Loss Landscape and Performance in Deep Learning

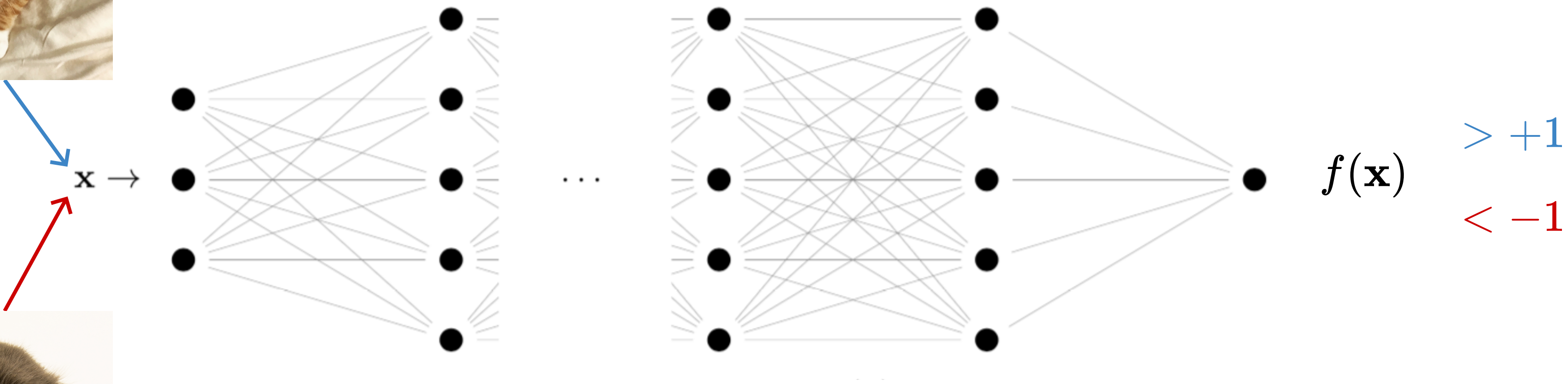
EPFL

Stefano Spigler

M. Geiger, A. Jacot, S. d'Ascoli, M. Baity-Jesi,
L. Sagun, G. Biroli, C. Hongler, M. Wyart

arXivs: 1901.01608; 1810.09665; 1809.09349

(Supervised) Deep Learning



- Learning from examples: **train set**

- Is able to predict: **test set**

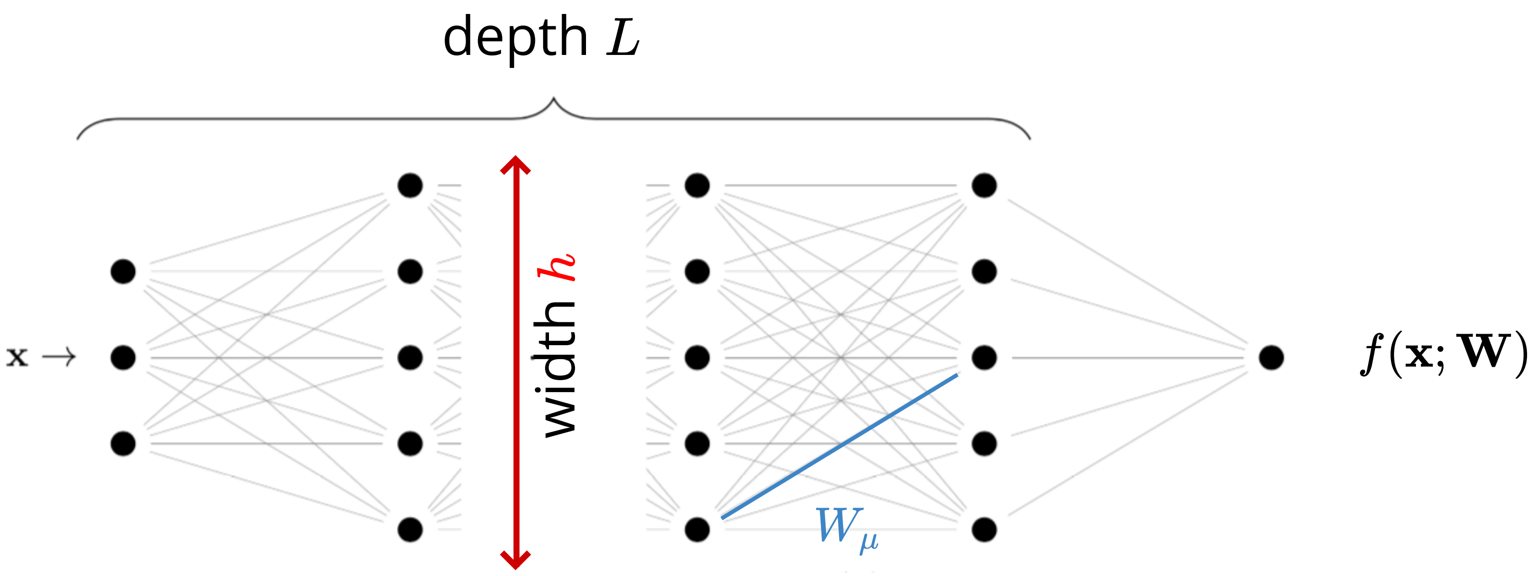
- Not understood why it works so well!

- How many data are needed to learn?

- What network size?

Set-up: Architecture

- Deep net $f(\mathbf{x}; \mathbf{W})$ with $N \sim h^2 L$ parameters



- Alternating *linear* and *nonlinear* operations!

Set-up: Dataset

- P training data:

$$\mathbf{x}_1, \ldots, \mathbf{x}_P$$

- Binary classification:

$$\mathbf{x}_i \to \text{label } y_i = \pm 1$$

$\pm 1 =$ cats/dogs, yes/no, even/odd...

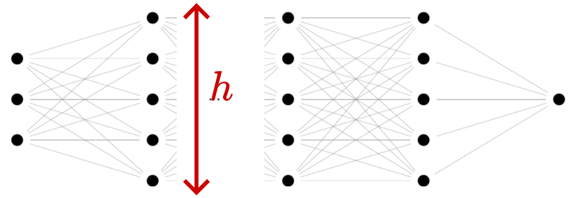
- Independent test set to evaluate performance

Example - MNIST (parity):

70k pictures, digits $0, \ldots, 9$;

use parity as label

Outline



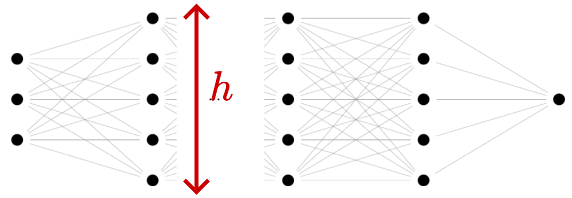
Vary **network size** N ($\sim h^2$):

1. Can networks fit **all** the P training data?

2. Can networks overfit? Can N be too large?

$\rightarrow$ Long term goal: how to choose N?

Outline



Vary **network size** N ($\sim h^2$):

1. Can networks fit **all** the P training data?

2. Can networks overfit? Can N be too large?

$\rightarrow$ Long term goal: how to choose N?

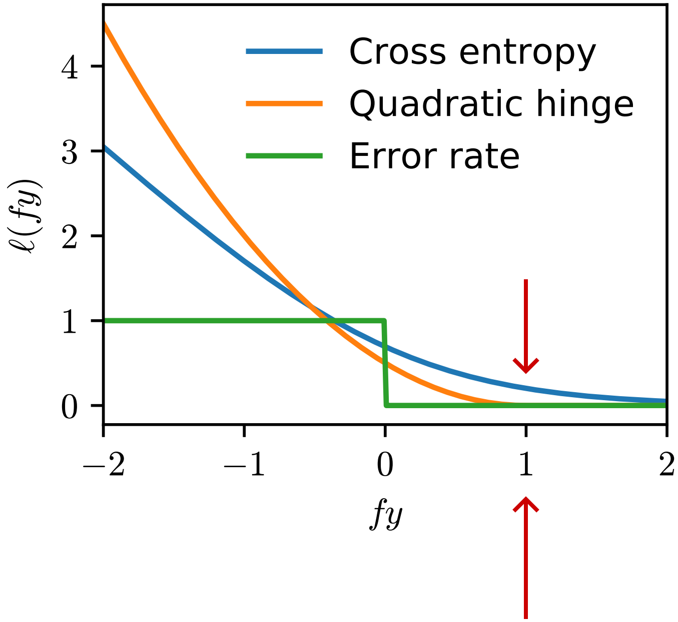
Learning

- Find parameters $\mathbf{W}$ such that $\text{sign} f(\mathbf{x}_i; \mathbf{W}) = y_i$ for $i \in$ train set

Binary classification:
$y_i = \pm 1$

- **Minimize some loss!**

$$\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{P} \ell\left(y_i f(\mathbf{x}_i; \mathbf{W})\right)$$



(classified correctly with some margin)

Hinge loss:

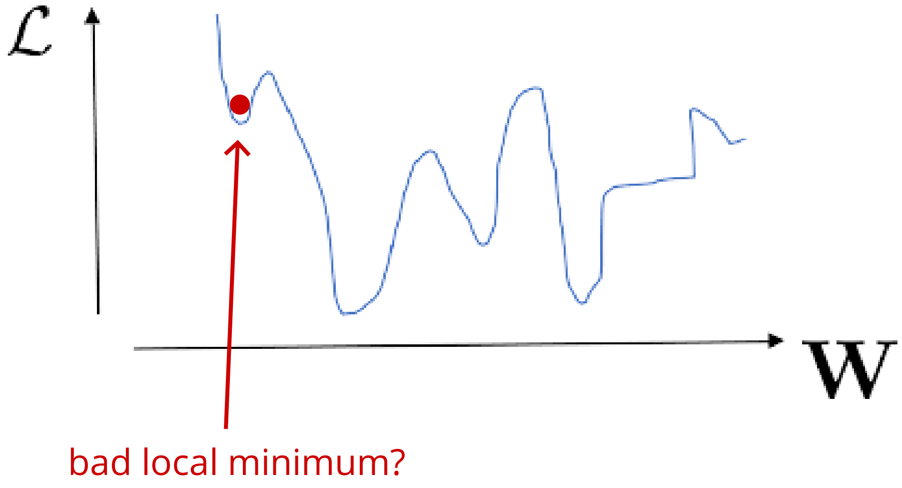
- $\mathcal{L}(\mathbf{W}) = 0$ if and only if $y_i f(\mathbf{x}_i; \mathbf{W}) > 1$ for all patterns

Learning dynamics = descent in loss landscape

- Minimize loss $\longleftrightarrow$ **gradient descent**

- Start with **random initial conditions!**

Random, high dimensional, not convex landscape!



- Why not stuck in bad local minima?

- What is the landscape geometry?

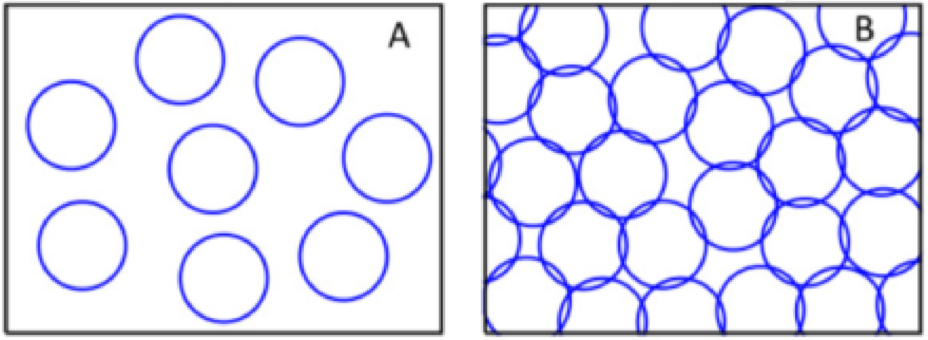
in practical settings:

- Many flat directions are found!

Soudry, Hoffer '17; Sagun et al. '17; Cooper '18; Baity-Jesy et al. '18 - arXiv:1803.06969

Analogy with granular matter: Jamming

Random packing:

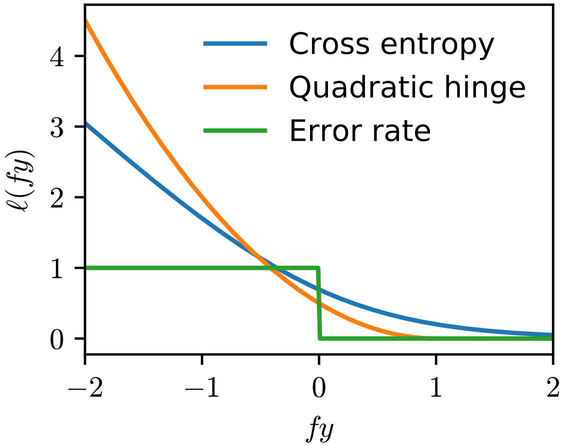


- random initial conditions
- minimize energy $\mathcal{L}$
- either find $\mathcal{L} = 0$ or $\mathcal{L} > 0$

Upon increasing density $\rightarrow$ transition

sharp transition with **finite-range** interactions

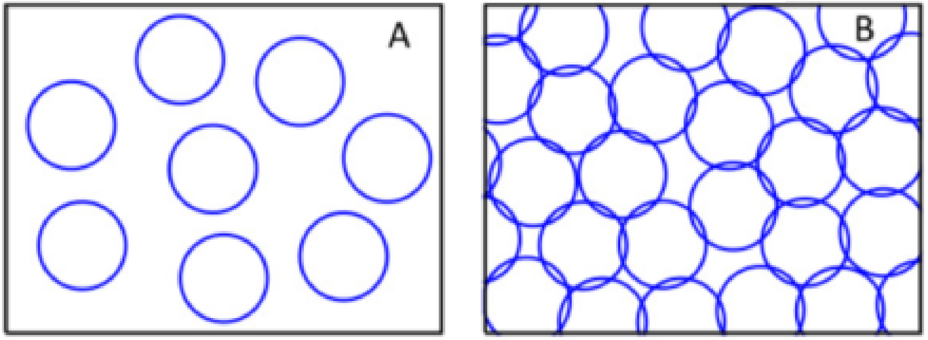
this is why we use the **hinge loss**!



Analogy with granular matter: Jamming

Random packing:

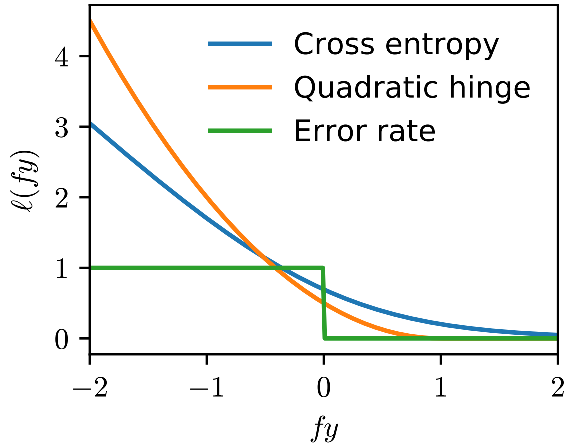
- random initial conditions
- minimize energy $\mathcal{L}$
- either find $\mathcal{L} = 0$ or $\mathcal{L} > 0$



Upon increasing density $\rightarrow$ transition

sharp transition with **finite-range** interactions

this is why we use the **hinge loss**!



Shallow networks $\longleftrightarrow$ packings of **spheres**: Franz and Parisi, '16

Deep nets $\longleftrightarrow$ packings of **ellipsoids**!

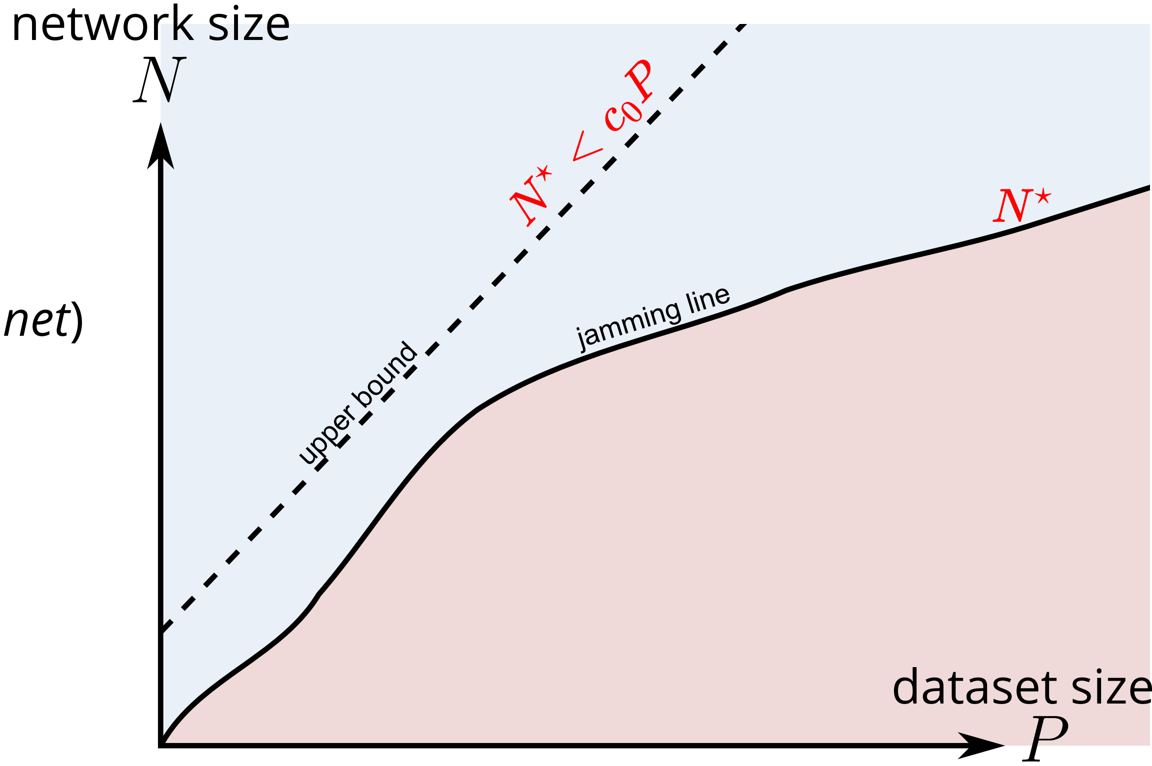
Theoretical results: Phase diagram

- When N is large, $\mathcal{L} = 0$
- Transition at $N^\star$

(*if signals propagate through the net*)

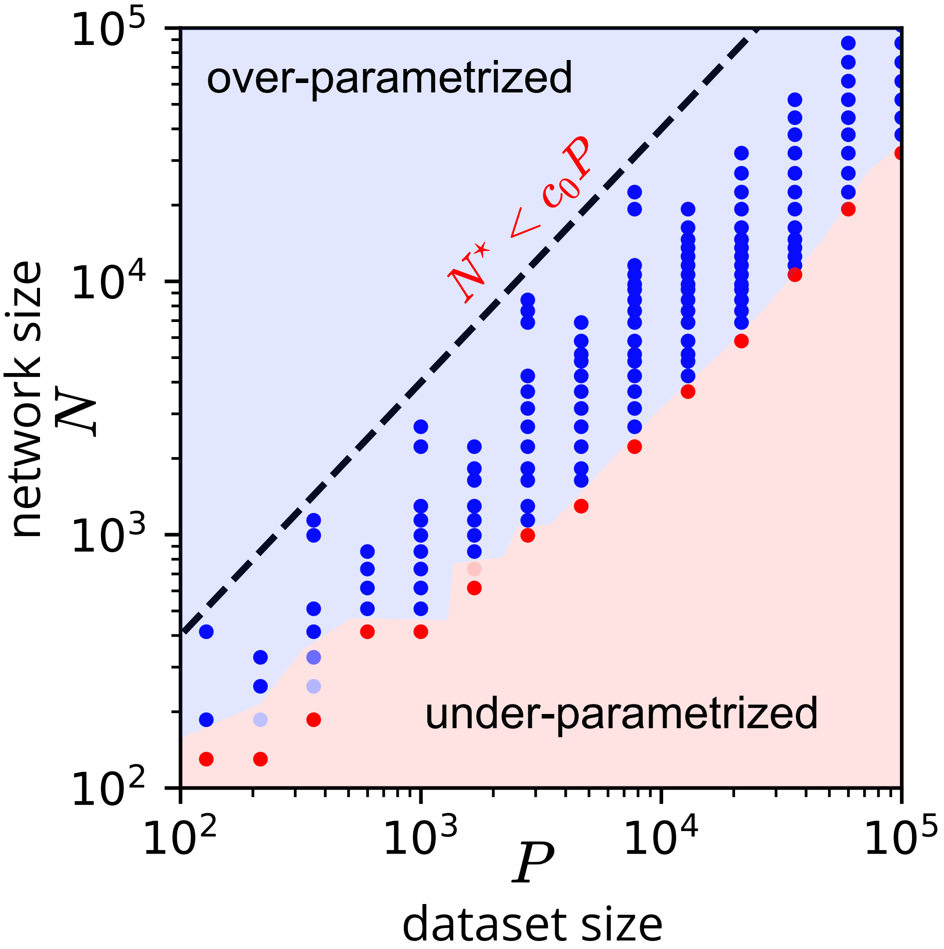
$N^\star < c_0 P$

typically $c_0 = \mathcal{O}(1)$



Empirical tests: MNIST (parity)

Geiger et al. '18 - arXiv:1809.09349;
Spigler et al. '18 - arXiv:1810.09665



No local minima are found when **overparametrized**!

- Above N^* we have $\mathcal{L} = 0$
- Solid line is the bound $N^* < c_0 P$

Landscape curvature

Geiger et al. '18 - arXiv:1809.09349

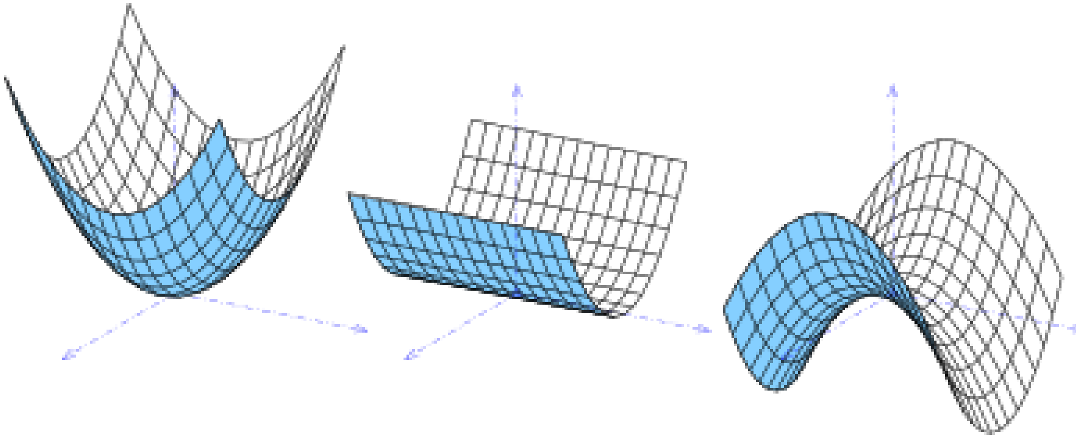
We don't find local minima when overparametrized...

...shape of the landscape?

w.r.t parameters W

Local curvature:

second order approximation



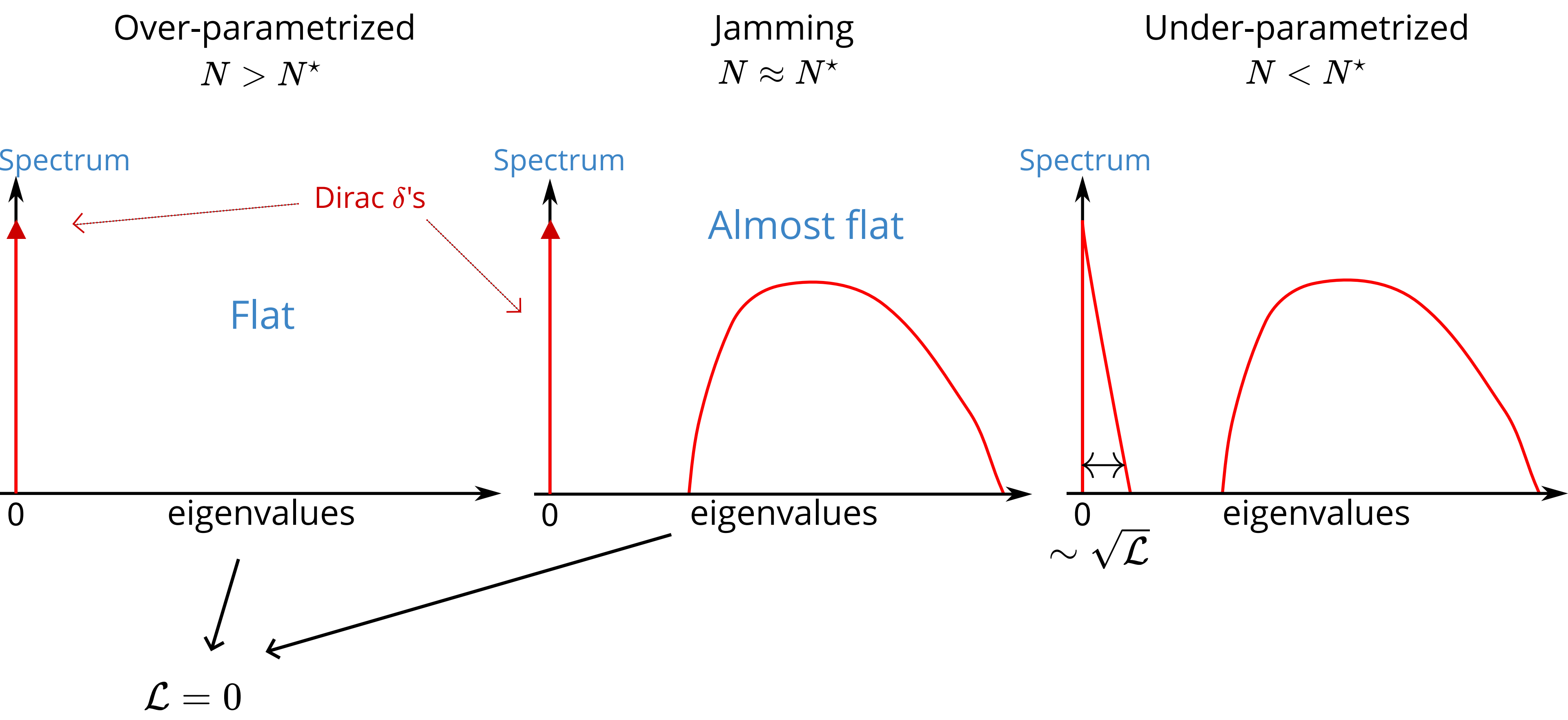
Information captured by **Hessian matrix**: $\mathcal{H}_{\mu\nu} = \frac{\partial^2}{\partial \mathbf{w}_\mu \partial \mathbf{w}_\nu} \mathcal{L}(\mathbf{W})$

↓

Spectrum of the Hessian (eigenvalues)

Flat directions

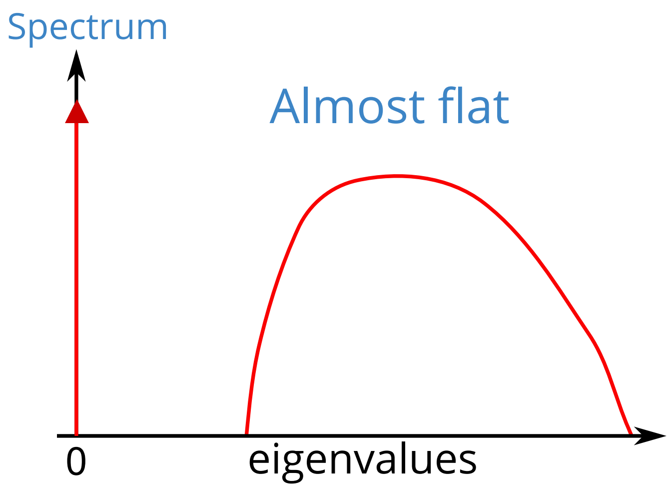
Geiger et al. '18 - arXiv:1809.09349



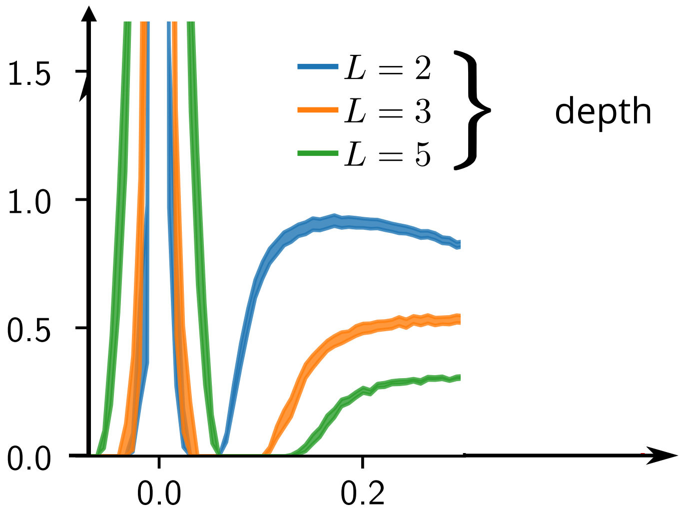
Flat directions

Geiger et al. '18 - arXiv:1809.09349

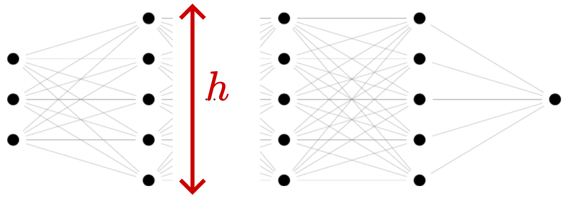
Jamming
$N \approx N^{\star}$



From numerical simulations:
(*at the transition*)



Outline



Vary **network size** N ($\sim h^2$):

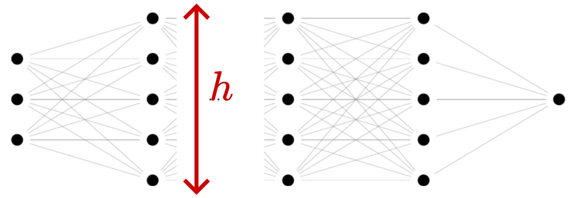
1. Can networks fit **all** the P training data?

Yes, deep networks **fit all data** if $N > N^*$ $\longrightarrow$ *jamming transition*

2. Can networks overfit? Can N be too large?

$\rightarrow$ Long term goal: how to choose N?

Outline



Vary **network size** N ($\sim h^2$):

1. Can networks fit **all** the P training data?

Yes, deep networks **fit all data** if $N > N^*$ $\longrightarrow$ *jamming transition*

2. Can networks overfit? Can N be too large?

$\rightarrow$ Long term goal: how to choose N?

Generalization

Spigler et al. '18 - arXiv:1810.09665

Ok, so just crank up N and fit everything?

Generalization? $\rightarrow$ Compute **test error** ϵ

But wait... what about **overfitting**?

Generalization

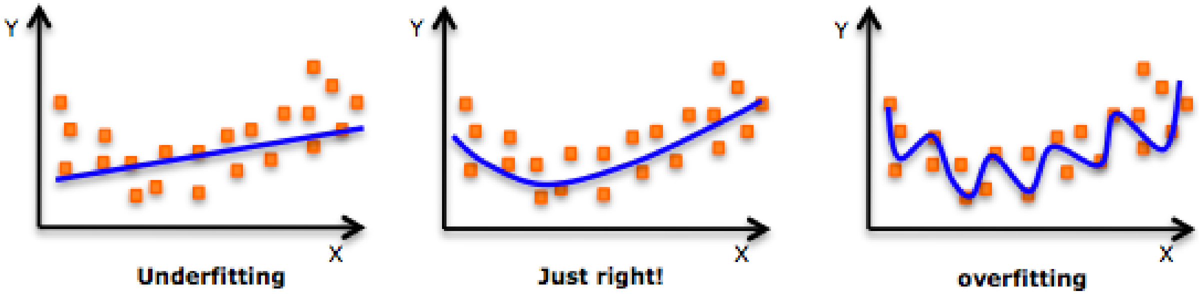
Spigler et al. '18 - arXiv:1810.09665

Ok, so just crank up N and fit everything?

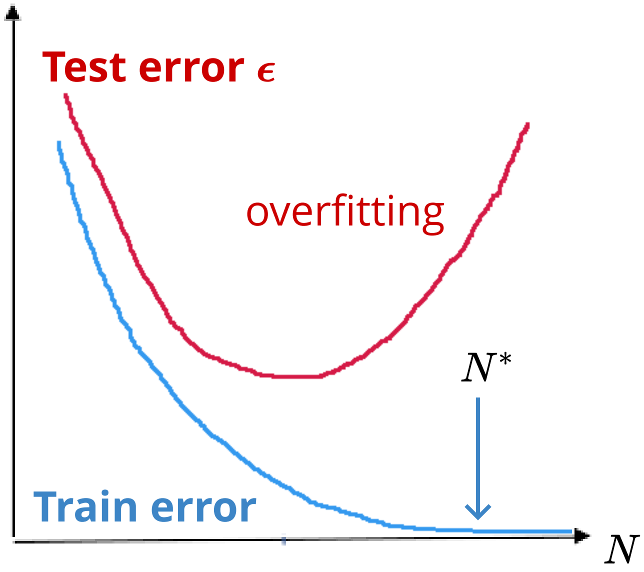
Generalization? $\rightarrow$ Compute **test error** ϵ

But wait... what about **overfitting**?

example: polynomial fitting

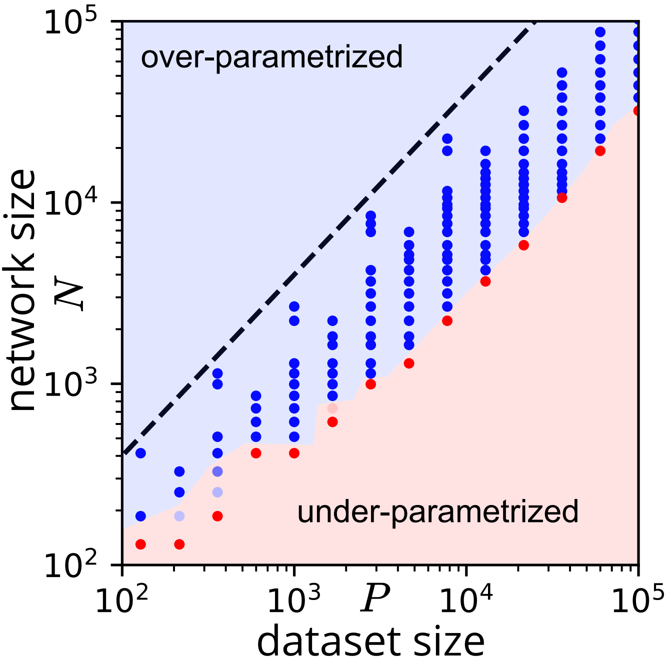


$N \sim$ polynomial degree



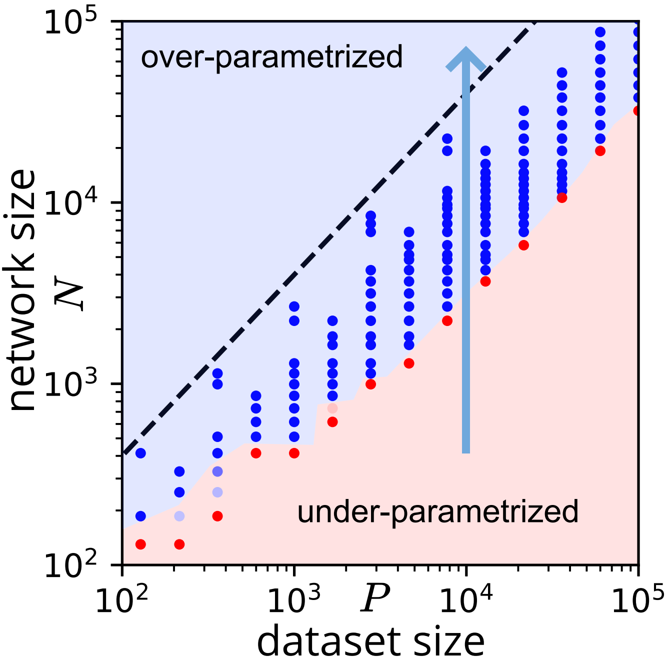
Overfitting?

Spigler et al. '18 - arXiv:1810.09665



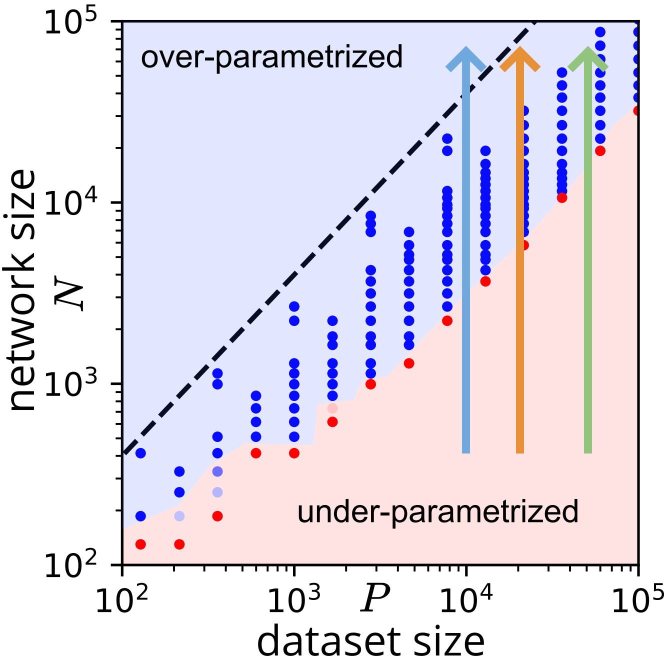
Overfitting?

Spigler et al. '18 - arXiv:1810.09665



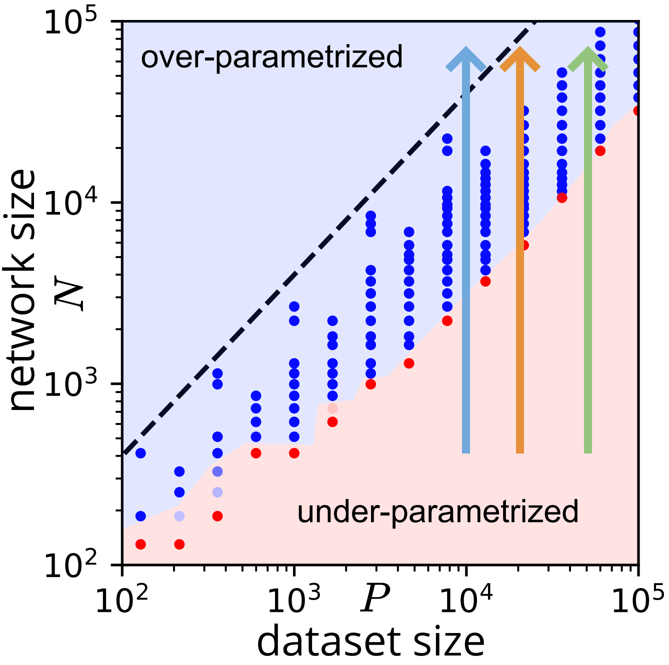
Overfitting?

Spigler et al. '18 - arXiv:1810.09665

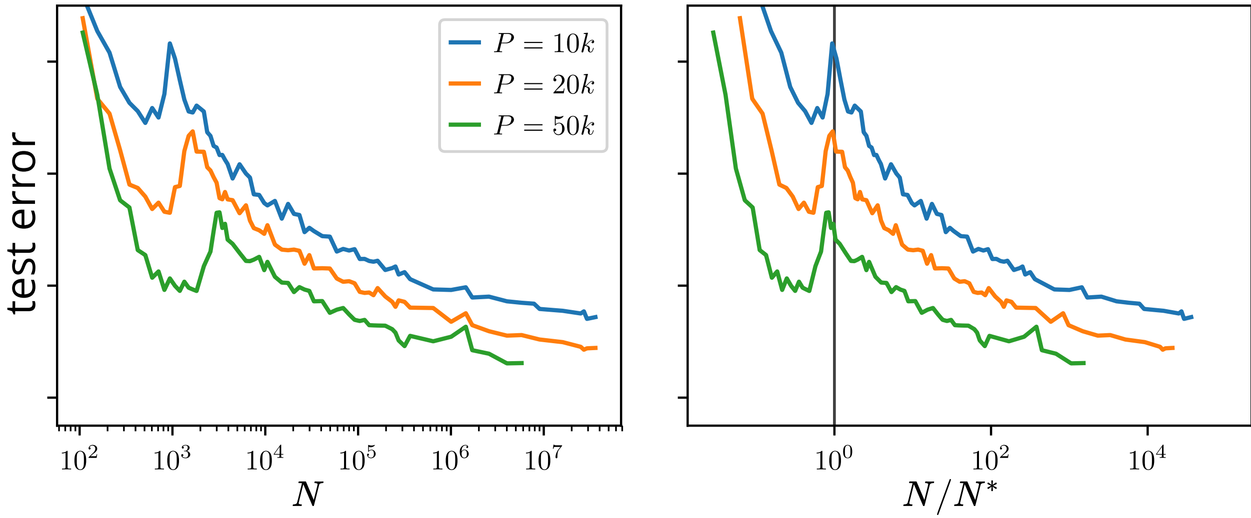


Overfitting?

Spigler et al. '18 - arXiv:1810.09665



"Double descent"



- **Test error decreases monotonically** with N!

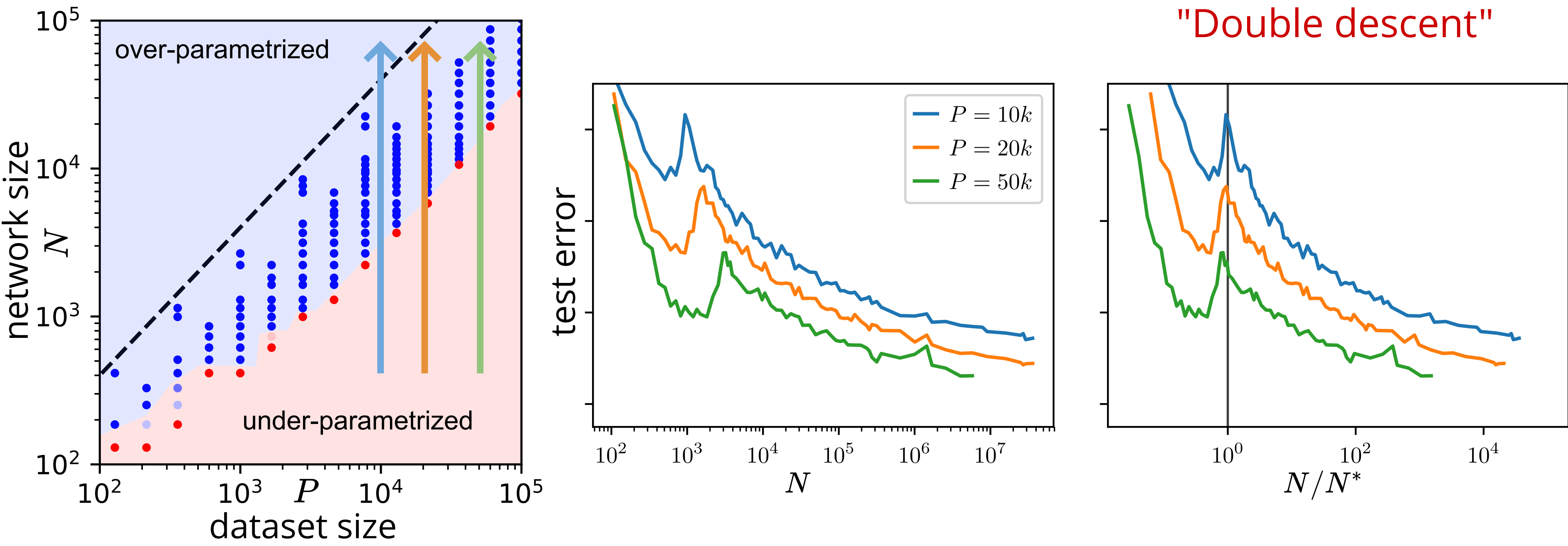
(after the peak)

- **Cusp** at the jamming transition

Advani and Saxe '17;
Spigler et al. '18 - arXiv:1810.09665;
Geiger et al. '19 - arXiv:1901.01608

Overfitting?

Spigler et al. '18 - arXiv:1810.09665



- **Test error decreases monotonically** with N!

We know why: Fluctuations!

(after the peak)

- **Cusp** at the jamming transition

Advani and Saxe '17;
Spigler et al. '18 - arXiv:1810.09665;
Geiger et al. '19 - arXiv:1901.01608

Ensemble average

- Random initialization $\rightarrow$ output function f_N is **stochastic**

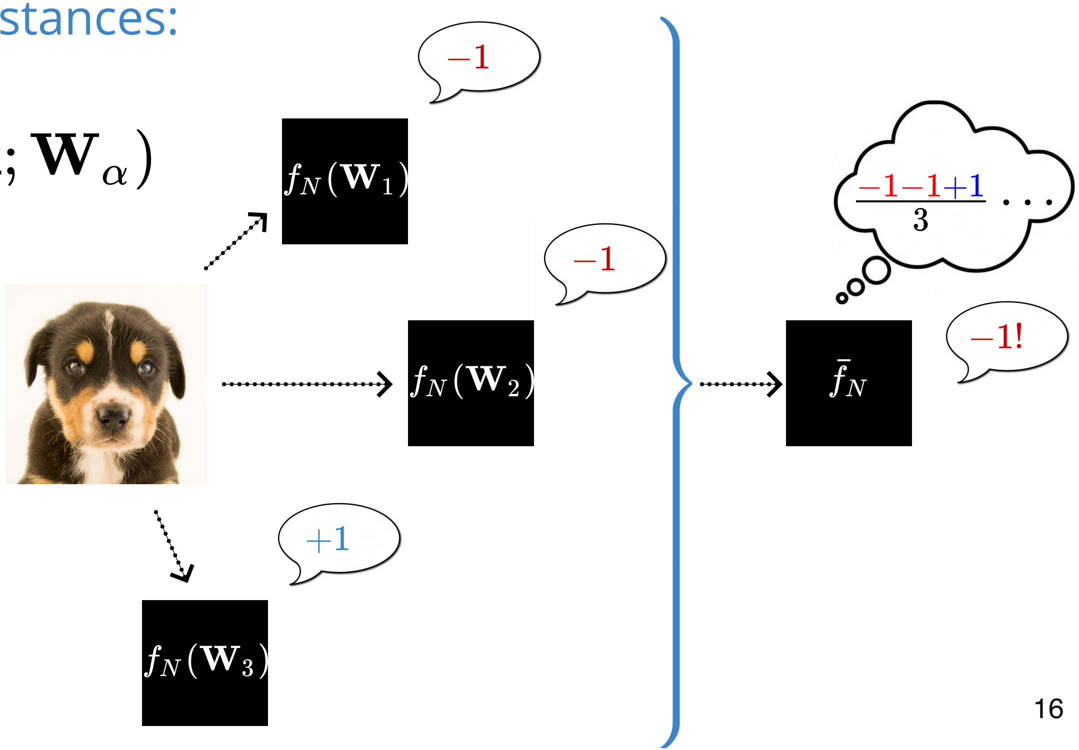
- Fluctuations: quantified by $\boxed{\textbf{average}}$ and **variance**

Ensemble average

- Random initialization $\rightarrow$ output function f_N is **stochastic**
- Fluctuations: quantified by **average** and **variance**

ensemble average over n instances:

$$\bar{f}_N^n(\mathbf{x}) \equiv \frac{1}{n}\sum_{\alpha=1}^{n} f_N(\mathbf{x};\mathbf{W}_\alpha)$$



Ensemble average

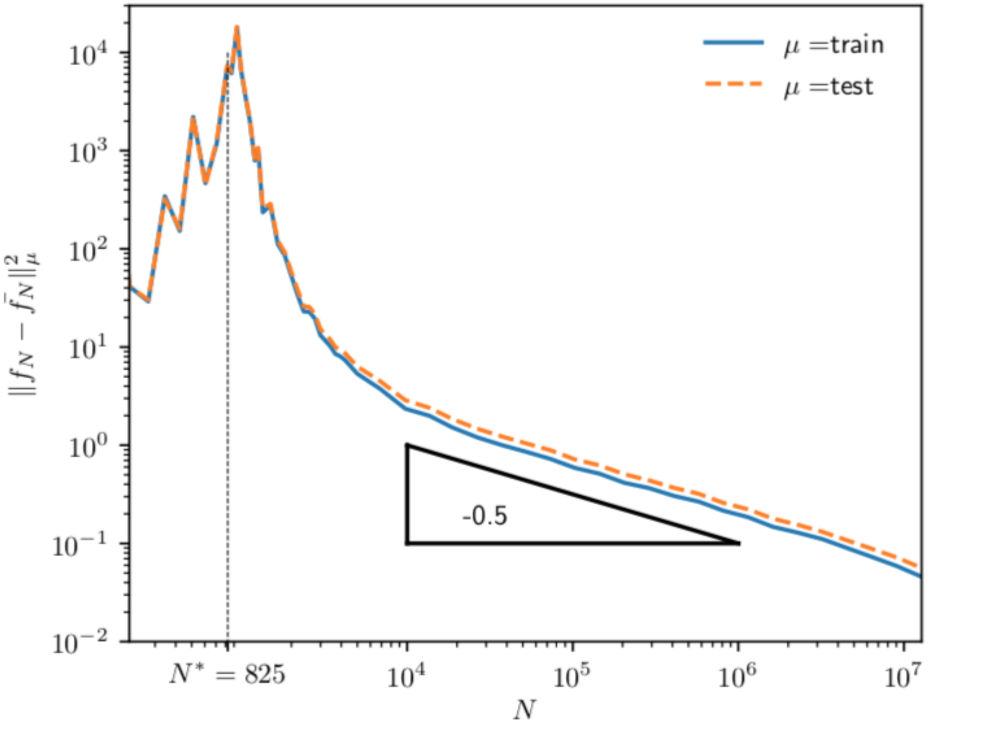
- Random initialization $\rightarrow$ output function f_N is **stochastic**

- Fluctuations: quantified by **average** and $\boxed{\textbf{variance}}$

Define some norm over the output functions:

ensemble variance (fixed n):

$$\boxed{\|f_N - \bar{f}_N^n\|^2 \sim N^{-\frac{1}{2}}}$$



Ensemble average

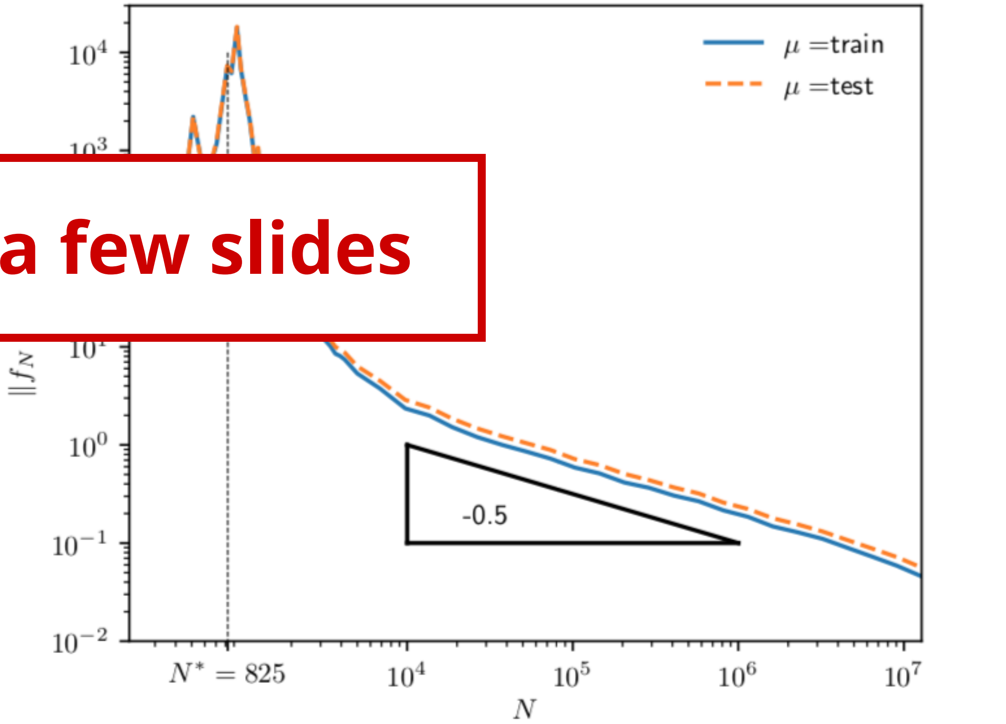
- Random initialization $\rightarrow$ output function f_N is **stochastic**
- Fluctuations: quantified by **average** and $\boxed{\textbf{variance}}$

Define some norm over the output functions:

ensemble va

Explained in a few slides

$$\boxed{\|f_N - \bar{f}_N^n\|^2 \sim N^{-\frac{1}{2}}}$$



Fluctuations increase error

Geiger et al. '19 - arXiv:1901.01608

Remark: *test error of ensemble average* $\neq$ *average test error*

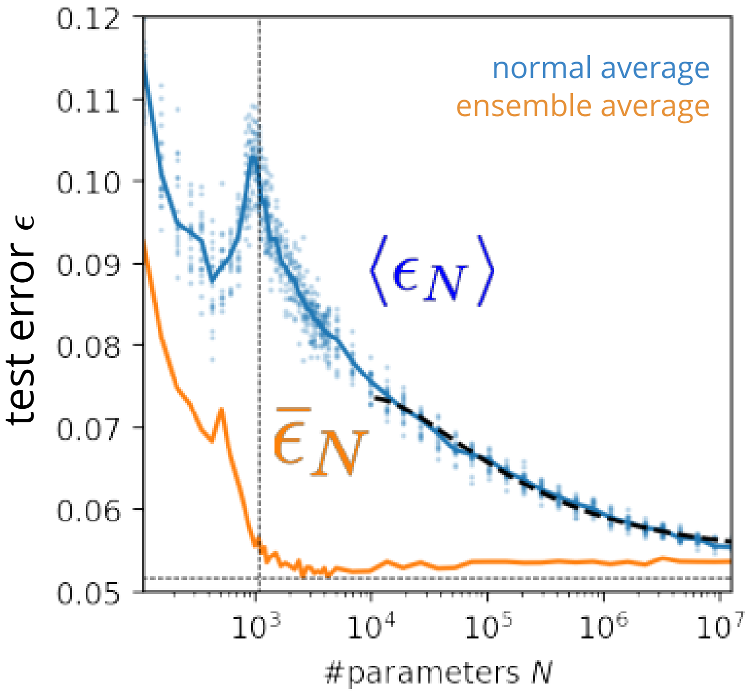
$$\bar{f}_N^n(\mathbf{x}) \to \bar{\epsilon}_N \qquad \{f(\mathbf{x};\mathbf{W}_\alpha)\} \to \langle \epsilon_N \rangle$$

Fluctuations increase error

Geiger et al. '19 - arXiv:1901.01608

Remark: *test error of ensemble average* $\neq$ *average test error*

$\bar{f}^n_N(\mathbf{x}) \to \bar{\epsilon}_N$ $\{f(\mathbf{x}; \mathbf{W}_\alpha)\} \to \langle \epsilon_N \rangle$



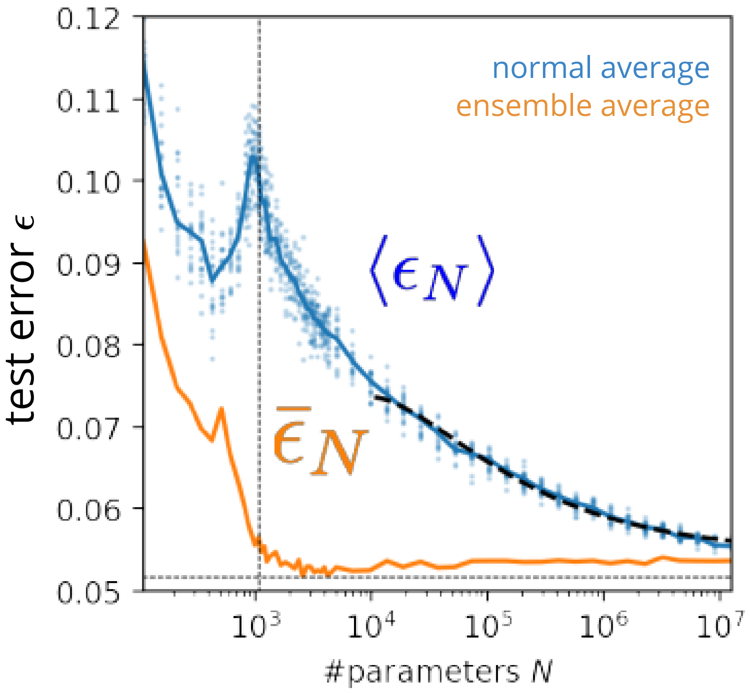
- **Test error** increases with fluctuations
- **Ensemble test error** is nearly flat after N^*!

Fluctuations increase error

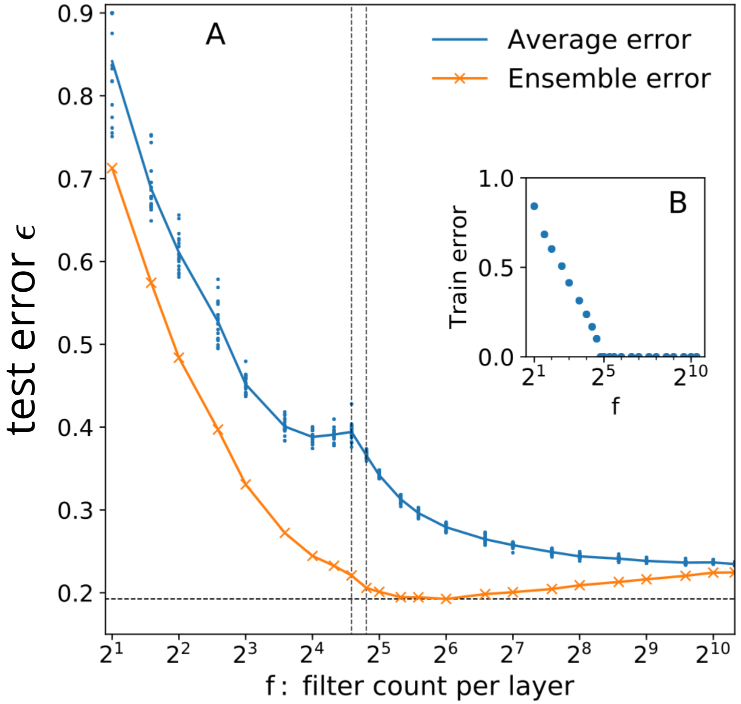
Geiger et al. '19 - arXiv:1901.01608

Remark: *test error of ensemble average* $\neq$ *average test error*

$$\bar{f}_N^n(\mathbf{x}) \to \bar{\epsilon}_N \qquad \{f(\mathbf{x};\mathbf{W}_\alpha)\} \to \langle \epsilon_N \rangle$$



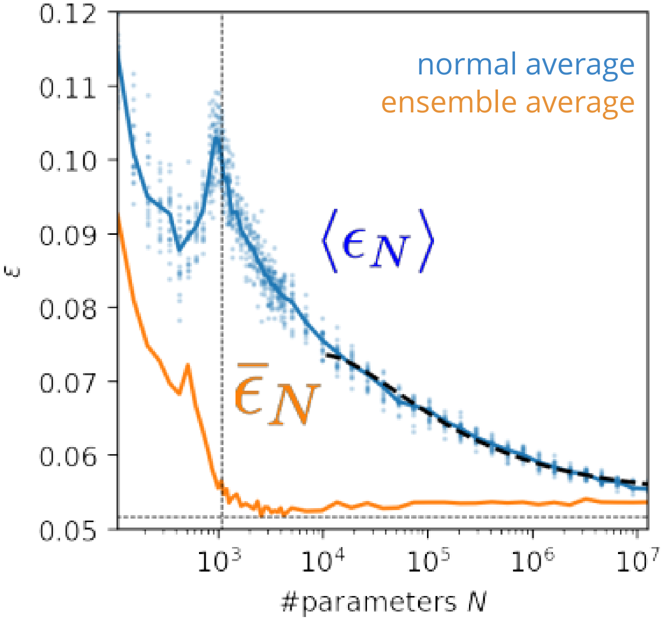
(MNIST parity)



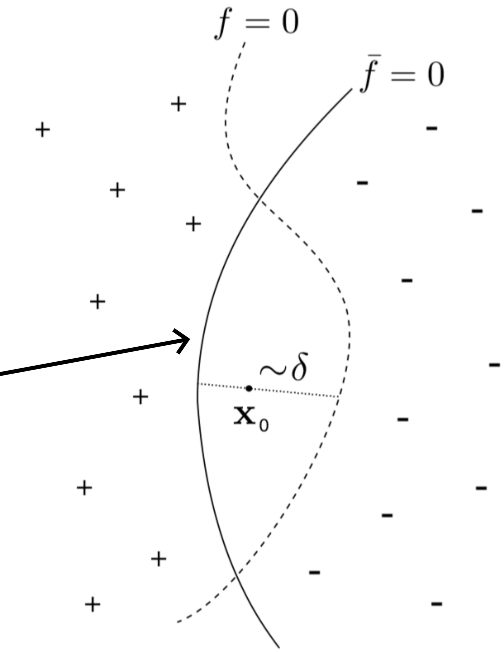
(CIFAR-10 $\to$ regrouped in 2 classes)

Scaling argument!

Geiger et al. '19 - arXiv:1901.01608

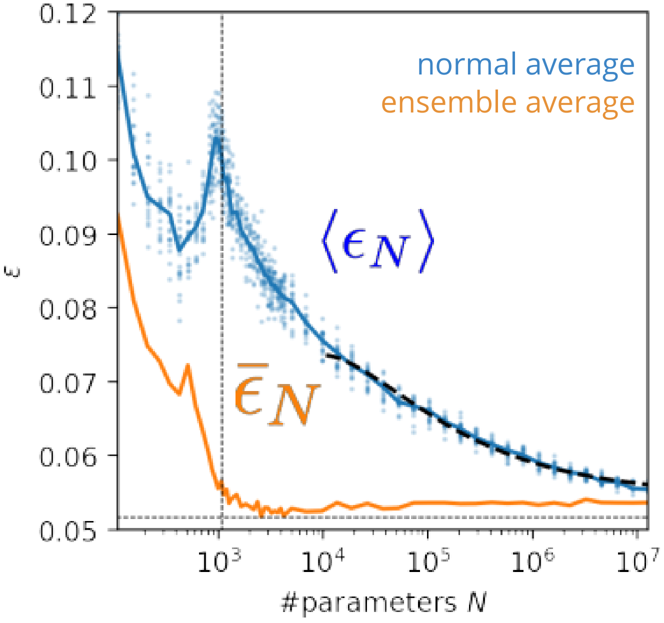


decision boundaries:

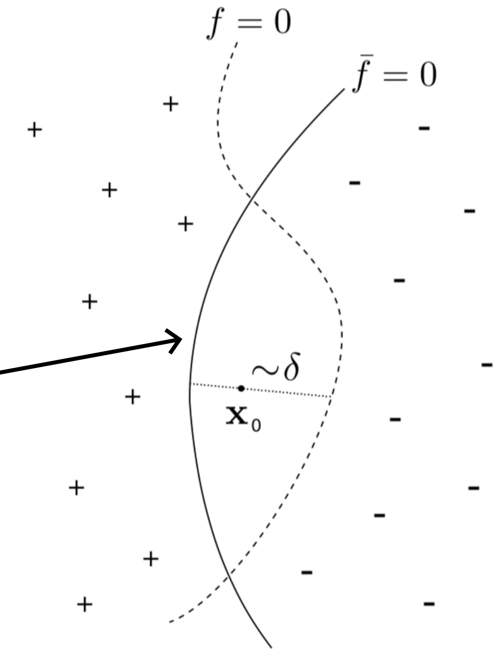


Scaling argument!

Geiger et al. '19 - arXiv:1901.01608



decision boundaries:

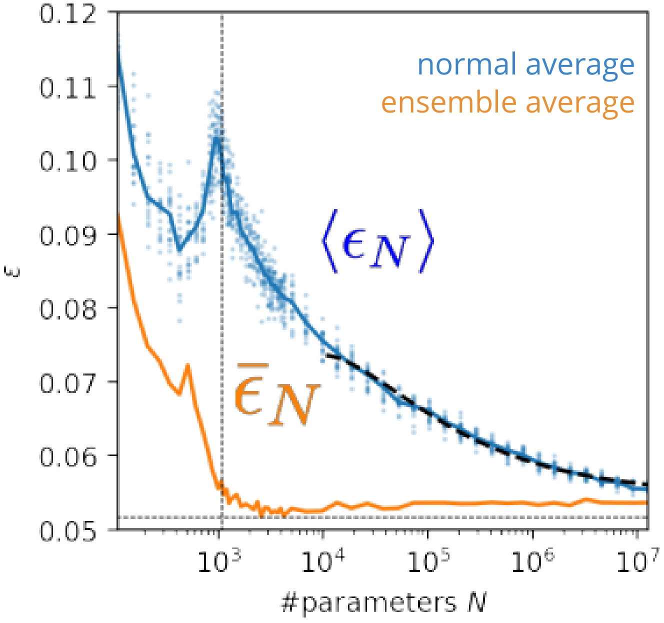


Smoothness of test error as function of
decision boundary + symmetry:

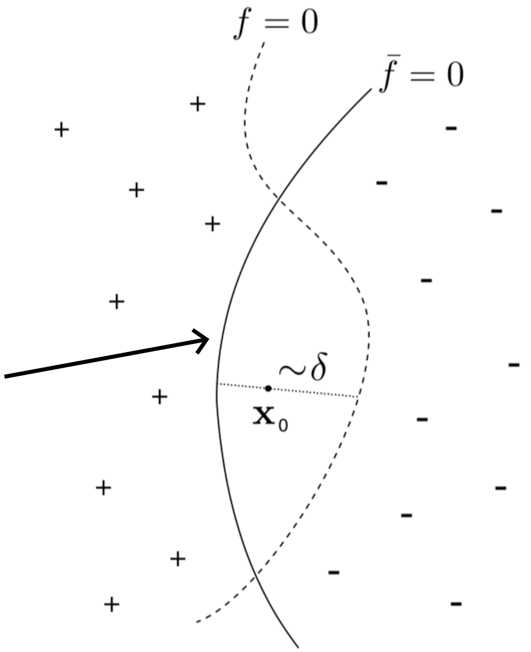
$$\boxed{\langle \epsilon_N \rangle - \bar{\epsilon}_N \sim \| f_N - \bar{f}_N \|^2 \sim N^{-\frac{1}{2}}}$$

Scaling argument!

Geiger et al. '19 - arXiv:1901.01608

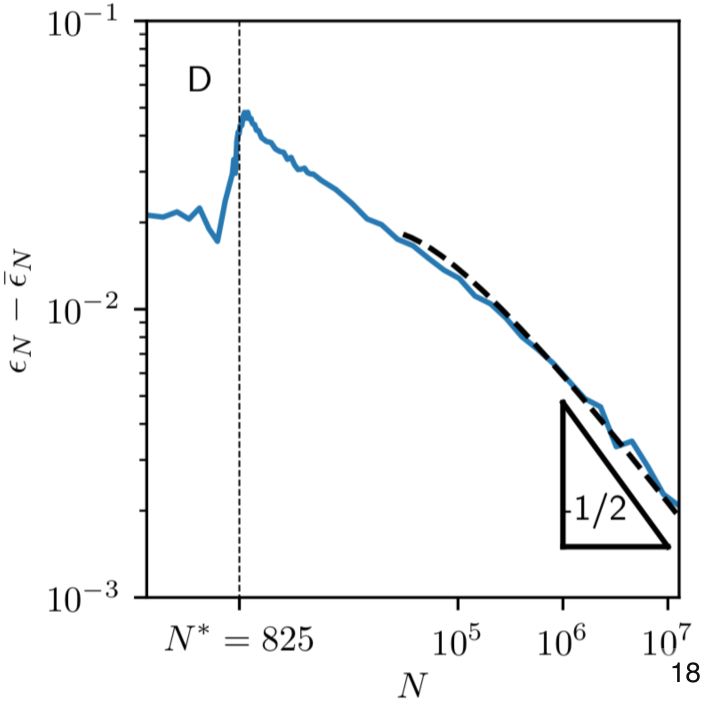


decision boundaries:



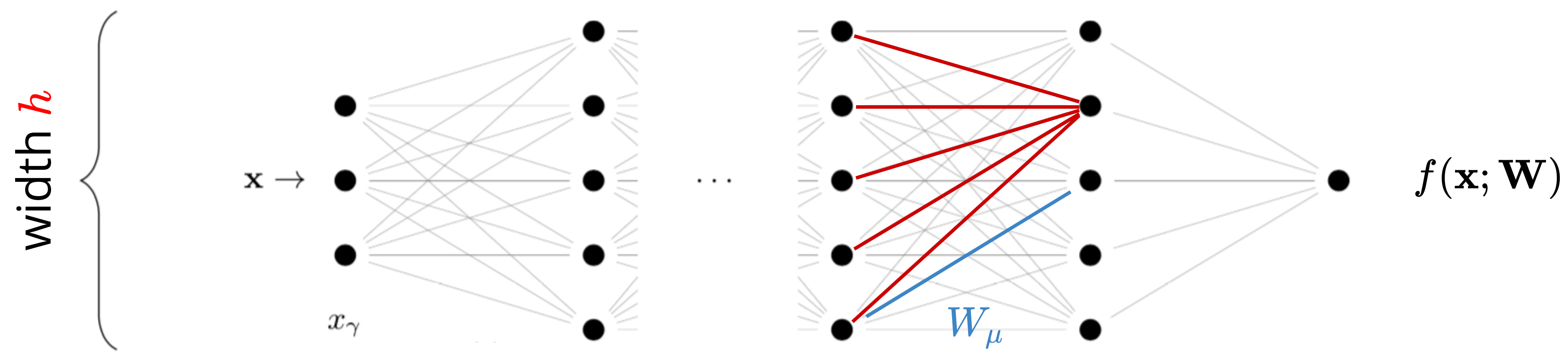
Smoothness of test error as function of decision boundary + symmetry:

$$\boxed{\langle \epsilon_N \rangle - \bar{\epsilon}_N \sim \| f_N - \bar{f}_N \|^2 \sim N^{-\frac{1}{2}}}$$



Infinitely-wide networks: Initialization

Neal '96; Williams '98; Lee et al '18; Schoenholz et al. '16



- Weights: each initialized as $W_\mu \sim h^{-\frac{1}{2}} \mathcal{N}(0,1)$

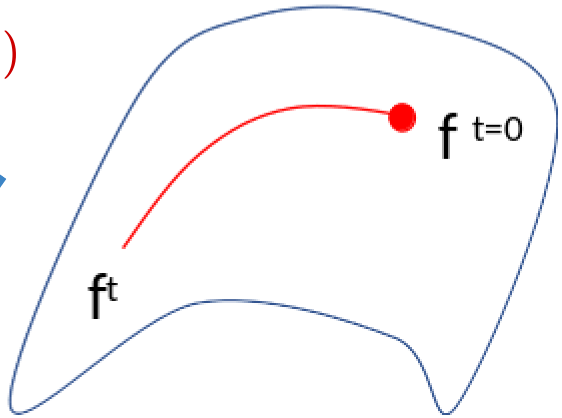
- Neurons sum h signals of order $h^{-\frac{1}{2}}$ $\longrightarrow$ **Central Limit Theorem**

- Output function becomes a **Gaussian Random Field** as $h \rightarrow \infty$

Infinitely-wide networks: Learning

Jacot et al. '18

For an input $\mathbf{x}$ the function $f(\mathbf{x};\mathbf{W})$ lives on a curved manifold

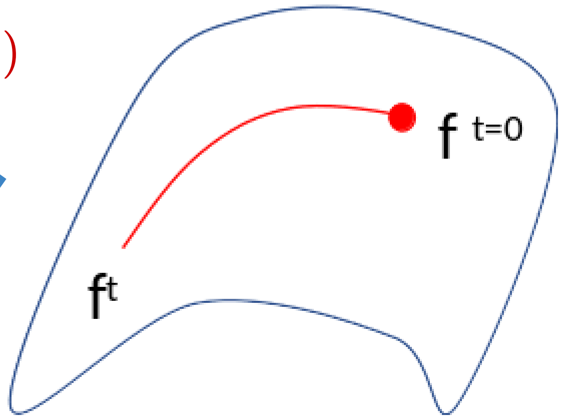


- For **small** width h: $\nabla_{\mathbf{W}} f$ evolves during training

Infinitely-wide networks: Learning

Jacot et al. '18

For an input $\mathbf{x}$ the function $f(\mathbf{x};\mathbf{W})$ lives on a curved manifold

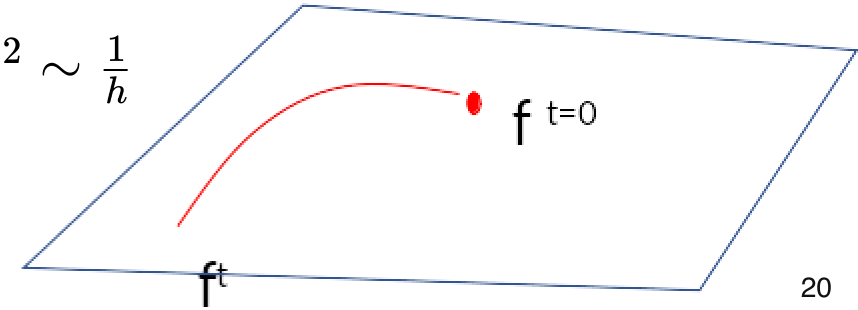


- For **small** width h: $\nabla_{\mathbf{W}} f$ evolves during training
- For **large** width h: $\nabla_{\mathbf{W}} f$ is constant during training

The manifold becomes linear!

Lazy learning:

- weights don't change much: $\|\mathbf{W}^t - \mathbf{W}^{t=0}\|^2 \sim \frac{1}{h}$
- enough to change the output f by $\sim \mathcal{O}(1)$!



Neural Tangent Kernel

- Gradient descent implies:

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\mathbf{x}; \mathbf{W}^t) = \sum_{i=1}^{P} \Theta^t(\mathbf{x}, \mathbf{x}_i) \;\; y_i \ell'(y_i f(\mathbf{x}_i; \mathbf{W}^t))$$

Neural Tangent Kernel

- Gradient descent implies:

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\mathbf{x}; \mathbf{W}^t) = \overbrace{\sum_{i=1}^{P} \boxed{\Theta^t(\mathbf{x}, \mathbf{x}_i)} \; y_i \ell'(y_i f(\mathbf{x}_i; \mathbf{W}^t))}^{\text{convolution with a kernel}}$$

$$\Theta^t(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}^t) \cdot \nabla_{\mathbf{W}} f(\mathbf{x}'; \mathbf{W}^t)$$

The formula for the *kernel* Θ^t is useless, unless…

Neural Tangent Kernel

- Gradient descent implies:

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\mathbf{x}; \mathbf{W}^t) = \overbrace{\sum_{i=1}^{P} \boxed{\Theta^t(\mathbf{x}, \mathbf{x}_i)} \; y_i \ell'(y_i f(\mathbf{x}_i; \mathbf{W}^t))}^{\text{convolution with a kernel}}$$

$$\Theta^t(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}^t) \cdot \nabla_{\mathbf{W}} f(\mathbf{x}'; \mathbf{W}^t)$$

The formula for the *kernel* Θ^t is useless, unless...

<u>*Theorem*</u>. (informal) $\lim_{\text{width } h \to \infty} \Theta^t(\mathbf{x}, \mathbf{x}') \equiv \Theta_\infty(\mathbf{x}, \mathbf{x}')$

Jacot et al. '18

Deep learning $=$ learning with a **kernel** as $h \to \infty$

Finite N asymptotics?

Geiger et al. '19 - arXiv:1901.01608;
Hanin and Nica '19;
Dyer and Gur-Ari '19

- **Evolution in time** is **small**: $\|\Theta^t - \Theta^{t=0}\|_F \sim 1/h \sim N^{-\frac{1}{2}}$

- **Fluctuations** are much **larger**: $\Delta\Theta^{t=0} \sim 1/\sqrt{h} \sim N^{-\frac{1}{4}}$
  at $t = 0$

$$f(\mathbf{x}; \mathbf{W}^t) = \int \mathrm{d}t \sum_{i=1}^{P} \Theta^t(\mathbf{x}, \mathbf{x}_i) \; y_i \ell'(y_i f(\mathbf{x}_i; \mathbf{W}^t))$$

Finite N asymptotics?

Geiger et al. '19 - arXiv:1901.01608;
Hanin and Nica '19;
Dyer and Gur-Ari '19

- **Evolution in time** is **small**: $\|\Theta^t - \Theta^{t=0}\|_F \sim 1/h \sim N^{-\frac{1}{2}}$

- **Fluctuations** are much **larger**: $\Delta\Theta^{t=0} \sim 1/\sqrt{h} \sim N^{-\frac{1}{4}}$
  at $t = 0$

$$f(\mathbf{x}; \mathbf{W}^t) = \int \mathrm{d}t \sum_{i=1}^{P} \Theta^t(\mathbf{x}, \mathbf{x}_i) \; y_i \ell'(y_i f(\mathbf{x}_i; \mathbf{W}^t))$$

Then: $$\boxed{\|f_N - \bar{f}_N\|^2 \sim \left(\Delta\Theta^{t=0}\right)^2 \sim N^{-\frac{1}{2}}}$$

The output function fluctuates similarly to the kernel

Conclusion

1. Can networks fit **all** the P training data?

- **Yes**, deep networks **fit all data** if $N > N^*$ $\longrightarrow$ *jamming transition*

2. Can networks overfit? Can N be too large?

- *Initialization* induces *fluctuations* in output that increase *test error*
- **No overfitting:** error keeps decreasing past N^* because *fluctuations diminish*

check Geiger et al. '19 - arXiv:1906.08034 for more!

$\rightarrow$ Long term goal: how to choose N?

(*tentative*) **Right after jamming**, and do **ensemble averaging**!

3. How does the test error scale with P?

check Spigler et al. '19 - arXiv:1905.10843 !