks”. In: *arXiv preprint arXiv:1810.02054* (2018).

[6] A. Jacot, F. Gabriel, and C. Hongler. “Neural tangent kernel: Convergence and generalization in neural networks”. In: *Advances in neural information processing systems*. 2018, pp. 8571–8580.

[7] K. Kawaguchi. “Deep learning without poor local minima”. In: *Advances in neural information processing systems*. 2016, pp. 586–594.

[8] Y. Li and Y. Liang. “Learning overparameterized neural networks via stochastic gradient descent on structured data”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8157–8166.