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

Models

- **Two-layers Neural Network (NN) model:**

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

Lazy training regime or *Kernel limit* [Du et al.,18], [Chizat, Bach,18]

- **Random Features (RF) model:** (second layer linearization)

$$f_{\mathsf{RF}}(x; \boldsymbol{a}) = \sum_{i=1}^{N} a_i \sigma(\boldsymbol{w}_i^{\top} x) \qquad [Rahimi, Recht, 08]$$

- **Neural Tangent (NT) model:** (first layer linearization)

$$f_{\mathsf{NT}}(x; \boldsymbol{a}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, x \rangle \sigma'(\boldsymbol{w}_i^{\top} x) \qquad [Jacot\ et\ al., 18]$$

Models

- **Two-layers Neural Network (NN) model:**

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

Lazy training regime or *Kernel limit* [Du et al.,18], [Chizat, Bach,18]

- **Random Features (RF) model:** (second layer linearization)

$$f_{\mathsf{RF}}(\boldsymbol{x}; \boldsymbol{a}) = \sum_{i=1}^{N} a_i \sigma(\boldsymbol{w}_i{}^{\mathsf{T}} \boldsymbol{x}) \qquad [\textit{Rahimi}, \textit{Recht}, 08]$$

- **Neural Tangent (NT) model:** (first layer linearization)

$$f_{\mathsf{NT}}(\boldsymbol{x}; \boldsymbol{a}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \sigma'(\boldsymbol{w}_i{}^{\mathsf{T}} \boldsymbol{x}) \qquad [\textit{Jacot et al.}, 18]$$

Models

- **Two-layers Neural Network** (NN) **model:**

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

Lazy training regime or *Kernel limit* [Du et al.,18], [Chizat, Bach,18]

- **Random Features** (RF) **model:** (second layer linearization)

$$f_{\mathsf{RF}}(\boldsymbol{x};\boldsymbol{a}) = \sum_{i=1}^{N} a_i \sigma(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x}) \qquad [\textit{Rahimi}, \textit{Recht}, 08]$$

- **Neural Tangent** (NT) **model:** (first layer linearization)

$$f_{\mathsf{NT}}(\boldsymbol{x};\boldsymbol{a}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \sigma'(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x}) \qquad [\textit{Jacot et al.}, 18]$$

Models

- **Two-layers Neural Network** (NN) **model:**

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

Lazy training regime or *Kernel limit* [Du et al.,18], [Chizat, Bach,18]

- **Random Features** (RF) **model:** (second layer linearization)

$$f_{\mathsf{RF}}(\boldsymbol{x}; \boldsymbol{a}) = \sum_{i=1}^{N} a_i \sigma(\boldsymbol{w}_i{}^{\mathsf{T}} \boldsymbol{x}) \qquad [\textit{Rahimi, Recht}, 08]$$

- **Neural Tangent** (NT) **model:** (first layer linearization)

$$f_{\mathsf{NT}}(\boldsymbol{x}; \boldsymbol{a}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \sigma'(\boldsymbol{w}_i{}^{\mathsf{T}} \boldsymbol{x}) \qquad [\textit{Jacot et al.}, 18]$$

Models

- **Two-layers Neural Network** (NN) **model:**

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

Lazy training regime or *Kernel limit* [Du et al.,18], [Chizat, Bach,18]

- **Random Features** (RF) **model:** (second layer linearization)

$$f_{\mathsf{RF}}(\boldsymbol{x};\boldsymbol{a}) = \sum_{i=1}^{N} a_i \sigma(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x}) \qquad [Rahimi, Recht, 08]$$

- **Neural Tangent** (NT) **model:** (first layer linearization)

$$f_{\mathsf{NT}}(\boldsymbol{x};\boldsymbol{a}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \sigma'(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x}) \qquad [Jacot\ et\ al., 18]$$

Goal: compare these three models.

Setting:

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

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

- Activation function $\sigma(x) = x^2$
- N number of neurons: $N/d = \rho$ and d large (high-dimensional regime)

We compare the **population squared loss**:

$$R_{M,N}(f^*) = \min_{\hat{f} \in \mathcal{F}_M} \mathbb{E}_x \Big\{ \Big(f_*(x) - \hat{f}(x) \Big)^2 \Big\}, \qquad M \in \{\mathsf{RF}, \mathsf{NT}\}$$

$$R_{\mathsf{NN},N}(f^*) = \lim_{t \to \infty} \lim_{\varepsilon \to 0} \mathbb{E}_x \Big\{ \Big(f_*(x) - \hat{f}_{\mathsf{SGD}}(x; t/\varepsilon, \varepsilon) \Big)^2 \Big\}$$

Goal: compare these three models.

Setting:

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

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

- Activation function $\sigma(x) = x^2$
- N number of neurons: $N/d = \rho$ and d large (high-dimensional regime)

We compare the **population squared loss:**

$$R_{M,N}(f^*) = \min_{\hat{f} \in \mathcal{F}_M} \mathbb{E}_{\boldsymbol{x}} \Big\{ \Big(f_*(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \Big)^2 \Big\}, \qquad M \in \{\mathsf{RF}, \mathsf{NT}\}$$

$$R_{\mathsf{NN},N}(f^*) = \lim_{t \to \infty} \lim_{\varepsilon \to 0} \mathbb{E}_{\boldsymbol{x}} \Big\{ \Big(f_*(\boldsymbol{x}) - \hat{f}_{\mathsf{SGD}}(\boldsymbol{x}; t/\varepsilon, \varepsilon) \Big)^2 \Big\}$$

Goal: compare these three models.

Setting:

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

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

- Activation function $\sigma(x) = x^2$
- N number of neurons: $N/d = \rho$ and d large (high-dimensional regime)

We compare the **population squared loss**:

$$R_{M,N}(f^*) = \min_{\hat{f} \in \mathcal{F}_M} \mathbb{E}_{\boldsymbol{x}} \left\{ \left(f_*(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2 \right\}, \qquad M \in \{\mathsf{RF}, \mathsf{NT}\}$$

$$R_{\mathsf{NN},N}(f^*) = \lim_{t \to \infty} \lim_{\varepsilon \to 0} \mathbb{E}_{\boldsymbol{x}} \left\{ \left(f_*(\boldsymbol{x}) - \hat{f}_{\mathsf{SGD}}(\boldsymbol{x}; t/\varepsilon, \varepsilon) \right)^2 \right\}$$

Results

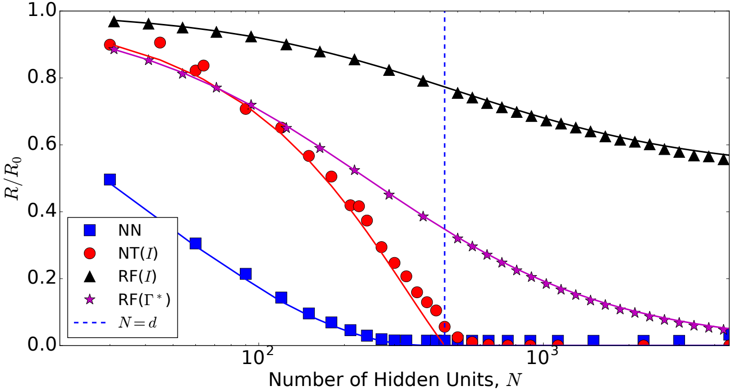

Figure: Population error in fitting a quadratic function in $d = 450$ dimensions for random features (RF), neural tangent (NT), and SGD trained neural networks (NN). Lines are analytical predictions and dots are empirical results.

RF vs NT vs NN

- RF model does not capture quadratic functions
- NN model:

$$R_{\mathsf{NN},N}(f_*) = \min_{\boldsymbol{W} \in \mathbb{R}^{N\times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{W}^\mathsf{T}\|_F^2 = 2 \sum_{i=N\wedge d+1}^{d} \lambda_i(\boldsymbol{B})^2$$

with $\lambda_1(\boldsymbol{B}) \geq \lambda_2(\boldsymbol{B}) \geq \ldots \geq \lambda_d(\boldsymbol{B}) \geq 0$.

- NT model: $\boldsymbol{W} = [\boldsymbol{w}_1^\mathsf{T}, \ldots, \boldsymbol{w}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{N\times d}$ fixed at initialization

$$R_{\mathsf{NT},N}(f_\star) = \min_{\boldsymbol{A} \in \mathbb{R}^{N\times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{A}^\mathsf{T} - \boldsymbol{A}\boldsymbol{W}^\mathsf{T}\|_F^2$$

for $N < d$, fit along $\boldsymbol{W}$ random subspace of dimension N.

RF vs NT vs NN

- RF model does not capture quadratic functions
- NN model:

$$R_{\mathsf{NN},N}(f_*) = \min_{\boldsymbol{W} \in \mathbb{R}^{N \times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{W}^\mathsf{T}\|_F^2 = 2 \sum_{i=N \wedge d+1}^{d} \lambda_i(\boldsymbol{B})^2$$

with $\lambda_1(\boldsymbol{B}) \geq \lambda_2(\boldsymbol{B}) \geq \ldots \geq \lambda_d(\boldsymbol{B}) \geq 0$.

- NT model: $\boldsymbol{W} = [\boldsymbol{w}_1^\mathsf{T}, \ldots, \boldsymbol{w}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{N \times d}$ fixed at initialization

$$R_{\mathsf{NT},N}(f_\star) = \min_{\boldsymbol{A} \in \mathbb{R}^{N \times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{A}^\mathsf{T} - \boldsymbol{A}\boldsymbol{W}^\mathsf{T}\|_F^2$$

for $N < d$, fit along $\boldsymbol{W}$ random subspace of dimension N.

RF vs NT vs NN

- RF model does not capture quadratic functions
- NN model:

$$R_{\mathsf{NN},N}(f_*) = \min_{\boldsymbol{W}\in\mathbb{R}^{N\times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{W}^\mathsf{T}\|_F^2 = 2\sum_{i=N\wedge d+1}^{d} \lambda_i(\boldsymbol{B})^2$$

with $\lambda_1(\boldsymbol{B}) \geq \lambda_2(\boldsymbol{B}) \geq \ldots \geq \lambda_d(\boldsymbol{B}) \geq 0$.

- NT model: $\boldsymbol{W} = [\boldsymbol{w}_1^\mathsf{T}, \ldots, \boldsymbol{w}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{N\times d}$ fixed at initialization

$$R_{\mathsf{NT},N}(f_\star) = \min_{\boldsymbol{A}\in\mathbb{R}^{N\times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{A}^\mathsf{T} - \boldsymbol{A}\boldsymbol{W}^\mathsf{T}\|_F^2$$

for $N < d$, fit along $\boldsymbol{W}$ random subspace of dimension N.

RF vs NT vs NN

- RF model does not capture quadratic functions
- NN model:

$$R_{\mathsf{NN},N}(f_*) = \min_{\boldsymbol{W} \in \mathbb{R}^{N \times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{W}^\mathsf{T}\|_F^2 = 2 \sum_{i=N \wedge d+1}^{d} \lambda_i(\boldsymbol{B})^2$$

with $\lambda_1(\boldsymbol{B}) \geq \lambda_2(\boldsymbol{B}) \geq \ldots \geq \lambda_d(\boldsymbol{B}) \geq 0$.

- NT model: $\boldsymbol{W} = [\boldsymbol{w}_1^\mathsf{T}, \ldots, \boldsymbol{w}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{N \times d}$ fixed at initialization

$$R_{\mathsf{NT},N}(f_\star) = \min_{\boldsymbol{A} \in \mathbb{R}^{N \times d}} 2\|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{A}^\mathsf{T} - \boldsymbol{A}\boldsymbol{W}^\mathsf{T}\|_F^2$$

for $N < d$, fit along $\boldsymbol{W}$ random subspace of dimension N.

Interpretation

- Fully trained NN **learns the most important eigendirections**, while the NT model remains confined to a **random set of directions**.

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

Mechanism more general: mixture of Gaussians, ReLu activation...

Interpretation

- Fully trained NN **learns the most important eigendirections**, while the NT model remains confined to a **random set of directions**.

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

Mechanism more general: mixture of Gaussians, ReLu activation...

Interpretation

- Fully trained NN **learns the most important eigendirections**, while the NT model remains confined to a **random set of directions**.

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

Mechanism 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