Gradient Descent for Deep Neural Networks: New Perspectives from Mean-field and NTK

Marco Mondelli

Institute of Science and Technology Austria (IST Austria)

Math ML seminar MPI MiS + UCLA, 3 March 2022

Understanding Gradient Descent for Over-parameterized Deep Neural Networks

Marco Mondelli

Institute of Science and Technology Austria (IST Austria)

Mathematics of Data Seminar, MPI MiS, 4 August 2020

New Perspectives from Mean-field and NTK

[Picture credit: F. Draxler et al.]

mode connectivity

[Picture credit: F. Draxler et al.]

mode connectivity

[Picture credit: K. Tanir, *Getty Images*]

memorization

[Picture credit: F. Draxler et al.]

[Picture credit: K. Tanir, *Getty Images*]

mode connectivity

memorization

convergence

[Picture credit: F. Draxler et al.]

[Picture credit: K. Tanir, *Getty Images*]

mode connectivity

memorization

convergence

implicit bias

A mean-field view

!

Two-layer networks

Data: $f(\boldsymbol{x}_1; y_1); : : : ; (\boldsymbol{x}_n; y_n)g \quad \text{i:i:d:} \; \mathrm{P}(\mathrm{R}^d \quad \mathrm{R})$

Goal: Minimize loss $L_N(\ ) = \mathrm{E} \quad y \quad \frac{1}{N} \sum_{i=1}^{N} a_i \ (\boldsymbol{x}; \boldsymbol{w}_i)^2 \ , \quad = (\boldsymbol{w}; a)$

Online SGD: $^{k+1} = \ ^{k} \quad N r_{\ k} \quad y_k \quad \frac{1}{N} \sum_{i=1}^{N} a_i^k \ (\boldsymbol{x}_k; \boldsymbol{w}_i^k)^2$

SGD minimizes loss $L_N(\ \ )$

$$L_N(\ \ ) = \mathrm{E}\left[y - \frac{1}{N}\sum_{i=1}^{N} a_i\ \ (\boldsymbol{x};\boldsymbol{w}_i) \right]^2; \qquad = (\boldsymbol{w};a)$$

Microscopic discrete dynamics of a gas with N particles

SGD minimizes loss $L_N(\ )$

$$L_N(\ ) = \mathrm{E}\left\{ y - \frac{1}{N}\sum_{i=1}^{N} a_i\ (\boldsymbol{x};\boldsymbol{w}_i)\ \right\}^2 ; \qquad = (\boldsymbol{w};a)$$

$L_N(\ )$ depends only on empirical distribution $\hat{\rho}^{(N)} = \frac{1}{N}\sum_{i=1}^{N} \delta_{\ i}$

SGD minimizes loss $L_N(\ )$

$$L_N(\ ) = \mathrm{E}\left[y - \frac{1}{N}\sum_{i=1}^{N} a_i\ (\boldsymbol{x};\boldsymbol{w}_i) \right]^2; \qquad = (\boldsymbol{w}; a)$$

$L_N(\ )$ depends only on empirical distribution $\mathrm{b}^{(N)} = \frac{1}{N}\sum_{i=1}^{N} {}_i$

$$\frac{1}{N}\sum_{i=1}^{N} a_i\ (\boldsymbol{x};\boldsymbol{w}_i) = \underbrace{\int a\ (\boldsymbol{x};\boldsymbol{w})\, \mathrm{b}^{(N)}(\mathrm{d}a\, \mathrm{d}\boldsymbol{w})}_{\text{expected } a \text{ w.r.t. empirical distribution } \mathrm{b}^{(N)}}$$

SGD minimizes loss $L_N(\ )$

$$L_N(\ ) = \mathrm{E}_{y}\left[\left(\frac{1}{N}\sum_{i=1}^{N} a_i\ (\boldsymbol{x};\boldsymbol{w}_i)\right)^2\right]; \qquad = (\boldsymbol{w};a)$$

expected a w.r.t. empirical distribution $b^{(N)}$

relaxation

$$L(\ ) = \mathrm{E}_{y}\left[\int a\ (\boldsymbol{x};\boldsymbol{w})\ (\mathrm{d}a\,\mathrm{d}\boldsymbol{w})\right]^2$$

expected a w.r.t.

L takes as input a probability distribution

Already exploited in [Bartlett, 1998; Bengio et al., 2006]

SGD ! gradient flow PDE

As N ! 1 and ! 0, we have:

$$b_k^{(N)} = \frac{1}{N} \sum_{i=1}^{N} \quad {}_i^k \quad) \quad k'' = \textbf{gradient flow PDE} \text{ for } L(\ )$$

Propagation-of-chaos argument [Sznitman, 1991; Mei et al., 2018]

SGD ! gradient flow PDE

[Picture credit: Peletier]

!

Discrete dynamics of SGD

Continuous dynamics of gradient flow

Two layers [Mei et al., 2018, 2019; Rotskoff et al., 2018; Chizat et al., 2018; Sirignano et al., 2018; : : :]

Multiple layers [Nguyen, 2019; Sirignano et al., 2019; Araujo et al., 2019; Pham et al., 2020; Fang et al., 2021; : : :]

Three vignettes

1. SGD weights close to **i.i.d.** particles
) *dropout stability and connectivity*

A. Shevchenko and M. Mondelli, "Landscape Connectivity and Dropout Stability of SGD Solutions for Over-parameterized Neural Networks", *ICML*, 2020.

Three vignettes

1. SGD weights close to **i.i.d.** particles
) *dropout stability and connectivity*

2. Particles evolve with **gradient flow**
) *exponential convergence rate*

A. Javanmard, M. Mondelli, and A. Montanari, "Analysis of a Two-Layer Neural Network via Displacement Convexity", *Annals of Statistics*, 2020.

Three vignettes

1. SGD weights close to **i.i.d.** particles
) *dropout stability and connectivity*

2. Particles evolve with **gradient flow**
) *exponential convergence rate*

3. **Stationary distribution** has Gibbs form
) *implicit bias*

A. Shevchenko, V. Kungurtsev, and M. Mondelli, "Mean-field Analysis of Piecewise Linear Solutions for Wide ReLU Networks", arXiv preprint, 2021.

Vignette 1: Dropout stability and connectivity

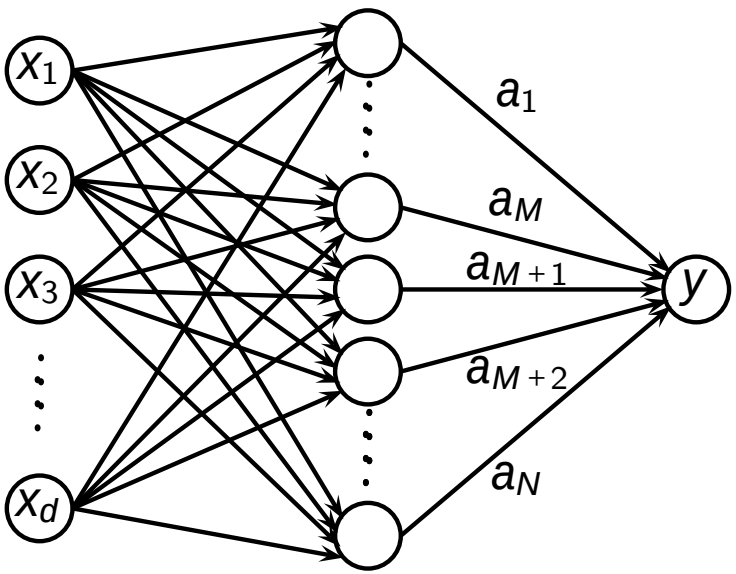
SGD minima connected via piecewise linear path with low loss [Garipov et al., 2018; Draxler et al., 2018; Fort et al., 2019; Anokhin et al., 2020]

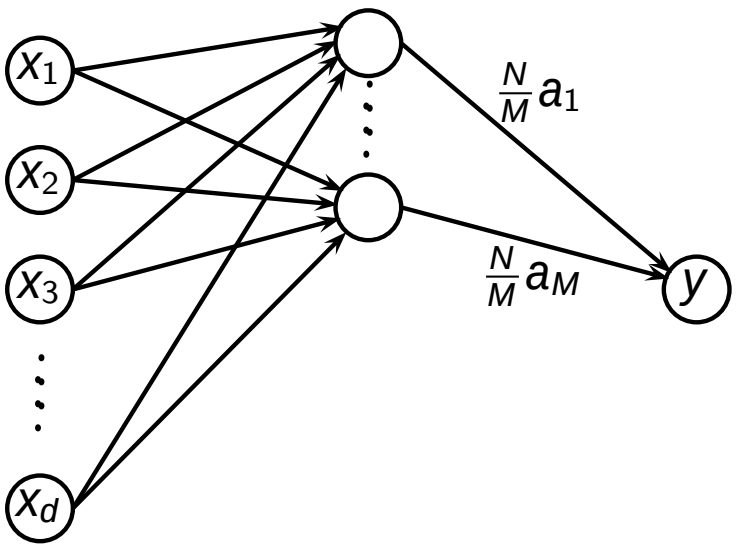
Mode connectivity proved assuming properties of well-trained networks (dropout stability and noise stability) [Kuditipudi et al., 2019]

Dropout stability

$$L_M(\ ) = \mathrm{E}\quad y\quad \frac{1}{M}\sum_{i=1}^{M} a_i\ \ (\boldsymbol{x};\boldsymbol{w}_i)\ \ ^2$$

is $"_{\mathrm{D}}$**-dropout stable** if $jL_N(\ )\quad L_M(\ )j\quad "_{\mathrm{D}}$





Dropout stability and connectivity

$$L_M(\ ) = \mathrm{E}\quad y\quad \frac{1}{M}\sum_{i=1}^{M} a_i\ \ (\boldsymbol{x};\boldsymbol{w}_i)\ \ ^2$$

is $"_{\mathrm{D}}$**-dropout stable** if $jL_N(\ )\quad L_M(\ )j\quad "_{\mathrm{D}}$

and $\ ^{0}$ are $"_{\mathrm{C}}$**-connected** if there exists a continuous path connecting them where the loss does not increase more than $"_{\mathrm{C}}$

Dropout stability, provably

N = # neurons of full network — = step size of SGD

M = # neurons after dropout — D = dimension of weights

Theorem (Shevchenko and M., 2020)

Let ^{k} be obtained after k SGD iterations. Then, with probability $1 \quad e^{\ z^2}$, for all $k\ 2\ [T = \ ]$, ^{k} is $''_{\mathrm{D}}$-dropout stable with

$$''_{\mathrm{D}} = Ke^{KT^3} \quad \frac{p}{p}\frac{\overline{\log M} + z}{\overline{M}} + p_{-}\ \mathrm{p}\frac{}{D + \log N + z} \quad :$$

Change in loss scales as $\quad \mathrm{r}\frac{}{\frac{\log M}{M}} + \mathrm{p}\frac{}{(D + \log N)}$

Dropout stability, provably

N = # neurons of full network = step size of SGD

M = # neurons after dropout D = dimension of weights

Theorem (Shevchenko and M., 2020)

Let ^{k} be obtained after k SGD iterations. Then, with probability $1 \quad e^{\ z^2}$, for all $k \ 2 \ [T = \ ]$, ^{k} is $"_{\mathrm{D}}$-dropout stable with

$$"_{\mathrm{D}} = Ke^{KT^3} \quad \frac{p\overline{\log M} + z}{p\overline{M}} + p_{-} \quad \mathrm{p}\frac{}{D + \log N + z} \quad :$$

^{k} close to N i.i.d. particles that evolve with gradient flow

$L_N(\ ^k)$ and $L_M(\ ^k)$ concentrate to the same limit

Connectivity, provably

Theorem (Shevchenko and M., 2020)

Let ^{k} be obtained after k SGD iterations using $f(\boldsymbol{x}_j; y_j)g_{j=0}^{k}$ P, and $(\ ^{0})^{k^0}$ after k^0 SGD iterations using $f(\boldsymbol{x}_j^0; y_j^0)g_{j=0}^{k^0}$ P. Then, with probability $1 \quad e^{\ z^2}$, for all $k\ 2\ [T = \ ]$ and $k^0\ 2\ [T^0 = \ ]$, ^{k} and $(\ ^{0})^{k^0}$ are $''_{\mathrm{C}}$-connected with

$$''_{\mathrm{C}} = Ke^{K \max(T;T^0)^3} \quad \frac{p\overline{\log N + z}}{p\overline{N}} + p - \mathrm{p}\frac{}{D + \log N + z} \quad :$$

Change in loss scales as $\mathrm{r}\frac{}{\frac{\log N}{N}} + \mathrm{p}\frac{}{(D + \log N)}$

Dropout stability with $M = N{=}2\ )$ connectivity

Connectivity, provably

Theorem (Shevchenko and M., 2020)

Let ^{k} be obtained after k SGD iterations using $f(\boldsymbol{x}_j; y_j)g_{j=0}^{k}$ P, and $(\ ^{0})^{k^0}$ after k^0 SGD iterations using $f(\boldsymbol{x}_j^0; y_j^0)g_{j=0}^{k^0}$ P. Then, with probability $1 \quad e^{\ z^2}$, for all $k\ 2\ [T = \ ]$ and $k^0\ 2\ [T^0 = \ ]$, ^{k} and $(\ ^{0})^{k^0}$ are $''_{\mathrm{C}}$-connected with

$$''_{\mathrm{C}} = Ke^{K\max(T;T^0)^3} \quad \frac{p\sqrt{\log N} + z}{p\sqrt{N}} + p - \mathrm{P}\frac{}{D + \log N + z} \quad :$$

Bounds similar in spirit for the **multilayer** case

Vignette 2: Exponential convergence rate

Data model

Data: $f(\boldsymbol{x}_1; y_1); : : : ; (\boldsymbol{x}_n; y_n)g \quad \text{i:i:d:} \quad \mathrm{P}(\mathrm{R}^d \quad \mathrm{R})$

$\boldsymbol{x}_i \quad \text{Unif}(\quad)$ and $y_i = f(\boldsymbol{x}_i) + \quad _i$

Data model

Data: $f(\boldsymbol{x}_1; y_1); : : : ; (\boldsymbol{x}_n; y_n)g$ $\overset{\text{i:i:d:}}{}$ $\mathrm{P}(\mathrm{R}^d \quad \mathrm{R})$

$\boldsymbol{x}_i$ Unif( ) and $y_i = f(\boldsymbol{x}_i) +$ $_i$

bounded and convex

f positive and smooth

f -strongly concave
(lower bound on curvature)

$_i$ sub-Gaussian

Problem statement

Goal: Minimize loss $L_N(\boldsymbol{w}) = \mathrm{E}\left[y - \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}; \boldsymbol{w}_i) \right]^2$

Problem statement

Goal: Minimize loss $L_N(\boldsymbol{w}) = \mathrm{E}\left[y - \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}; \boldsymbol{w}_i) \right]^2$

'bump'-like

$(\boldsymbol{x}; \boldsymbol{w}_i) = K(\boldsymbol{x} - \boldsymbol{w}_i)$

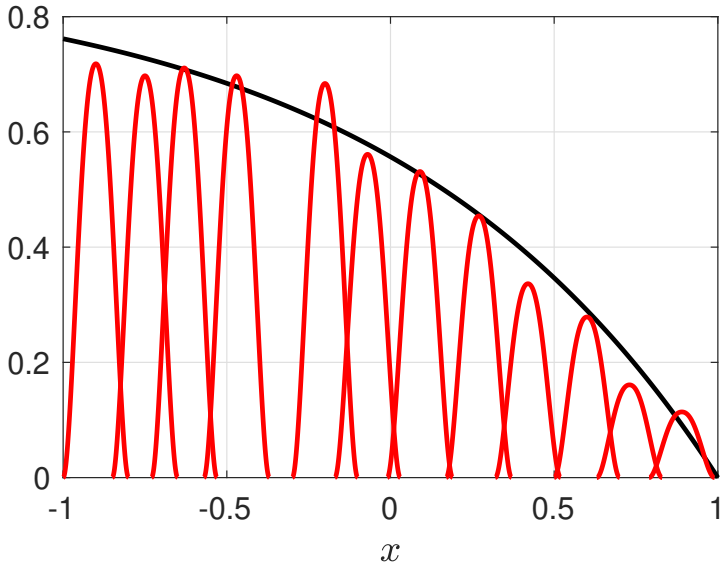
$\boldsymbol{w}_i$ = center of bump

= width of the bump

$K(\boldsymbol{x}) = {}^{d}K(\boldsymbol{x} =)$

Problem statement

Goal: Minimize loss $L_N(\boldsymbol{w}) = \mathrm{E}\left[\left(y - \frac{1}{N}\sum_{i=1}^{N} K(\boldsymbol{x}, \boldsymbol{w}_i)\right)^2\right]$



Fit data with combination of bumps

Learn N centers $\boldsymbol{w}$ via gradient descent

Non-convex optimization problem in $N \cdot d$ dimensions

Exponential and dimension-free convergence rate

Theorem (Javanmard, M. and Montanari, 2020)

Let f be -strongly concave, a bump of width , and $\boldsymbol{w}^k$ obtained after k SGD iterations with constant step size . Then, with high probability,

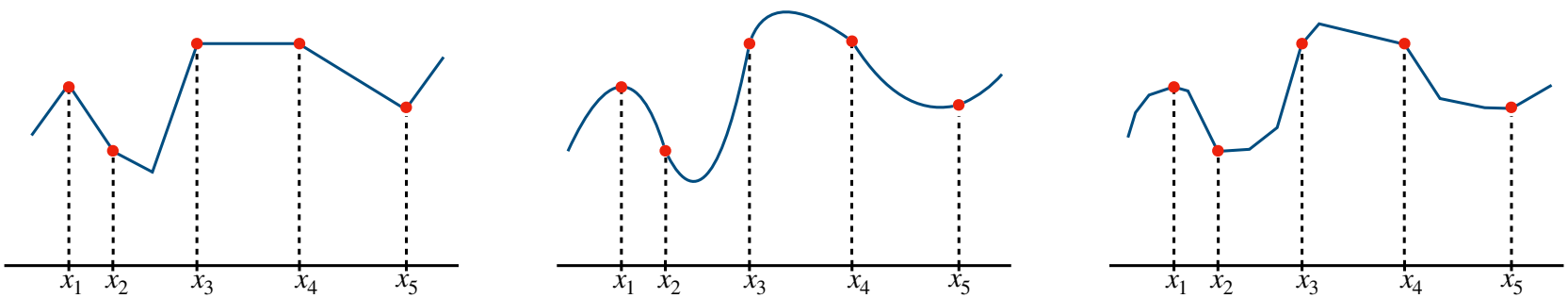
$$\underbrace{L_N(\boldsymbol{w}^k)}_{\text{loss at step } k} \quad L_N(\boldsymbol{w}^0)\, e^{\quad k}_{\text{exp. rate}} + \underbrace{(N; \quad ; \quad)}_{\text{distance to opt.}};$$

where $(N; \quad ; \quad) \; ! \; 0$ as $N \; ! \; 1$, $\quad ! \; 0$, and $\quad ! \; 0$.

As $\quad ! \; 0$, gradient flow optimizes a **displacement convex** loss.

Vignette 3: Implicit bias

Which solution does SGD choose?



Related work

Solution **implicitly biased** towards: : :

Total variation of curvature [Savarese et al., 2019; Ongie et al., 2020]

Max-margin [Soudry et al., 2018; Chizat et al., 2020]

Cubic splines [Williams et al., 2019; Jin et al., 2020]

Linear splines [Williams et al., 2019; Ergen et al., 2021]

Minimum number of tangent changes [Blanc et al., 2020]

Best fit closest to initialization [Zhang et al., 2020]

SGD + ReLU = piecewise linear solution

Theorem (Shevchenko, Kungurtsev and M., 2021)

(Informal) Consider training a two-layer ReLU network via noisy-SGD, in the low noise regime. Then, the resulting estimator converges to a **piecewise linear solution**, and the number of **knot points** – i.e., points at which distinct linear pieces connect – between two consecutive training inputs is **at most 3**.

SGD + ReLU = piecewise linear solution

Theorem (Shevchenko, Kungurtsev and M., 2021)

(Informal) Consider training a two-layer ReLU network via noisy-SGD, in the low noise regime. Then, the resulting estimator converges to a **piecewise linear solution**, and the number of **knot points** – i.e., points at which distinct linear pieces connect – between two consecutive training inputs is **at most 3**.

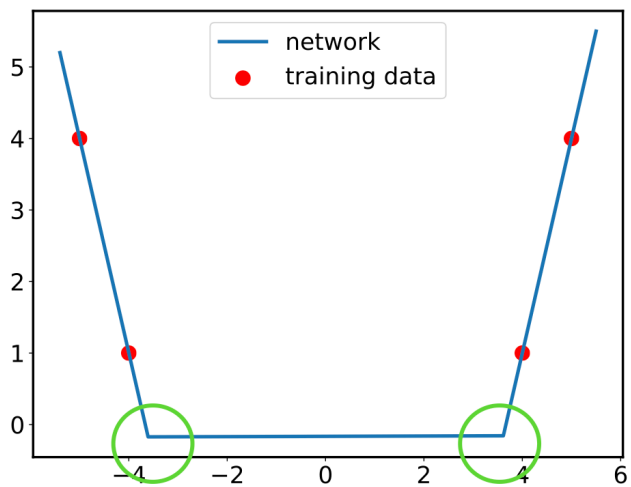
Qualitatively different behaviour from [Williams et al., 2019; Blanc et al., 2020; Jin et al., 2020; Ergen et al., 2021].

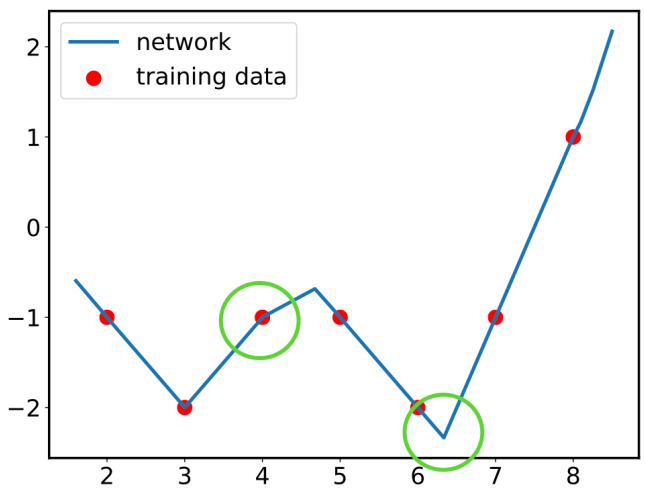
Analysis holds for both constant and vanishing ℓ_2 regularization.

SGD + ReLU = piecewise linear solution

Theorem (Shevchenko, Kungurtsev and M., 2021)

(Informal) Consider training a two-layer ReLU network via noisy-SGD, in the low noise regime. Then, the resulting estimator converges to a **piecewise linear solution**, and the number of **knot points** – i.e., points at which distinct linear pieces connect – between two consecutive training inputs is **at most 3**.





Two-layer network trained on a 1D dataset

Data: $f(x_1; y_1); : : : ; (x_n; y_n)g \quad \text{i:i:d:} \quad \mathrm{P}(\mathrm{R} \quad \mathrm{R})$

Data distribution supported on M points: $\mathrm{P} = \frac{1}{M} \sum_{j=1}^{M} \quad (x_j; y_j)$

Goal: Minimize loss

$$L_N(\ ) = \mathrm{E} \quad y \quad \frac{1}{N} \sum_{i=1}^{N} a_i \ (x; \boldsymbol{w}_i)^{\ 2} \quad ; \qquad = (\boldsymbol{w}; \boldsymbol{a})$$

Online SGD:

$$^{k+1} = \ ^{k} \quad N r_{\ k} \quad y_k \quad \frac{1}{N} \sum_{i=1}^{N} a_i^k \ (x_k; \boldsymbol{w}_i^k)^{\ 2}$$

Two-layer network trained on a 1D dataset

Data: $f(x_1; y_1); : : : ; (x_n; y_n)g \quad \text{i:i:d:} \; \mathrm{P}(\mathrm{R} \quad \mathrm{R})$

Data distribution supported on M points: $\mathrm{P} = \frac{1}{M} \sum_{j=1}^{M} \quad (x_j; y_j)$

Goal: Minimize **regularized** loss

$$L_N(\ ) = \frac{1}{M} \sum_{j=1}^{M} \quad y_j \quad \frac{1}{N} \sum_{i=1}^{N} a_i \quad (x_j; \boldsymbol{w}_i) \quad ^2 + \frac{}{N} \sum_{i=1}^{N} k \quad _i k^2; \qquad = (\boldsymbol{w}; \boldsymbol{a})$$

Online SGD:

$$^{k+1} = \quad ^k \qquad N r \quad _k \qquad y_k \quad \frac{1}{N} \sum_{i=1}^{N} a_i^k \quad (x_k; \boldsymbol{w}_i^k) \quad ^2$$

Two-layer network trained on a 1D dataset

Data: $f(x_1; y_1); : : : ; (x_n; y_n)g \quad \text{i:i:d:} \quad \mathrm{P}(\mathrm{R} \quad \mathrm{R})$

Data distribution supported on M points: $\mathrm{P} = \frac{1}{M} \sum_{j=1}^{M} \quad (x_j; y_j)$

Goal: Minimize **regularized** loss

$$L_N(\ ) = \frac{1}{M} \sum_{j=1}^{M} \quad y_j \quad \frac{1}{N} \sum_{i=1}^{N} a_i \quad (x_j; \boldsymbol{w}_i) \quad ^2 + \frac{}{N} \sum_{i=1}^{N} k \quad _i k^2; \qquad = (\boldsymbol{w}; \boldsymbol{a})$$

Online noisy-SGD:

$$^{k+1} = (1 \qquad) \quad ^k \quad N r \quad _k \quad y_k \quad \frac{1}{N} \sum_{i=1}^{N} a_i^k \quad (x_k; \boldsymbol{w}_i^k) \quad ^2 \quad + \quad \sqrt{r} \; \boldsymbol{g}^k$$

$$\boldsymbol{g}^k \qquad N(0; \boldsymbol{I})$$

Mean-field analysis

As $N \; ! \; 1$ and $\quad ! \; 0$, ^{k} close to N i.i.d. particles $\quad _{k} =$ solution of gradient flow

Mean-field analysis

As $N \to \infty$ and $\to 0$, ^{k} close to N i.i.d. particles $_{k}$ = solution of gradient flow

As $t = k \to \infty$, $_{t}$ close to = minimizer of free-energy

$$( ) = \frac{1}{Z} \exp (\quad (;))$$

$$(;) = \frac{1}{M} \sum_{j=1}^{M} \int a^{0} (x_j; \boldsymbol{w}^{0}) (\mathrm{d}a^{0} \mathrm{d}\boldsymbol{w}^{0}) \quad y_j \quad a (x_j; \boldsymbol{w}) + \frac{}{2} k \quad k_2^2$$

Main technical result

Theorem (Shevchenko, Kungurtsev and M., 2021)

Consider the **curvature** of the neural network estimator:

$$C(x) = \frac{d^2}{dx^2} \int a \quad (x; \boldsymbol{w}) \quad (\mathrm{d}a\, \mathrm{d}\boldsymbol{w}):$$

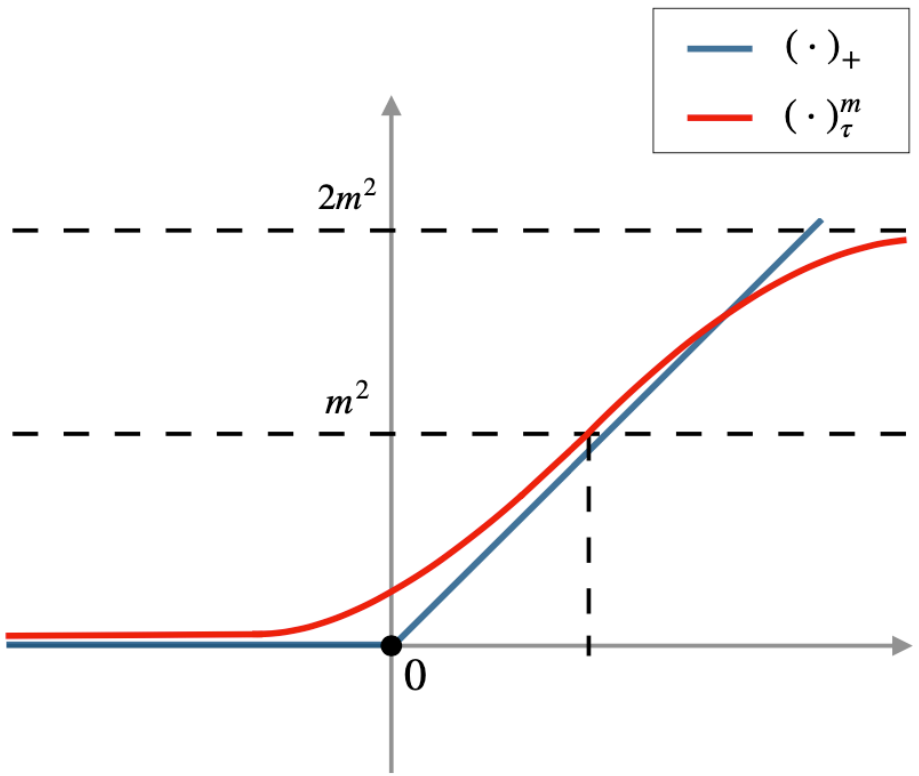
Then, as ! 1 , $C(x) = 0$ for all x except a set of **vanishing measure** which concentrates on **at most 3 points** per consecutive training inputs.

Characterization of capturing the location of the knot points

Proof ideas (1)

$a\ \ (x;\boldsymbol{w}) = a(x\ \ w + b)_+$ neither bounded nor smooth $\ )$

need a sequence of approximations



Proof ideas (2)

Taking two derivatives of ReLU gives Dirac delta

$$C(x) = \frac{d^2}{dx^2} \int a \quad (x; \boldsymbol{w}) \quad (\mathrm{d}a\, \mathrm{d}\boldsymbol{w}) \int a\, w^2 \quad (a; w; \quad w\ x)\mathrm{d}a\, \mathrm{d}w$$

Proof ideas (2)

Taking two derivatives of ReLU gives Dirac delta) analysis of 2D 'Gaussian-like' integral

$$C(x) = \frac{d^2}{dx^2} \int a \quad (x; \boldsymbol{w}) \quad (\mathrm{d}a\, \mathrm{d}\boldsymbol{w}) \int a\, w^2 \quad (a; w; \quad w\ x)\mathrm{d}a\, \mathrm{d}w$$

$$(a; w; \quad w\ x) = \frac{\exp\left\{ \frac{}{2} \quad 2aw(A^j x \quad B^j) + a^2 + w^2 + w^2 x^2 \right\}}{Z}$$

$w > 0$, x 2 $[x_j; x_{j+1}]$

A^j and B^j coefficients (independent of $x; a; w$)

Z normalization constant

Proof ideas (3)

Taking two derivatives of ReLU gives Dirac delta) analysis of 2D 'Gaussian-like' integral

$$C(x) \qquad \int aw^2 \frac{\exp\left\{ -\frac{}{2} \; 2aw(A^j x \quad B^j) + a^2 + w^2 + w^2 x^2 \right\}}{Z} \mathrm{d}a \, \mathrm{d}w$$

Proof ideas (3)

Taking two derivatives of ReLU gives Dirac delta) analysis of 2D 'Gaussian-like' integral

$$C(x) \quad \int aw^2 \frac{\exp\left\{ -\frac{}{2} \; 2aw(A^j x \quad B^j) + a^2 + w^2 + w^2 x^2 \right\}}{Z} \mathrm{d}a \, \mathrm{d}w$$

$C(x)$ blows up when covariance of Gaussian not PSD; otherwise $C(x)$! 0

A Neural Tangent Kernel (NTK) view

$$\boldsymbol{K}^{(L)} = \sum_{l=1}^{L} \frac{@\boldsymbol{F}_L}{@\mathrm{vec}(\boldsymbol{W}_l)} \frac{@\boldsymbol{F}_L}{@\mathrm{vec}(\boldsymbol{W}_l)}^T \quad 2 \mathrm{R}^{N \times N}$$

Neural Tangent Kernel (NTK)

N training samples in R^d

Neural network of depth L

$\boldsymbol{W}_l \in \mathrm{R}^{n_{l-1} \times n_l}$ weights of the network ($n_0 = d$)

$\boldsymbol{F}_L \in \mathrm{R}^N$ network output corresponding to training data

$$\boldsymbol{K}^{(L)} = \sum_{l=1}^{L} \frac{\partial \boldsymbol{F}_L}{\partial \mathrm{vec}(\boldsymbol{W}_l)} \frac{\partial \boldsymbol{F}_L}{\partial \mathrm{vec}(\boldsymbol{W}_l)}^T \in \mathrm{R}^{N \times N}$$

NTK properties

Convergence to **non-random limit** [Jacot et al., 2018]

$$\left(\prod_{l=1}^{L-1} \frac{2}{n_l} \right) \boldsymbol{K}^{(L)} \xrightarrow{p} \boldsymbol{K}^{(L)}$$

Connection to **memorization** and **generalization bounds** [Arora et al., 2019; Montanari et al., 2020]

Connection to **convergence of gradient descent**

$$\| r(\theta) \|_2^2 \geq \lambda_{\min}\left(\boldsymbol{K}^{(L)} \right) \cdot 2 \cdot \ell(\theta)$$

NTK properties

Convergence to **non-random limit** [Jacot et al., 2018]

$$\left(\sum_{l=1}^{L} \frac{2}{n_l} \right) \boldsymbol{K}^{(L)} \xrightarrow{p} \boldsymbol{K}^{(L)}$$

Connection to **memorization** and **generalization bounds** [Arora et al., 2019; Montanari et al., 2020]

Connection to **convergence of gradient descent**

$$\| r(\cdot) \|_2^2 \geq \lambda_{\min}\left(\boldsymbol{K}^{(L)} \right) \| \cdot \|_2$$

Only **two layers** or **all layers with poly(N) neurons** /

Two vignettes

1. Global convergence for deep networks with one wide layer + pyramidal topology

Q. Nguyen and M. Mondelli, "Global Convergence of Deep Networks with One Wide Layer Followed by Pyramidal Topology", *NeurIPS*, 2020.

Two vignettes

1. Global convergence for deep networks with one wide layer + pyramidal topology

2. Bounds on NTK spectrum for deep ReLU networks with one wide layer

Q. Nguyen, M. Mondelli and G. Montufar, "Tight Bounds on the Smallest Eigenvalue of the Neural Tangent Kernel for Deep ReLU Networks", *ICML*, 2021.

Convergence for (very) wide networks

	Deep?	Activation	Layer Width	Train All Layers?	# Wide Layers
[Oymak et al., '20]	No	Smooth	$(N^2 \quad {}_0{}^2)$	No	x
[Song et al., '21]	No	Smooth	$(N^{3=2})$	Yes	x
[Allen-Zhu et al., '18]	Yes	General	$(N^{24}L^{12} \quad {}^4)$	No	All
[Allen-Zhu et al., '18]	Yes	General	$(N^{24}L^{12} \quad {}^4)$	No	All
[Zou et al., '19]	Yes	ReLU	$(N^8L^{12} \quad {}^4)$	No	All
[Du et al., '19]	Yes	Smooth	$(\frac{N^4 2^{\,0\,(L)}}{{}^4_{\min}(\boldsymbol{K}^{(L)})})$	Yes	All

Vignette 1: Global convergence

	Deep?	Activation	Layer Width	Train All Layers?	# Wide Layers
[Oymak et al., '20]	No	Smooth	$(N^2 \quad {}_0{}^2)$	No	x
[Song et al., '21]	No	Smooth	$(N^{3=2})$	Yes	x
[Allen-Zhu et al., '18]	Yes	General	$(N^{24}L^{12} \quad {}^4)$	No	All
[Zou et al., '19]	Yes	ReLU	$(N^8L^{12} \quad {}^4)$	No	All
[Du et al., '19]	Yes	Smooth	$(\frac{N^4 2^{\,O\,(L)}}{\min^4(\boldsymbol{K}^{(L)})})$	Yes	All
[Nguyen & M., '20] (Gen.)	**Yes**	**Smooth**	$\boldsymbol{N}$	**Yes**	**One**
[Nguyen & M., '20] (LeCun)	**Yes**	**Smooth**	$\Omega(\boldsymbol{N^2 2^{\mathcal{O}(L)}})$	**Yes**	**One**

Need only one wide layer + pyramidal topology

Connection to the NTK

General recipe

Lipschitz gradient + PL inequality $\Rightarrow$ Linear convergence of GD

$$k\,r \quad k_2^2 \qquad \min \quad \boldsymbol{K}^{(L)} \quad 2$$

Connection to the NTK

General recipe

Lipschitz gradient + PL inequality $\Rightarrow$ Linear convergence of GD

$$k\,r \quad k_2^2 \qquad \min \quad \boldsymbol{K}^{(L)} \quad 2$$

$$+ \qquad \text{(pyramidal topology)}$$

$$k\,r \quad k \qquad \min\,(\boldsymbol{F}_1) \prod_{l=3} \min\,(\boldsymbol{W}_l)^{p-} \qquad \text{(weak PL inequality)}$$

= loss function

$\boldsymbol{F}_1$ feature matrix of first layer

$\boldsymbol{W}_l$ weight matrix of layer l

Maintaining the (weak) PL inequality

General recipe

Lipschitz gradient + PL inequality $\Rightarrow$ Linear convergence of GD

$$k\,r \quad k \qquad \min(\boldsymbol{F}_1) \prod_{l=3}^{L} \min(\boldsymbol{W}_l)^{p} \qquad \text{(weak PL inequality)}$$

$\boldsymbol{F}_1; \boldsymbol{W}_3; \ldots; \boldsymbol{W}_L$ are **full rank** at initialization

Maintaining the (weak) PL inequality

General recipe

Lipschitz gradient + PL inequality $\Rightarrow$ Linear convergence of GD

$$\|r\| \quad \|\quad \min(\boldsymbol{F}_1) \prod_{l=3} \min(\boldsymbol{W}_l)^{p} \quad \text{(weak PL inequality)}$$

$\boldsymbol{F}_1; \boldsymbol{W}_3; \ldots; \boldsymbol{W}_L$ stay **full rank** during training

$$\|r\| \quad (\quad) \quad r \quad (\quad^{0})\| \quad Q(\quad;\quad^{0})\| \quad \|^{0} \quad \text{(locally Lipschitz gradient)}$$

Open questions

ReLU activations?

Arbitrary topologies?

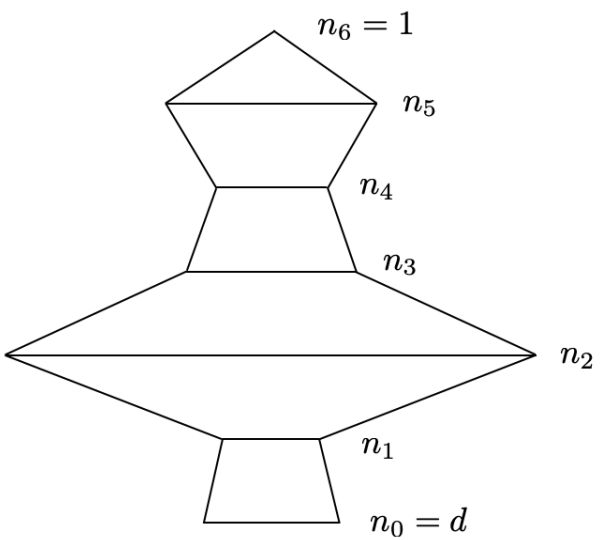
Trade-off width for depth (e.g., two layers with $(\sqrt{N})$ neurons, rather than one layer with (N) neurons)?

Vignette 2: Bounds on NTK spectrum

Main results (Nguyen, M. and Montufar, 2021)

Tight **lower bounds on the smallest eigenvalue** of NTK matrices for deep ReLU networks:

1. **Asymptotic** setting with infinite widths
2. Finite widths: (at least) **one wide layer** with (N) neurons + all other layers with arbitrary sublinear widths



$$n_2 = \Omega(N)$$

$$d, n_1, n_3, n_4, n_5 \ll N$$

NTK with one wide layer

Theorem 4.1 (Nguyen, M. and Montufar, 2021)

Let $(\boldsymbol{W}_l)_{i,j} \overset{\text{i.i.d.}}{\sim} N(0, \frac{2}{n_l})$. Assume that the data points $\{\boldsymbol{x}_i\}_{i=1}^N$ are i.i.d. with $\|\boldsymbol{x}_i\| \sim \sqrt{d}$ and satisfying Lipschitz concentration. Let $_l = 1$ if $n_l = (N \operatorname{polylog}(N))$ and $_l = 0$, otherwise. Then, with high probability,

$$O\left(d \left(\prod_{l=1}^{L-1} n_l\right) \left(\sum_{l=1}^{L} \frac{1}{l}\right) \left(\sum_{l=1}^{L} l \left(\prod_{l}^{L-1} l\right)^2\right)\right) \geq \min \boldsymbol{K}^L$$

$$\geq d \left(\prod_{l=1}^{L-1} n_l\right) \left(\sum_{l=1}^{L} \frac{1}{l}\right) \left(\sum_{l=1}^{L} l \left(\prod_{l}^{L-1} l\right)^2\right)$$

NTK with one wide layer

Theorem 4.1 (Nguyen, M. and Montufar, 2021)

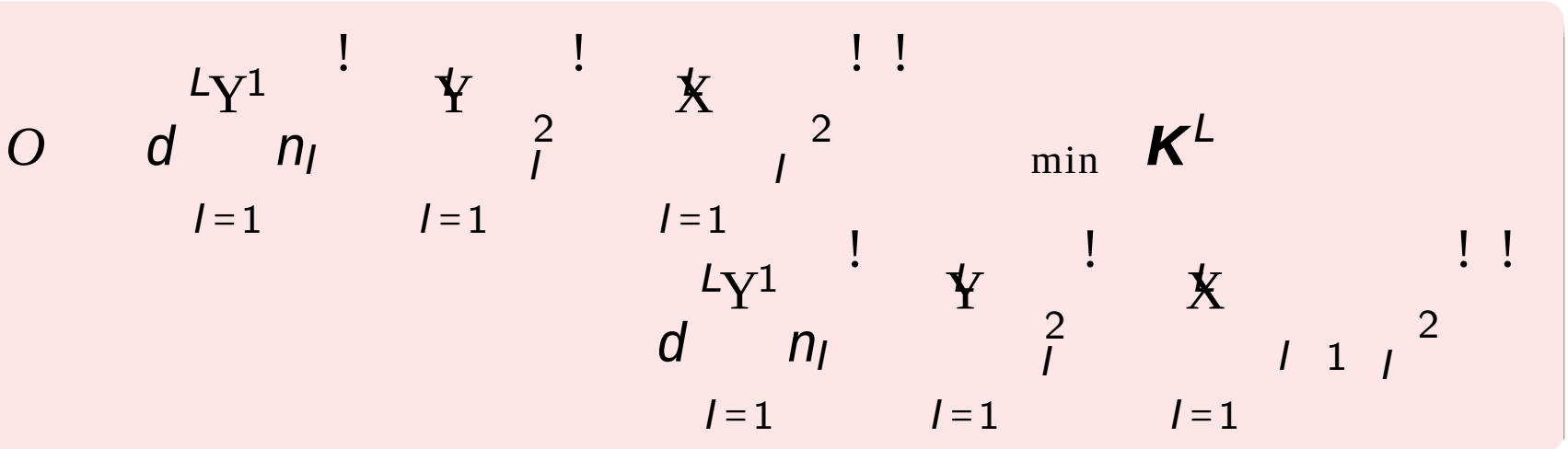
Let $(\boldsymbol{W}_l)_{i,j} \overset{\text{i.i.d.}}{\sim} N(0, \frac{2}{n_l})$. Assume that the data points $\{\boldsymbol{x}_i\}_{i=1}^N$ are i.i.d. with $\|\boldsymbol{x}_i\| \sim \sqrt{d}$ and satisfying Lipschitz concentration. Let $_l = 1$ if $n_l = (N \operatorname{polylog}(N))$ and $_l = 0$, otherwise. Then, with high probability,

$$O\left(d^{\left(\prod_{l=1}^{L-1} n_l\right)} \left(\prod_{l=1}^{L} {}_l^2\right) \left(\sum_{l=1}^{L} {}_l^2 \min \boldsymbol{K}^L\right)\right)$$

$$d^{\left(\prod_{l=1}^{L-1} n_l\right)} \left(\prod_{l=1}^{L} {}_l^2\right) \left(\sum_{l=1}^{L} {}_l {}_{l-1} {}_l^2\right)$$

9 layer with $\tilde{}(N)$ neurons $)$ **NTK bounded away from 0**

NTK with one wide layer



9 layer with $\tilde{}(N)$ neurons $)$ **NTK bounded away from 0**

Regardless of position of wide layer

Allow for arbitrary (polynomial) bottlenecks

NTK with one wide layer

$$O\left(d\prod_{l=1}^{L-1} n_l\left(\sum_{l=1}^{L}\left(\prod_{l'=l}^{L}\dot{\sigma}_{l'}\right)^2\right)\left(\sum_{l=1}^{L}\left(\prod_{l'=l}^{L}\dot{\sigma}_{l'}\right)^2\right)\min_{l}\lambda_{\min}\left(\boldsymbol{K}^{L}\right)\right)$$

$$O\left(d\prod_{l=1}^{L-1} n_l\left(\sum_{l=1}^{L}\left(\prod_{l'=l}^{L}\dot{\sigma}_{l'}\right)^2\right)\left(\sum_{l=1}^{L}\left(\prod_{l'=l}^{L}\dot{\sigma}_{l'}\right)^2\right)\left(\prod_{l=1}^{L}\dot{\sigma}_{l}^2\right)\lambda_{\min}\left(\boldsymbol{K}^{L}\right)\right)$$

Immediate consequence on **memorization capacity**

Matching bounds if 9 layer with $\tilde{}(N)$ neurons + LeCun initialization ($\dot{\sigma}_l^2 \quad 1=n_l \;_1$)

Numerical results

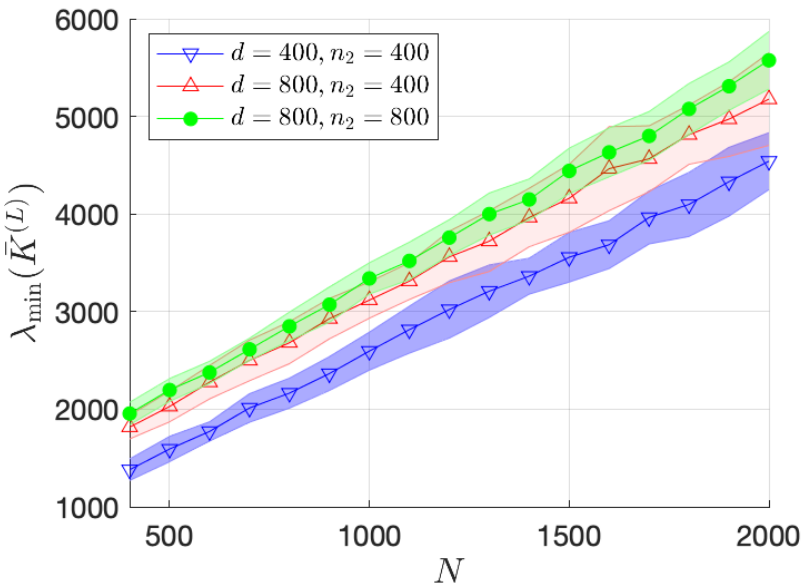
$$\min K^L = \max_{l \in \{0,1,2\}} n_l$$

$L = 3$

$(W_l)_{ij} \quad N(0; 2/n_{l-1})$

3 choices of $(d; n_2)$

$n_1 = 8N$



Smallest eigenvalue of NTK **scales linearly** in N (and in n_1)