Limitations of Lazy Training of Two-layers Neural Networks

Theodor Misiakiewicz*

Stanford University

December 11, 2019

Joint work with Behrooz Ghorbani*, Song Mei*, Andrea Montanari

*Equal contributions

Lazy Training Regime

Two-layers Neural Network (NN):

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$$

Lazy Training regime: for some initialization $\boldsymbol{\theta}_i^0 = (a_i^0, \boldsymbol{w}_i^0)$

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^t) \approx f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) + \langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^0, \nabla_{\boldsymbol{\theta}} f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) \rangle$$

$$\approx 0 + \underbrace{\sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Random Features (RF) model}} + \underbrace{\sum_{i=1}^{N} \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Neural Tangent (NT) model}}$$

[Jacot, Gabriel, Hongler, 2018; Du, Zhai, Poczos, Singh, 2018; Chizat, Bach, 2018b; Arora, Du, Hu, Li, Wang, 2019; Allen-Zhu, Li, Song, 2018; Yehudai, Shamir, 2019; ...]

Lazy Training Regime

Two-layers Neural Network (NN):

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$$

Lazy Training regime: for some initialization $\boldsymbol{\theta}_i^0 = (a_i^0, \boldsymbol{w}_i^0)$

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^t) \approx f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) + \langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^0, \nabla_{\boldsymbol{\theta}} f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) \rangle$$

$$\approx 0 + \underbrace{\sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Random Features (RF) model}} + \underbrace{\sum_{i=1}^{N} \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Neural Tangent (NT) model}}$$

[Jacot, Gabriel, Hongler, 2018; Du, Zhai, Poczos, Singh, 2018; Chizat, Bach, 2018b; Arora, Du, Hu, Li, Wang, 2019; Allen-Zhu, Li, Song, 2018; Yehudai, Shamir, 2019; ...]

Lazy Training Regime

Two-layers Neural Network (NN):

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$$

Lazy Training regime: for some initialization $\boldsymbol{\theta}_i^0 = (a_i^0, \boldsymbol{w}_i^0)$

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^t) \approx f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) + \langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^0, \nabla_{\boldsymbol{\theta}} f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) \rangle$$

$$\approx 0 + \underbrace{\sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Random Features (RF) model}} + \underbrace{\sum_{i=1}^{N} \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Neural Tangent (NT) model}}$$

[Jacot, Gabriel, Hongler, 2018; Du, Zhai, Poczos, Singh, 2018; Chizat, Bach, 2018b; Arora, Du, Hu, Li, Wang, 2019; Allen-Zhu, Li, Song, 2018; Yehudai, Shamir, 2019; ...]

Lazy Training Regime

Two-layers Neural Network (NN):

$$f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$$

Lazy Training regime: for some initialization $\boldsymbol{\theta}_i^0 = (a_i^0, \boldsymbol{w}_i^0)$

$$\begin{aligned} f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^t) &\approx f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) + \langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^0, \nabla_{\boldsymbol{\theta}} f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) \rangle \\ &\approx 0 + \underbrace{\sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Random Features (RF) model}} + \underbrace{\sum_{i=1}^{N} \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Neural Tangent (NT) model}} \end{aligned}$$

[Jacot, Gabriel, Hongler, 2018; Du, Zhai, Poczos, Singh, 2018; Chizat, Bach, 2018b; Arora, Du, Hu, Li, Wang, 2019; Allen-Zhu, Li, Song, 2018; Yehudai, Shamir, 2019; ...]

Lazy Training Regime

Two-layers Neural Network (NN):

$$f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{W}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle)$$

Lazy Training regime: for some initialization $\boldsymbol{\theta}_i^0 = (a_i^0, \boldsymbol{w}_i^0)$

$$\begin{aligned} f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^t) \approx & f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) + \langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^0, \nabla_{\boldsymbol{\theta}} f_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\theta}^0) \rangle \\ \approx & 0 + \underbrace{\sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Random Features (RF) model}} + \underbrace{\sum_{i=1}^{N} \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i^0, \boldsymbol{x} \rangle)}_{\text{Neural Tangent (NT) model}} \end{aligned}$$

[Jacot, Gabriel, Hongler, 2018; Du, Zhai, Poczos, Singh, 2018; Chizat, Bach, 2018b; Arora, Du, Hu, Li, Wang, 2019; Allen-Zhu, Li, Song, 2018; Yehudai, Shamir, 2019; ...]

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $x_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(x_i) \equiv \langle x_i, \boldsymbol{B} x_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $x_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(x_i) \equiv \langle x_i, B x_i \rangle + b_0, \qquad \text{with } B \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $x_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(x_i) \equiv \langle x_i, Bx_i \rangle + b_0, \qquad \text{with } B \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $x_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(x_i) \equiv \langle x_i, \boldsymbol{B} x_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $\boldsymbol{x}_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(\boldsymbol{x}_i) \equiv \langle \boldsymbol{x}_i, \boldsymbol{B}\boldsymbol{x}_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $\boldsymbol{x}_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(\boldsymbol{x}_i) \equiv \langle \boldsymbol{x}_i, \boldsymbol{B}\boldsymbol{x}_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $\boldsymbol{x}_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(\boldsymbol{x}_i) \equiv \langle \boldsymbol{x}_i, \boldsymbol{B}\boldsymbol{x}_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Questions:

- Do RF/NT provide a good approximation to effectively trained NN?
- Do RF/NT learn effective/smart representations of the data?

Setting:

- $\boldsymbol{x}_i \sim \mathrm{N}(0, \mathbf{I}_d)$,

$$y_i = f_*(\boldsymbol{x}_i) \equiv \langle \boldsymbol{x}_i, \boldsymbol{B}\boldsymbol{x}_i \rangle + b_0, \qquad \text{with } \boldsymbol{B} \succeq 0$$

- Here $\sigma(x) = x^2$ (cf. paper for generalization)
- The neural network NN is trained by SGD
- Compare population squared error loss

Results

- $\boldsymbol{B} \in \mathbb{R}^{450\times 450}$, $\lambda_i(\boldsymbol{B}) \sim_{iid} \exp(1)$
- N varies in $\{30, \ldots, 4500\}$

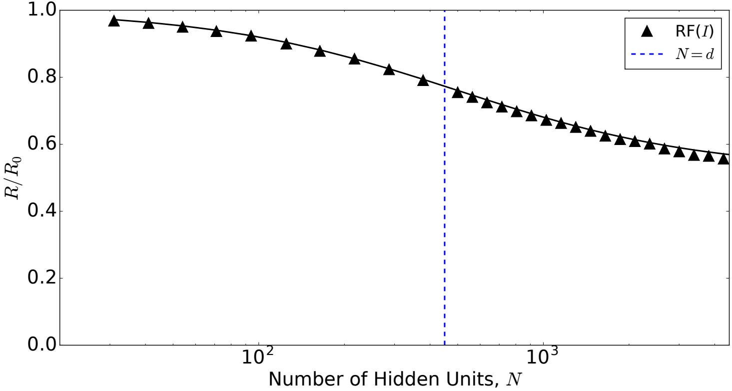

Figure: Lines are analytical predictions and dots are empirical results.

Results

- $\boldsymbol{B} \in \mathbb{R}^{450\times 450}$, $\lambda_i(\boldsymbol{b}) \sim_{iid} \exp(1)$
- N varies in $\{30, \dots, 4500\}$

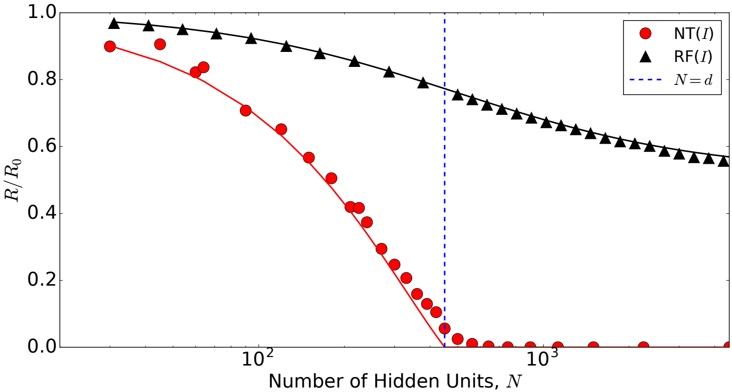

Figure: Lines are analytical predictions and dots are empirical results.

Results

- $B \in \mathbb{R}^{450 \times 450}$, $\lambda_i(b) \sim_{iid} \exp(1)$
- N varies in $\{30, \ldots, 4500\}$

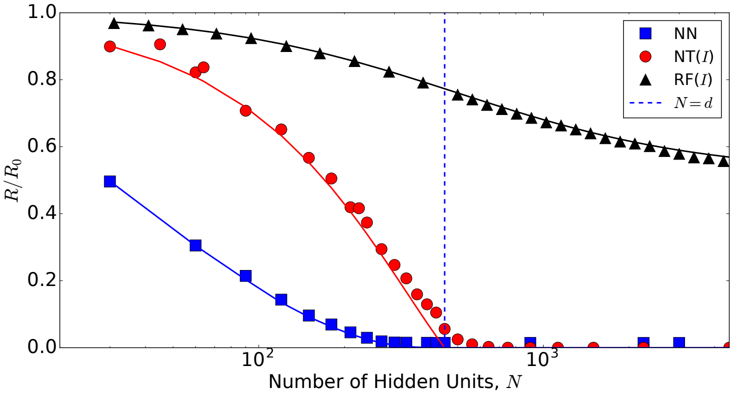

Figure: Lines are analytical predictions and dots are empirical results.

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \dots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT model fits random directions spanned by $(\boldsymbol{w}_1^0, \ldots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \ldots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \dots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \ldots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \dots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists \boldsymbol{B}$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Interpretation

- RF model does not capture quadratic functions (regardless of the non-linearity)
- The NT **model fits random directions** spanned by $(\boldsymbol{w}_1^0, \dots, \boldsymbol{w}_N^0)$
- Fully trained NN **learns the most important eigendirections**
- $\exists B$ arbitrarily large gap between NN and NT

Neural networks are superior to linearized model such as RF and NT, because they can learn a good representation of the data

These phenomena are more general: mixture of Gaussians, ReLu activation...

Thank you!

For further discussions, you can visit our poster:

Poster # 230
East Exhibition Hall B + C
5:00 - 7:00pm, Wednesday 11th

If you have any questions: please email us at misiakie@stanford.edu

“Limitations of Lazy Training of Two-layers Neural Networks”

Behrooz Ghorbani, Song Mei, **Theodor Misiakiewicz**, Andrea Montanari