# 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

#### Two-layers Neural Network (NN):

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

$$f_{\mathsf{NN}}(oldsymbol{x}; oldsymbol{ heta}^t) pprox f_{\mathsf{NN}}(oldsymbol{x}; oldsymbol{ heta}^0) + \langle oldsymbol{ heta}^t - oldsymbol{ heta}^0, 
abla_{oldsymbol{ heta}} f_{\mathsf{NN}}(oldsymbol{x}; oldsymbol{ heta}^0) 
angle \\ pprox 0 + \sum_{i=1}^N oldsymbol{t}_i \sigma(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle) \\ + \sum_{i=1}^N \langle oldsymbol{b}_i, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{x} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w} 
angle \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w} \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w} \sigma'(\langle oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, oldsymbol{w}_i^0, o$$

### Two-layers Neural Network (NN):

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

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

$$\begin{split} f_{\mathsf{NN}}(x; \boldsymbol{\theta^t}) \approx & f_{\mathsf{NN}}(x; \boldsymbol{\theta^0}) + \langle \boldsymbol{\theta^t} - \boldsymbol{\theta^0}, \nabla_{\boldsymbol{\theta}} f_{\mathsf{NN}}(x; \boldsymbol{\theta^0}) \rangle \\ \approx & 0 + \sum_{i=1}^{N} t_i \sigma(\langle \boldsymbol{w}_i^0, x \rangle) + \sum_{i=1}^{N} \langle \boldsymbol{b}_i, x \rangle \sigma'(\langle \boldsymbol{w}_i^0, x \rangle) \end{split}$$

Two-layers Neural Network (NN):

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

Lazy Training regime: for some initialization  $\theta_i^0 = (a_i^0, w_i^0)$ 

Random Features (RF) model 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; ...]

Two-layers Neural Network (NN):

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

Lazy Training regime: for some initialization  $\theta_i^0 = (a_i^0, w_i^0)$ 

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

[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; ...]

Two-layers Neural Network (NN):

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

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

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

[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; ...]

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

### Setting

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

$$y_i = f_*(oldsymbol{x}_i) \equiv \langle oldsymbol{x}_i, oldsymbol{B} oldsymbol{x}_i 
angle + b_0, \qquad ext{ with } oldsymbol{B} \succeq$$

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting

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

$$y_i = f_*(oldsymbol{x}_i) \equiv \langle oldsymbol{x}_i, oldsymbol{B} oldsymbol{x}_i 
angle + b_0, \qquad ext{with } oldsymbol{B} \succeq 0$$

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting

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

$$y_i = f_*(oldsymbol{x}_i) \equiv \langle oldsymbol{x}_i, oldsymbol{B} oldsymbol{x}_i 
angle + b_0, \qquad ext{with } oldsymbol{I}$$

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting:

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

$$g_i = f_*(oldsymbol{x}_i) \equiv \langle oldsymbol{x}_i, oldsymbol{B} oldsymbol{x}_i 
angle + b_0, \qquad ext{with } oldsymbol{B} \succeq oldsymbol{x}_i$$

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting:

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

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

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting:

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

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

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- ► Compare population squared error loss

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

### Setting:

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

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

ightharpoonup Here  $\sigma(x)=x^2$ 

- ► The neural network NN is trained by SGD
- Compare population squared error loss

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

### Setting:

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

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

ightharpoonup Here  $\sigma(x)=x^2$ 

- ▶ The neural network NN is trained by SGD
- ► Compare population squared error loss

### Results

- $lackbox{\textbf{B}} \in \mathbb{R}^{450 \times 450}, \, \lambda_i(oldsymbol{B}) \sim_{iid} \exp(1)$
- ▶ *N* varies in {30,...,4500}

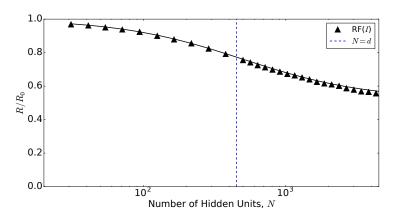


Figure: Lines are analytical predictions and dots are empirical results.

### Results

- $lackbox{\textbf{B}} \in \mathbb{R}^{450 imes 450}, \, \lambda_i(b) \sim_{iid} \exp(1)$
- ► *N* varies in {30,...,4500}

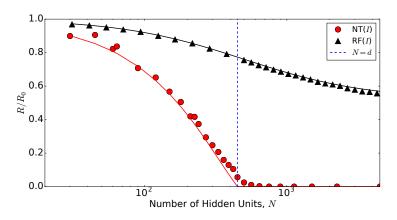


Figure: Lines are analytical predictions and dots are empirical results.

### Results

- $lackbox{\textbf{B}} \in \mathbb{R}^{450 imes 450}, \, \lambda_i(b) \sim_{iid} \exp(1)$
- ► *N* varies in {30,...,4500}

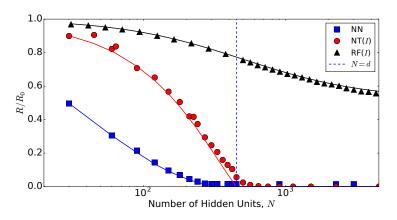


Figure: Lines are analytical predictions and dots are empirical results.

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

- ▶ RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

- ▶ RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

- ► RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

- ► RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

- ► RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

- ► RF model does not capture quadratic functions (regardless of the non-linearity)
- ightharpoonup 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
- $ightharpoonup \exists B$  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

# Thank you!

For further discussions, you can visit our poster:

 $\begin{array}{c} \text{Poster} \ \# \ 230 \\ \text{East Exhibition Hall B} + \text{C} \\ 5:00 \ \text{-} \ 7:00 \text{pm}, \ \text{Wednesday 11th} \end{array}$ 

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

8/8