Limitations of Lazy Training of Two-layers Neural Networks

Theodor Misiakiewicz
[with Behrooz Ghorbani, Song Mei, Andrea Montanari]

Stanford University

December 2019

Linearized two-layers neural network

We focus on the functional class of two-layers neural networks on $\boldsymbol{x} \in \mathbb{R}^d$:

$$\mathcal{F}_{\mathrm{NN}} = \left\{ f_{\mathrm{NN}}(\boldsymbol{x}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle), \quad a_i \in \mathbb{R}, \boldsymbol{w}_i \in \mathbb{R}^d \right\}$$

Consider two 'linearized' function sub-classes of $\mathcal{F}_{\mathrm{NN}}$: fix $\boldsymbol{W} = \{\boldsymbol{w}_i\}_{i \in [N]}$ with $\boldsymbol{w}_i \in \mathbb{R}^d$

- *Random Features* [Rahimi-Recht 2008]

$$\mathcal{F}_{\mathrm{RF}}(\boldsymbol{W}) = \left\{ f_{\mathrm{RF}}(\boldsymbol{x}) = \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle), \quad a_i \in \mathbb{R} \right\}$$

- *Neural Tangent Kernel* [Jacot et al. 2018]

$$\mathcal{F}_{\mathrm{NT}}(\boldsymbol{W}) = \left\{ f_{\mathrm{NT}}(\boldsymbol{x}) = \sum_{i=1}^{N} \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \sigma'(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle), \quad \boldsymbol{a}_i \in \mathbb{R}^d \right\}$$

Random features and lazy training

Two interpretations:

- Random features as a computationally efficient approximation of a Kernel
- First order approximation of the model around initialization:

$$
\begin{aligned}
f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{a}^t, \boldsymbol{w}^t) \approx & f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{a}^0, \boldsymbol{w}^0) + ((\boldsymbol{a}^t, \boldsymbol{w}^t) - (\boldsymbol{a}^0, \boldsymbol{w}^0))^T \nabla f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{a}^0, \boldsymbol{w}^0) \\
\approx & 0 + f_{\mathrm{RF}}(\boldsymbol{x}; \boldsymbol{a}^t - \boldsymbol{a}^0, \boldsymbol{w}^0) + f_{\mathrm{NT}}(\boldsymbol{x}; \boldsymbol{w}^t - \boldsymbol{w}^0, \boldsymbol{w}^0)
\end{aligned}
$$

Recent string of articles: gradient descent on NN

right scaling at initialization + sufficient overparametrization

$\Rightarrow$ parameters change hardly during training (aka *lazy training* [Chizat and Bach 2018]) and model behaves like the above linear model

[Du et al 2018, Allen-Zhu et al. 2018, Arora et al. 2019]

Lazy training vs fully trained

Are deep neural networks simply efficient linear models?

Limitations of RF **and** NT **in high-dimension [Ghorbani, Mei, M., Montanari 2019]:**

If number of neurons $N \leq d^{k+1-\delta}$ for RF or $Nd \leq d^{k+1-\delta}$ for NN, then one cannot fit more than a degree-k polynomial approximation.

Class of NN is in principle much richer than random-features models and can give way better fits. But what about *effective networks* trained by gradient descent? What are the advantage of learning features against keeping the features fixed in high dimension?

Concretely, can we characterize the gap between NN trained with SGD and random features?

Setting

Setting: data $\boldsymbol{x} \sim N(0, \boldsymbol{I}_d)$, fit the target function

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

Look at the population squared loss (number of samples $\to \infty$)

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

$$R_{\mathrm{NN},N}(f^*; k, \varepsilon) = \mathbb{E}\Big\{ \Big(f_*(\boldsymbol{x}) - \hat{f}_{\mathrm{SGD}}(\boldsymbol{x}; k, \varepsilon) \Big)^2 \Big\}$$

where $\hat{f}_{\mathrm{SGD}}(\boldsymbol{x}; k, \varepsilon)$ is the two layers neural network with parameters $(a_i, \boldsymbol{w}_i)_{i \in [N]}$ obtained after k steps of SGD with step size ε.

Theoretical limits

We consider $N = \rho d$ and $\sigma(x) = x^2$.

Random Features:

$$\frac{R_{\mathrm{RF},N}(f_*)}{\|f_*\|_{L_2}^2} = 1 - \frac{\rho}{1+\rho}\frac{\mathrm{Tr}(\boldsymbol{B})^2}{d\|\boldsymbol{B}\|_F^2}$$

Neural Tangent model:

$$\frac{R_{\mathrm{NT},N}(f_*)}{\|f_*\|_{L_2}^2} = (1-\rho)_+^2 + \rho(1-\rho)_+\frac{\mathrm{Tr}(\boldsymbol{B})^2}{d\|\boldsymbol{B}\|_F^2}$$

SGD trained two-layer neural network:

$$\lim_{t\to\infty}\lim_{\varepsilon\to 0}\frac{R_{\mathrm{NN},N}(f^*; t/\varepsilon, \varepsilon)}{\|f_*\|_{L_2}^2} = 1 - \frac{\sum_{i=1}^{d\wedge N}\lambda_i(\boldsymbol{B})^2}{\|\boldsymbol{B}\|_F^2}$$

Simulations

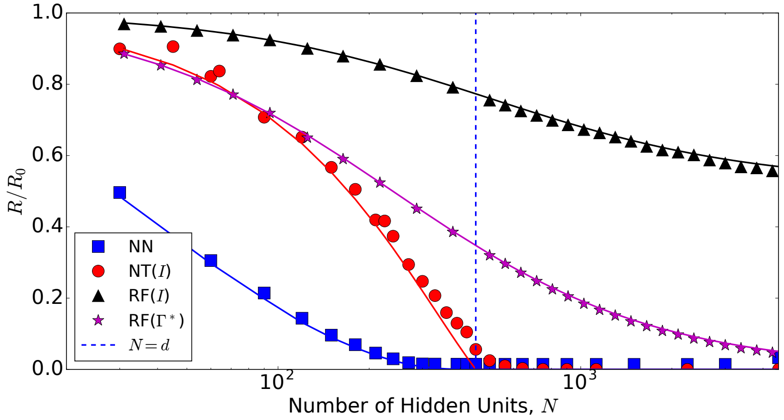

Figure: Population error in fitting a quadratic function in $d = 450$ dimensions for random features (RF), neural tangent (NT), and SGD trained neural networks (NN). Lines are analytical predictions and dots are empirical results.

Interpretation of results

- RF do not capture quadratic functions in this regime.
- NT fits the quadratic function along a random subspace determined by the fixed random weight $\boldsymbol{w}_i$
- Fully trained NN fits the N principal eigendirections of $\boldsymbol{B}$.

Fully trained NN *learns the most important eigendirections, surpassing the* NT *model which is confined to a random set of directions. Hence in this model, neural networks are superior to linearized models such as* RF *or* NT, *because they can learn a good representation of the data.*

Mechanism potentially more general. Next slides: results for mixture of two-gaussians and ReLu activation function.

Mixture of two Gaussians

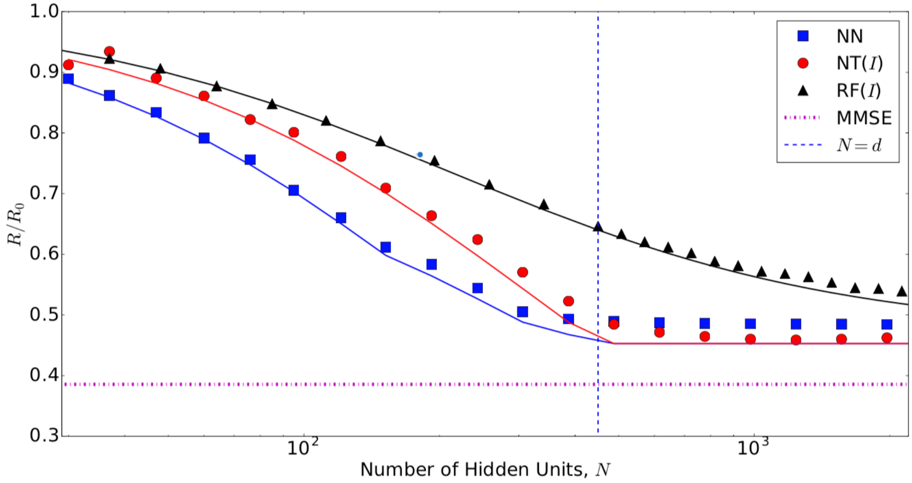

Figure: Population error in fitting a mixture of two Gaussians in $d = 450$ dimensions. Lines are analytical predictions and dots are empirical results.

ReLu activation function

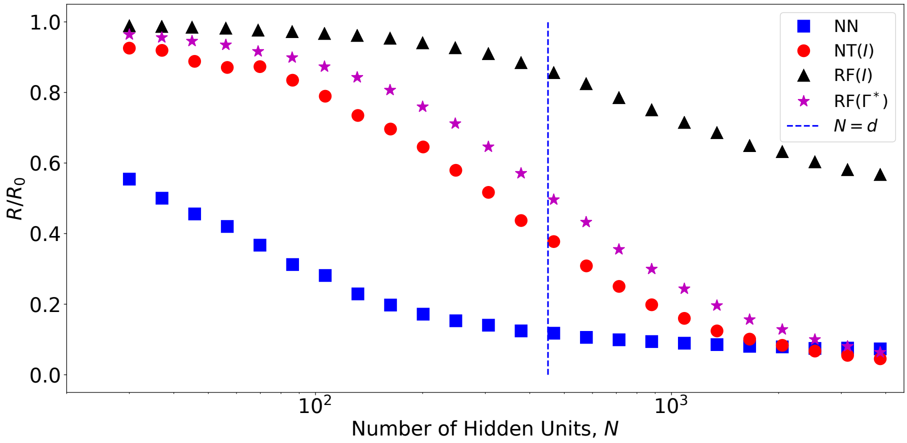

Figure: Population error in fitting a quadratic function in $d = 450$ dimensions with ReLu activation functions.