Lecture 13: DNN Optimization Theory

Haizhao Yang

Department of Mathematics
University of Maryland College Park

2022 Summer Mini Course
Tianyuan Mathematical Center in Central China

Supervised deep learning

Conditions

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map $f(x)$ defined on Ω
- $\{x_i\}_{i=1}^{n}$ are sampled randomly from an unknown distribution $U(x)$ on Ω

Goal

Recover the unknown map $f(x)$

Deep learning in practice

- Only the empirical loss is available:

$$R_S(\theta) := \frac{1}{N}\sum_{i=1}^{N}(h(x_i;\theta) - y_i)^2$$

- The best empirical solution is $h(x;\theta_S)$ with

$$\theta_S = \text{argmin}\, R_S(\theta)$$

- Numerical optimization to obtain a numerical solution $h(x;\theta_N)$.
- In practice, $\theta_N \neq \theta_S$ and how good θ_N is?

Main Algorithms

- First order methods: stochastic gradient descent and its variants
- Second order methods: Newton method
- Quasi-second order methods: quasi-Newton methods

In Practice

First order methods usually have the best performance in terms of the computational speed and generalization error.

Main Analysis Tools for First Order Methods

- Neural tangent kernel (lazy training)
- Mean-field analysis

Neural tangent kernel/Lazy training

- Idea: in the limit of infinite width, deep learning becomes kernel methods
- Global optimization convergence:
  - Jacot et al. 2018 (two layers);
  - Du et al. 2019 (L layers, DNN);
  - Z Allen-Zhu, Y Li, Z Song 2018 (L layers, DNN, RNN);
  - D Zou*, Y Cao*, D. Zhou, and Q Gu 2018 (L layers, DNN, milder conditions)
  - Chizat et al. 2018
- Generalization theory
  - Y Cao and Q Gu, 2019a (GD)
  - Y Cao and Q Gu, 2019b (SGD)
- Consistent optimization and generalization for classification
  - Z Ji and M Telgarsky 2020
  - Z Chen*, Y Cao*, D Zou, and Q Gu 2020

Neural Tangent Kernel of Deep Learning Optimization

- **Optimization objective function**:

$$R_S(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N}(h(\boldsymbol{x}_i;\boldsymbol{\theta}) - f(\boldsymbol{x}_i))^2$$

- Introduce $\mathcal{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N\times d}$, then
  - $h(\mathcal{X};\boldsymbol{\theta}(t)) := [h(\boldsymbol{x}_i;\boldsymbol{\theta}(t))] \in \mathbb{R}^N$
  - $\nabla_{\boldsymbol{\theta}} h(\mathcal{X};\boldsymbol{\theta}(t)) := [\nabla_{\boldsymbol{\theta}_j} h(\boldsymbol{x}_i;\boldsymbol{\theta}(t))] \in \mathbb{R}^{N\times W}$
  - $\nabla_{h(\mathcal{X};\boldsymbol{\theta}(t))} R_S := \frac{2}{N}(h(\mathcal{X};\boldsymbol{\theta}(t)) - f(\mathcal{X})) := [\frac{2}{N}(h(\boldsymbol{x}_i;\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_i))] \in \mathbb{R}^N$
- **Gradient descent**

$$\begin{aligned}\boldsymbol{\theta}(t+1) &= \boldsymbol{\theta}(t) - \tau\frac{2}{N}\sum_{i=1}^{N}(h(\boldsymbol{x}_i;\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_i))\nabla_{\boldsymbol{\theta}(t)} h(\boldsymbol{x}_i;\boldsymbol{\theta}) \\ &= \boldsymbol{\theta}(t) - \tau\nabla_{\boldsymbol{\theta}} h(\mathcal{X};\boldsymbol{\theta}(t))^T \nabla_{h(\mathcal{X};\boldsymbol{\theta}(t))} R_S,\end{aligned}$$

- **Gradient flow**

$$\partial_t\boldsymbol{\theta}(t) = -\nabla_{\boldsymbol{\theta}} h(\mathcal{X};\boldsymbol{\theta}(t))^T \nabla_{h(\mathcal{X};\boldsymbol{\theta}(t))} R_S,$$

Neural Tangent Kernel of Deep Learning Optimization

- **Gradient flow**

$$\partial_t \boldsymbol{\theta}(t) = -\nabla_{\boldsymbol{\theta}} h(\mathcal{X}; \boldsymbol{\theta}(t))^T \nabla_{h(\mathcal{X};\boldsymbol{\theta}(t))} R_S,$$

- **DNN evolution**

$$\partial_t h(\mathcal{X}; \boldsymbol{\theta}(t)) = \nabla_{\boldsymbol{\theta}} h(\mathcal{X}; \boldsymbol{\theta}(t)) \partial_t \boldsymbol{\theta}(t) = -\hat{\boldsymbol{K}}_t(\mathcal{X}, \mathcal{X}) \nabla_{h(\mathcal{X};\boldsymbol{\theta}(t))} R_S$$

  with the neural tangent kernel (NTK)

$$\hat{\boldsymbol{K}}_t = \nabla_{\boldsymbol{\theta}} h(\mathcal{X}; \boldsymbol{\theta}(t)) \nabla_{\boldsymbol{\theta}} h(\mathcal{X}; \boldsymbol{\theta}(t))^T.$$

- Nonlinear ODEs and challenging to analyze

- **Linearization**

  $h^{\text{lin}}(\boldsymbol{x};\theta(t)) := h(\boldsymbol{x};\theta(0)) + \nabla_{\boldsymbol{\theta}} h(\boldsymbol{x};\theta(0))(\theta(t)-\theta(0)) \approx h(\boldsymbol{x};\theta(t)),$

- **Approximate DNN evolution**

$$\begin{aligned}
\partial_t h^{\text{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t)) &= -\hat{\boldsymbol{K}}_0(\boldsymbol{x},\mathcal{X})\nabla_{h^{\text{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))} R_S \\
&= -\hat{\boldsymbol{K}}_0(\boldsymbol{x},\mathcal{X})\frac{2}{N}(h^{\text{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t)) - f(\mathcal{X}))
\end{aligned}$$

- **Linear ODE with a solution**

  $h^{\text{lin}}(\boldsymbol{x};\theta(t)) = h(\boldsymbol{x};\theta(0)) - \hat{\boldsymbol{K}}_0(\boldsymbol{x},\mathcal{X})\hat{\boldsymbol{K}}_0^{-1}\left(I - e^{-\hat{\boldsymbol{K}}_0 t}\right)(h(\mathcal{X};\theta(0)) - \mathcal{Y})$

  and

$$h^{\text{lin}}(\mathcal{X};\theta(t)) = \left(I - e^{-\hat{\boldsymbol{K}}_0 t}\right)\mathcal{Y} + e^{-\hat{\boldsymbol{K}}_0 t} h(\mathcal{X};\theta(0)).$$

  with $\mathcal{Y} := [y_1,\ldots,y_N]^T \in \mathbb{R}^N$.

Neural Tangent Kernel of Deep Learning Optimization

Question:

How to go through the details to show the convergence to a global minimizer rigorously?

Neural Tangent Kernel of Deep Learning Optimization

Notations:

$$a_k^t := a_k(t), \quad \boldsymbol{w}_k^t := \boldsymbol{w}_k(t), \quad \boldsymbol{\theta}^t := \boldsymbol{\theta}(t) := \mathrm{vec}\{a_k^t, \boldsymbol{w}_k^t\}_{k=1}^N.$$

$$\bar{a}_k^t := \bar{a}_k(t) := \gamma^{-1} a_k(t)$$

with $0 < \gamma < 1$, e.g., $\gamma = \frac{1}{\sqrt{N}}$ or $\gamma = \frac{1}{N}$.

$$\bar{\boldsymbol{\theta}}(t) \text{ means } \mathrm{vec}\{\bar{a}_k^t, \boldsymbol{w}_k^t\}_{k=1}^N.$$

Initialization:

$$a_k^0 := a_k(0) \sim \mathcal{N}(0, \gamma^2), \quad \boldsymbol{w}_k^0 := \boldsymbol{w}_k(0) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d),$$

$$\boldsymbol{\theta}^0 := \boldsymbol{\theta}(0) := \mathrm{vec}\{a_k^0, \boldsymbol{w}_k^0\}_{k=1}^N.$$

Neural Tangent Kernel of Deep Learning Optimization

- Activation function: $\sigma(x) = \max\{x, 0\}$ for our two-layer neural network $h(\boldsymbol{x}_i; \boldsymbol{\theta})$.
- Prediction error at one sample: $e_i = h(\boldsymbol{x}_i; \boldsymbol{\theta}) - f(\boldsymbol{x}_i)$ and $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)^\intercal$.
- Empirical risk:

$$R_S(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h(\boldsymbol{x}_i; \boldsymbol{\theta}) - f(\boldsymbol{x}_i)\right)^2 = \frac{1}{2n} \boldsymbol{e}^\intercal \boldsymbol{e}.$$

- GD dynamics:

$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}), \tag{1}$$

or equivalently in terms of a_k and $\boldsymbol{w}_k$ as follows:

$$\dot{a}_k = -\nabla_{a_k} R_S(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} e_i \sigma(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i),$$

$$\dot{\boldsymbol{w}}_k = -\nabla_{\boldsymbol{w}_k} R_S(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} e_i a_k \sigma'(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i) \boldsymbol{x}_i.$$

Neural Tangent Kernel of Deep Learning Optimization

- NTK $k^{(a)}$ for parameters in the last linear transform
$$k^{(a)}(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)} g^{(a)}(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{x}'),$$
where
$$g^{(a)}(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{x}') := [\sigma(\boldsymbol{w}^\top \boldsymbol{x})] \cdot [\sigma(\boldsymbol{w}^\top \boldsymbol{x}')].$$
- NTK $k^{(w)}$ for parameters in the first layer
$$k^{(w)}(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}_{(a, \boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d+1})} g^{(w)}(a, \boldsymbol{w}; \boldsymbol{x}, \boldsymbol{x}'),$$
where
$$g^{(w)}(a, \boldsymbol{w}; \boldsymbol{x}, \boldsymbol{x}') := a^2 \left[\sigma'(\boldsymbol{w}^\top \boldsymbol{x}) \boldsymbol{x}\right] \cdot \left[\sigma'(\boldsymbol{w}^\top \boldsymbol{x}') \boldsymbol{x}'\right].$$
- Gram matrices $\boldsymbol{K}^{(a)}$ and $\boldsymbol{K}^{(w)}$ with $\boldsymbol{K}_{ij}^{(a)} = k^{(a)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{K}_{ij}^{(w)} = k^{(w)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, respectively.

Assumption

We assume that
$$\lambda_S := \lambda_{\min}\left(\boldsymbol{K}^{(a)}\right) > 0.$$

Neural Tangent Kernel of Deep Learning Optimization

- The discrete NTK matrix $\hat{\boldsymbol{K}}(\boldsymbol{\theta}) = \hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}) + \hat{\boldsymbol{K}}^{(w)}(\boldsymbol{\theta})$ with

$$\hat{\boldsymbol{K}}_{ij}^{(a)}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{k=1}^{N} g^{(a)}(\boldsymbol{w}_k; \boldsymbol{x}_i, \boldsymbol{x}_j),$$

$$\hat{\boldsymbol{K}}_{ij}^{(w)}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{k=1}^{N} g^{(w)}(a_k, \boldsymbol{w}_k; \boldsymbol{x}_i, \boldsymbol{x}_j).$$

- $\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta})$ and $\hat{\boldsymbol{K}}^{(w)}(\boldsymbol{\theta})$ are both Gram matrices (positive semi-definite).
- The dynamics of GD:

$$\frac{\mathrm{d}}{\mathrm{d}t} h(\boldsymbol{x}_i; \boldsymbol{\theta}) = -\frac{1}{n}\sum_{j=1}^{n} \hat{\boldsymbol{K}}_{ij}(\boldsymbol{\theta})(h(\boldsymbol{x}_j; \boldsymbol{\theta}) - f(\boldsymbol{x}_j))$$

  and

$$\frac{\mathrm{d}}{\mathrm{d}t} R_S(\boldsymbol{\theta}) = -\|\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta})\|_2^2 = -\frac{N}{n^2}\boldsymbol{e}^\intercal \hat{\boldsymbol{K}}(\boldsymbol{\theta})\boldsymbol{e} \leq -\frac{N}{n^2}\boldsymbol{e}^\intercal \hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta})\boldsymbol{e}.$$

Neural Tangent Kernel of Deep Learning Optimization

- The dynamics of GD:
$$\frac{\mathrm{d}}{\mathrm{d}t} R_S(\boldsymbol{\theta}) \leq -\frac{N}{n^2} \boldsymbol{e}^{\intercal} \hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}) \boldsymbol{e}. \tag{2}$$
- Goal: $h(\boldsymbol{x}_i; \boldsymbol{\theta}(t)) \to f(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i \Leftrightarrow R_S(\boldsymbol{\theta}) \to 0$.
- True if the smallest eigenvalue $\lambda_{\min}\left(\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta})\right)$ has a positive lower bound uniformly in t

Neural Tangent Kernel of Deep Learning Optimization

Introduce a stopping time $t^* = \inf\{t \mid \boldsymbol{\theta}(t) \notin \mathcal{M}(\boldsymbol{\theta}^0)\}$, where

$$\mathcal{M}(\boldsymbol{\theta}^0) := \left\{ \boldsymbol{\theta} \mid \|\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}) - \hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}^0)\|_{\mathrm{F}} \leq \frac{1}{4}\lambda_S \right\}. \tag{3}$$

For any $t \in [0, t^*)$, we have:

Approximation

- (**Initialization**) $\lambda_{\min}\left(\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}(0))\right) \approx \lambda_S$?
- (**Evolution**) $\lambda_{\min}\left(\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}(0))\right) \approx \lambda_{\min}\left(\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}(t))\right)$?

Loss bound

- The GD dynamics

$$\frac{\mathrm{d}}{\mathrm{d}t} R_S(\boldsymbol{\theta}) \leq -\frac{N}{n^2}\boldsymbol{e}^{\intercal}\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta})\boldsymbol{e}$$

- And hence

$$R_S(\boldsymbol{\theta}(t)) \leq \exp\left(-\frac{N\lambda_S t}{n}\right) R_S(\boldsymbol{\theta}^0)$$

Overparametrization:

When N is large enough, t^* is in fact equal to infinity?

Theorem (Linear convergence rate (arXiv:2106.06682))

Let $\boldsymbol{\theta}^0 := \mathrm{vec}\{a_k^0, \boldsymbol{w}_k^0\}_{k=1}^N$ at the GD initialization for minimizing MSE, where $a_k^0 \sim \mathcal{N}(0, \gamma^2)$ and $\boldsymbol{w}_k^0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ with any $\gamma \in (0,1)$. Let $C_d := \mathbb{E}\|\boldsymbol{w}\|_1^4 < +\infty$ with $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and λ_S be a positive constant in our assumption. For any $\delta \in (0,1)$, if

$$N \geq \max\left\{\frac{32n^4 C_d}{\lambda_S^2 \delta}, \frac{4\sqrt{2}dn\sqrt{R_S(\boldsymbol{\theta}^0)}}{\lambda_S}, \frac{256\sqrt{2}d^3 n^2(\log(4N(d+1)/\delta))\sqrt{R_S(\boldsymbol{\theta}^0)}}{\lambda_S^2}\right\}, \tag{4}$$

then with probability at least $1-\delta$ over the random initialization $\boldsymbol{\theta}^0$, we have, for all $t \geq 0$,

$$R_S(\boldsymbol{\theta}(t)) \leq \exp\left(-\frac{N\lambda_S t}{n}\right) R_S(\boldsymbol{\theta}^0).$$

Lemma

For any $\delta \in (0,1)$ with probability at least $1-\delta$ over the random initialization, we have

$$R_S(\boldsymbol{\theta}^0) \leq \frac{1}{2}\left(1 + 6\gamma\sqrt{N}d\left(\log\frac{4N(d+1)}{\delta}\right)\left(\sqrt{2\log(2d)} + \sqrt{2\log(8/\delta)}\right)\right)^2.$$

$R_S(\boldsymbol{\theta}^0)$ is not large.

Neural Tangent Kernel of Deep Learning Optimization

Lemma

For any $\delta \in (0,1)$, *if* $N \geq \frac{16n^4 C_d}{\lambda_S^2 \delta}$, *then with probability at least* $1-\delta$ *over the random initialization, we have*

$$\lambda_{\min}\left(\hat{\boldsymbol{K}}^{(a)}(\boldsymbol{\theta}^0)\right) \geq \frac{3}{4}\lambda_S,$$

where $C_d := \mathbb{E}\|\boldsymbol{w}\|_1^4 < +\infty$ *with* $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$.

The law of large number:

- $\lambda_S := \lambda_{\min}\left(\boldsymbol{K}^{(a)}\right) > 0$.
- Gram matrices $\boldsymbol{K}^{(a)}$ with
  $K_{ij}^{(a)} := \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_d)} g^{(a)}(\boldsymbol{w};\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_d)}\left[\sigma(\boldsymbol{w}^\intercal\boldsymbol{x})\right]\cdot\left[\sigma(\boldsymbol{w}^\intercal\boldsymbol{x}')\right].$
- $\hat{\boldsymbol{K}}_{ij}^{(a)}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{k=1}^{N} g^{(a)}(\boldsymbol{w}_k;\boldsymbol{x}_i,\boldsymbol{x}_j)$.

Neural Tangent Kernel of Deep Learning Optimization

Lemma

For any $\delta \in (0,1)$*, if* $N \geq \max\left\{\frac{32n^4 C_d}{\lambda_S^2 \delta}, \frac{4\sqrt{2}dn\sqrt{R_S(\boldsymbol{\theta}^0)}}{\lambda_S}\right\}$*, then with probability at least* $1-\delta$ *over the random initialization, for any* $t \in [0, t^*)$ *and any* $k \in [N]$*,*

$$
\begin{aligned}
|a_k(t) - a_k(0)| &\leq q, & \|\boldsymbol{w}_k(t) - \boldsymbol{w}_k(0)\|_\infty &\leq q, \\
|a_k(0)| &\leq \gamma\eta, & \|\boldsymbol{w}_k(0)\|_\infty &\leq \eta,
\end{aligned}
$$

where

$$
q := \frac{8dn\sqrt{R_S(\boldsymbol{\theta}^0)\log\frac{4N(d+1)}{\delta}}}{N\lambda_S}
$$

and

$$
\eta := \sqrt{2\log\frac{4N(d+1)}{\delta}}.
$$

- Concentration inequality.
- Parameters stay around initialization when width $N \to \infty$.

Neural Tangent Kernel of Deep Learning Optimization

Lemma: $t^* = \infty$

- $t \to t^*$, $\theta(t)$ will go out of a neighborhood of $\theta(0)$.
- When $t \in [0, t^*)$, $|\theta(t) - \theta(0)| \leq O(\frac{1}{N})$.
- Therefore, when N is large enough, $t^* = \infty$.

Optimization for PDE Solvers

Question: can we apply existing optimization analysis for PDE solvers?

A simple example

- Two-layer network: $h(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{N} a_k \sigma(\boldsymbol{w}_k^T \boldsymbol{x})$.
- A second order differential equation: $\mathcal{L}u = f$ with

$$\mathcal{L}u = \sum_{\alpha,\beta=1}^{d} A_{\alpha\beta}(\boldsymbol{x}) u_{x_\alpha x_\beta}.$$

- $f(\boldsymbol{x};\boldsymbol{\theta}) := \mathcal{L}h(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{N} a_k \boldsymbol{w}_k^T A(\boldsymbol{x}) \boldsymbol{w}_k \sigma''(\boldsymbol{w}_k^T \boldsymbol{x})$ to fit $f(\boldsymbol{x})$
- Much more difficult nonlinearity in $\boldsymbol{x}$ and $\boldsymbol{w}$ in the fitting than the original NN fitting.

Assumption

- Two-layer network: $h(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{N} a_k \sigma(\boldsymbol{w}_k^T \boldsymbol{x})$ on $[0,1]^d$.
- A second order differential equation: $\mathcal{L}u = f$ with

$$\mathcal{L}u = \sum_{\alpha,\beta=1}^{d} A_{\alpha\beta}(\boldsymbol{x}) u_{x_\alpha x_\beta} + \sum_{\alpha=1}^{d} b_\alpha(\boldsymbol{x}) u_{x_\alpha} + c(\boldsymbol{x}) u.$$

- $\mathcal{L}$ satisfies the condition: there exists $M \geq 1$ such that for all $\boldsymbol{x} \in \Omega = [0,1]^d$, $\alpha, \beta \in [d]$, we have $A_{\alpha\beta} = A_{\beta\alpha}$

$$|A_{\alpha\beta}(\boldsymbol{x})| \leq M, \quad |b_\alpha(\boldsymbol{x})| \leq M, \quad \text{and} \quad |c(\boldsymbol{x})| \leq M.$$

- Fixed n samples in the PDE domain.
- Empirical loss

$$R_S(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{\{\boldsymbol{x}_i\}_{i=1}^{n}} |\mathcal{L}h(\boldsymbol{x}_i;\boldsymbol{\theta}) - f(\boldsymbol{x}_i)|^2$$

with h satisfying boundary conditions.

Optimization for PDE Solvers

Luo and Y., arXiv:2006.15733

Theorem (Linear convergence rate)

Let $\boldsymbol{\theta}^0 := \mathrm{vec}\{a_k^0, \boldsymbol{w}_k^0\}_{k=1}^N$ be the GD initialization, where $a_k^0 \sim \mathcal{N}(0, \gamma^2)$ and $\boldsymbol{w}_k^0 \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ with any $\gamma \in (0,1)$. Let $C_d := \mathbb{E}\|\boldsymbol{w}\|_1^{12} < +\infty$ with $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ and λ_S be a positive constant. For any $\delta \in (0,1)$, if width

$$N \geq \max\left\{ \frac{512 n^4 M^4 C_d}{\lambda_S^2 \delta}, \frac{200\sqrt{2} M d^3 n \log(4N(d+1)/\delta)\sqrt{R_S(\boldsymbol{\theta}^0)}}{\lambda_S}, \right. \\ \left. \frac{2^{23} M^3 d^9 n^2 (\log(4N(d+1)/\delta))^4 \sqrt{R_S(\boldsymbol{\theta}^0)}}{\lambda_S^2} \right\},$$

then with probability at least $1-\delta$ over the random initialization $\boldsymbol{\theta}^0$, we have, for all $t \geq 0$,

$$R_S(\boldsymbol{\theta}(t)) \leq \exp\left(-\frac{N\lambda_S t}{n}\right) R_S(\boldsymbol{\theta}^0).$$

Mean-Field Analysis

- Chizat and Bach 2018; Mei et al. 2018; Mei et al. 2019, Lu et al. 2020, etc.
- Idea:
  1) a two-layer neural network can be seen as an approximation to an infinitely wide neural network with parameters following a distribution p_t;
  2) understanding network training via the evolution of p_t.