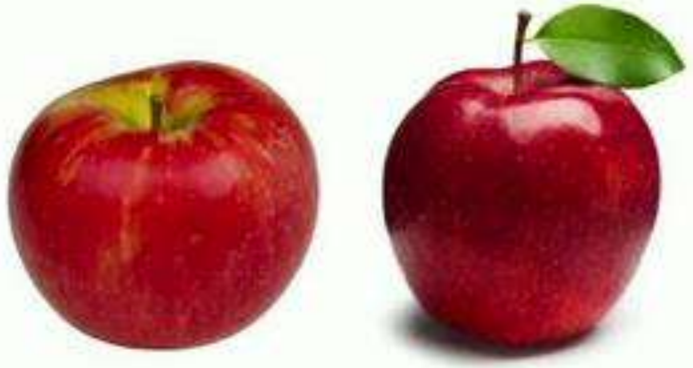
Universality and limitations of deep learning

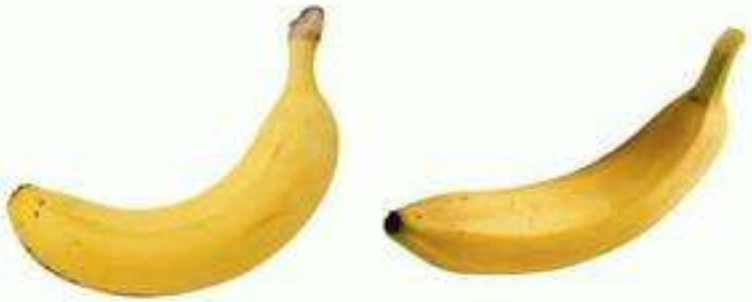
Emmanuel Abbe (EPFL/Princeton) and Colin Sandon (MIT)

MSR, 08.19

Class 1



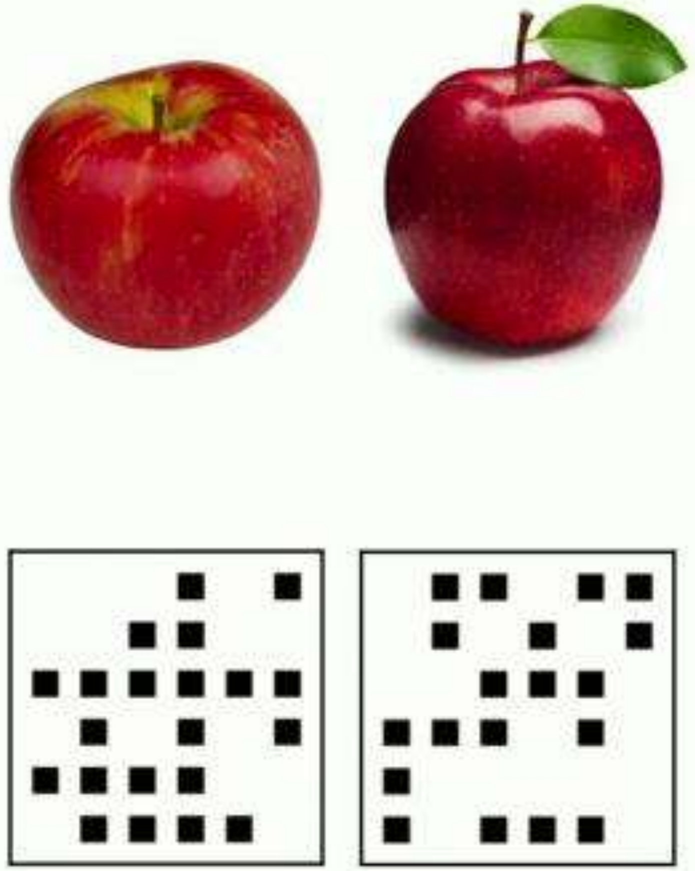
Class 2



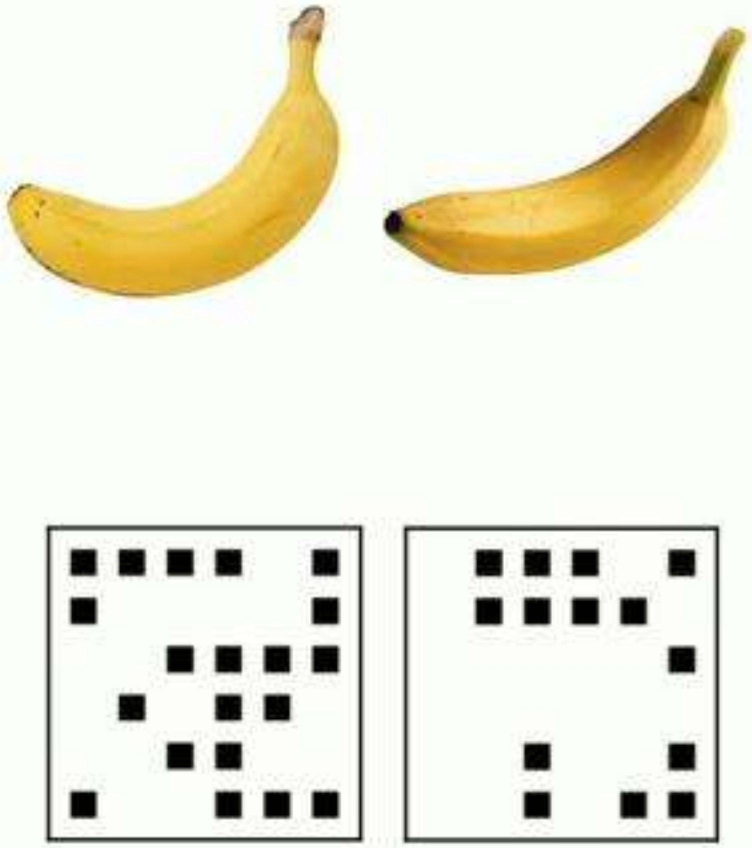
Class 1

Class 2

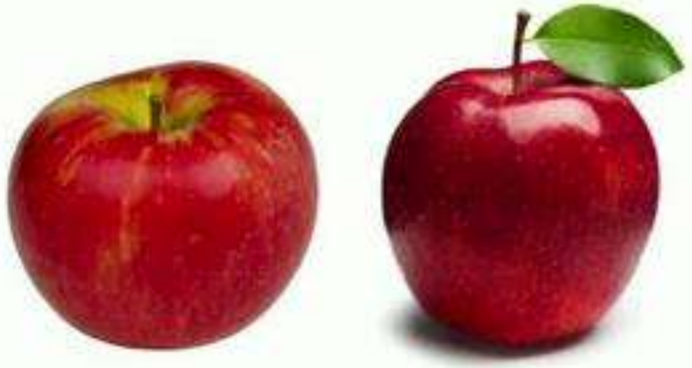
Class 1



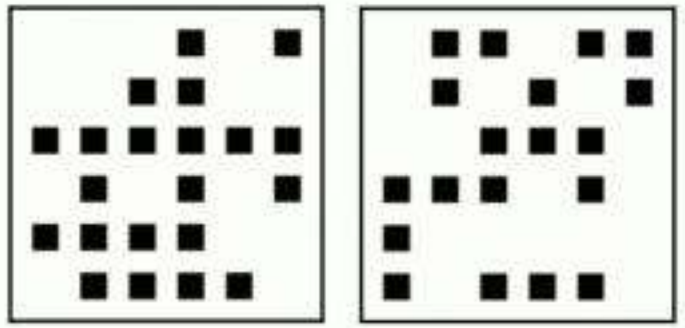
Class 2



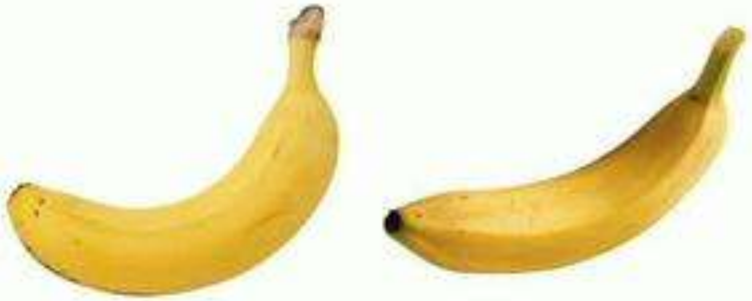
Class 1

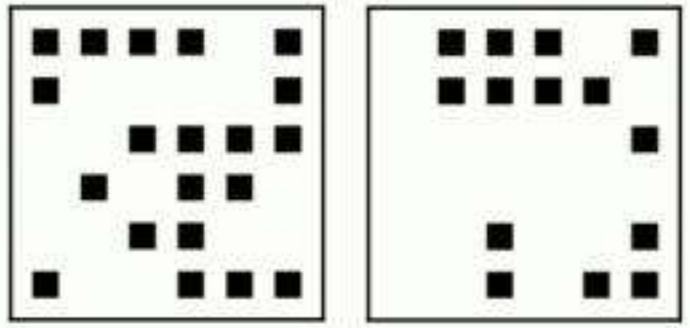


odd



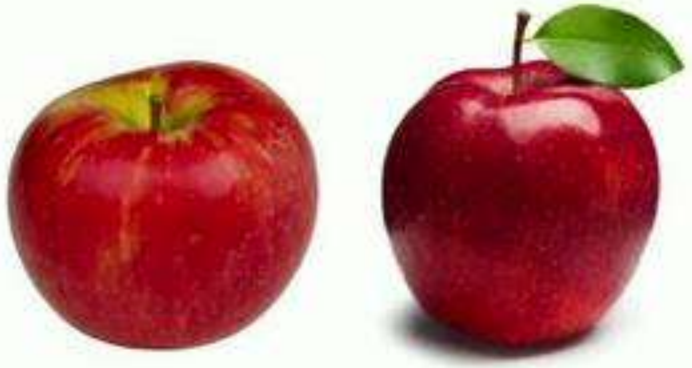
Class 2



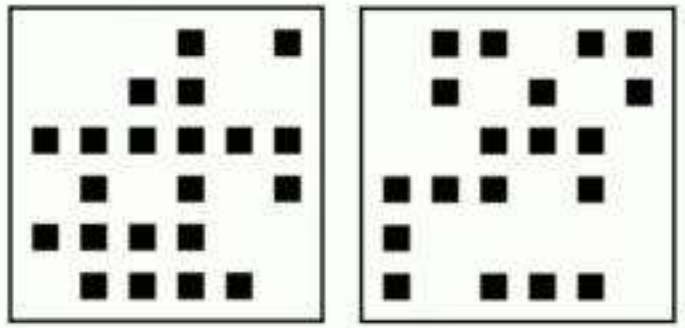


even

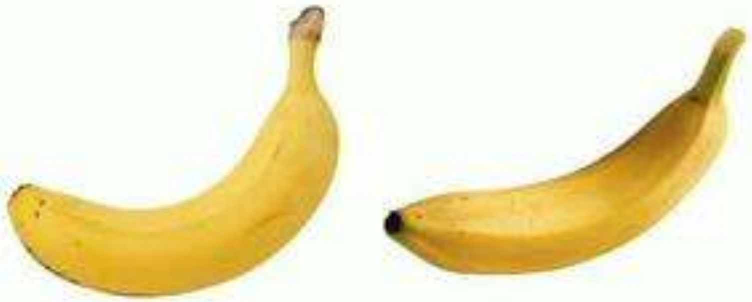
Class 1

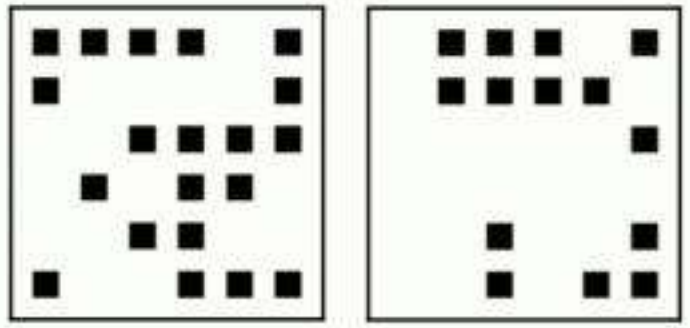


odd



Class 2

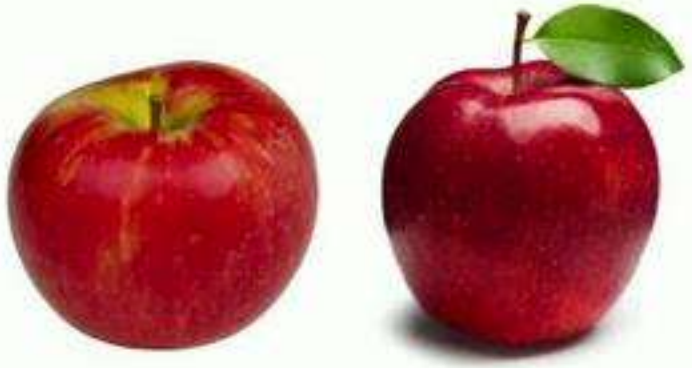




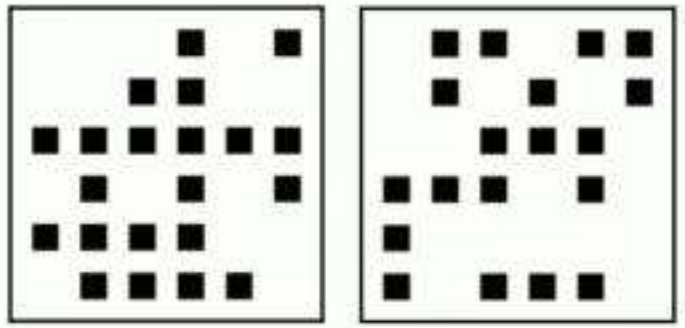
even

M. Minsky and S. Papert, Perceptrons: an introduction to computational geometry, MIT Press, 1969.

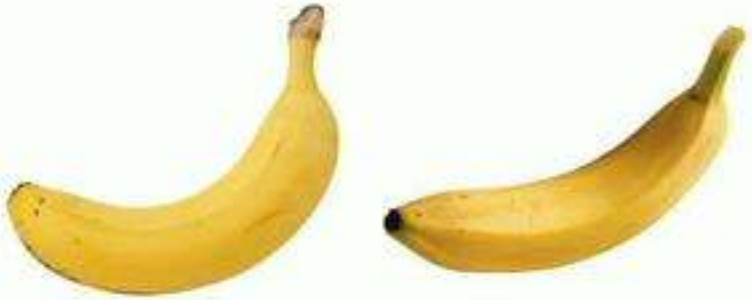
Class 1

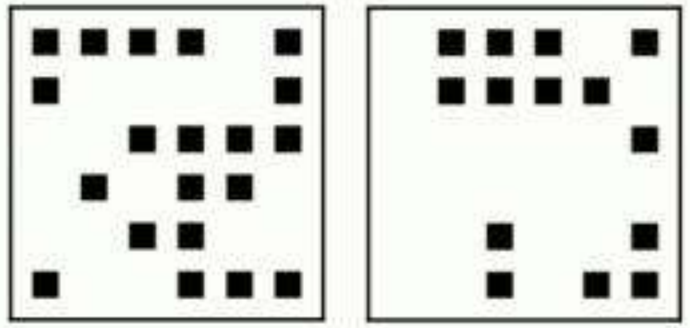


odd



Class 2

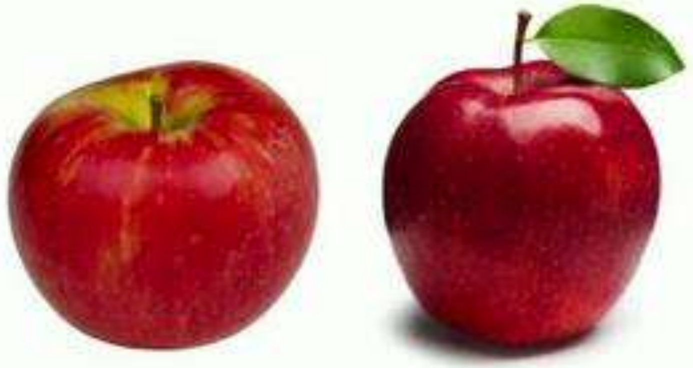


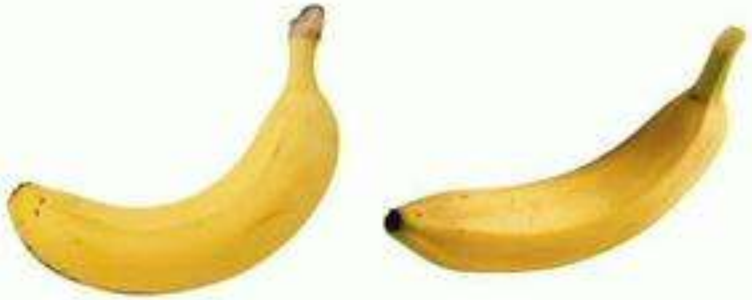


even

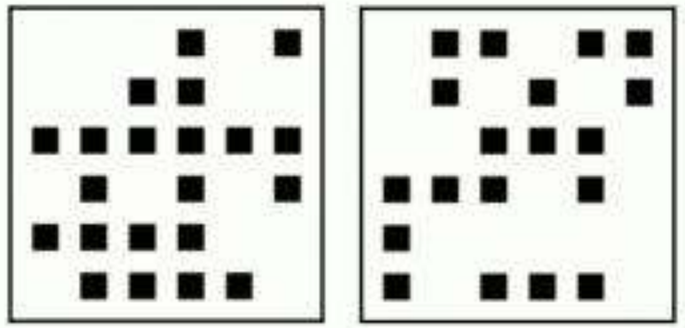
M. Minsky and S. Papert, Perceptrons: an introduction to computational geometry, MIT Press, 1969.

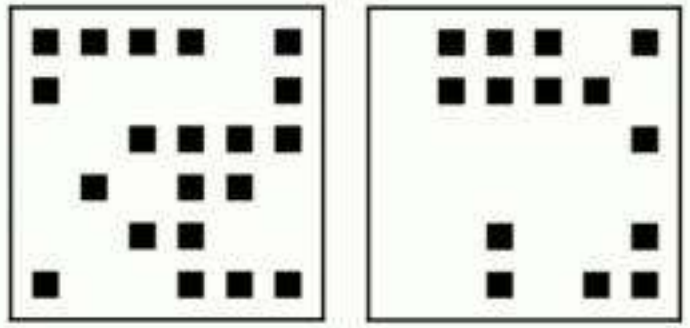
Class 1	Class 2





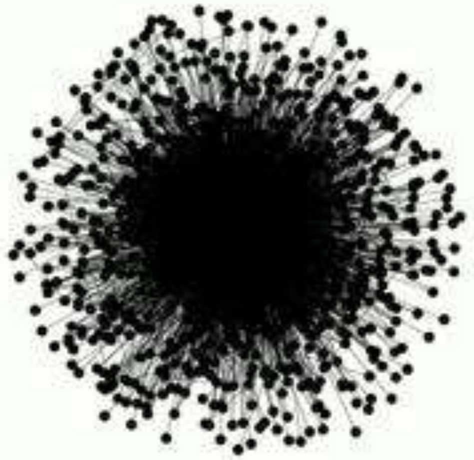
odd

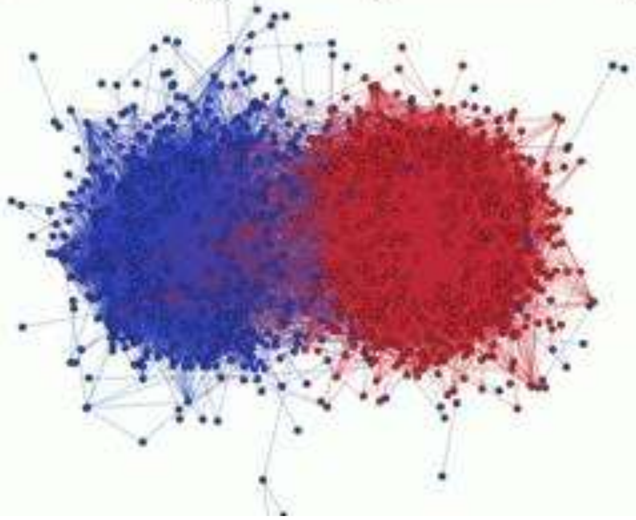




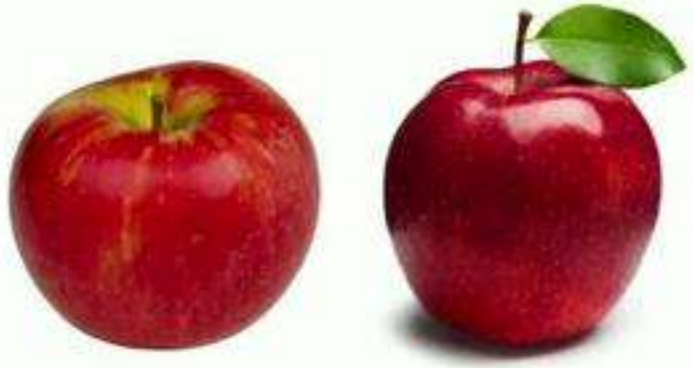
even

M. Minsky and S. Papert, Perceptrons: an introduction to computational geometry, MIT Press, 1969.

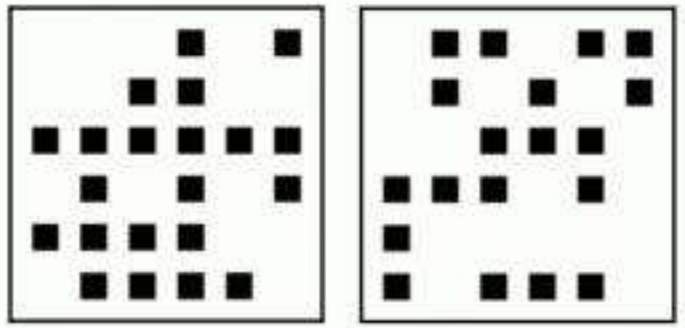




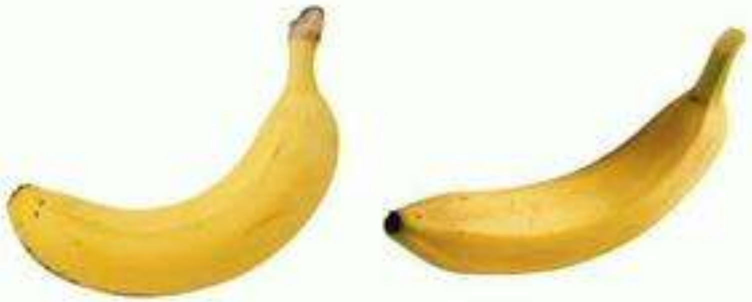
Class 1

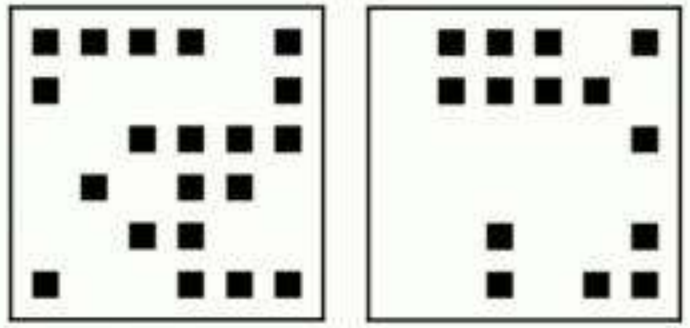


odd



Class 2



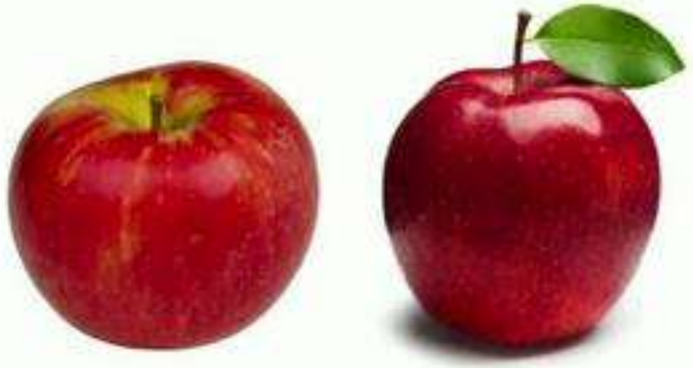


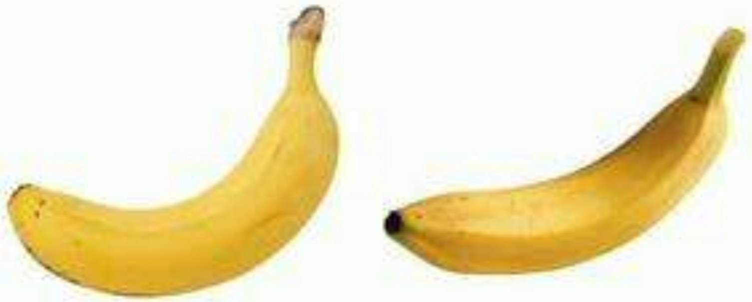
even

M. Minsky and S. Papert, Perceptrons: an introduction to computational geometry, MIT Press, 1969.

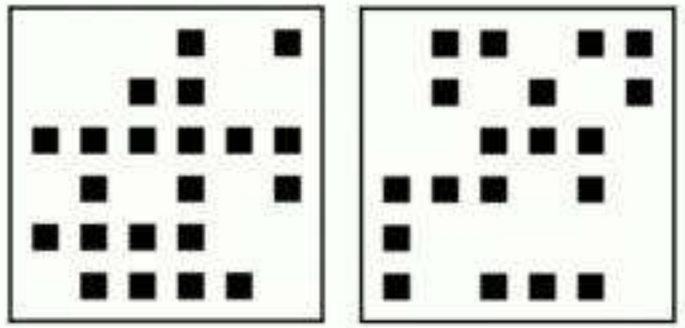
Class 1

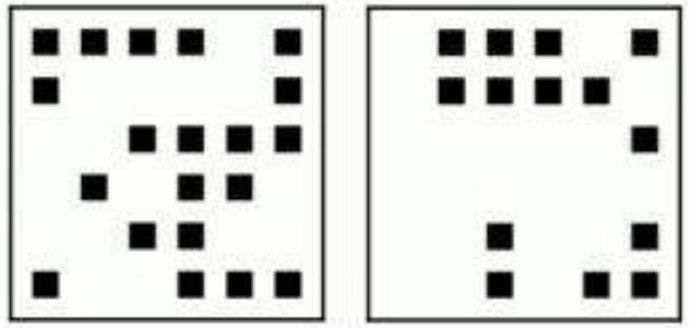
Class 2





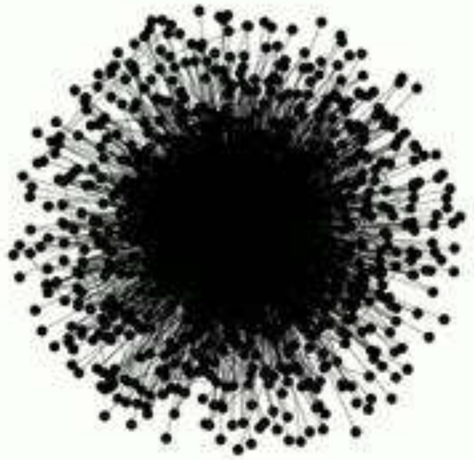
odd

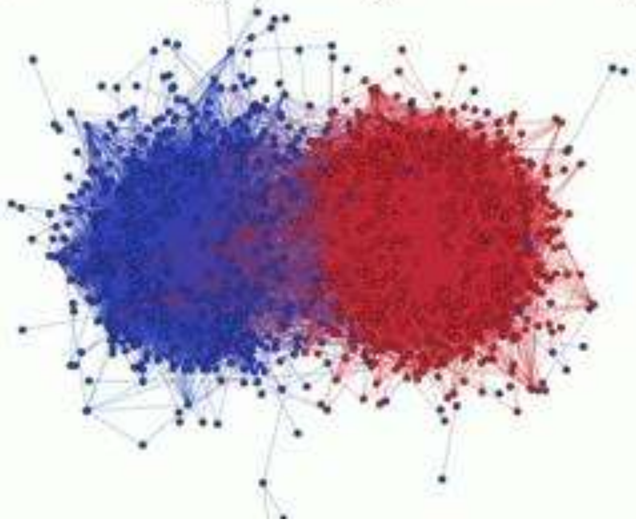




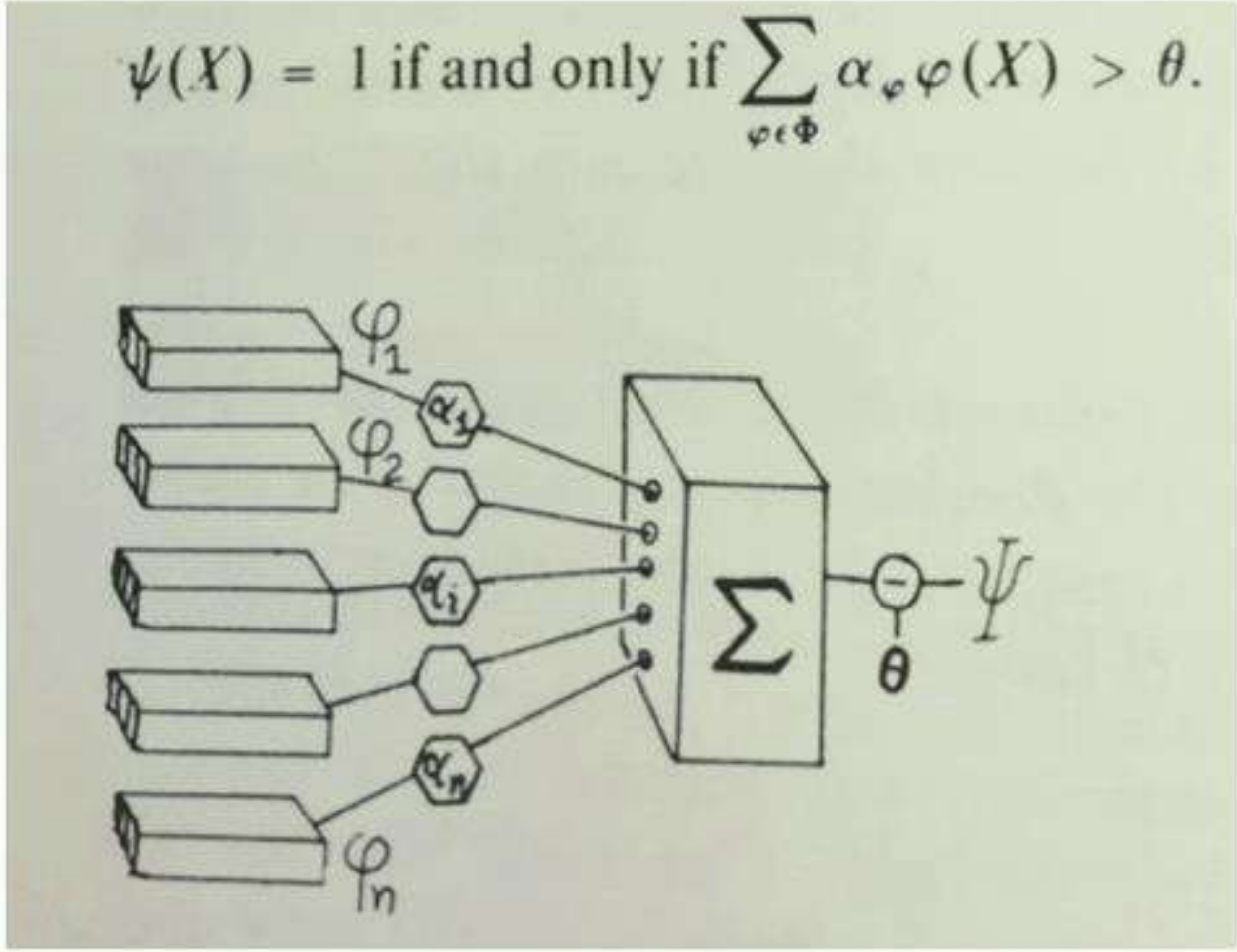
even

M. Minsky and S. Papert, Perceptrons: an introduction to computational geometry, MIT Press, 1969.

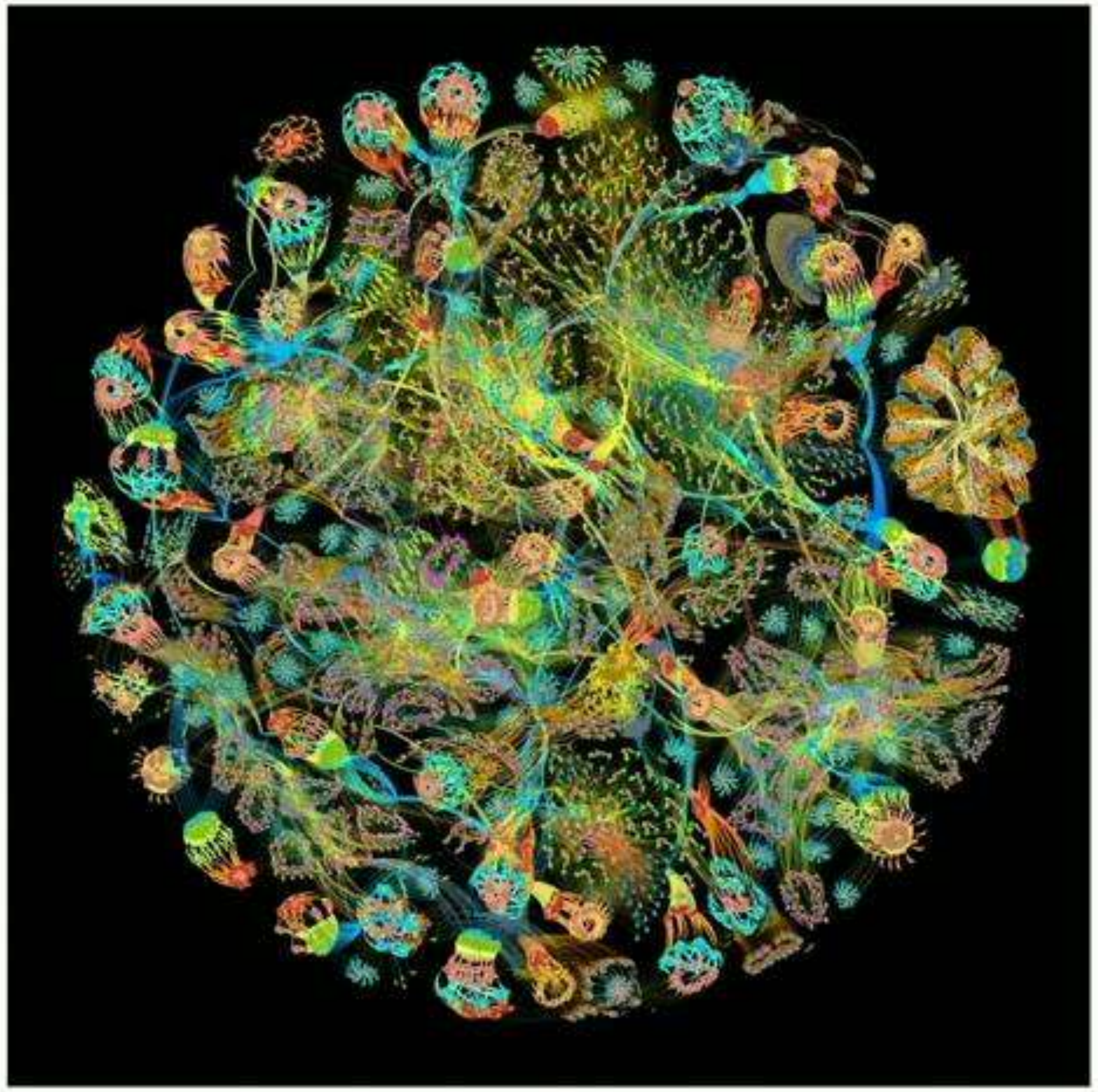




1969


$\psi(X) = 1$ if and only if $\sum_{\varphi \in \Phi} \alpha_\varphi \varphi(X) > \theta$.
φ_1
φ_2
φ_n
α_1
α_i
α_n
Σ
θ
Ψ

2019



u/mattfyles: ResNet-50 neural network from Microsoft Research.
~3 million nodes and ~10 million edges (layout in Gephi)

https://i.redd.it/55tffb8uh6uz.jpg

Can deep learning learn any function (say Boolean) that is learnable in poly-time?

Can deep learning learn any function (say Boolean)
that is learnable in poly-time?

What do we mean by ‘deep learning’
and ‘can learn’?

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

↯ **Optimization.** ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Optimization. ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Are poly-size NN with **SGD** the **"ultimate learning paradigm"**?
[Ben-David,Shalev-Shwartz]

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Optimization. ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Are poly-size NN with **SGD** the **"ultimate learning paradigm"**?
[Ben-David,Shalev-Shwartz]

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Optimization. ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Are poly-size NN with **SGD** the **"ultimate learning paradigm"**?
[Ben-David,Shalev-Shwartz]

1. Can we learn a given function with a random initialization?
2. Can we learn a random function with a chosen initialization?

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Optimization. ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Are poly-size NN with **SGD** the **"ultimate learning paradigm"**?
[Ben-David,Shalev-Shwartz]

1. Can we learn a given function with a random initialization?
2. Can we learn a random function with a chosen initialization?

Example. Parities: $f_S(x) = \prod_{i \in S} x_i, \quad S \sim_U 2^{[n]}$

Formalizing the problem

Approximation. Any function on n variables that can be implemented in poly(n)-time can be expressed by a poly(n)-size NN [Parberry 94, Sipser 06]

Estimation. Poly(n)-size NN can be learned with empirical risk minimization (ERM) with poly(n)-samples [VC 71, Anthony-Bartlett 99]

Optimization. ERM is NP-hard [Bartlett-Ben-David 02, Klivans et al. 09]

Are poly-size NN with **SGD** the **"ultimate learning paradigm"**?
[Ben-David,Shalev-Shwartz]

1. Can we learn a given function with a random initialization?
2. Can we learn a random function with a chosen initialization?

Example. Parities: $f_S(x) = \prod_{i \in S} x_i, \quad S \sim_U 2^{[n]}$

under symmetry: failure at 2 implies failure at 1 for a typical function

Formalizing the problem

$\mathcal{X}$: the data domain ($\{+1,-1\}^n$)

$\mathcal{Y}$: the label domain ($\{+1,-1\}$)

$P_{\mathcal{X}}$: prob. dist. on $\mathcal{X}$

$P_{\mathcal{F}}$: prob. dist. on $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$

Formalizing the problem

$\mathcal{X}$: the data domain ($\{+1,-1\}^n$)

$\mathcal{Y}$: the label domain ($\{+1,-1\}$)

$P_{\mathcal{X}}$: prob. dist. on $\mathcal{X}$

$P_{\mathcal{F}}$: prob. dist. on $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$

balanced classes: $\mathbb{P}_{(F,X)\sim P_{\mathcal{F}}\times P_{\mathcal{X}}}(F(X)=1) = 1/2 + o_n(1)$

Formalizing the problem

$\mathcal{X}$: the data domain ($\{+1,-1\}^n$)

$\mathcal{Y}$: the label domain ($\{+1,-1\}$)

$P_{\mathcal{X}}$: prob. dist. on $\mathcal{X}$

$P_{\mathcal{F}}$: prob. dist. on $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$

balanced classes: $\mathbb{P}_{(F,X)\sim P_{\mathcal{F}}\times P_{\mathcal{X}}}(F(X)=1) = 1/2 + o_n(1)$

Definition. Weak learning of $(P_{\mathcal{X}}, P_{\mathcal{F}})$ in t time-steps:

- $F \sim P_{\mathcal{F}}$
- Access t times an oracle relying on $(P_{\mathcal{X}}, F) \rightarrow (X_i, F(X_i))$, $X_i \sim P_{\mathcal{X}}$
- Output $\hat{F}^{(t)}$ such that $\mathbb{P}(\hat{F}^{(t)}(X_{t+1}) = F(X_{t+1})) = 1/2 + \Omega_n(1)$

Formalizing the problem

$\mathcal{X}$: the data domain ($\{+1,-1\}^n$)
$P_{\mathcal{X}}$: prob. dist. on $\mathcal{X}$

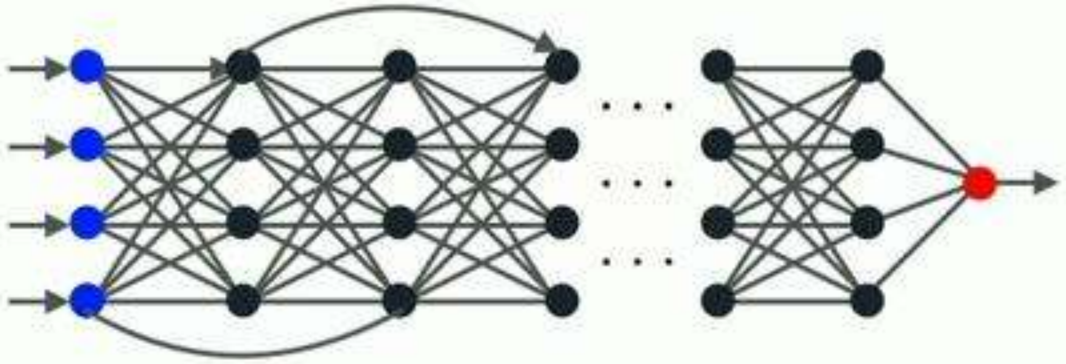
$\mathcal{Y}$: the label domain ($\{+1,-1\}$)
$P_{\mathcal{F}}$: prob. dist. on $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$

balanced classes: $\mathbb{P}_{(F,X)\sim P_{\mathcal{F}}\times P_{\mathcal{X}}}(F(X)=1) = 1/2 + o_n(1)$

Definition. Weak learning of $(P_{\mathcal{X}}, P_{\mathcal{F}})$ in t time-steps:

- $F \sim P_{\mathcal{F}}$
- Access t times an oracle relying on $(P_{\mathcal{X}}, F) \;\to (X_i, F(X_i))$, $X_i \sim P_{\mathcal{X}}$
- Output $\hat{F}^{(t)}$ such that $\mathbb{P}(\hat{F}^{(t)}(X_{t+1}) = F(X_{t+1})) = 1/2 + \Omega_n(1)$

Definition. Neural nets = weighted DAG, $n{+}1$ roots (inputs), one leaf (output) plus some non-linearity on other vertices



results

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

“average-case” v.s. “worst-case” SQ (E. Boix)

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Parities [Shalev-Shwartz, Shamir, Shammah 17]

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Parities [Shalev-Shwartz, Shamir, Shammah 17]

Corollary. GD can learn efficiently monomials of degree k if and only if k is finite

Results

population gradient

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

“average-case” v.s. “worst-case” SQ (E. Boix)

Parities [Shalev-Shwartz, Shamir, Shammah 17]

Corollary. GD can learn efficiently monomials of degree k if and only if k is finite

Results

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

- poly-size NN
- poly-time initialization
- poly-steps
- poly-rate
- poly-range
- **poly-noise**

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

- poly-size NN
- poly-time initialization
- poly-steps
- poly-rate
- poly-range
- **poly-noise /poly-precision**

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

- poly-size NN
- poly-time initialization
- poly-steps
- poly-rate
- poly-range
- **poly-noise /poly-precision**

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Corollary. SGD can learn efficiently parities while Perceptron, GD or SQ cannot

Formalizing the problem

$\mathcal{X}$: the data domain ($\{+1,-1\}^n$) $\quad$ $P_{\mathcal{X}}$: prob. dist. on $\mathcal{X}$

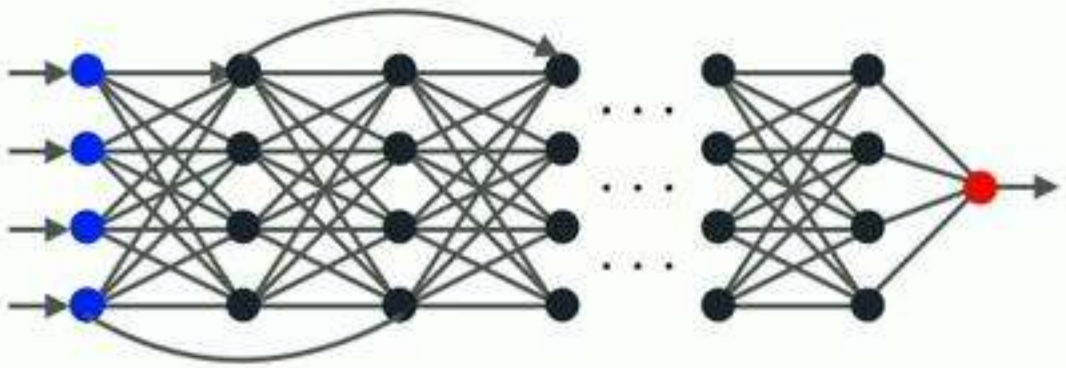
$\mathcal{Y}$: the label domain ($\{+1,-1\}$) $\quad$ $P_{\mathcal{F}}$: prob. dist. on $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$

$$\text{balanced classes: } \mathbb{P}_{(F,X)\sim P_{\mathcal{F}}\times P_{\mathcal{X}}}(F(X)=1) = 1/2 + o_n(1)$$

Definition. Weak learning of $(P_{\mathcal{X}}, P_{\mathcal{F}})$ in t time-steps:

- $F \sim P_{\mathcal{F}}$
- Access t times an oracle relying on $(P_{\mathcal{X}}, F) \rightarrow (X_i, F(X_i))$, $X_i \sim P_{\mathcal{X}}$
- Output $\hat{F}^{(t)}$ such that $\mathbb{P}(\hat{F}^{(t)}(X_{t+1}) = F(X_{t+1})) = 1/2 + \Omega_n(1)$

Definition. Neural nets = weighted DAG, $n+1$ roots (inputs), one leaf (output) plus some non-linearity on other vertices



Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Parities [Shalev-Shwartz, Shamir, Shammah 17]

Corollary. GD can learn efficiently monomials of degree k if and only if k is finite

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Results

Theorem 1. GD cannot learn efficiently function distributions having low CP

- poly-size NN
- any initialization
- poly-steps
- poly-rate
- poly-range
- poly-noise

$$\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2$$

$$(X, F, F') \sim P_{\mathcal{X}} \times P_{\mathcal{F}} \times P_{\mathcal{F}}$$

low = super-poly decay in n

related to statistical dim. [Kearns 98, Blum et al. 01]

"average-case" v.s. "worst-case" SQ (E. Boix)

Parities [Shalev-Shwartz, Shamir, Shammah 17]

Corollary. GD can learn efficiently monomials of degree k if and only if k is finite

Results

Theorem 2. SGD can learn efficiently any efficiently learnable distribution

- poly-size NN
- poly-time initialization
- poly-steps
- poly-rate
- poly-range
- **poly-noise /poly-precision**

any $(P_{\mathcal{X}}, P_{\mathcal{F}})$ that can be learned by some algorithm in poly-time with poly-samples

Corollary. SGD can learn efficiently parities while Perceptron, GD or SQ cannot

proof techniques

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \quad D_\star : \text{junk data}$$

(Gradient descent)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \leq \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \leq \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV} \leq T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\star^{(1)})_{TV}$$

(Data-processing + triangular inequality)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \leq \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV} \leq T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\star^{(1)})_{TV}$$

(Data-processing + triangular inequality)

$$\mathbb{E}_F d(Q_F, Q_\star)_{TV} \leq \frac{\gamma}{2\sigma} (\mathbb{E}_F \| \mathbb{E}_{D_F} \nabla - \mathbb{E}_{D_\star} \nabla \|_2^2)^{1/2}$$

(Pinsker's inequality)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \le \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV} \le T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\star^{(1)})_{TV}$$

(Data-processing + triangular inequality)

$$\mathbb{E}_F d(Q_F, Q_\star)_{TV} \le \frac{\gamma}{2\sigma} (\mathbb{E}_F \|\mathbb{E}_{D_F} \nabla - \mathbb{E}_{D_\star} \nabla\|_2^2)^{1/2}$$

(Pinsker's inequality)

$$\mathbb{E}_F (\mathbb{E}_{D_F} \nabla_e - \mathbb{E}_{D_\star} \nabla_e)^2 = \mathbb{E}_F (\mathbb{E}_{D_\star} \nabla_e (1 - D_F / D_\star))^2$$

(Radon-Nikodym)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \le \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV} \le T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\star^{(1)})_{TV}$$

(Data-processing + triangular inequality)

$$\mathbb{E}_F d(Q_F, Q_\star)_{TV} \le \frac{\gamma}{2\sigma} (\mathbb{E}_F \|\mathbb{E}_{D_F} \nabla - \mathbb{E}_{D_\star} \nabla\|_2^2)^{1/2}$$

(Pinsker's inequality)

$$\mathbb{E}_F (\mathbb{E}_{D_F} \nabla_e - \mathbb{E}_{D_\star} \nabla_e)^2 = \mathbb{E}_F (\mathbb{E}_{D_\star} \nabla_e (1 - D_F / D_\star))^2$$

(Radon-Nikodym)

$$= \mathbb{E}_F \mathbb{E}_{D_\star} \nabla_e^{\otimes 2} (1 - D_F / D_\star)^{\otimes 2}$$

(Tensorization)

CP: information-theoretic argument

$$W_H{}^{(t)} = W_H{}^{(t-1)} - \gamma \mathbb{E}_{(X,Y)\sim D_H} \nabla L(W_H{}^{(t-1)}(X), Y) + Z_\sigma^{(t)} \qquad H \in \{F, \star\}$$

$$W^{(0)} = W_F^{(0)} = W_\star^{(0)} \qquad D_F : \text{true data} \qquad D_\star : \text{junk data}$$

(Gradient descent)

$$\mathbb{P}\{W_F{}^{(T)}(X) = F(X)\} \le \mathbb{P}\{\underbrace{W_\star{}^{(T)}(X) = F(X)}_{1/2}\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV}$$

(Total-variation bound)

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\star^{(T)})_{TV} \le T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\star^{(1)})_{TV}$$

(Data-processing + triangular inequality)

$$\mathbb{E}_F d(Q_F, Q_\star)_{TV} \le \frac{\gamma}{2\sigma} (\mathbb{E}_F \| \mathbb{E}_{D_F} \nabla - \mathbb{E}_{D_\star} \nabla \|_2^2)^{1/2}$$

(Pinsker's inequality)

$$\mathbb{E}_F (\mathbb{E}_{D_F} \nabla_e - \mathbb{E}_{D_\star} \nabla_e)^2 = \mathbb{E}_F (\mathbb{E}_{D_\star} \nabla_e (1 - D_F / D_\star))^2$$

(Radon-Nikodym)

$$= \mathbb{E}_F \mathbb{E}_{D_\star} \nabla_e^{\otimes 2} (1 - D_F / D_\star)^{\otimes 2}$$

(Tensorization)

$$\le (\mathbb{E}_{D_\star} \nabla_e^2)(\mathbb{E}_{F,F'} \mathbb{E}_{D_\star} (1 - D_F / D_\star)^{\otimes 2} (1 - D_{F'} / D_\star)^{\otimes 2})^{1/2}$$

(CS+replica)

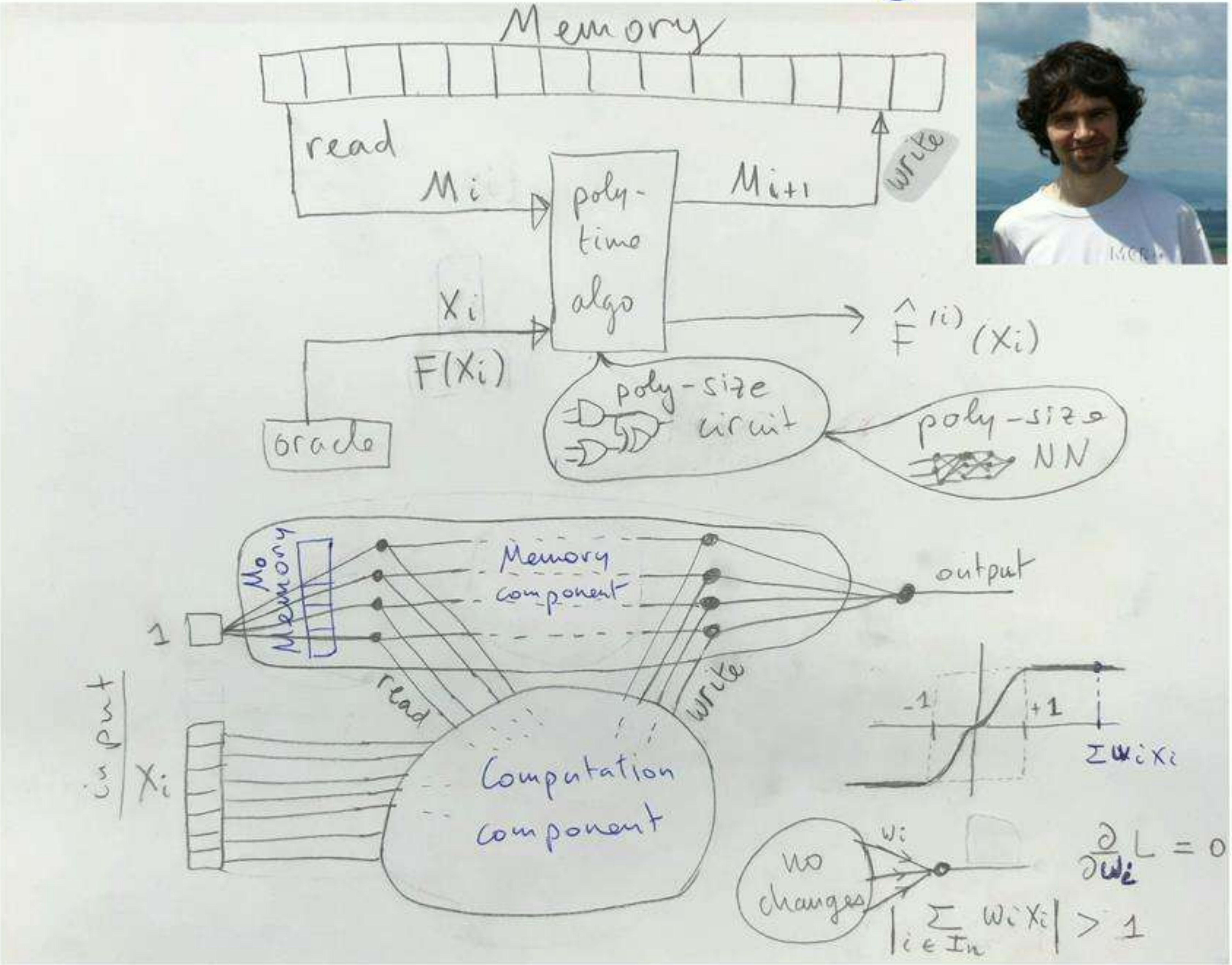
Junk-flow and cross-predictability

Theorem. $\mathbb{P}\{{W_F}^{(T)}(X) = F(X)\} \leq 1/2 + \frac{1}{\sigma} \cdot \mathrm{JF} \cdot \mathrm{CP}^{1/4}$

$$\mathrm{JF} := \sum_{t=1}^{T} \gamma_t \|\mathbb{E}_{D_\star} \nabla^{(t)}\|_2$$

$$\mathrm{CP} := \mathbb{E}_{F,F'} (\mathbb{E}_X F(X) F(X'))^2$$

The universal emulation argument



thank you

Towards demystifying Generalization and Early Stopping in Neural Networks

Mahdi Soltanolkotabi
Department of Electrical and Computer Engineering
USC University of Southern California

August 26, 2019
AI Institute “Geometry of Deep Learning”
Microsoft Research, Redmond, WA

Collaborators:

Samet Oymak, Zalan Fabian, Mingchen Li

Motivation: overparameterization without overfitting

Motivation: overparameterization without overfitting

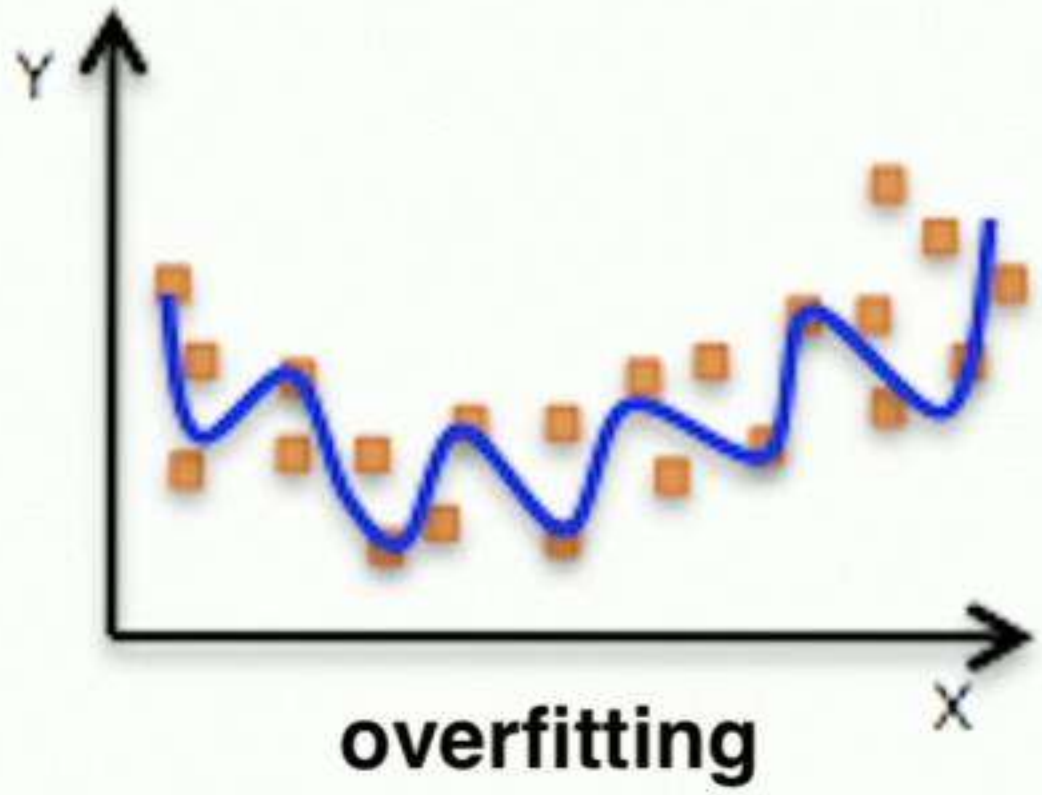
Mystery

$$\# \text{ of parameters} \quad >> \quad \# \text{ training data}$$

Motivation: overparameterization without overfitting

Mystery

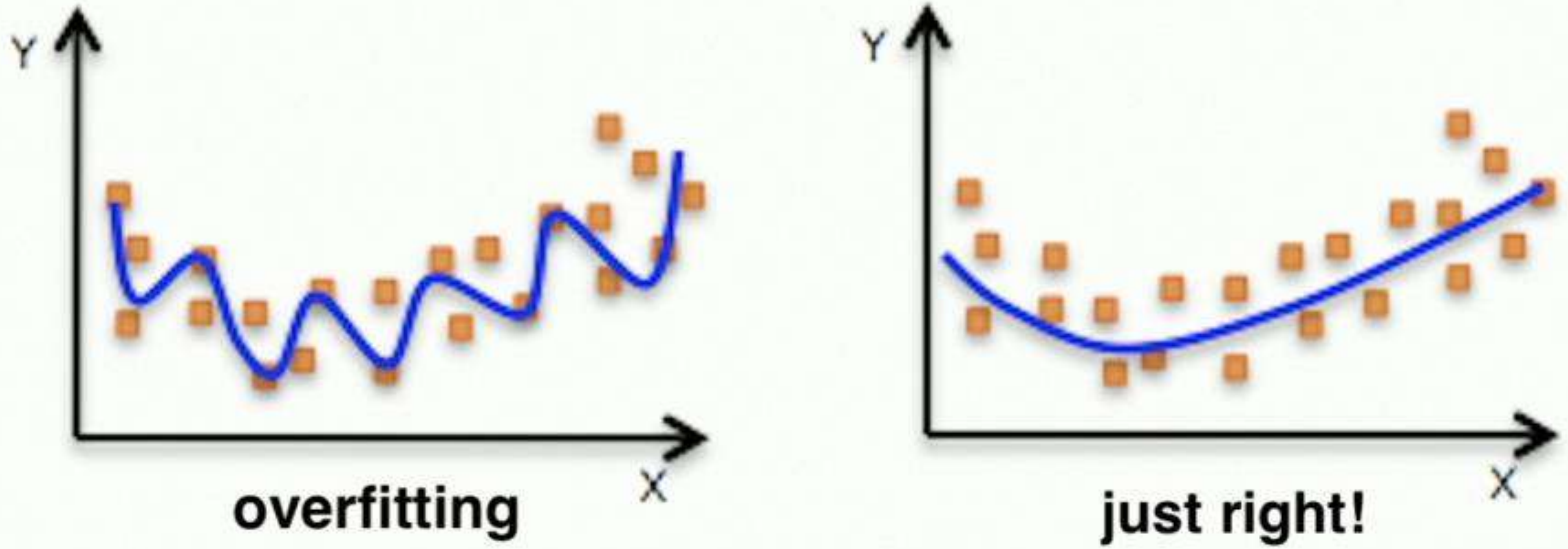
# of parameters >> # training data



Motivation: overparameterization without overfitting

Mystery

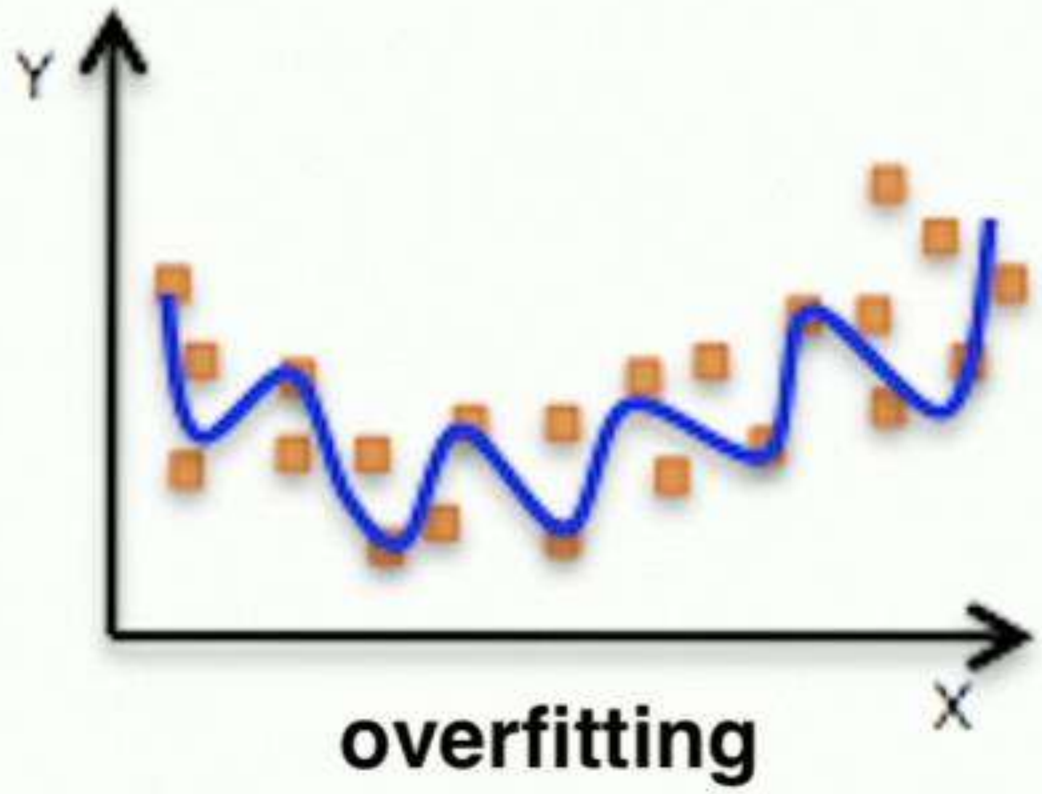
# of parameters >> # training data

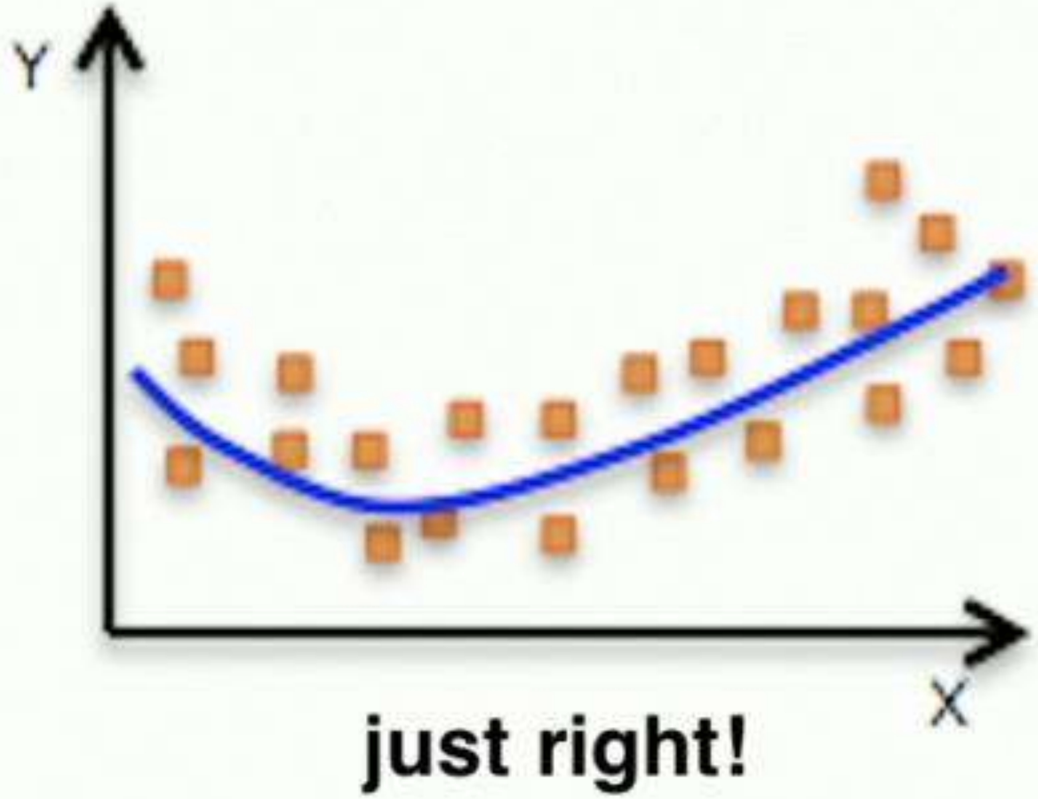


Motivation: overparameterization without overfitting

Mystery

# of parameters >> # training data





Challenges:

- *Optimization: Why can neural nets fit to the training data?*
- *Generalization: Why can neural nets predict?*

Experiment I: Overfitting to corruption

Experiment I: Overfitting to corruption

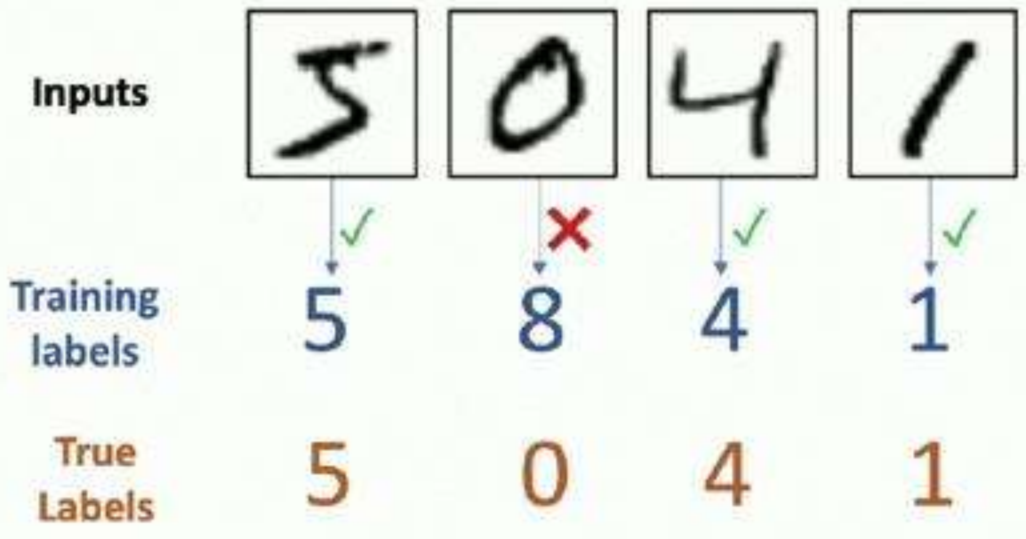
Add corruption

- Corrupt a fraction of training labels by replacing with another random label
- No corruption on test labels

Experiment I: Overfitting to corruption

Add corruption

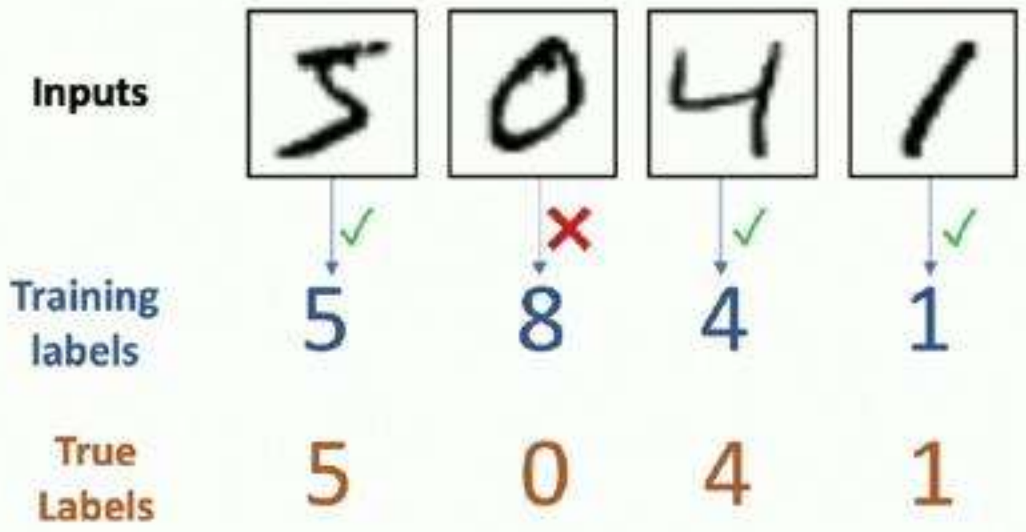
- Corrupt a fraction of training labels by replacing with another random label
- No corruption on test labels

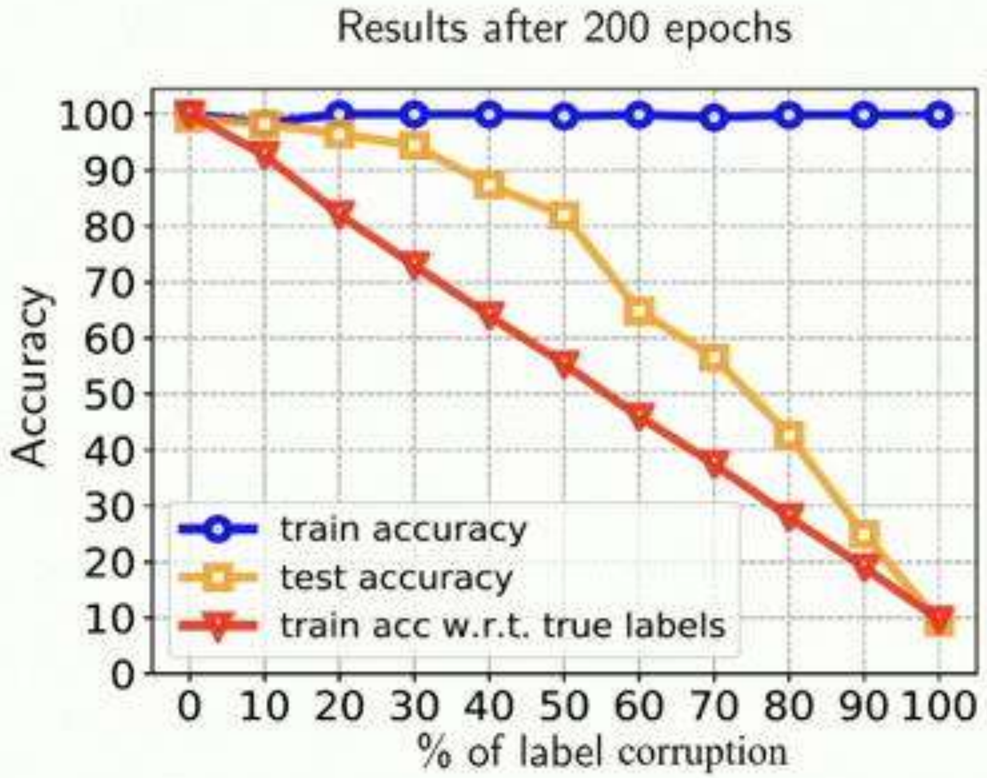


Experiment I: Overfitting to corruption

Add corruption

- Corrupt a fraction of training labels by replacing with another random label
- No corruption on test labels



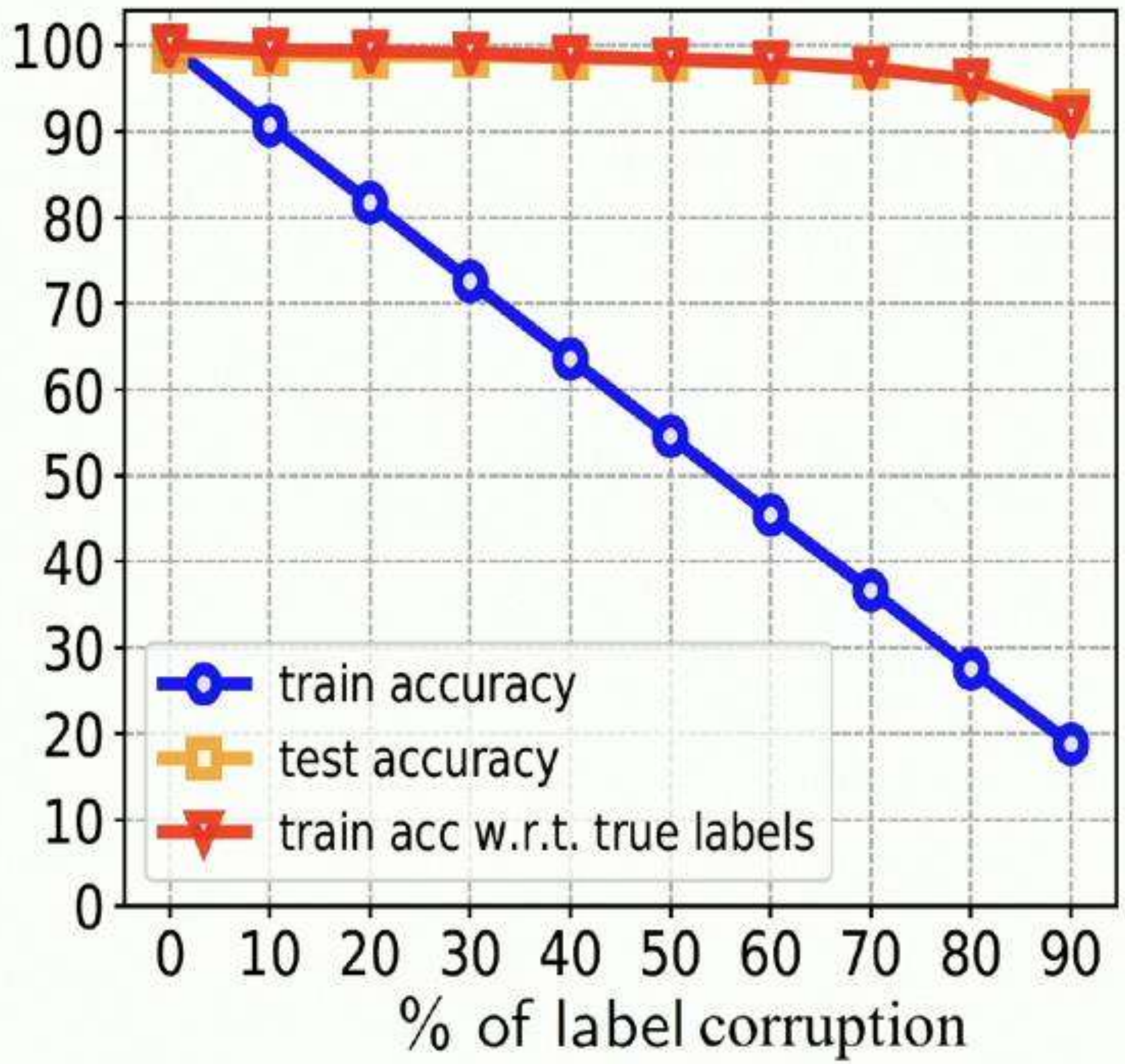


Experiment II-Early stopping and robustness

Repeat the same experiment but stop early

Experiment II-Early stopping and robustness

Repeat the same experiment but stop early

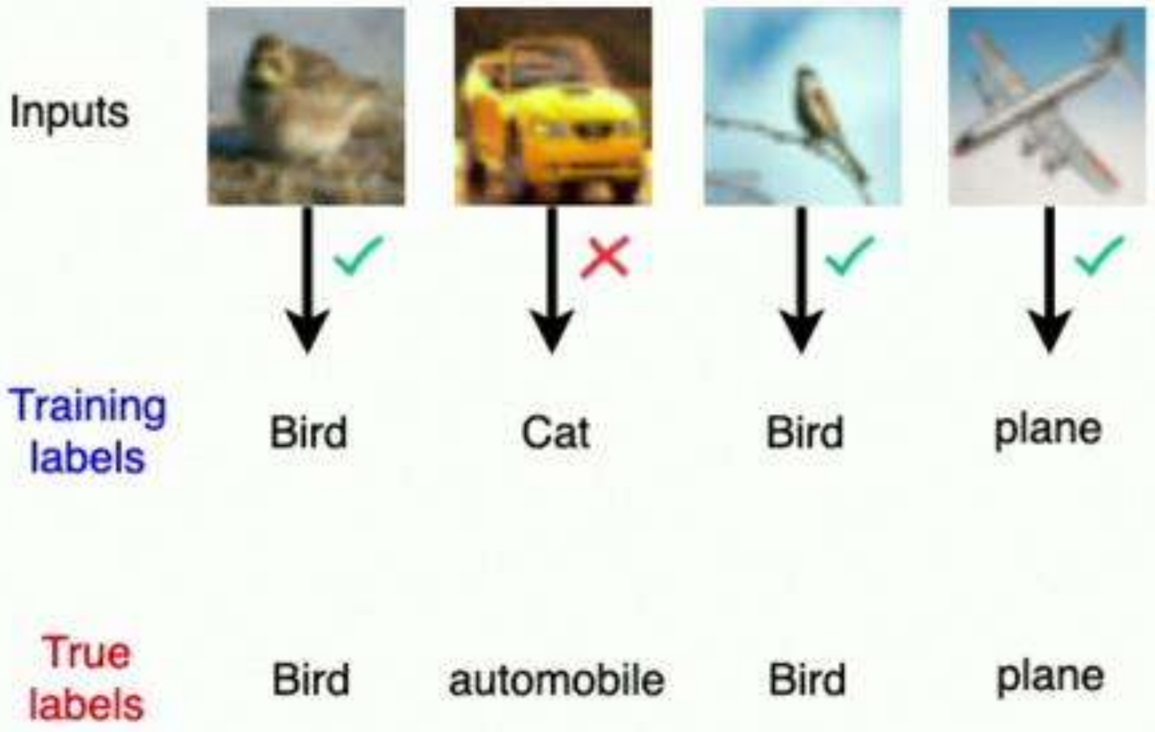


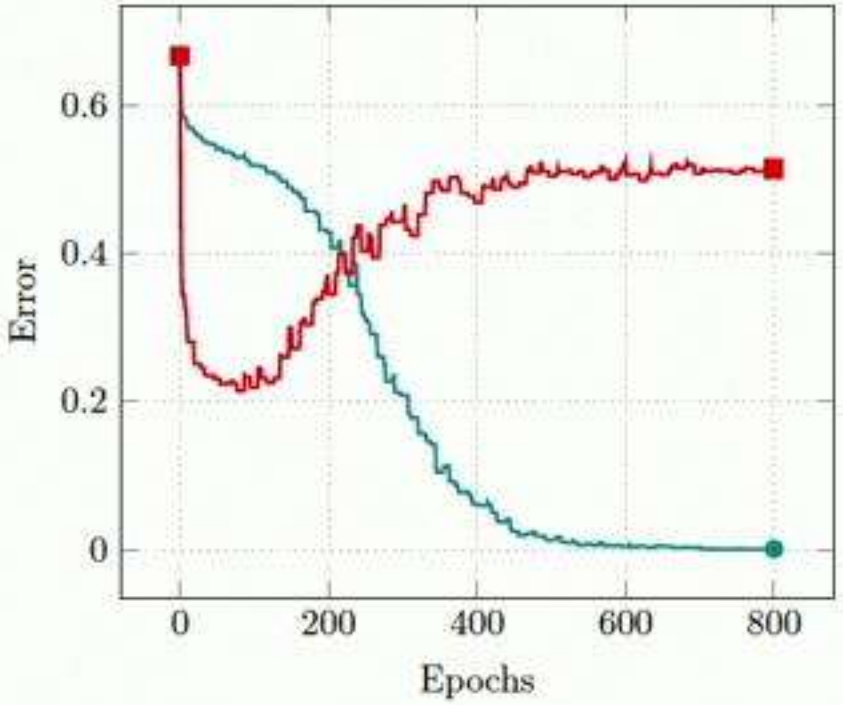
Focus: optimization/generalization dynamics

Focus: optimization/generalization dynamics

Add corruption

- Corrupt 50% of training labels by replacing with another random label
- No corruption on test labels





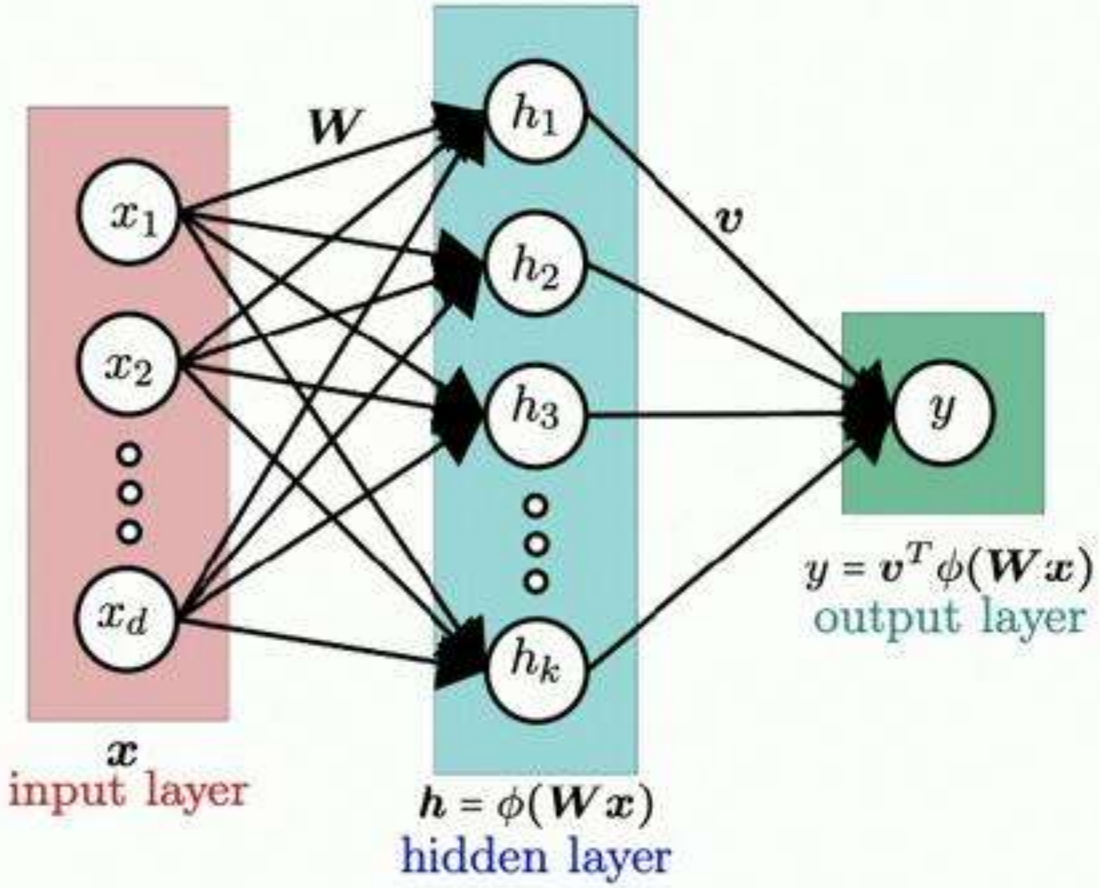
train error (green)
test error (red)

Theory for overparameterization without overfitting

- *Optimization*
- *Generalization*
- *Early stopping*

Optimization

One-hidden layer

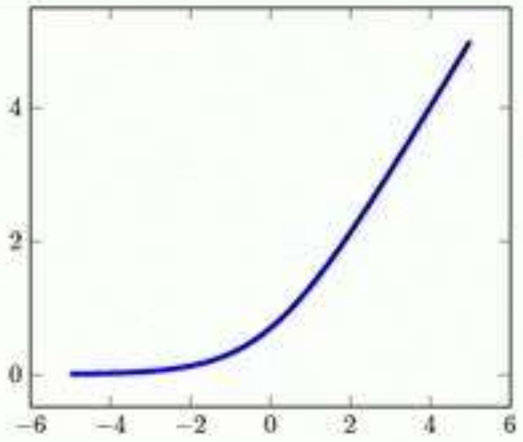


$$\boldsymbol{y}_i = \boldsymbol{v}^T\phi(\boldsymbol{W}\boldsymbol{x}_i)$$

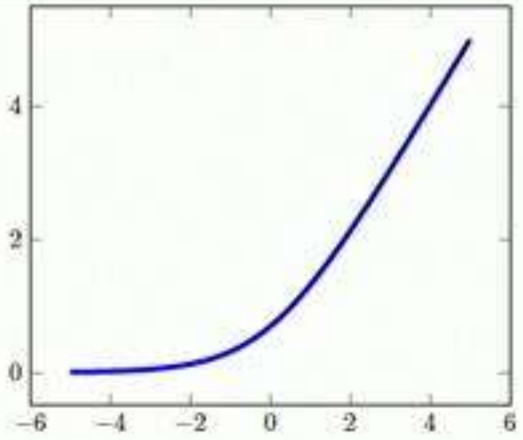
Theory for smooth activations

- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss

$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v},\boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$



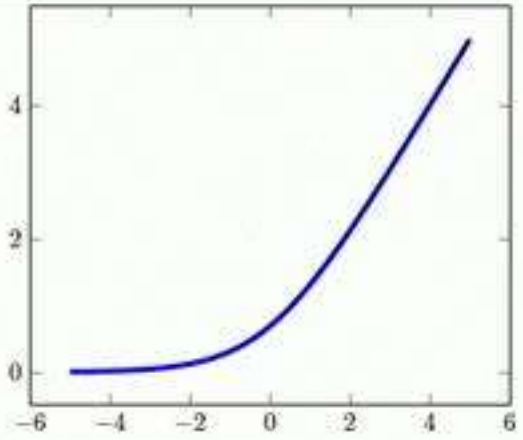
Theory for smooth activations



- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v}, \boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$
  - Run gradient descent
$$(\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) = (\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau)$$

Theory for smooth activations

- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v}, \boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$
  - Run gradient descent
  $(\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) = (\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau)$

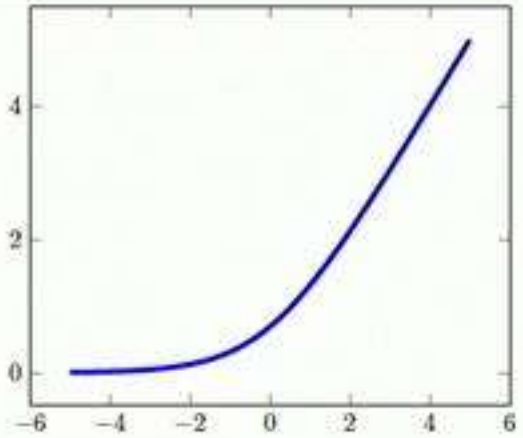


Theorem

Assume

- *Distinct data points* ($\boldsymbol{x}_i \neq \boldsymbol{x}_j$)
- *Smooth activations e.g.* $\phi(z) = \log(1 + e^z)$
- *Overparameterization: # training data* $\leq 2 \times$ *# width* ($k \geq \frac{n}{2}$)
- *Initialization* $\boldsymbol{v}_0$ *at random i.i.d.* $\mathcal{N}(0, \nu^2)$ *and* $\boldsymbol{W}_0$ *i.i.d.* $\mathcal{N}(0,1)$ *with* $\nu >> 1$

Theory for smooth activations



- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v},\boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$
    - Run gradient descent
    $(\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) = (\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau)$

Theorem

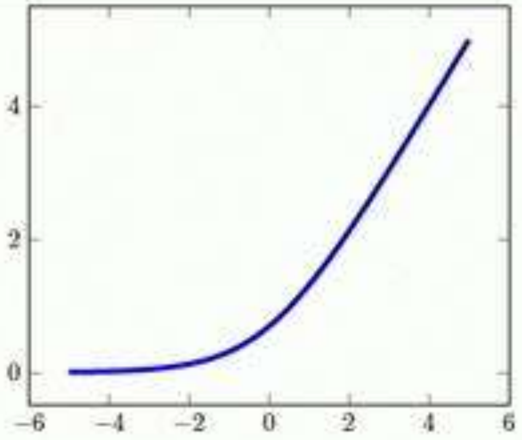
Assume

- *Distinct data points* ($\boldsymbol{x}_i \neq \boldsymbol{x}_j$)
- *Smooth activations e.g.* $\phi(z) = \log(1 + e^z)$
- *Overparameterization: # training data* $\leq 2 \times$ *# width* ($k \geq \frac{n}{2}$)
- *Initialization* $\boldsymbol{v}_0$ *at random i.i.d.* $\mathcal{N}\left(0, \nu^2\right)$ *and* $\boldsymbol{W}_0$ *i.i.d.* $\mathcal{N}(0,1)$ *with* $\nu >> 1$

Then, with high probability

- *Zero training error:* $\mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau) \leq (1-\rho)^\tau \mathcal{L}(\boldsymbol{v}_0, \boldsymbol{W}_0)$

Theory for smooth activations



- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v},\boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$
  - Run gradient descent
  $(\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) = (\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau)$

Theorem

Assume

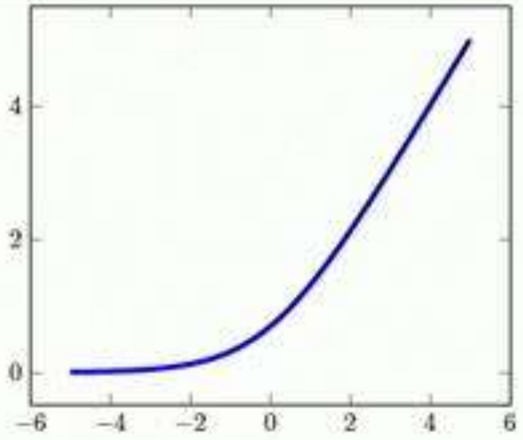
- *Distinct data points ($\boldsymbol{x}_i \neq \boldsymbol{x}_j$)*
- *Smooth activations e.g. $\phi(z) = \log(1 + e^z)$*
- *Overparameterization: # training data $\leq 2 \times$ # width ($k \geq \frac{n}{2}$)*
- *Initialization $\boldsymbol{v}_0$ at random i.i.d. $\mathcal{N}\left(0, \nu^2\right)$ and $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}\left(0, 1\right)$ with $\nu >> 1$*

Then, with high probability

- *Zero training error: $\mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau) \leq (1-\rho)^\tau \mathcal{L}(\boldsymbol{v}_0, \boldsymbol{W}_0)$*

Possible extension: If rank$(\boldsymbol{X}) = d$, # training data $\leq 2 \times$ # parameters*

Theory for smooth activations



- Data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
$$\min_{\boldsymbol{v},\boldsymbol{W}} \mathcal{L}(\boldsymbol{v}, \boldsymbol{W}) := \sum_{i=1}^{n} \left(\boldsymbol{v}^T \phi(\boldsymbol{W}\boldsymbol{x}_i) - y_i\right)^2$$
  - Run gradient descent
  $(\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) = (\boldsymbol{v}_{\tau+1}, \boldsymbol{W}_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau)$

Theorem

Assume

- *Distinct data points ($\boldsymbol{x}_i \neq \boldsymbol{x}_j$)*
- *Smooth activations e.g.* $\phi(z) = \log(1 + e^z)$
- *Overparameterization: # training data* $\leq 2 \times$ *# width* ($k \geq \frac{n}{2}$)
- *Initialization* $\boldsymbol{v}_0$ *at random i.i.d.* $\mathcal{N}\left(0, \nu^2\right)$ *and* $\boldsymbol{W}_0$ *i.i.d.* $\mathcal{N}(0, 1)$ *with* $\nu >> 1$

Then, with high probability

- *Zero training error:* $\mathcal{L}(\boldsymbol{v}_\tau, \boldsymbol{W}_\tau) \leq (1-\rho)^\tau \mathcal{L}(\boldsymbol{v}_0, \boldsymbol{W}_0)$

Possible extension: If rank$(\boldsymbol{X}) = d$, # training data $\leq 2 \times$ # parameters*
Prior work [Du et. al., Allen-Zhu et. al., Oymak et. al. ...]
require very wide networks $k \gtrsim \frac{n^4}{\lambda^4}$

Generalization

Model and training

- Data: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathbb{R}^d \times \mathbb{R}^K$
- Training Loss:
  $\mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\sum_{i=1}^{n} \|\boldsymbol{V}\phi(\boldsymbol{W}\boldsymbol{x}_i) - \boldsymbol{y}_i\|_{\ell_2}^2$

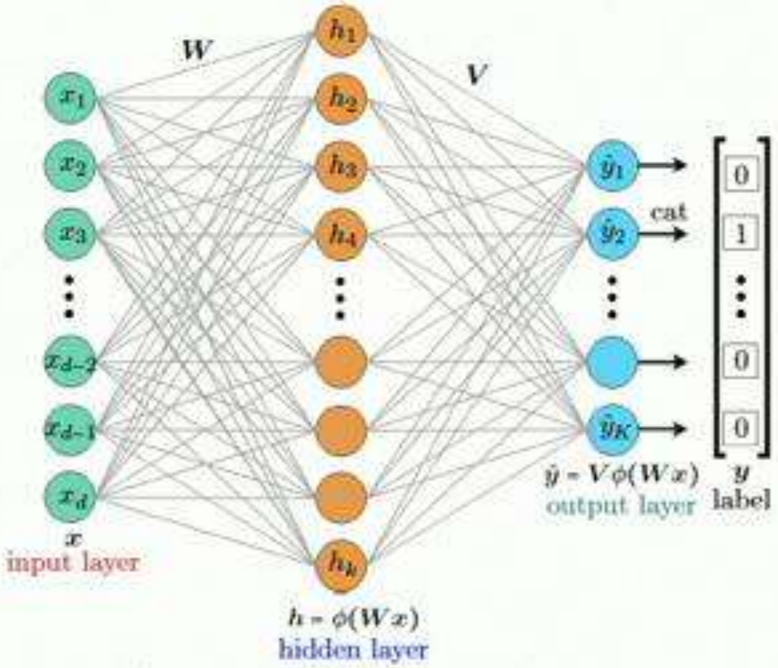
  Concatenate label and prediction vectors

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix}, f(\boldsymbol{W}) = \begin{bmatrix} \boldsymbol{V}f(\boldsymbol{x}_1; \boldsymbol{W}) \\ \vdots \\ \boldsymbol{V}f(\boldsymbol{x}_n; \boldsymbol{W}) \end{bmatrix} \in \mathbb{R}^{nK}.$$

$$\min_{\boldsymbol{W} \in \mathbb{R}^{k \times d}} \mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\|f(\boldsymbol{W}) - \boldsymbol{y}\|_{\ell_2}^2 .$$

- Algorithm: gradient descent from random initialization

$$\boldsymbol{W}_{\tau+1} = \boldsymbol{W}_{\tau} - \eta \nabla \mathcal{L}(\boldsymbol{W}_{\tau})$$

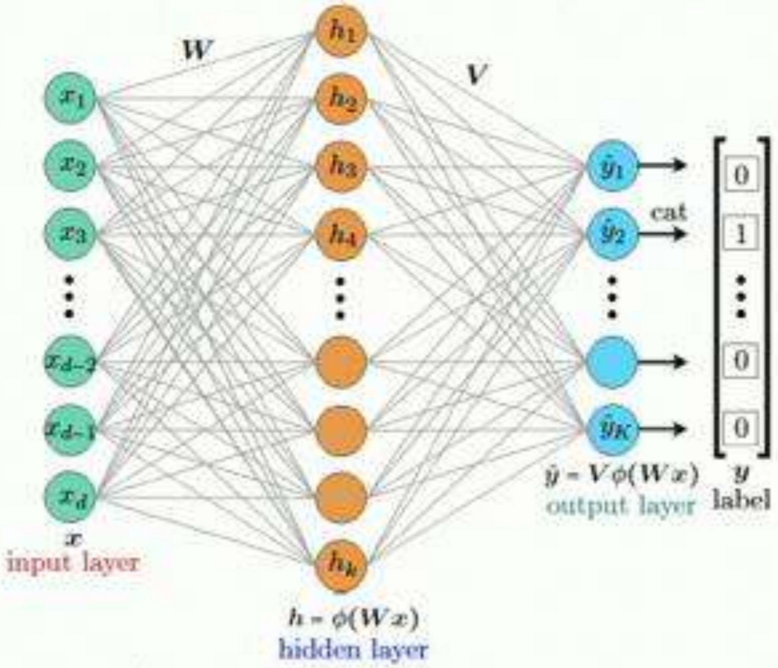


Model and training

- Data: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathbb{R}^d \times \mathbb{R}^K$
- Training Loss:
  $\mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\sum_{i=1}^n \|\boldsymbol{V}\phi(\boldsymbol{W}\boldsymbol{x}_i) - \boldsymbol{y}_i\|_{\ell_2}^2$

  Concatenate label and prediction vectors

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix}, f(\boldsymbol{W}) = \begin{bmatrix} \boldsymbol{V}f(\boldsymbol{x}_1; \boldsymbol{W}) \\ \vdots \\ \boldsymbol{V}f(\boldsymbol{x}_n; \boldsymbol{W}) \end{bmatrix} \in \mathbb{R}^{nK}.$$



$$\min_{\boldsymbol{W}\in\mathbb{R}^{k\times d}} \mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\|f(\boldsymbol{W}) - \boldsymbol{y}\|_{\ell_2}^2.$$

- Algorithm: gradient descent from random initialization

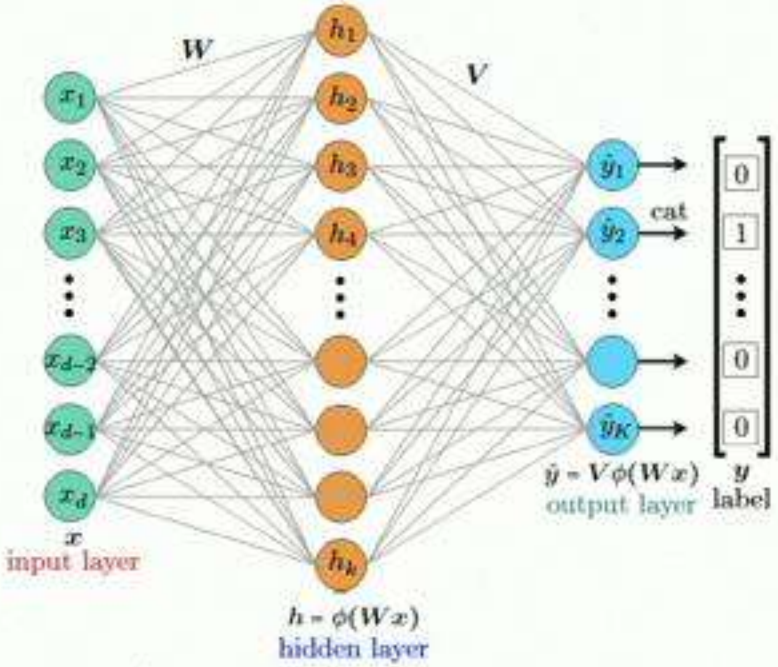
$$\boldsymbol{W}_{\tau+1} = \boldsymbol{W}_\tau - \eta\nabla\mathcal{L}(\boldsymbol{W}_\tau)$$

$$\nabla\mathcal{L}(\boldsymbol{W}) = \mathcal{J}^T(\boldsymbol{W})\left(f(\boldsymbol{W}) - \boldsymbol{y}\right) \quad \text{with} \quad \mathcal{J}(\boldsymbol{W}) = \frac{\partial f(\boldsymbol{W})}{\partial \text{vect}(\boldsymbol{W})}.$$

Model and training

- Data: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathbb{R}^d \times \mathbb{R}^K$
- Training Loss:
  $\mathcal{L}(\boldsymbol{W}) := \frac{1}{2} \sum_{i=1}^{n} \| \boldsymbol{V} \phi(\boldsymbol{W} \boldsymbol{x}_i) - \boldsymbol{y}_i \|_{\ell_2}^2$

  Concatenate label and prediction vectors



$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix}, f(\boldsymbol{W}) = \begin{bmatrix} \boldsymbol{V} f(\boldsymbol{x}_1; \boldsymbol{W}) \\ \vdots \\ \boldsymbol{V} f(\boldsymbol{x}_n; \boldsymbol{W}) \end{bmatrix} \in \mathbb{R}^{nK}.$$

$$\min_{\boldsymbol{W} \in \mathbb{R}^{k \times d}} \mathcal{L}(\boldsymbol{W}) := \frac{1}{2} \| f(\boldsymbol{W}) - \boldsymbol{y} \|_{\ell_2}^2 .$$

- Algorithm: gradient descent from random initialization

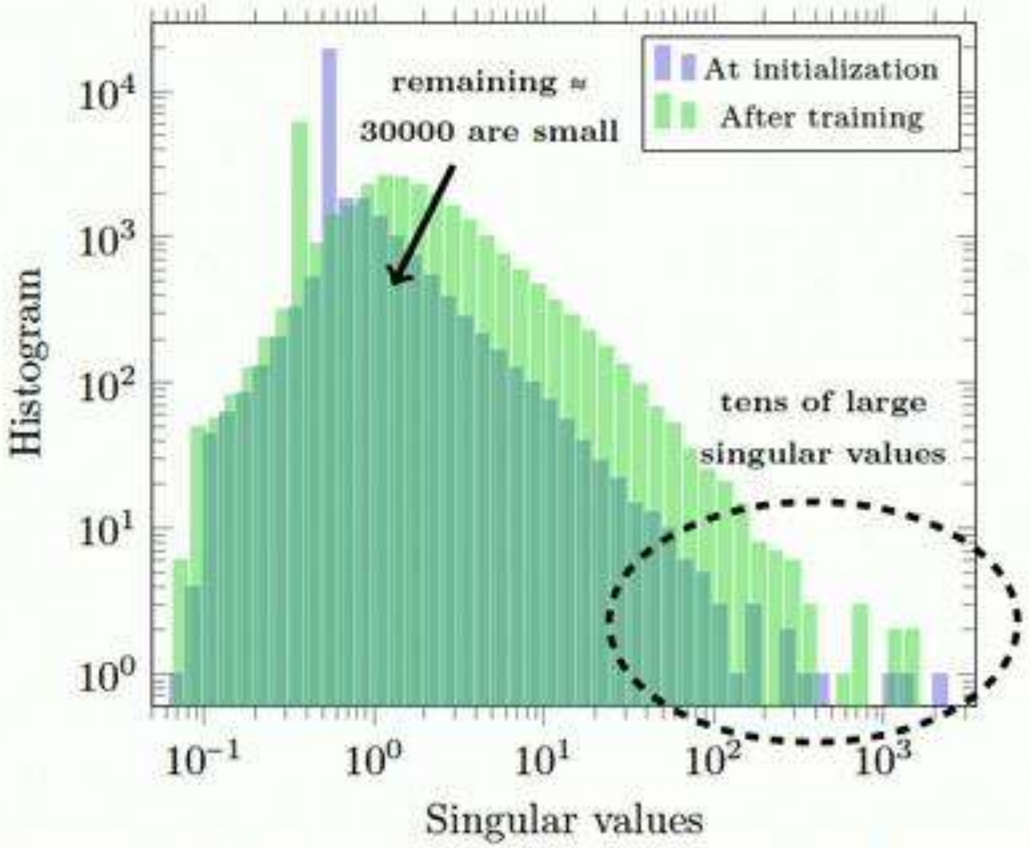
$$\boldsymbol{W}_{\tau+1} = \boldsymbol{W}_\tau - \eta \nabla \mathcal{L}(\boldsymbol{W}_\tau)$$

$$\nabla \mathcal{L}(\boldsymbol{W}) = \mathcal{J}^T(\boldsymbol{W}) (f(\boldsymbol{W}) - \boldsymbol{y}) \quad \text{with} \quad \mathcal{J}(\boldsymbol{W}) = \frac{\partial f(\boldsymbol{W})}{\partial \text{vect}(\boldsymbol{W})}.$$

- Prediction: pass through softmax and pick maximum

Key observation

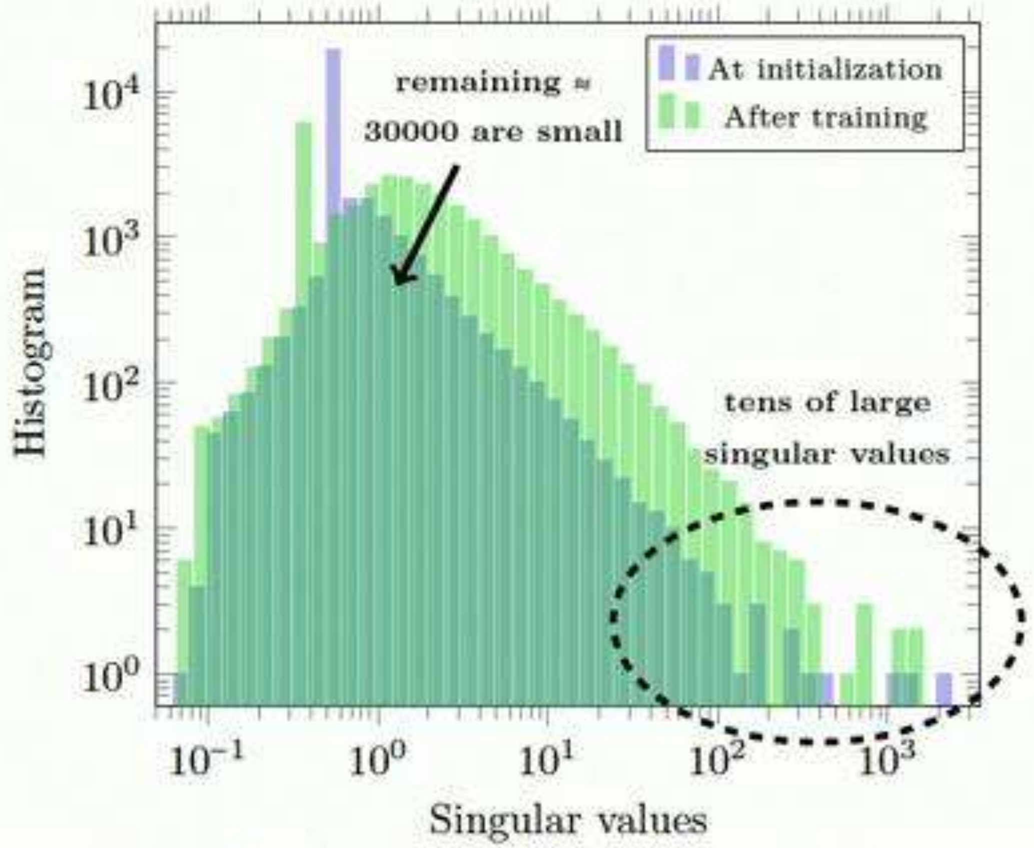
- Dataset: CIFAR10
- Model: ResNET20
- Task: Three-way classification (automobile, airplane, bird)
- $n = 10,000$ and $p = 270,000$



Histogram of the singular values of the initial final Jacobian of neural net during training.

Key observation

- Dataset: CIFAR10
- Model: ResNET20
- Task: Three-way classification (automobile, airplane, bird)
- $n = 10,000$ and $p = 270,000$



Histogram of the singular values of the initial final Jacobian of neural net during training.

Jacobian has low-rank structure

Information and nuisance spaces

$$\nabla \mathcal{L}(\boldsymbol{W}) = \mathcal{J}^T(\boldsymbol{W})\,(f(\boldsymbol{W}) - \boldsymbol{y}) \quad \text{with} \quad \mathcal{J}(\boldsymbol{W}) = \frac{\partial f(\boldsymbol{W})}{\partial \mathsf{vect}(\boldsymbol{W})}.$$

Information and nuisance spaces

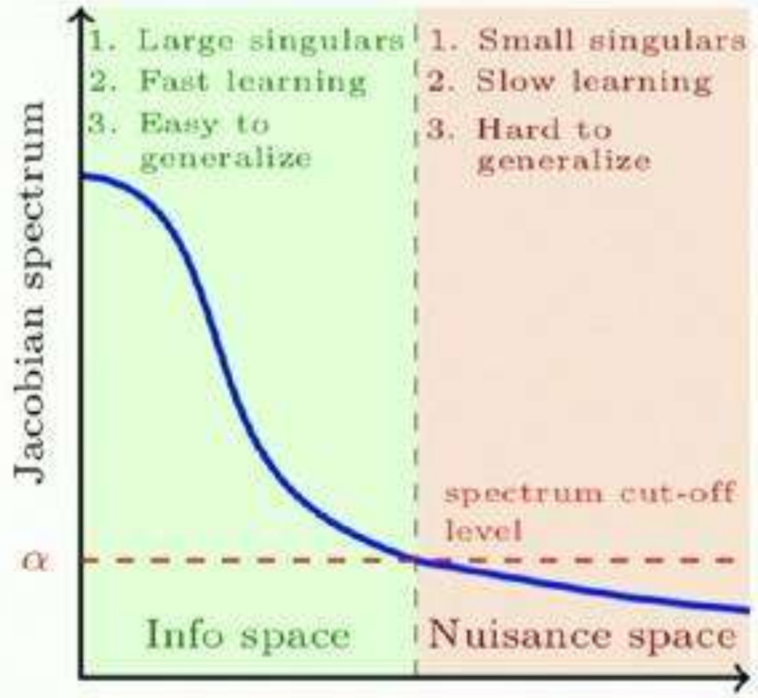
$$\nabla \mathcal{L}(\boldsymbol{W}) = \mathcal{J}^T(\boldsymbol{W})\,(f(\boldsymbol{W}) - \boldsymbol{y}) \quad \text{with} \quad \mathcal{J}(\boldsymbol{W}) = \frac{\partial f(\boldsymbol{W})}{\partial \mathsf{vect}(\boldsymbol{W})}.$$

Information and nuisance space of the Jacobian

Jacobian $\boldsymbol{J} \in \mathbb{R}^{nK \times p}$

$$\boldsymbol{J} = \sum_{s=1}^{nK} \lambda_s \boldsymbol{u}_s \boldsymbol{v}_s^T = \boldsymbol{U}\, diag\,(\lambda_1, \lambda_2, \ldots, \lambda_{nK})\, \boldsymbol{V}^T$$

- *Information space:* $\mathcal{I} =$*span*$(\{\boldsymbol{u}_s\}_{s=1}^r)$
- *nuisance space:* $\mathcal{N} =$*span*$(\{\boldsymbol{u}_s\}_{s=r+1}^n)$



Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theorem

Fix number $\bar{\alpha} \geq 0$ and $\Gamma \geq 1$.

- *Set $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ and $\mathcal{I}$ based on cut-off $\alpha = \sqrt{nK}\bar{\alpha}$*
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ iterations

$$missclass\left(f(\boldsymbol{W}_T)\right) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(\boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

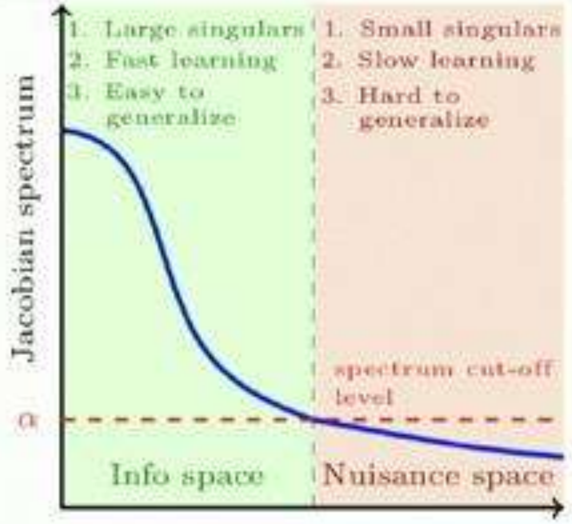
Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(\boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$



Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\left(f(\boldsymbol{W}_T)\right) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(\boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

- Structured datasets generalize easier and require smaller networks.

Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\left(f(\boldsymbol{W}_T)\right) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(\boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

- Structured datasets generalize easier and require smaller networks.
- with constant $\bar{\alpha}$, constant iterations and width is sufficient for learning.

Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK\log(K)}}$ and initialize $\boldsymbol{W}_0$ i.i.d. $\mathcal{N}(0,1)$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \left(\mathbb{E}[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)]\right)^{\frac{1}{2}}$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\left(f(\boldsymbol{W}_T)\right) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(\boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

- Structured datasets generalize easier and require smaller networks.
- with constant $\bar{\alpha}$, constant iterations and width is sufficient for learning.
- Picking cut-off small ($\mathcal{I} = \mathbb{R}^n$) and $K = 1$ improves upon [Arora et. al.]

$$\text{missclass error} \lesssim \sqrt{\frac{\boldsymbol{y}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}\boldsymbol{y}}{n}} \quad \text{with} \quad k \gtrsim \frac{n^4 \log n}{\lambda_{\min}^4\left(\boldsymbol{J}\boldsymbol{J}^T\right)}$$

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \mathcal{J}(\boldsymbol{W}_0)$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(f(\boldsymbol{W}_0) - \boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theorem

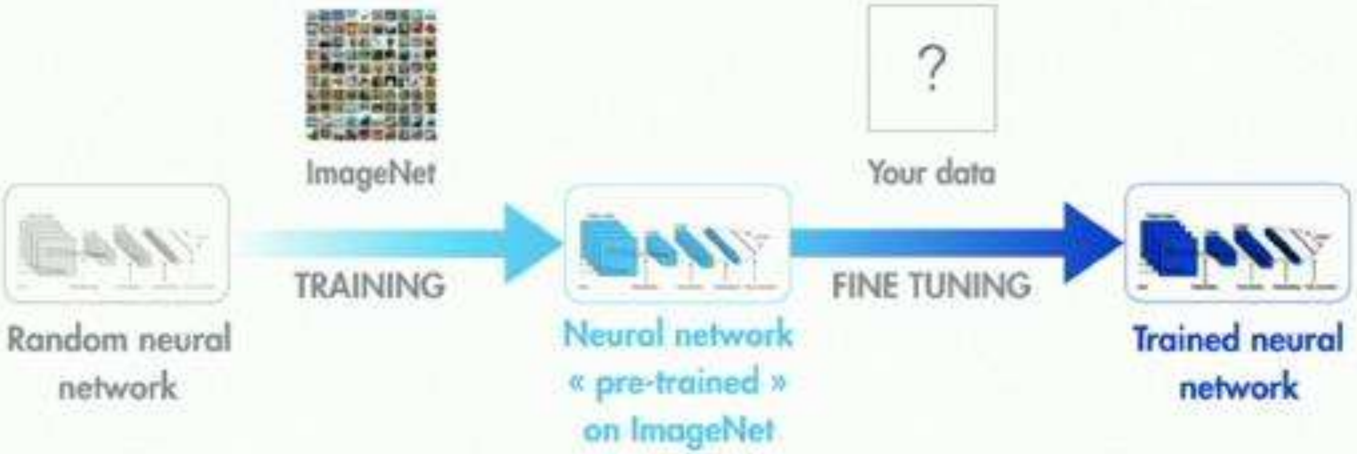
Fix number $\bar{\alpha} \geq 0$ and $\Gamma \geq 1$.

- *Set $\boldsymbol{J} = \mathcal{J}(\boldsymbol{W}_0)$ and $\mathcal{I}$ based on cut-off $\alpha = \sqrt{nK}\bar{\alpha}$*
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ iterations

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(f(\boldsymbol{W}_0) - \boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

- Applies to pre-trained models e.g. meta/transfer learning



Question: What can we say about pretrained networks vs random?

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \mathcal{J}(\boldsymbol{W}_0)$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(f(\boldsymbol{W}_0) - \boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theorem

Fix number $\bar{\alpha} \geq 0$ *and* $\Gamma \geq 1$.

- *Set* $\boldsymbol{J} = \mathcal{J}(\boldsymbol{W}_0)$ *and* $\mathcal{I}$ *based on cut-off* $\alpha = \sqrt{nK}\bar{\alpha}$
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ *iterations*

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(f(\boldsymbol{W}_0) - \boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

- Applies to any iteration $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{W}_\tau)$

Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $\boldsymbol{W}_0$

Theorem

Fix number $\bar{\alpha} \geq 0$ and $\Gamma \geq 1$.

- *Set $\boldsymbol{J} = \mathcal{J}(\boldsymbol{W}_0)$ and $\mathcal{I}$ based on cut-off $\alpha = \sqrt{nK}\bar{\alpha}$*
- $k \gtrsim \frac{\Gamma^4}{\bar{\alpha}^8}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^8}$ iterations

$$missclass\,(f(\boldsymbol{W}_T)) \lesssim \underbrace{e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(f(\boldsymbol{W}_0) - \boldsymbol{y})}{\sqrt{n}}}_{bias} + \underbrace{\frac{\Gamma}{\bar{\alpha}}\frac{1}{\sqrt{n}}}_{variance}$$

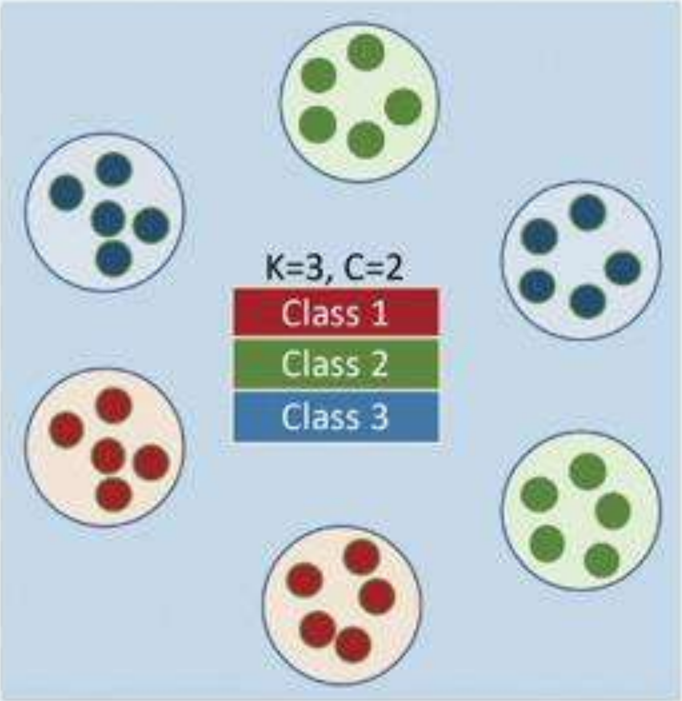
- Applies to any iteration $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{W}_\tau)$
- A step towards kernel adaptation $\mathcal{K}_\tau = \boldsymbol{J}(\boldsymbol{W}_\tau)\boldsymbol{J}^T(\boldsymbol{W}_\tau)$

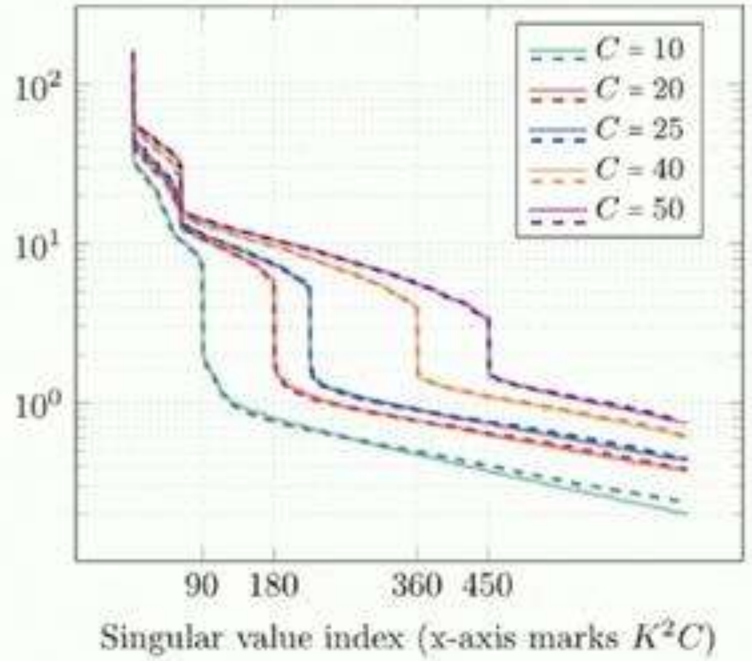
$$\mathcal{K}_0 \rightarrow \mathcal{K}_1 \rightarrow \mathcal{K}_2 \rightarrow \ldots \rightarrow$$

see [Bach and Chizat 2018], [Mei and Montanari], [Yang], [Soudry et. al.] and many others

Concrete example: Gaussian Mixture Model (GMM)

Data set a GMM with K classes each containing C components per class with small σ^2





Theorem

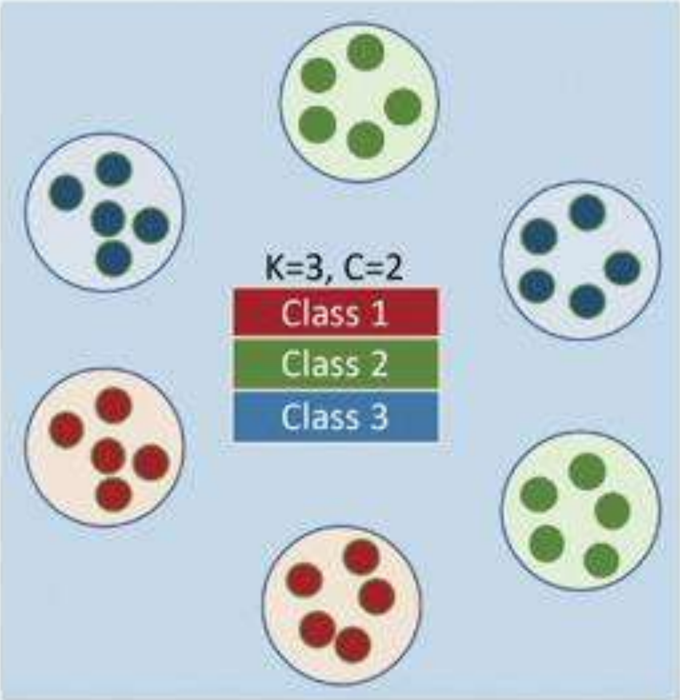
With high probability

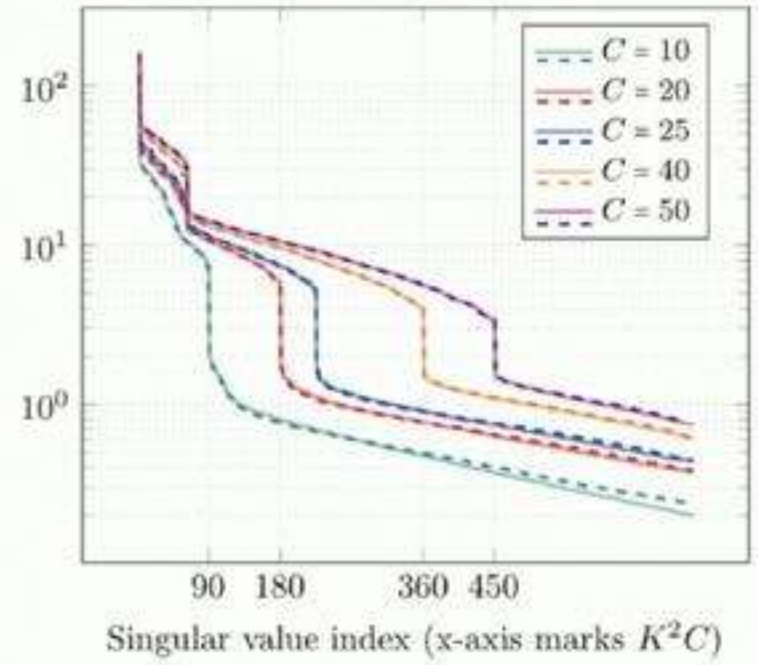
- *Jacobian has* K^2C *large singular values that grow* $\propto \sqrt{n}$
- *If* $k \gtrsim \Gamma^4 K^8 C^4$ *after* $T \propto \Gamma K^2 C$ *iterations,*

$$misclass\,(f(\boldsymbol{W}_T)) \lesssim \Gamma\sqrt{\frac{K^2C}{n}} + e^{-\Gamma}$$

Concrete example: Gaussian Mixture Model (GMM)

Data set a GMM with K classes each containing C components per class with small σ^2





Theorem

With high probability

- *Jacobian has* K^2C *large singular values that grow* $\propto \sqrt{n}$
- *If* $k \gtrsim \Gamma^4 K^8 C^4$ *after* $T \propto \Gamma K^2 C$ *iterations,*

$$misclass\,(f(\boldsymbol{W}_T)) \lesssim \Gamma \sqrt{\frac{K^2C}{n}} + e^{-\Gamma}$$

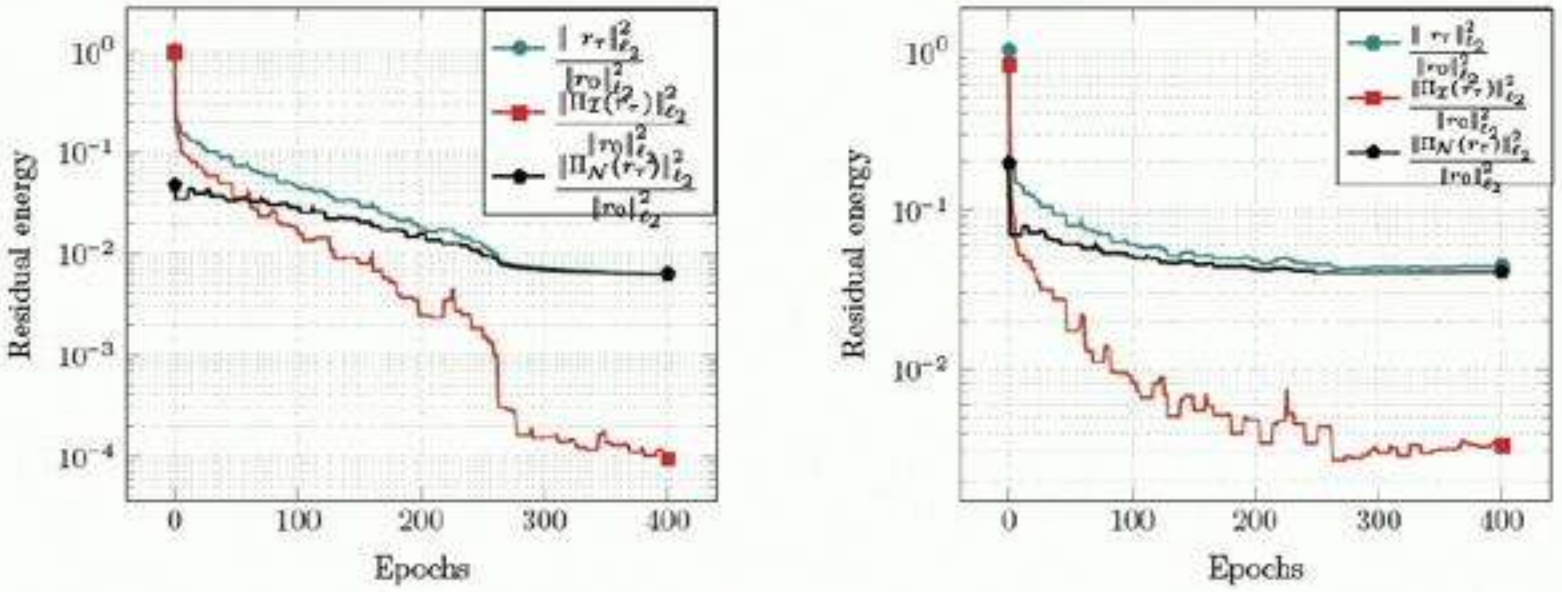
[Arora et. al. 2019] $k \to \infty$ as $\sigma \to 0$

Numerical experiments

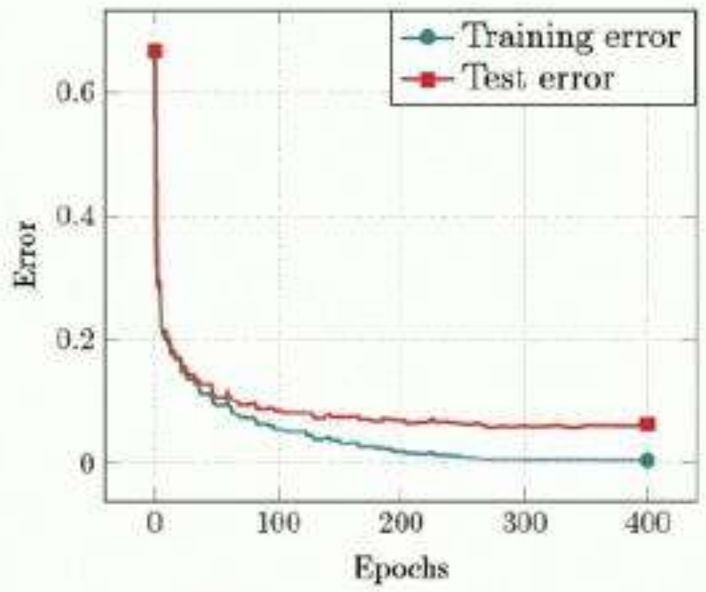
No label corruption

$$K = 3 \text{ classes}, \quad \dim(\mathcal{I}) = 50, \quad n = 10,000$$

Consider evolution of residual $\boldsymbol{r}_\tau = f(\boldsymbol{W}_\tau) - \boldsymbol{y}$



Residual along the information/nuisance spaces of the final Jacobian using (a) train data and (b) test data



Train and test error

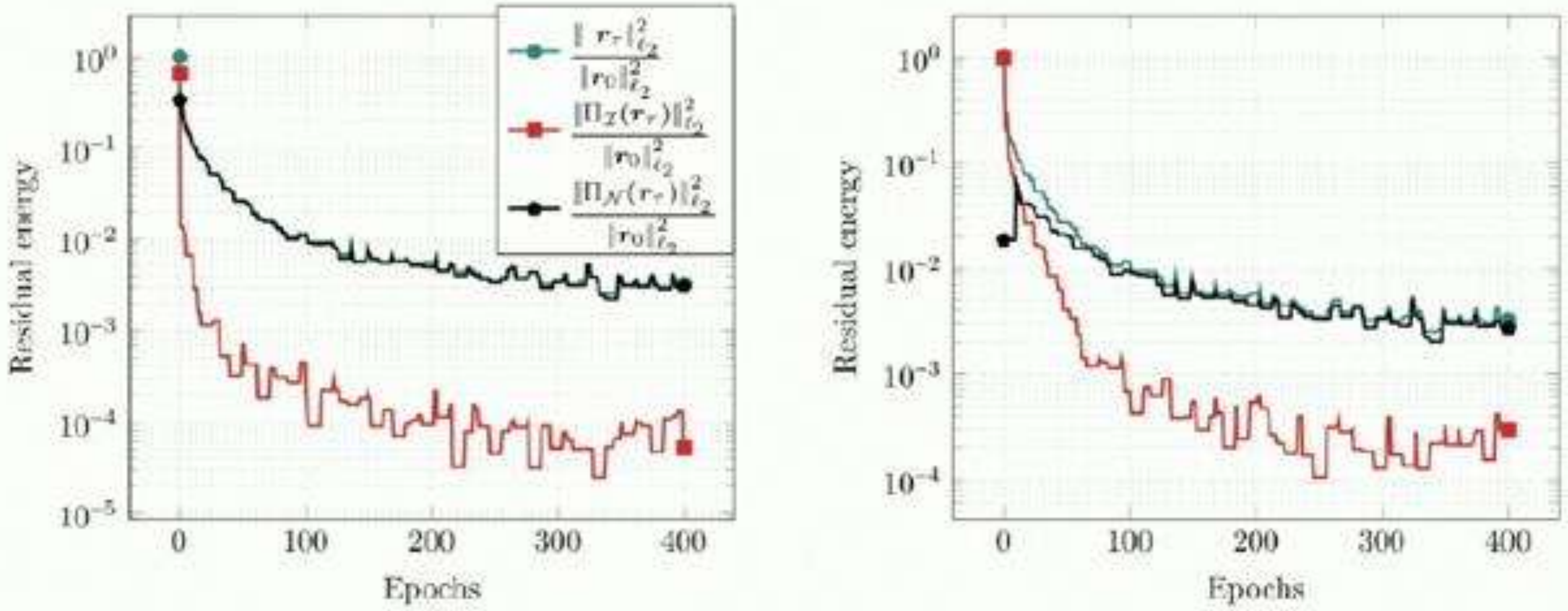
	$\frac{\lVert\Pi_{\mathcal{I}}(\boldsymbol{y})\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{N}}(\boldsymbol{y})\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{y}\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{I}}(\boldsymbol{r}_0)\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{N}}(\boldsymbol{r}_0)\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$	$\frac{\lVert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{r}_0\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$
$\boldsymbol{J}_{init}$	0.724	0.690	$5.44 \cdot 10^{-3}$	0.886	0.465	$4.10 \cdot 10^{-3}$
$\boldsymbol{J}_{final}$	0.987	0.158	$3.16 \cdot 10^{-3}$	0.976	0.217	$3.43 \cdot 10^{-3}$

Table 1: Depiction of the alignment of the initial label/residual with the information/nuisance space using uncorrupted data and a Multi-class ResNet20 model trained with SGD.

No label corruption with ADAM

$$K = 3 \text{ classes}, \quad \dim(\mathcal{I}) = 50, \quad n = 10,000$$

Consider evolution of residual $\boldsymbol{r}_\tau = f(\boldsymbol{W}_\tau) - \boldsymbol{y}$
green (total), red (on $\mathcal{I}$), black (on $\mathcal{N}$)



Residual along the information/nuisance spaces of the
(a) initial and (b) final Jacobian using ADAM

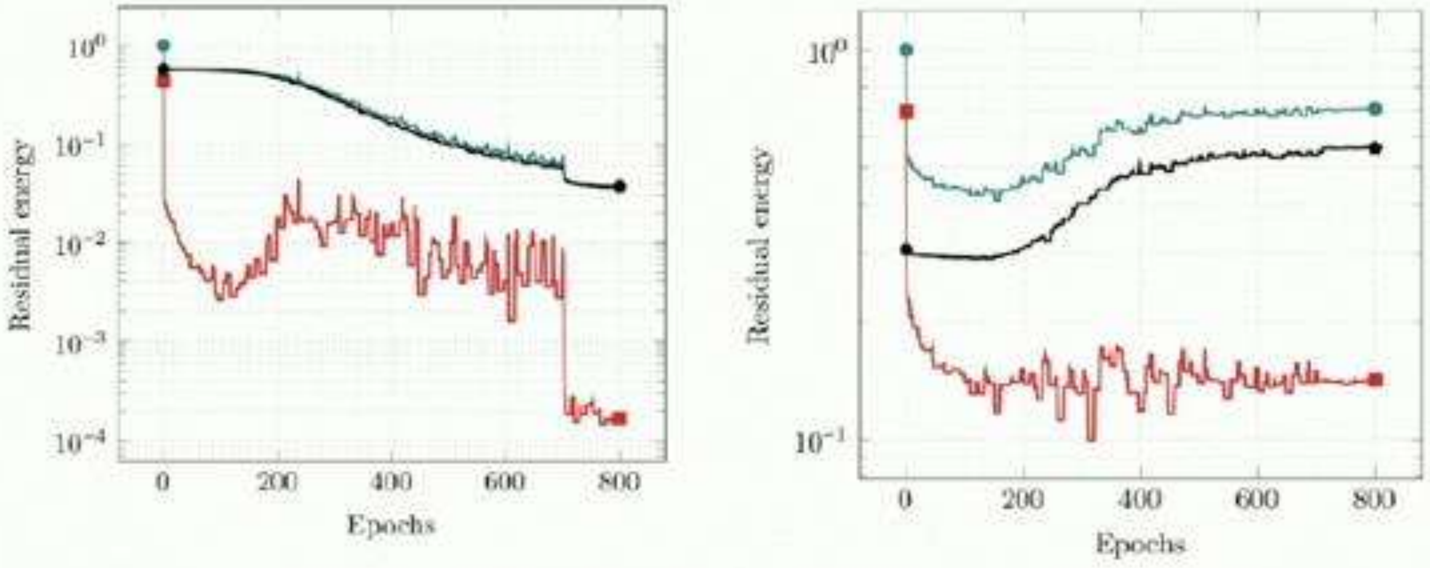
	$\frac{\lVert\Pi_{\mathcal{I}}(\boldsymbol{y})\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{N}}(\boldsymbol{y})\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{y}\rVert_{\ell_2}}{\lVert\boldsymbol{y}\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{I}}(\boldsymbol{r}_0)\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$	$\frac{\lVert\Pi_{\mathcal{N}}(\boldsymbol{r}_0)\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$	$\frac{\lVert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{r}_0\rVert_{\ell_2}}{\lVert\boldsymbol{r}_0\rVert_{\ell_2}}$
$\boldsymbol{J}_{init}$	0.702	0.712	$5.36 \cdot 10^{-3}$	0.814	0.582	$4.43 \cdot 10^{-3}$
$\boldsymbol{J}_{final}$	0.997	0.078	$3.10 \cdot 10^{-3}$	0.991	0.136	$3.06 \cdot 10^{-3}$

Table 2: Depiction of the alignment of the initial label/residual with the information/nuisance space using uncorrupted data and a Multi-class ResNet20 model trained with Adam.

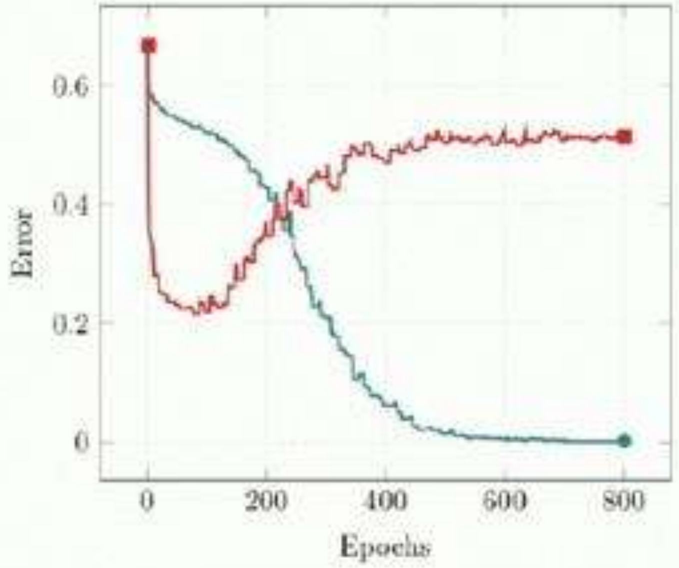
50% label corruption

$$K = 3 \text{ classes}, \quad \dim(\mathcal{I}) = 50, \quad n = 10,000$$

Consider evolution of residual $\boldsymbol{r}_\tau = f(\boldsymbol{W}_\tau) - \boldsymbol{y}$



Residual along the information/nuisance spaces of the final Jacobian using (a) train data and (b) test data

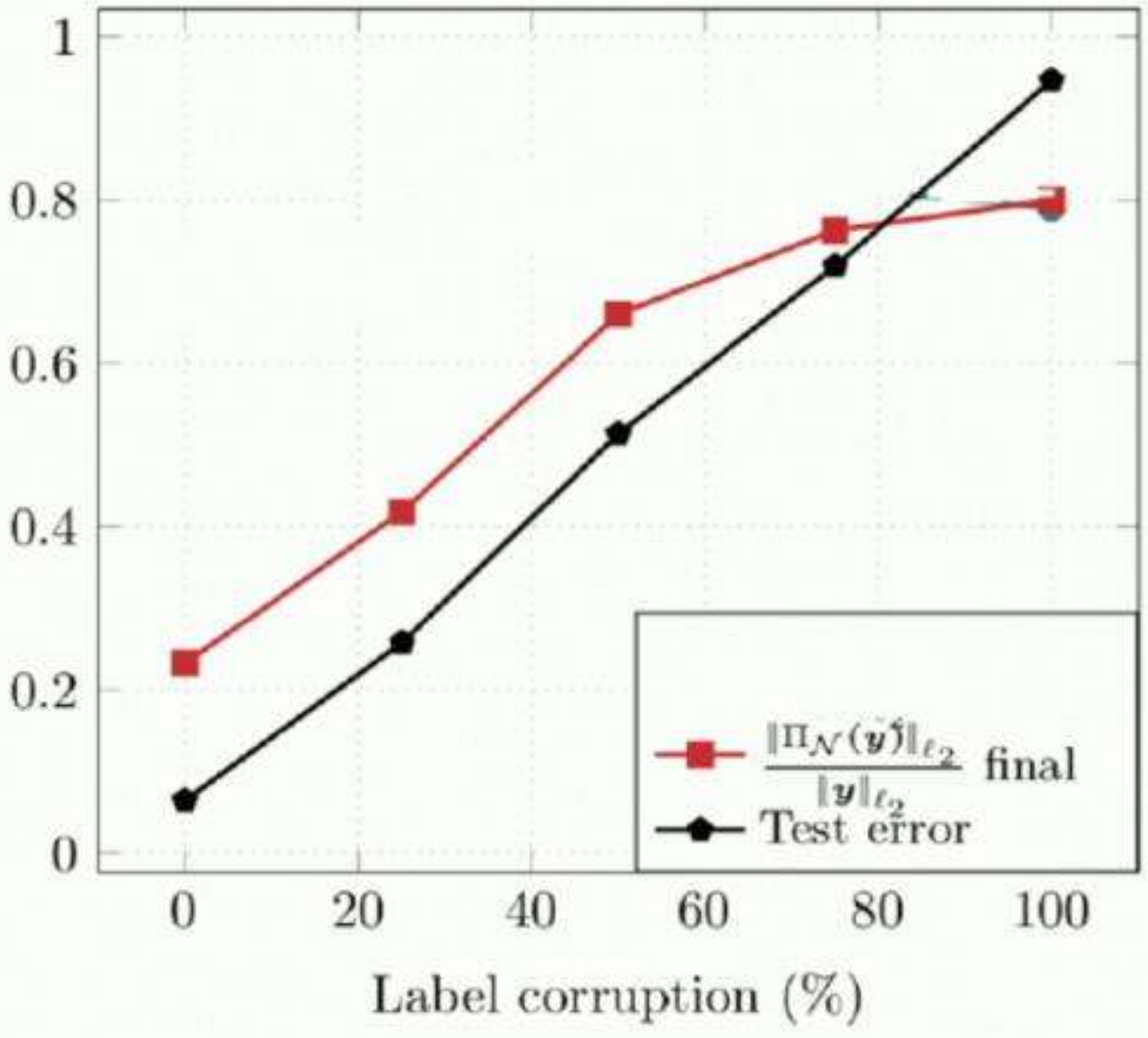


Train and test error

	$\frac{\Vert\Pi_{\mathcal{I}}(\boldsymbol{y})\Vert_{\ell_2}}{\Vert\boldsymbol{y}\Vert_{\ell_2}}$	$\frac{\Vert\Pi_{\mathcal{N}}(\boldsymbol{y})\Vert_{\ell_2}}{\Vert\boldsymbol{y}\Vert_{\ell_2}}$	$\frac{\Vert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{y}\Vert_{\ell_2}}{\Vert\boldsymbol{y}\Vert_{\ell_2}}$	$\frac{\Vert\Pi_{\mathcal{I}}(\boldsymbol{r}_0)\Vert_{\ell_2}}{\Vert\boldsymbol{r}_0\Vert_{\ell_2}}$	$\frac{\Vert\Pi_{\mathcal{N}}(\boldsymbol{r}_0)\Vert_{\ell_2}}{\Vert\boldsymbol{r}_0\Vert_{\ell_2}}$	$\frac{\Vert\boldsymbol{J}_{\mathcal{I}}^{\dagger}\boldsymbol{r}_0\Vert_{\ell_2}}{\Vert\boldsymbol{r}_0\Vert_{\ell_2}}$
$\boldsymbol{J}_{init}$	0.587	0.810	$1.72 \cdot 10^{-3}$	0.643	0.766	$1.98 \cdot 10^{-3}$
$\boldsymbol{J}_{final}$	0.751	0.660	$1.87 \cdot 10^{-3}$	0.763	0.646	$1.20 \cdot 10^{-3}$

Table 3: Depiction of the alignment of the initial label/residual with the information/nuisance space using 50% label corrupted data and a Multi-class ResNet20 model trained with SGD.

Test error grows in tandem with energy on nuisance space

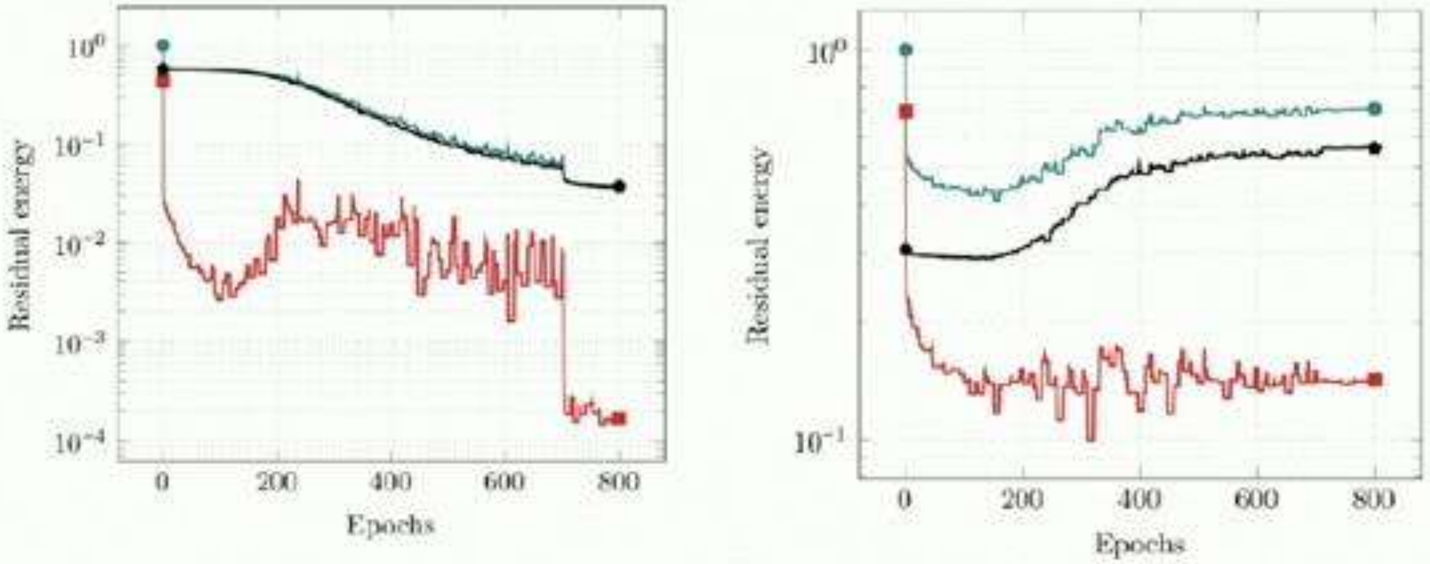


Fraction of the energy of the label vector
as a function of the amount of label corruption.

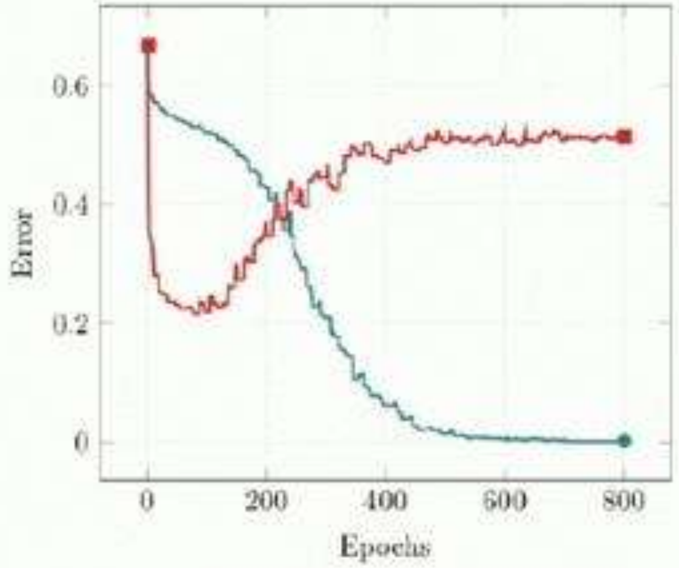
50% label corruption

$$K = 3 \text{ classes}, \quad \dim(\mathcal{I}) = 50, \quad n = 10,000$$

Consider evolution of residual $\boldsymbol{r}_\tau = f(\boldsymbol{W}_\tau) - \boldsymbol{y}$



Residual along the information/nuisance spaces of the final Jacobian using (a) train data and (b) test data

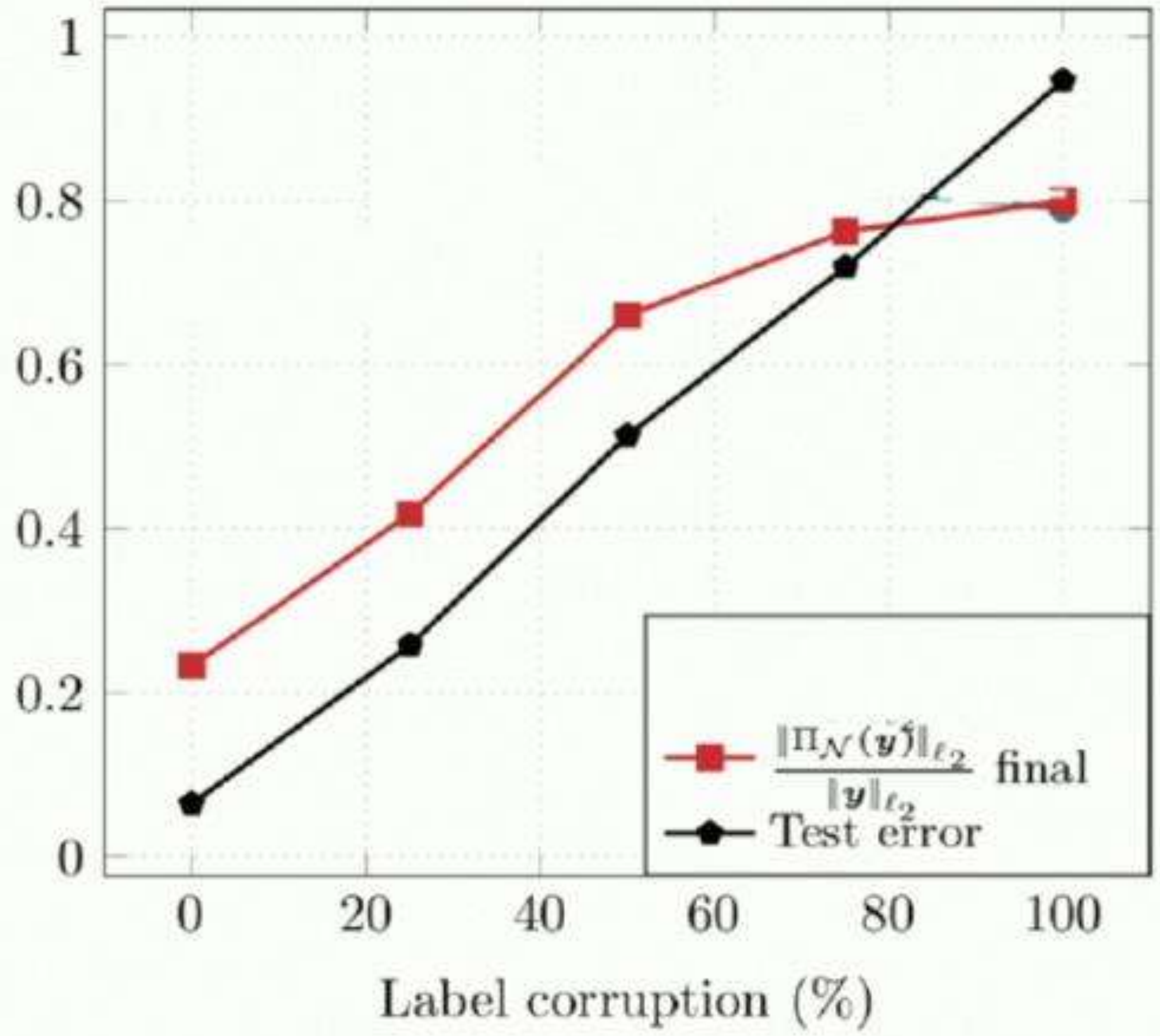


Train and test error

	$\frac{\lVert \Pi_{\mathcal{I}}(\boldsymbol{y}) \rVert_{\ell_2}}{\lVert \boldsymbol{y} \rVert_{\ell_2}}$	$\frac{\lVert \Pi_{\mathcal{N}}(\boldsymbol{y}) \rVert_{\ell_2}}{\lVert \boldsymbol{y} \rVert_{\ell_2}}$	$\frac{\lVert \boldsymbol{J}_{\mathcal{I}}^{\dagger} \boldsymbol{y} \rVert_{\ell_2}}{\lVert \boldsymbol{y} \rVert_{\ell_2}}$	$\frac{\lVert \Pi_{\mathcal{I}}(\boldsymbol{r}_0) \rVert_{\ell_2}}{\lVert \boldsymbol{r}_0 \rVert_{\ell_2}}$	$\frac{\lVert \Pi_{\mathcal{N}}(\boldsymbol{r}_0) \rVert_{\ell_2}}{\lVert \boldsymbol{r}_0 \rVert_{\ell_2}}$	$\frac{\lVert \boldsymbol{J}_{\mathcal{I}}^{\dagger} \boldsymbol{r}_0 \rVert_{\ell_2}}{\lVert \boldsymbol{r}_0 \rVert_{\ell_2}}$
$\boldsymbol{J}_{init}$	0.587	0.810	$1.72 \cdot 10^{-3}$	0.643	0.766	$1.98 \cdot 10^{-3}$
$\boldsymbol{J}_{final}$	0.751	0.660	$1.87 \cdot 10^{-3}$	0.763	0.646	$1.20 \cdot 10^{-3}$

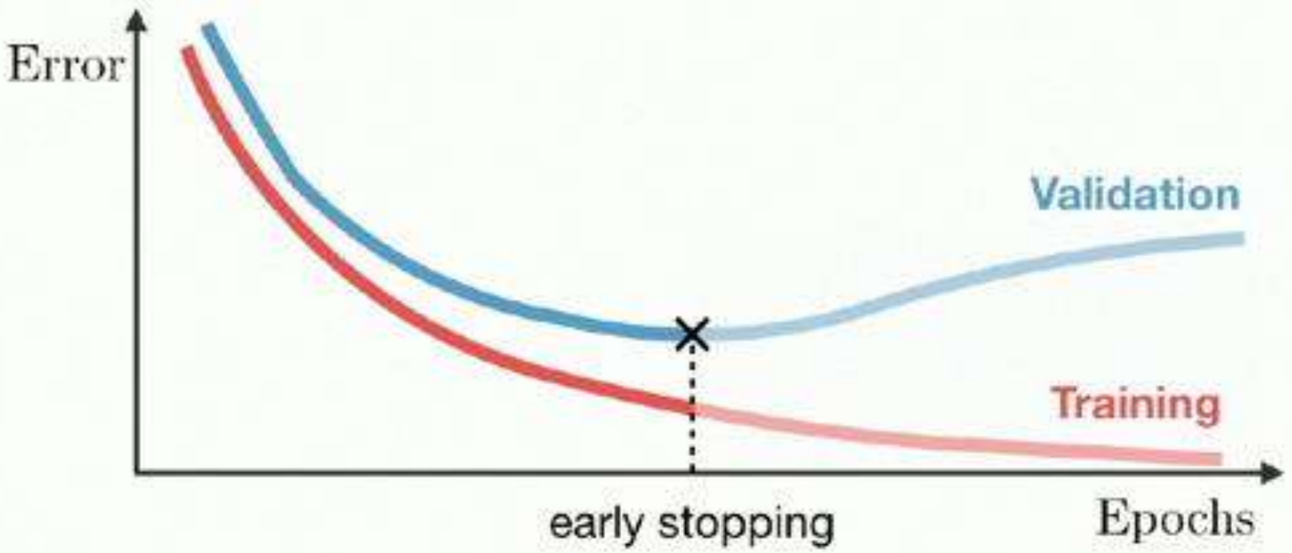
Table 3: Depiction of the alignment of the initial label/residual with the information/nuisance space using 50% label corrupted data and a Multi-class ResNet20 model trained with SGD.

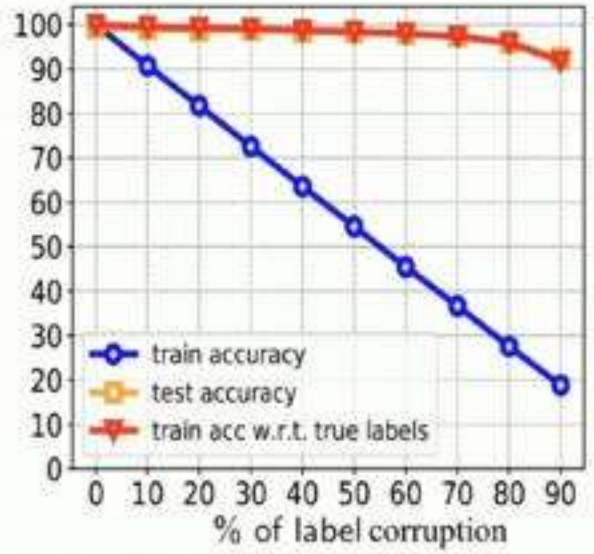
Test error grows in tandem with energy on nuisance space



Fraction of the energy of the label vector
as a function of the amount of label corruption.

Early stopping and robustness to label corruption



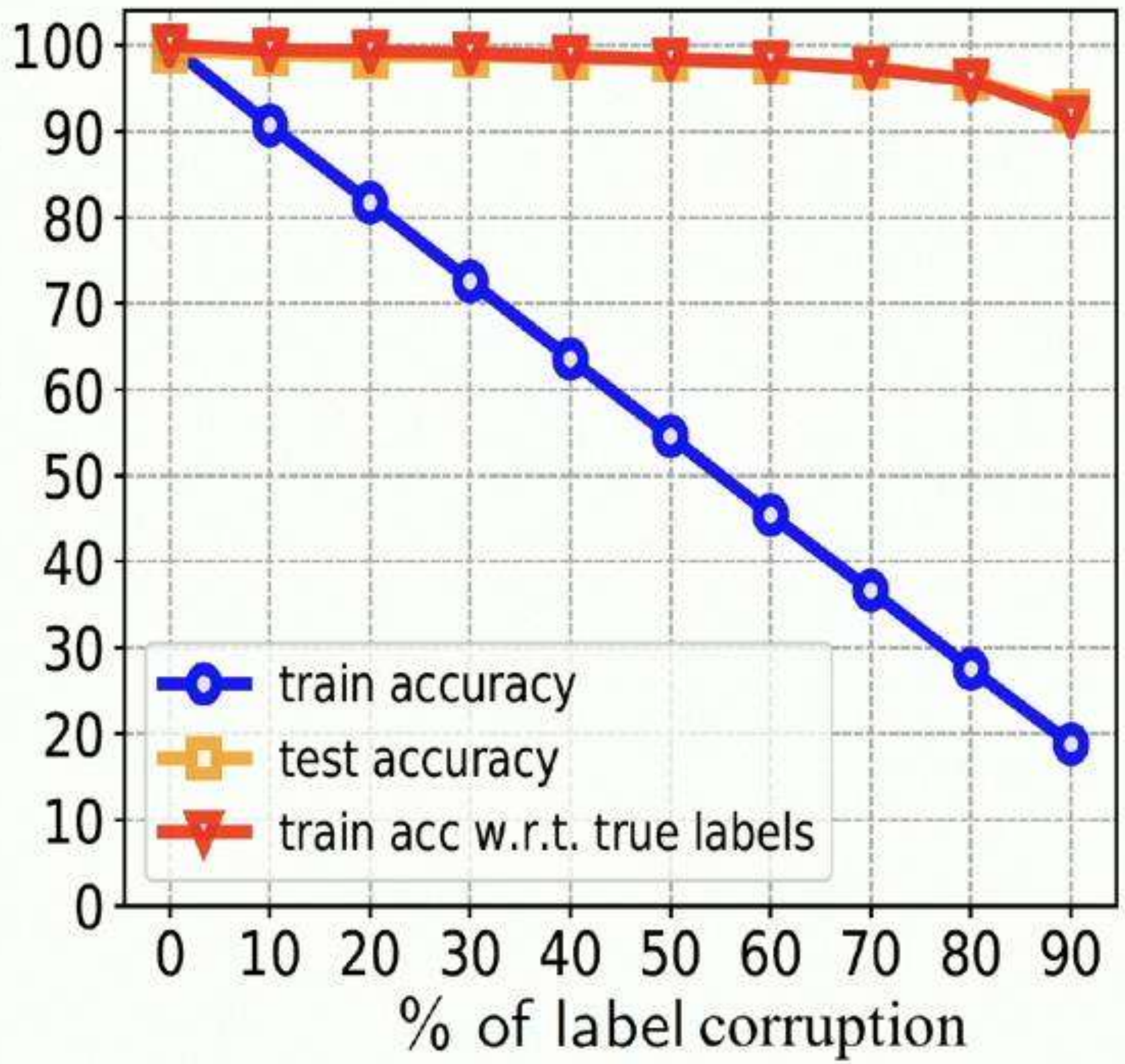


Experiment II-Early stopping and robustness

Repeat the same experiment but stop early

Experiment II-Early stopping and robustness

Repeat the same experiment but stop early

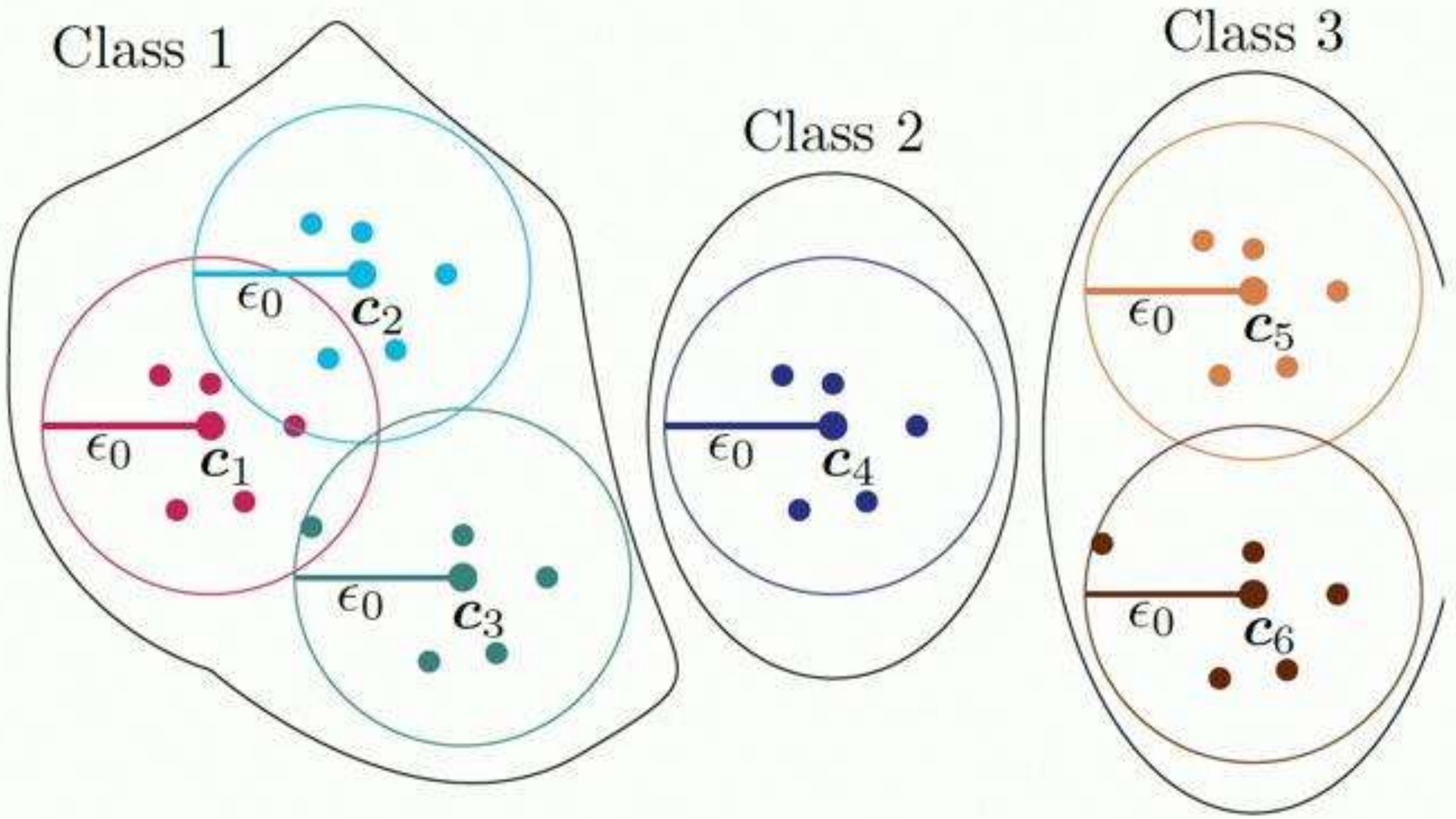


Model (without corruption)

clean data: clusterable data

input/label pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \in \mathbb{R}^d \times \mathbb{R}^K$

L clusters and K classes



Robustness to corruption

Clean data points $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$, corrupt $s := \rho n$ to get corrupted data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.

Robustness to corruption

Clean data points $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$, corrupt $s := \rho n$ to get corrupted data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
Fit

$$\mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\sum_{i=1}^{n} \|f(\boldsymbol{W}, \boldsymbol{x}_i) - \boldsymbol{y}_i\|_{\ell_2}^2$$

via gradient descent

Robustness to corruption

Clean data points $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$, corrupt $s := \rho n$ to get corrupted data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
Fit

$$\mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\sum_{i=1}^{n} \|f(\boldsymbol{W}, \boldsymbol{x}_i) - \boldsymbol{y}_i\|_{\ell_2}^2$$

via gradient descent

Theorem (Oymak and Soltanolkotabi 2019)

Assume

- *Corruption level* $\rho < \frac{1}{16}$
- *Overparameterization:* $\#parameters \gtrsim L^4$

Robustness to corruption

Clean data points $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$, corrupt $s := \rho n$ to get corrupted data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
Fit

$$\mathcal{L}(\boldsymbol{W}) := \frac{1}{2}\sum_{i=1}^{n} \|f(\boldsymbol{W}, \boldsymbol{x}_i) - \boldsymbol{y}_i\|_{\ell_2}^2$$

via gradient descent

Theorem (Oymak and Soltanolkotabi 2019)

Assume

- *Corruption level* $\rho < \frac{1}{16}$
- *Overparameterization:* $\#parameters \gtrsim L^4$

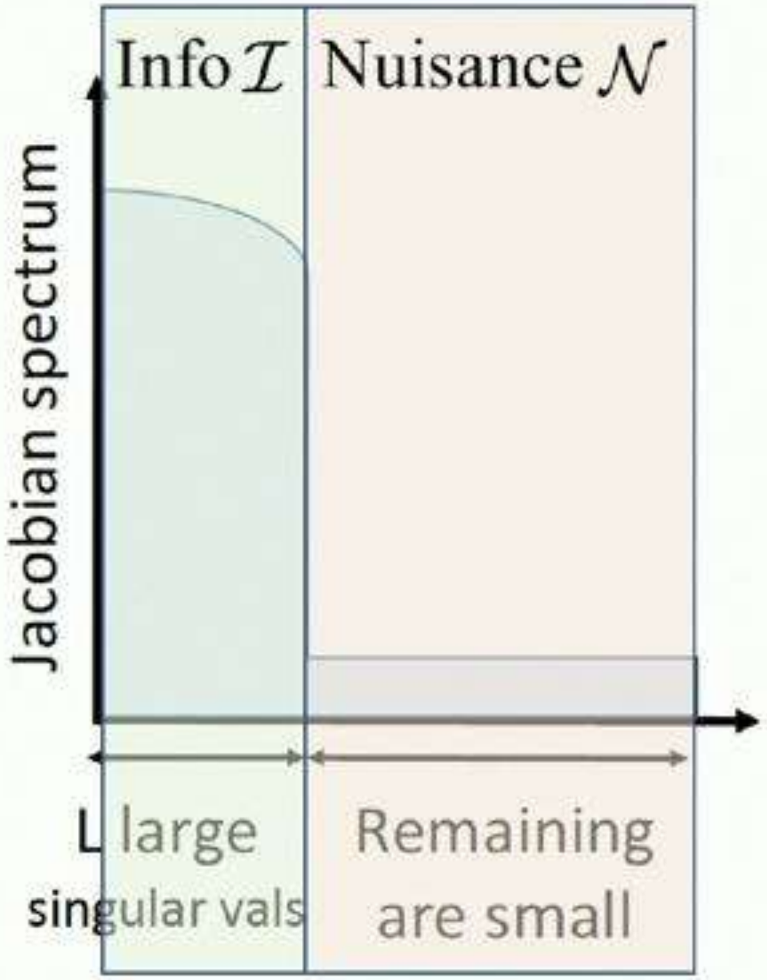
Starting from random initialization, after $\tau \sim L\log(1/\rho)$ *iterations, gradient descent finds a model with perfect accuracy, i.e.*

$$\textit{closest label to } f(\boldsymbol{W}_\tau, \boldsymbol{x}_i) = \textit{true label } \bar{y}_i$$

Key Intuition

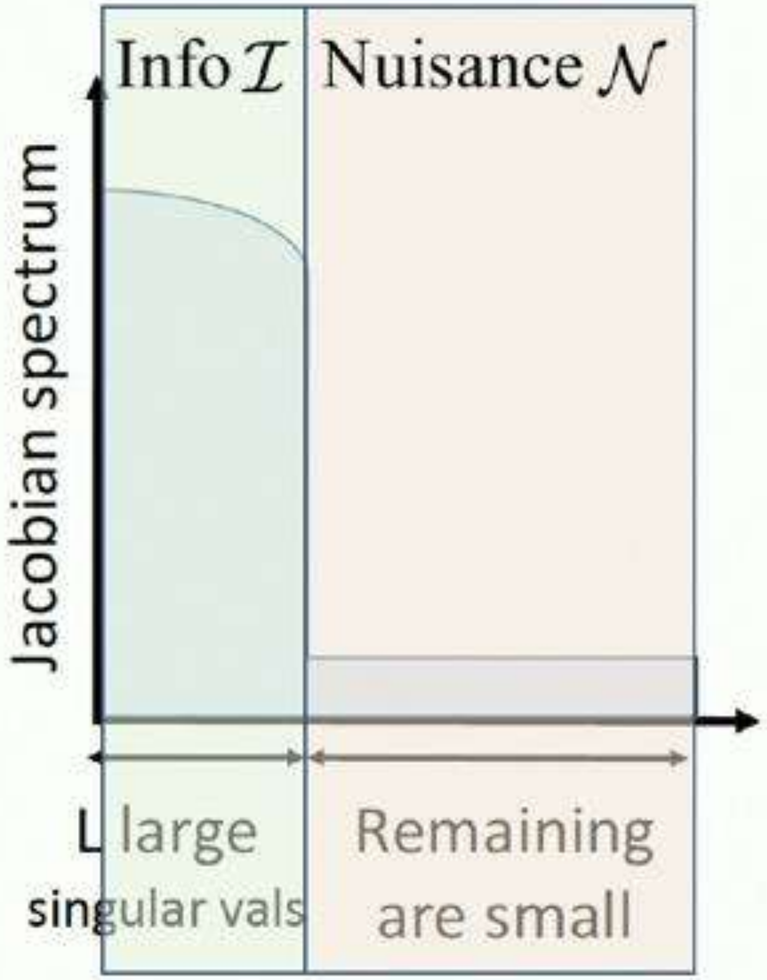
Key Intuition

Jacobian is low-rank

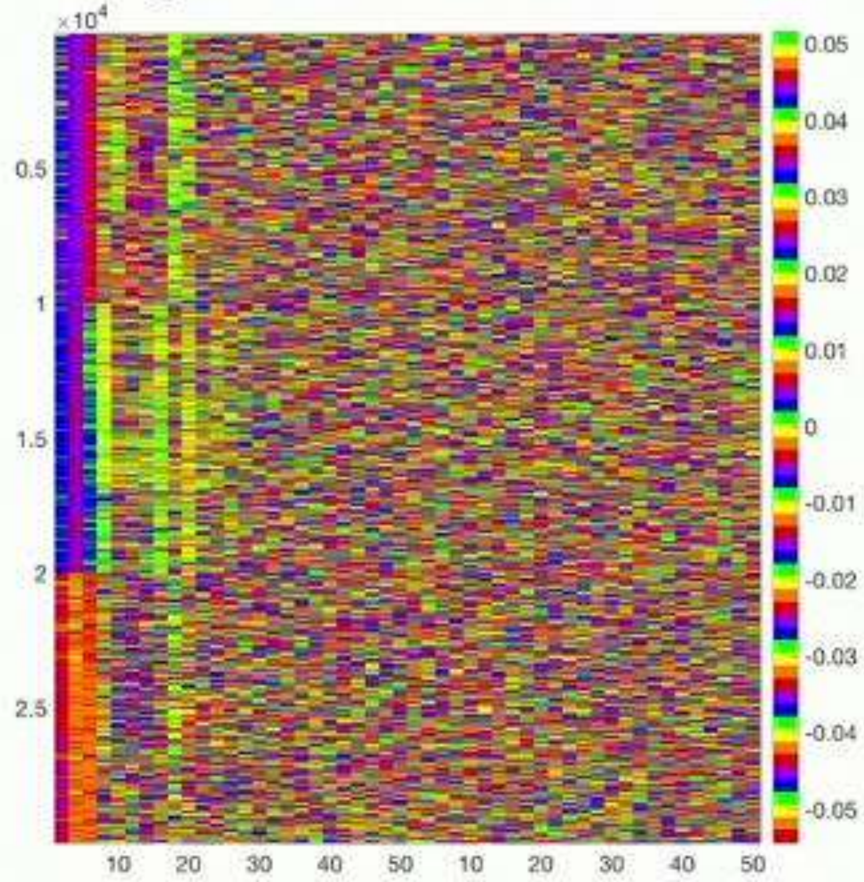


Key Intuition

Jacobian is low-rank

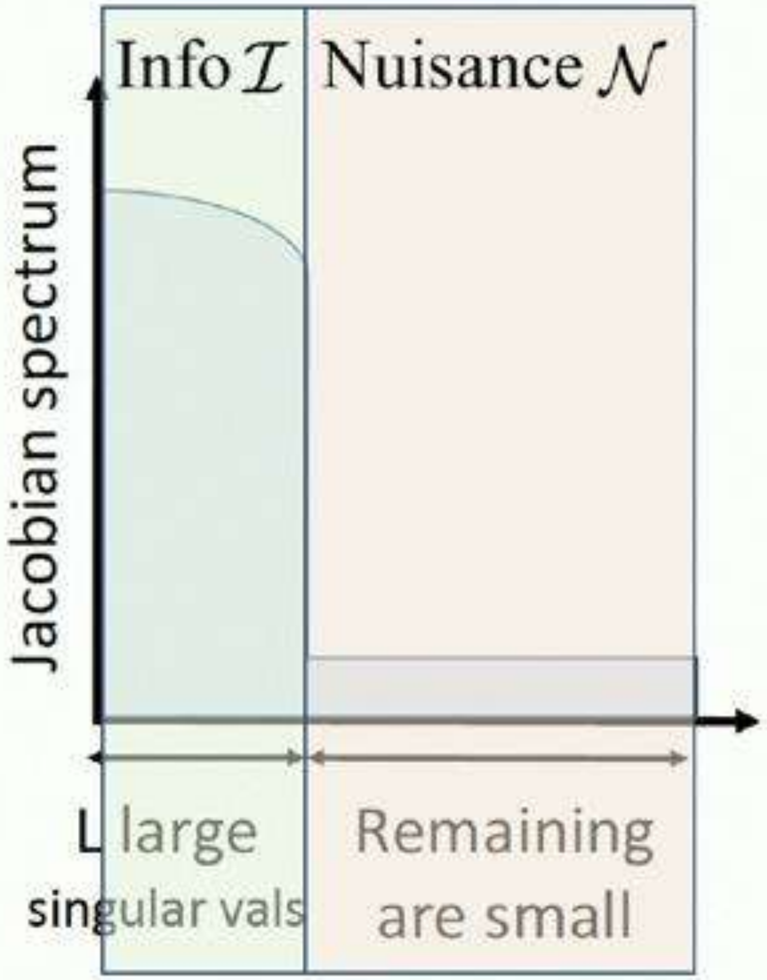


Prominent eigenvects of Jacobian are diffused

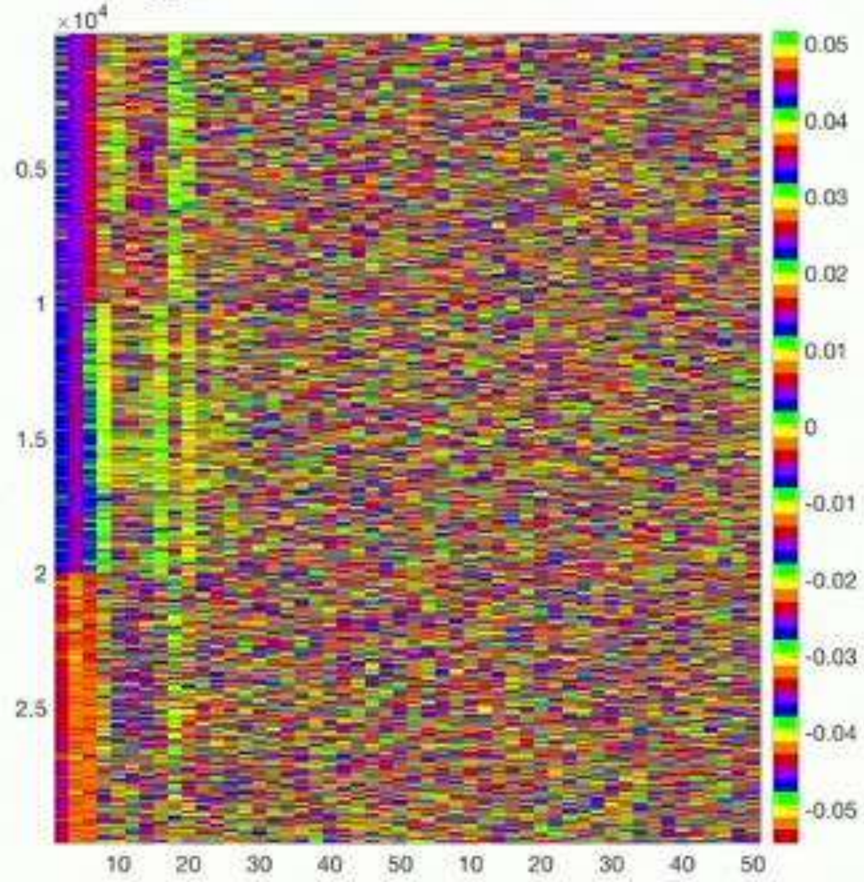


Key Intuition

Jacobian is low-rank



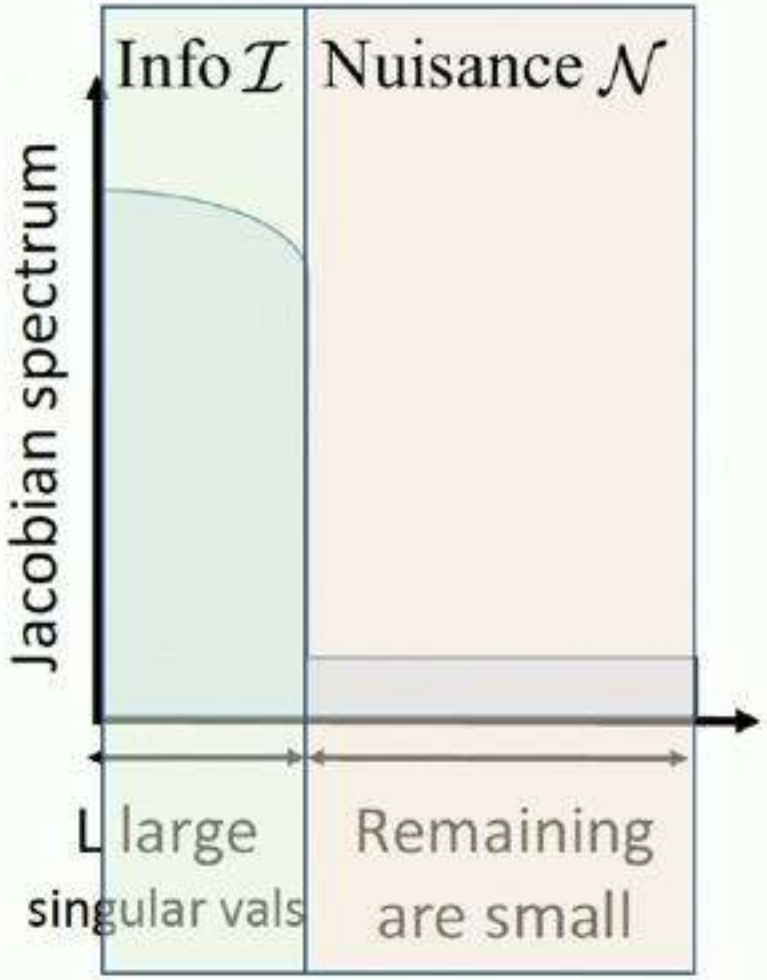
Prominent eigenvects of Jacobian are diffused



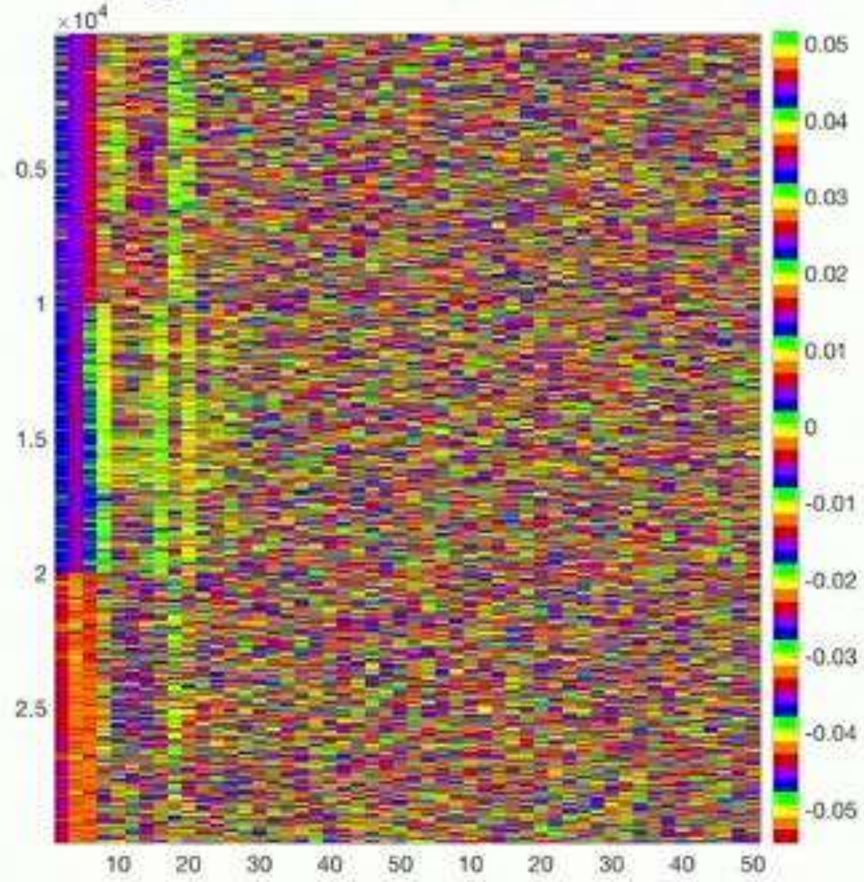
Interaction of Jacobian and residual $\nabla\mathcal{L}(\boldsymbol{\theta}) = \mathcal{J}^T(\boldsymbol{\theta})\,(f(\boldsymbol{\theta}, \boldsymbol{X}) - \boldsymbol{y})$

Key Intuition

Jacobian is low-rank



Prominent eigenvects of Jacobian are diffused



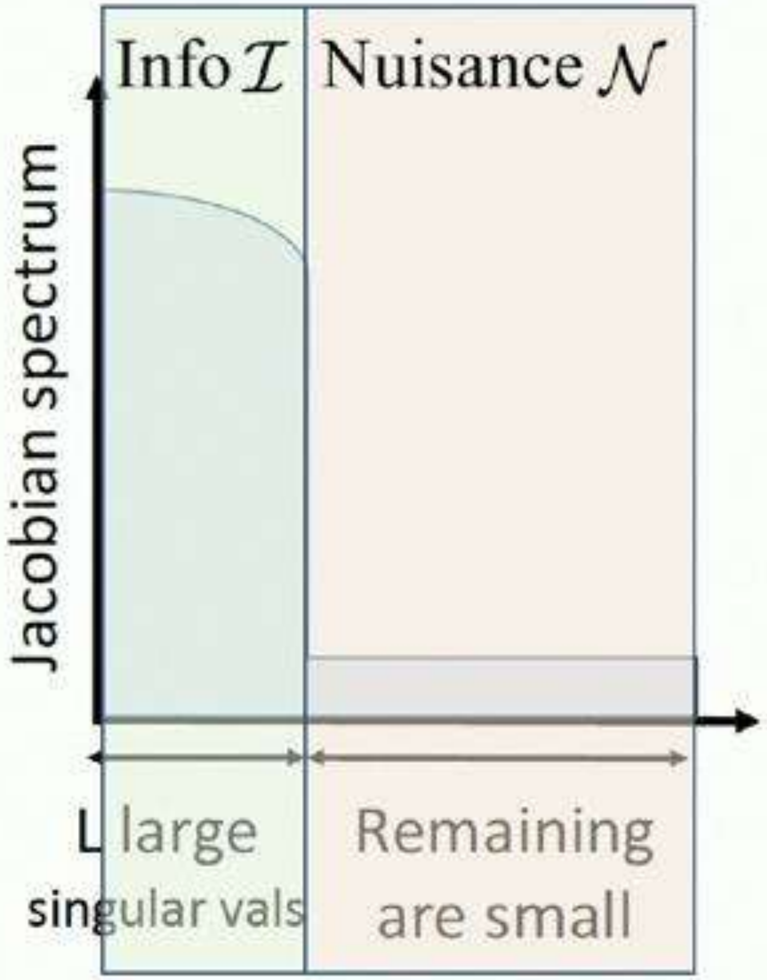
Interaction of Jacobian and residual $\nabla\mathcal{L}(\boldsymbol{\theta}) = \mathcal{J}^T(\boldsymbol{\theta})\,(f(\boldsymbol{\theta},\boldsymbol{X}) - \boldsymbol{y})$

Residual can be decomposed into two terms

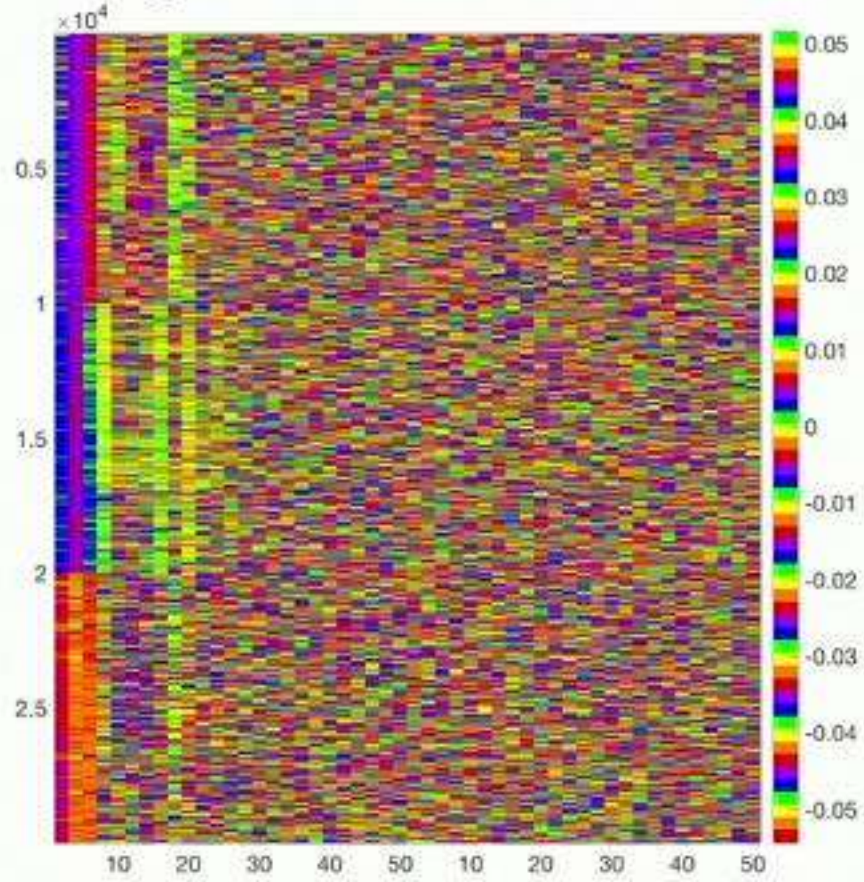
$$r(\boldsymbol{\theta}) := f(\boldsymbol{\theta},\boldsymbol{X}) - \boldsymbol{y} = \underbrace{f(\boldsymbol{\theta},\boldsymbol{X}) - \bar{\boldsymbol{y}}}_{\text{Residual w.r.t. true labels}} + \underbrace{\bar{\boldsymbol{y}} - \boldsymbol{y}}_{corruption}$$

Key Intuition

Jacobian is low-rank



Prominent eigenvects of Jacobian are diffused



Interaction of Jacobian and residual $\nabla \mathcal{L}(\boldsymbol{\theta}) = \mathcal{J}^T(\boldsymbol{\theta})\,(f(\boldsymbol{\theta}, \boldsymbol{X}) - \boldsymbol{y})$

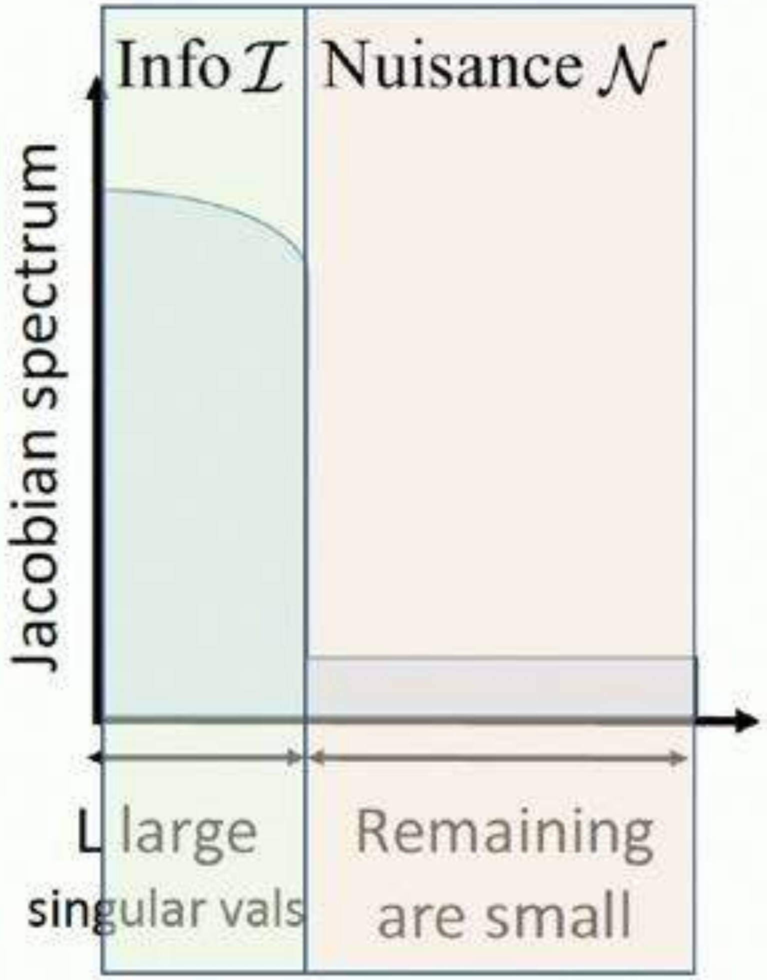
Residual can be decomposed into two terms

$$r(\boldsymbol{\theta}) := f(\boldsymbol{\theta}, \boldsymbol{X}) - \boldsymbol{y} = \underbrace{f(\boldsymbol{\theta}, \boldsymbol{X}) - \bar{\boldsymbol{y}}}_{\text{Residual w.r.t. true labels}} + \underbrace{\bar{\boldsymbol{y}} - \boldsymbol{y}}_{corruption}$$

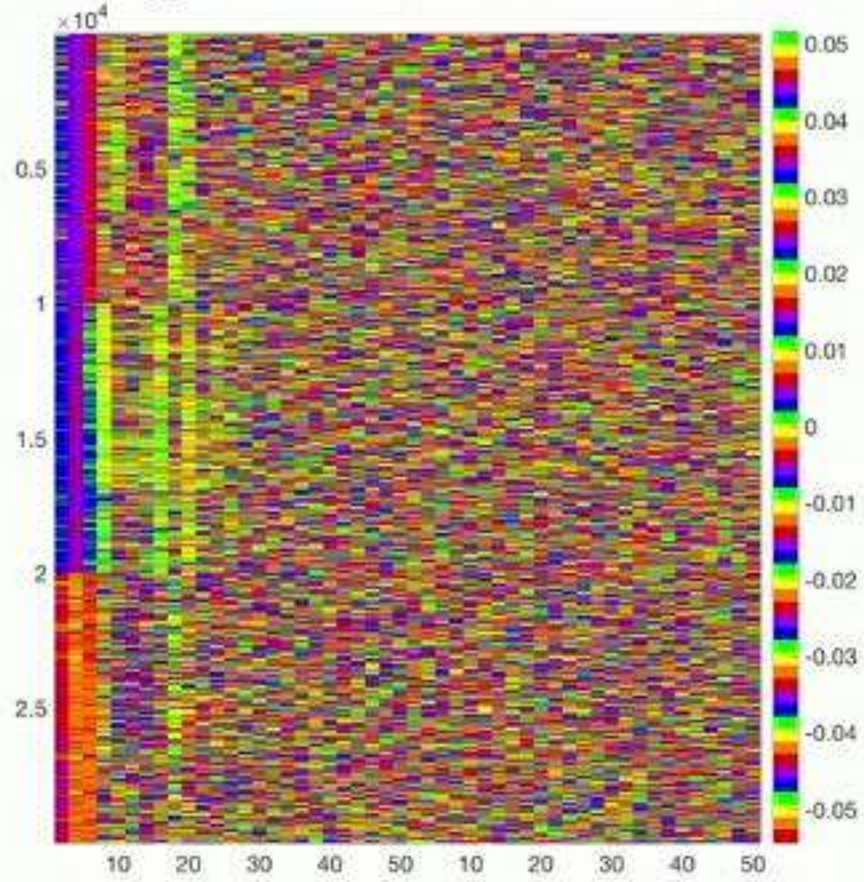
- Residual w.r.t. true labels falls mostly onto $\mathcal{I}$ and quickly goes to zero

Key Intuition

Jacobian is low-rank



Prominent eigenvects of Jacobian are diffused



Interaction of Jacobian and residual $\nabla \mathcal{L}(\boldsymbol{\theta}) = \mathcal{J}^T(\boldsymbol{\theta})\,(f(\boldsymbol{\theta}, \boldsymbol{X}) - \boldsymbol{y})$

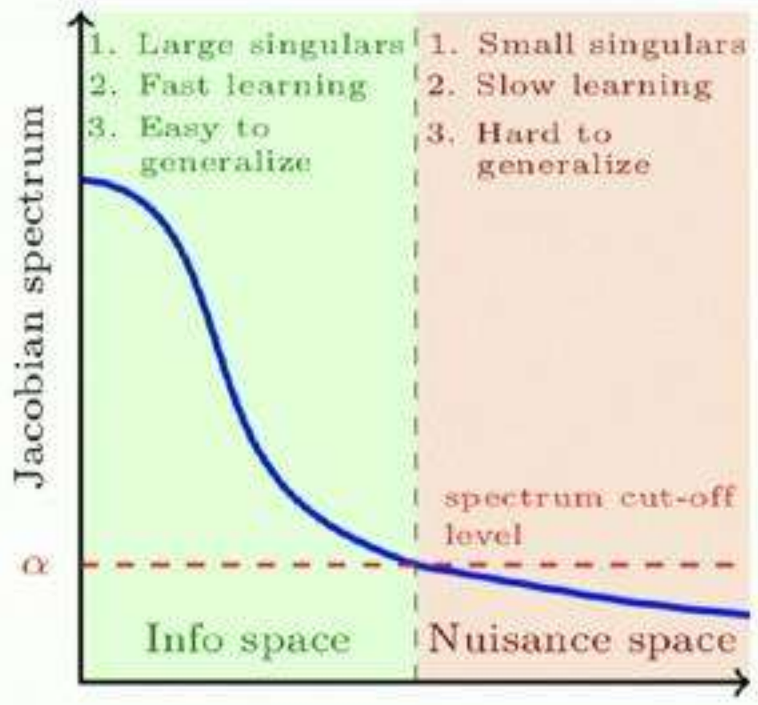
Residual can be decomposed into two terms

$$r(\boldsymbol{\theta}) := f(\boldsymbol{\theta}, \boldsymbol{X}) - \boldsymbol{y} = \underbrace{f(\boldsymbol{\theta}, \boldsymbol{X}) - \bar{\boldsymbol{y}}}_{\text{Residual w.r.t. true labels}} + \underbrace{\bar{\boldsymbol{y}} - \boldsymbol{y}}_{corruption}$$

- Residual w.r.t. true labels falls mostly onto $\mathcal{I}$ and quickly goes to zero
- Diffuseness $\Rightarrow$ corruption $\boldsymbol{y} - \bar{\boldsymbol{y}}$ falls mostly onto $\mathcal{N}$ and slowly goes to zero

Conclusion

- Global optimization: With modest overparameterization neural networks can fit any data
- Generalization: Neural networks can predict well on test data when the prominent eigenvectors of the Jacobian are aligned with the labels
- Early stopping and robustness to label corruption: Neural networks are robust to sparse label corruption when the Jacobian is low-rank and prominent eigenvectors are diffused



Thanks!

Funding acknowledgment

the David & Lucile Packard FOUNDATION

ALFRED P. SLOAN FOUNDATION 1934

NSF

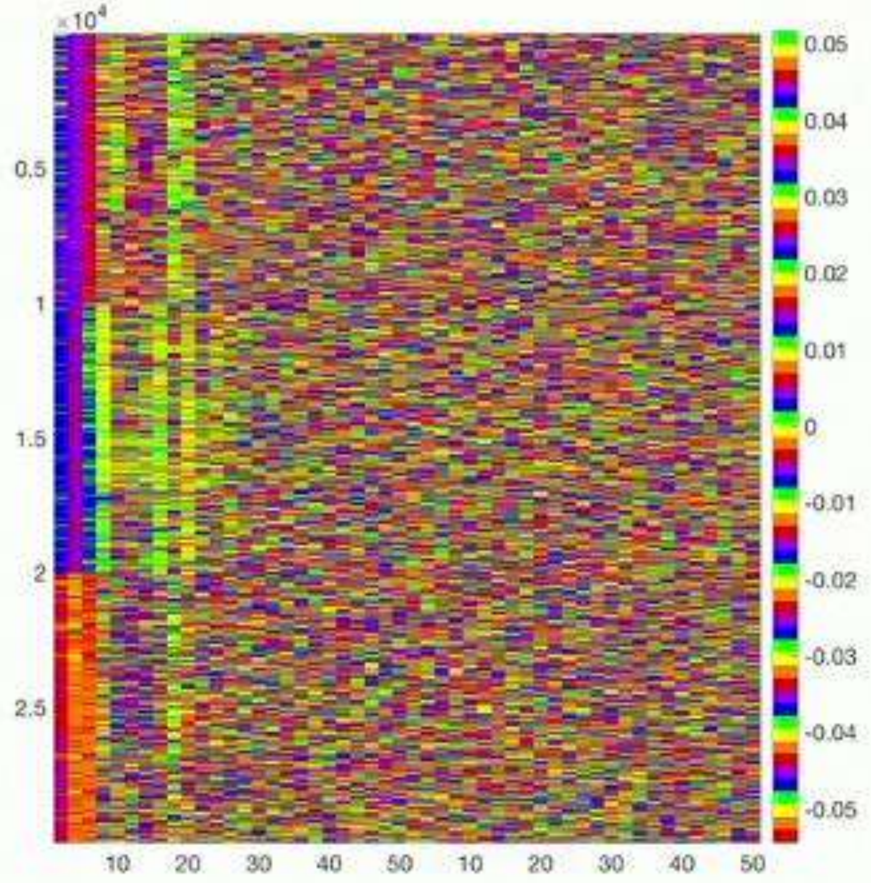
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
UNITED STATES AIR FORCE

NSF

Google

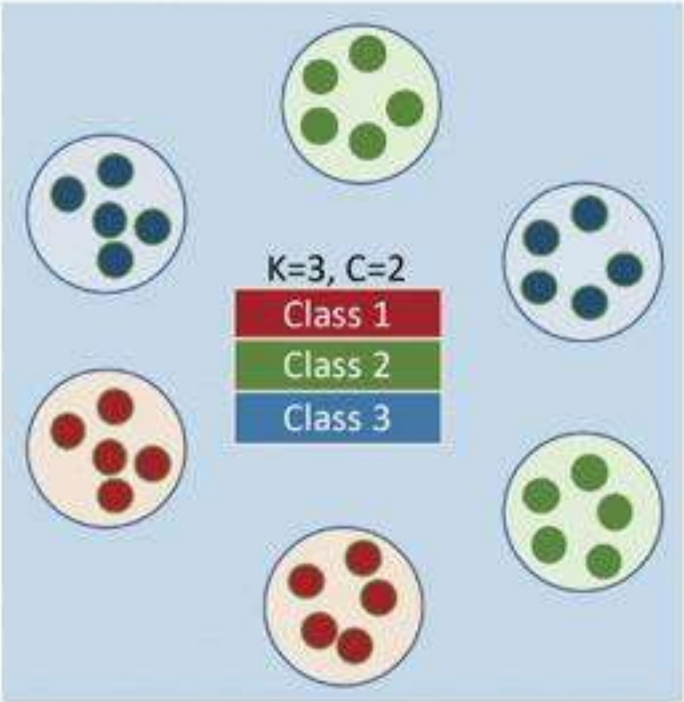
Key Idea II

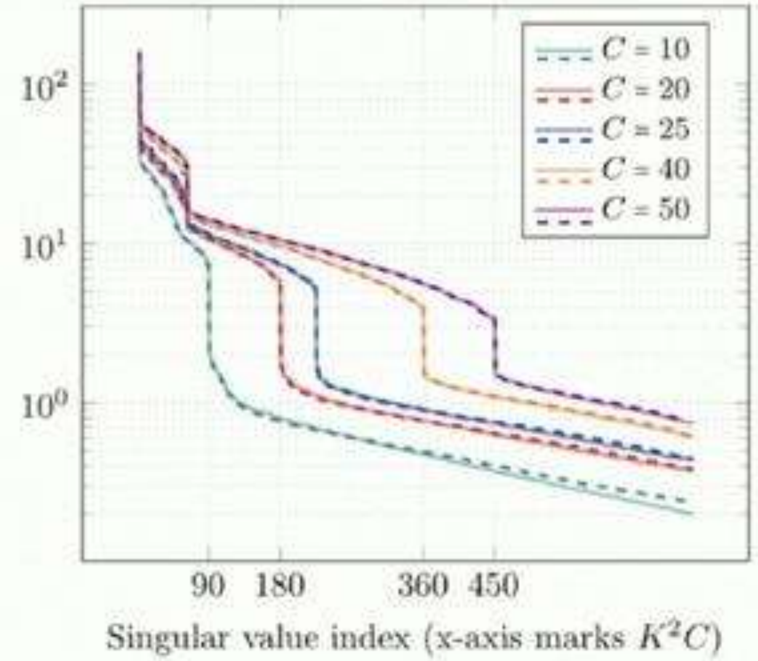
Prominent eigenvects of Jacobian are diffused



Concrete example: Gaussian Mixture Model (GMM)

Data set a GMM with K classes each containing C components per class with small σ^2





Theorem

With high probability

- *Jacobian has* K^2C *large singular values that grow* $\propto \sqrt{n}$
- *If* $k \gtrsim \Gamma^4 K^8 C^4$ *after* $T \propto \Gamma K^2 C$ *iterations,*

$$misclass\,(f(\boldsymbol{W}_T)) \lesssim \Gamma\sqrt{\frac{K^2C}{n}} + e^{-\Gamma}$$

Geometry of Optimization and Learning: Rich vs Kernel Inductive Bias

Nati Srebro (TTIC)

Based on work with **Suriya Gunasekar** (TTIC→MSR), Behnam Neyshabur (TTIC→Google),
Ryota Tomioka (TTIC→MSR), Srinadh Bhojanapalli (TTIC→Google),
Blake Woodworth, Pedro Savarese, Arturs Backurs, David McAllester (TTIC),
Daniel Soudry, Elad Hoffer, Mor Shpigel, Itay Sofer (Technion), **Jason Lee** (USC)
Russ Salakhutdinov (CMU), Ashia Wilson, Becca Roelofs, Mitchel Stern, Ben Recht (Berkeley),
Zhiyuan Li (Princeton), Yann LaCun (NYU/Facebook), Charlie Smart (UChicago).

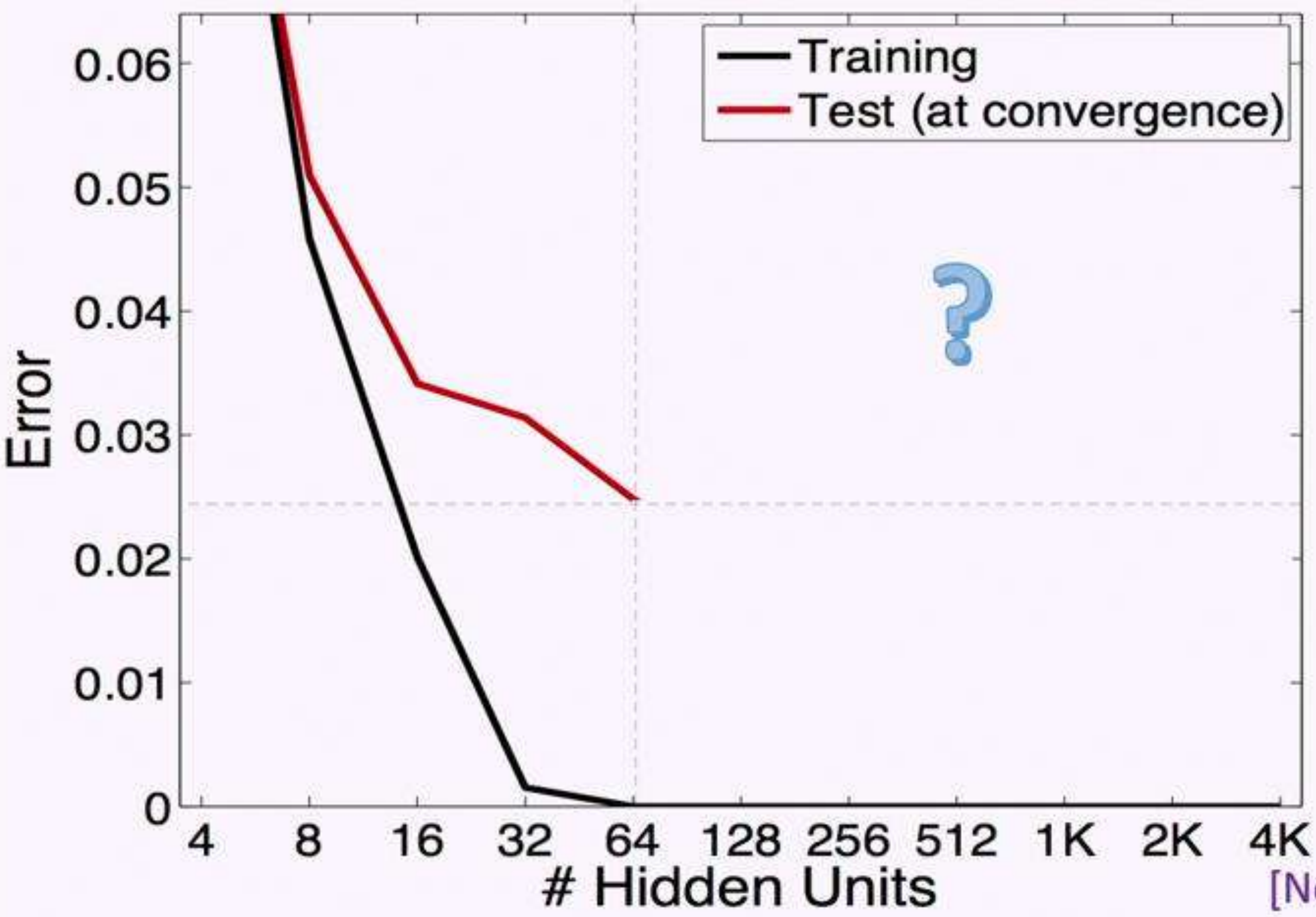

Last Time On...

Geometry of Optimization and Learning: Rich vs Kernel Inductive Bias

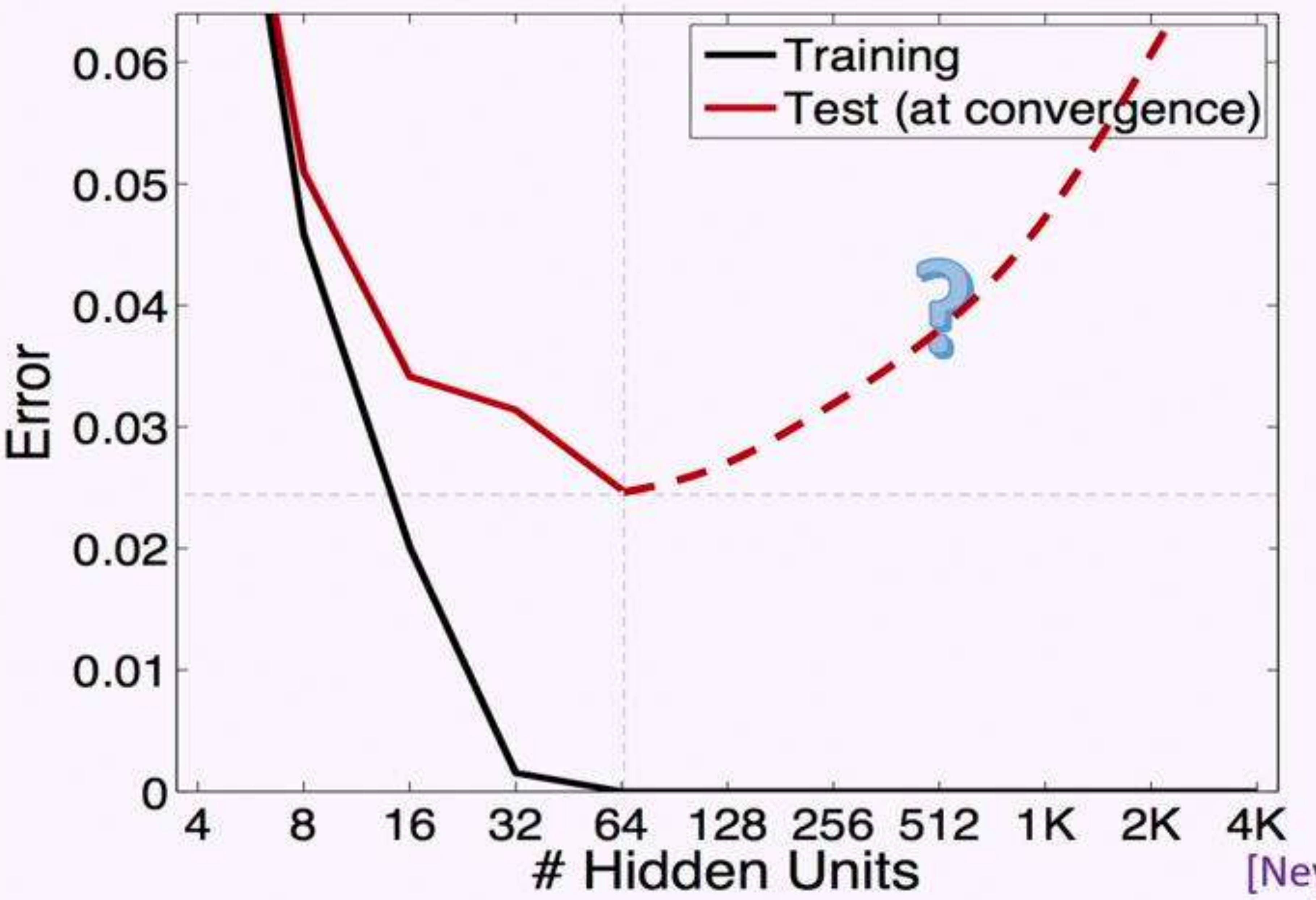
Nati Srebro (TTIC)

Based on work with **Suriya Gunasekar** (TTIC→MSR), Behnam Neyshabur (TTIC→Google),
Ryota Tomioka (TTIC→MSR), Srinadh Bhojanapalli (TTIC→Google),
Blake Woodworth, Pedro Savarese, Arturs Backurs, David McAllester (TTIC),
Daniel Soudry, Elad Hoffer, Mor Shpigel, Itay Sofer (Technion), **Jason Lee** (USC)
Russ Salakhutdinov (CMU), Ashia Wilson, Becca Roelofs, Mitchel Stern, Ben Recht (Berkeley),
Zhiyuan Li (Princeton), Yann LaCun (NYU/Facebook), Charlie Smart (UChicago).

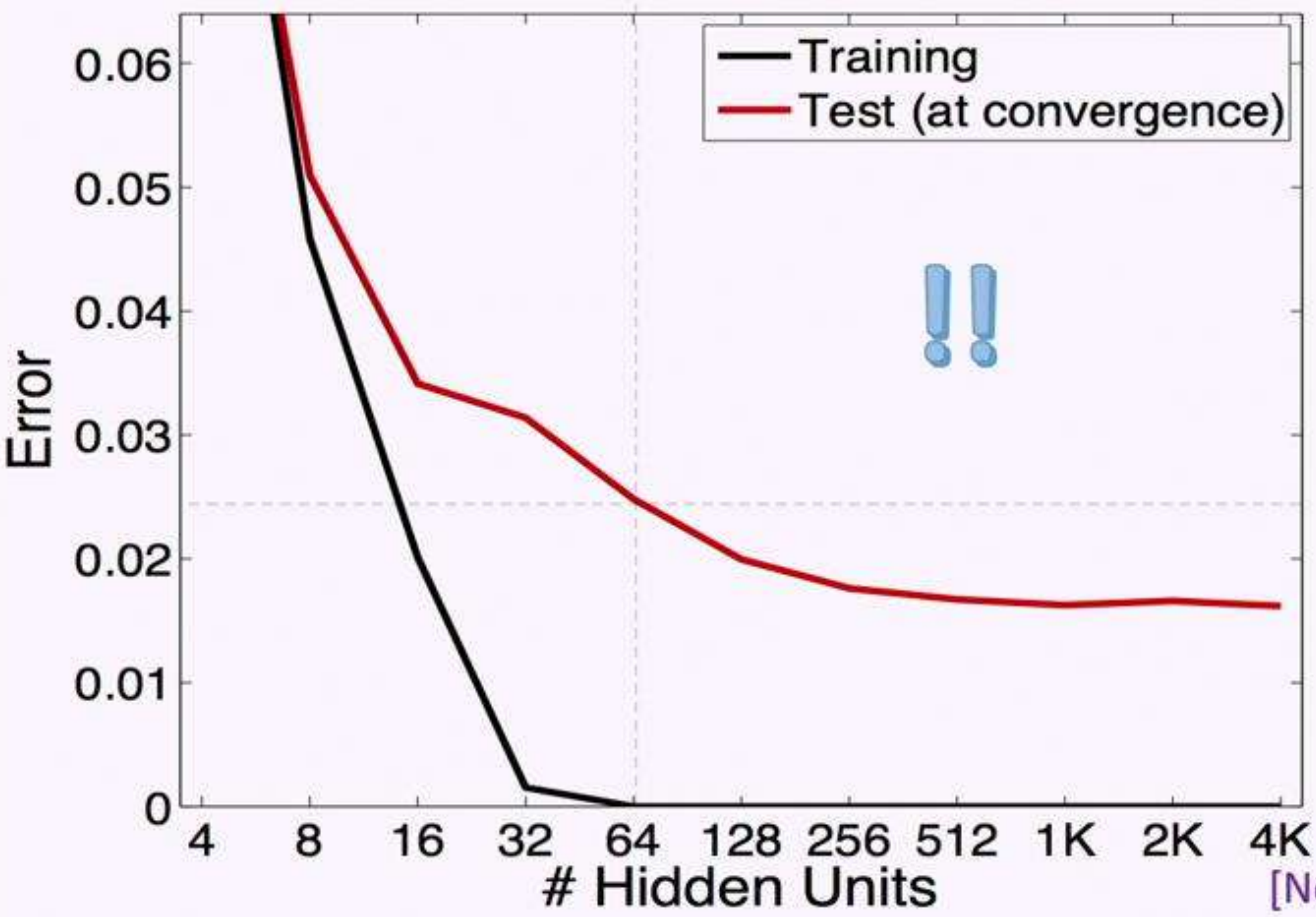

Last Time On...



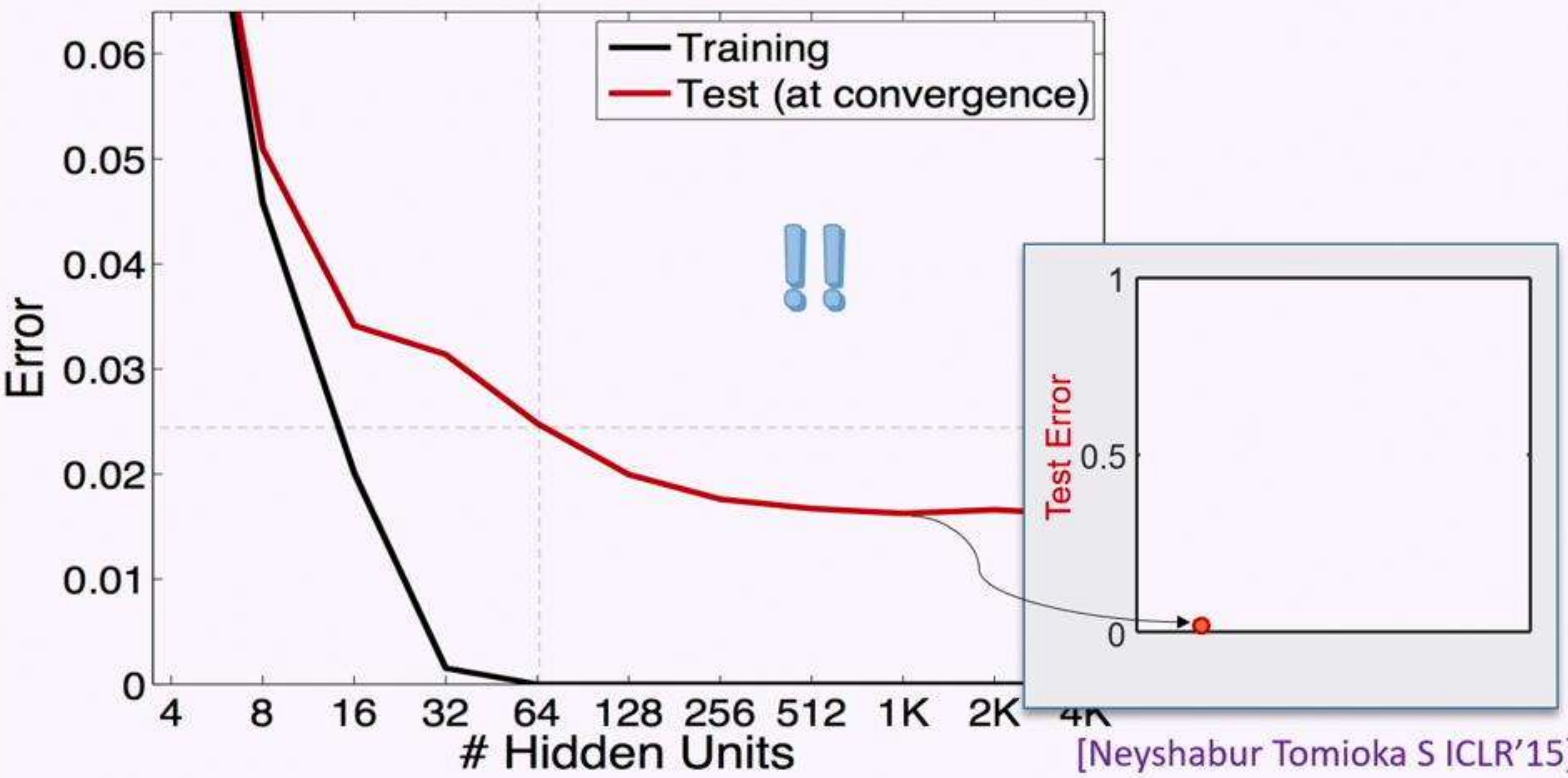
[Neyshabur Tomioka S ICLR'15]



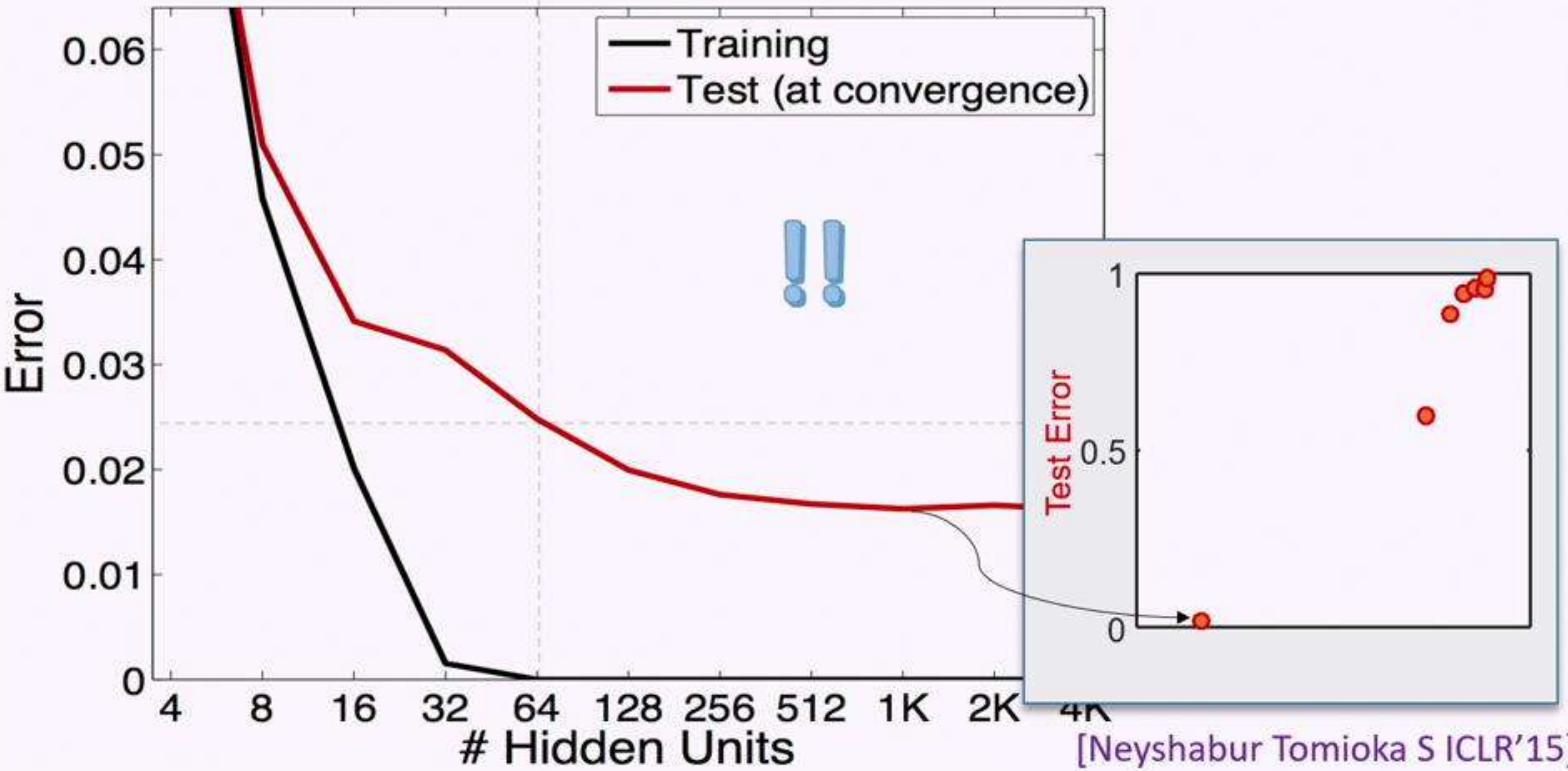
[Neyshabur Tomioka S ICLR'15]



[Neyshabur Tomioka S ICLR’15]



[Neyshabur Tomioka S ICLR'15]

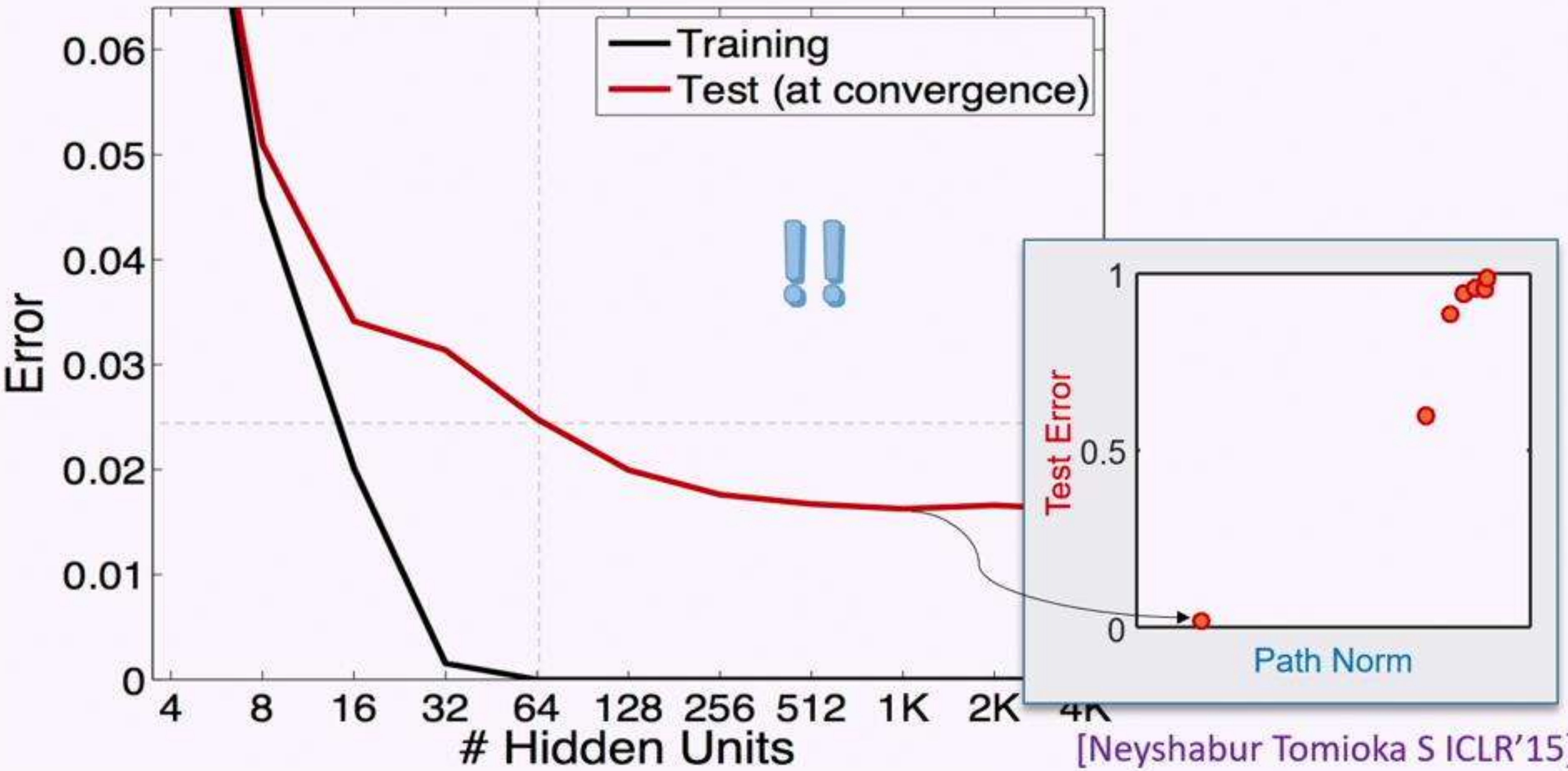


[Neyshabur Tomioka S ICLR'15]

For valid generalization, the size of the weights is more important than the size of the network

1997

Peter L. Bartlett

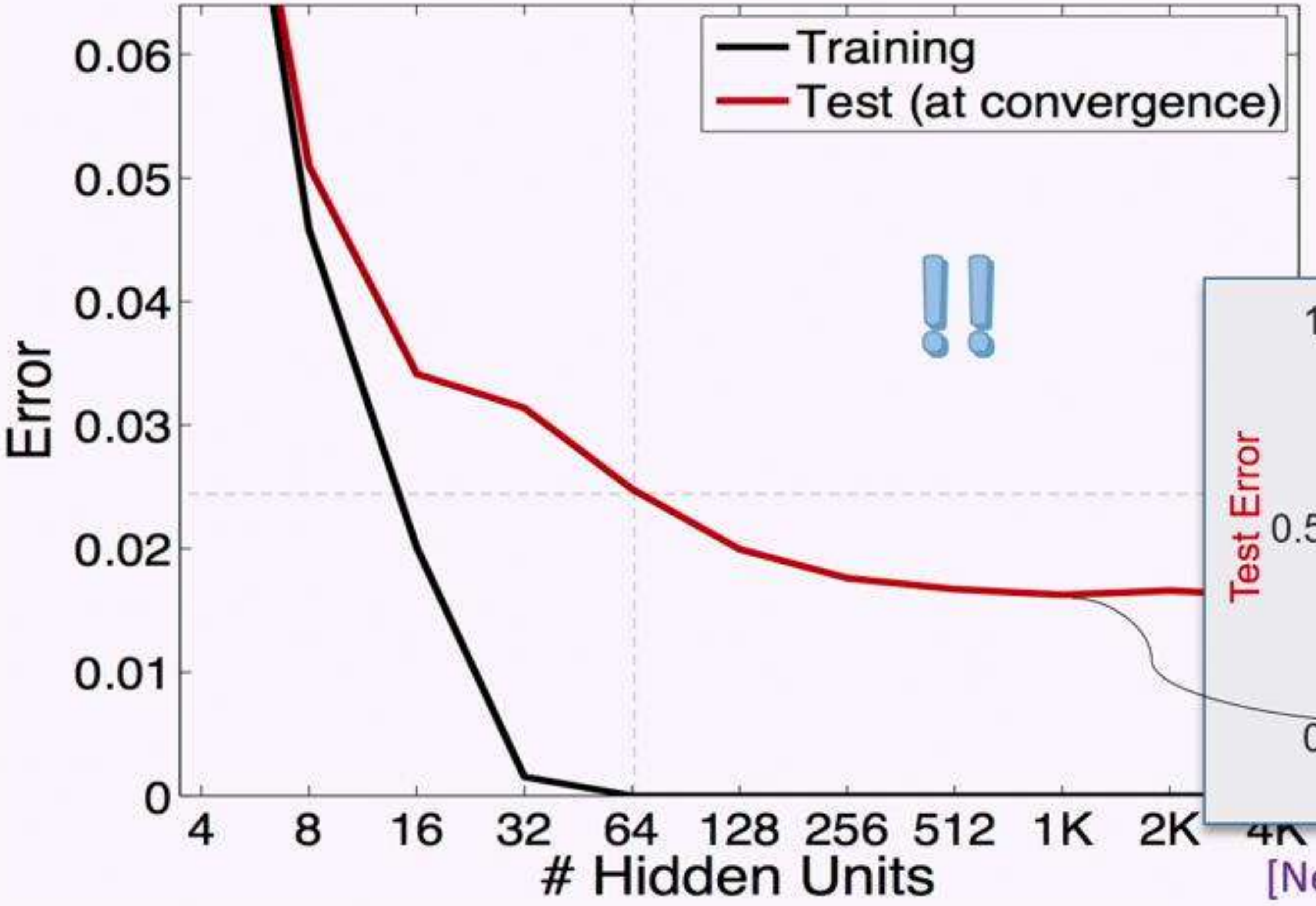


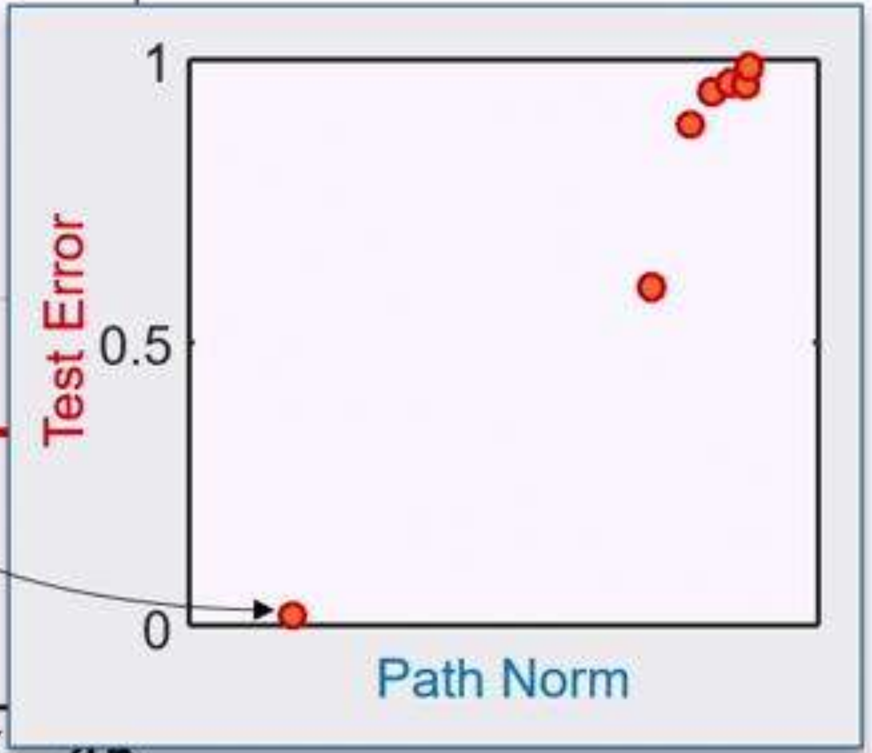
[Neyshabur Tomioka S ICLR'15]

For valid generalization, the size of the weights is more important than the size of the network

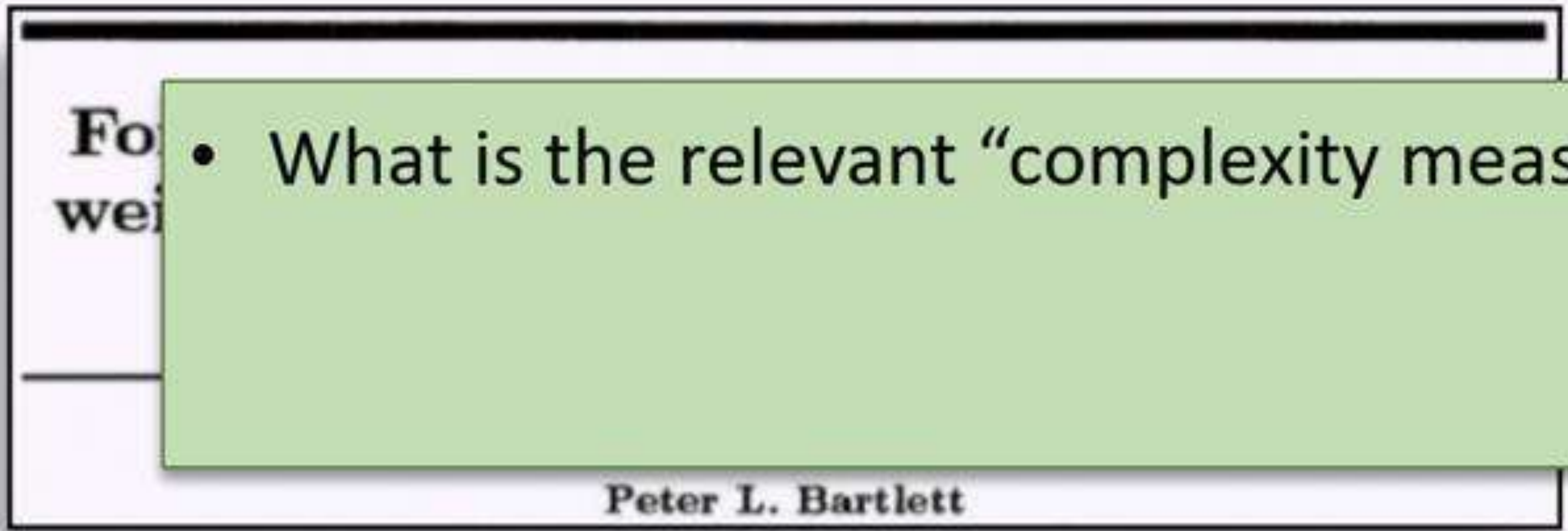
1997

Peter L. Bartlett

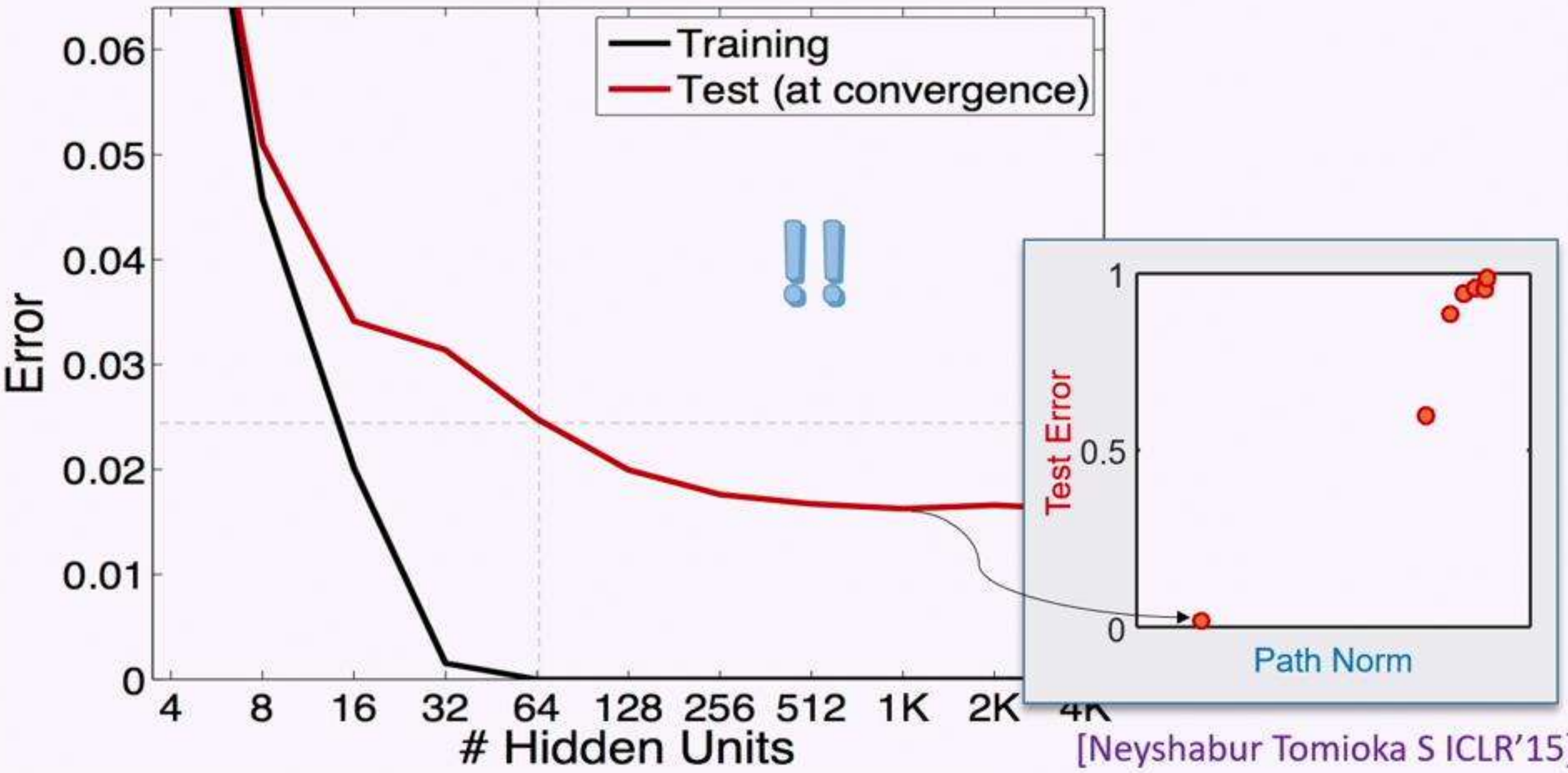




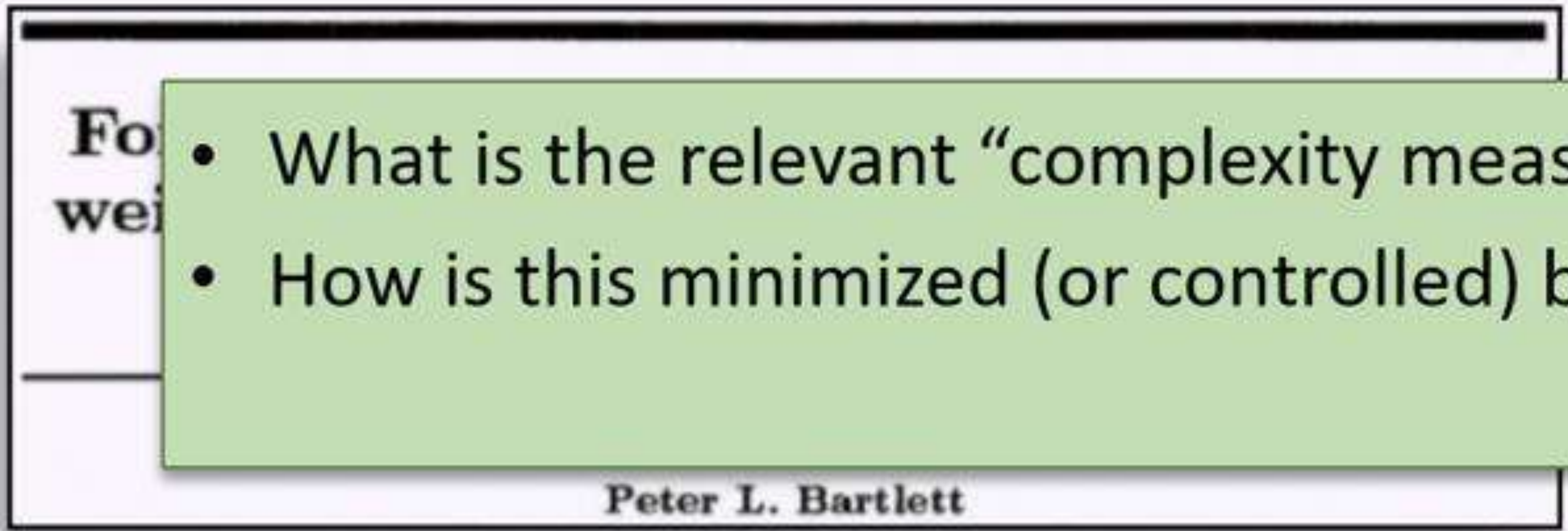
[Neyshabur Tomioka S ICLR’15]



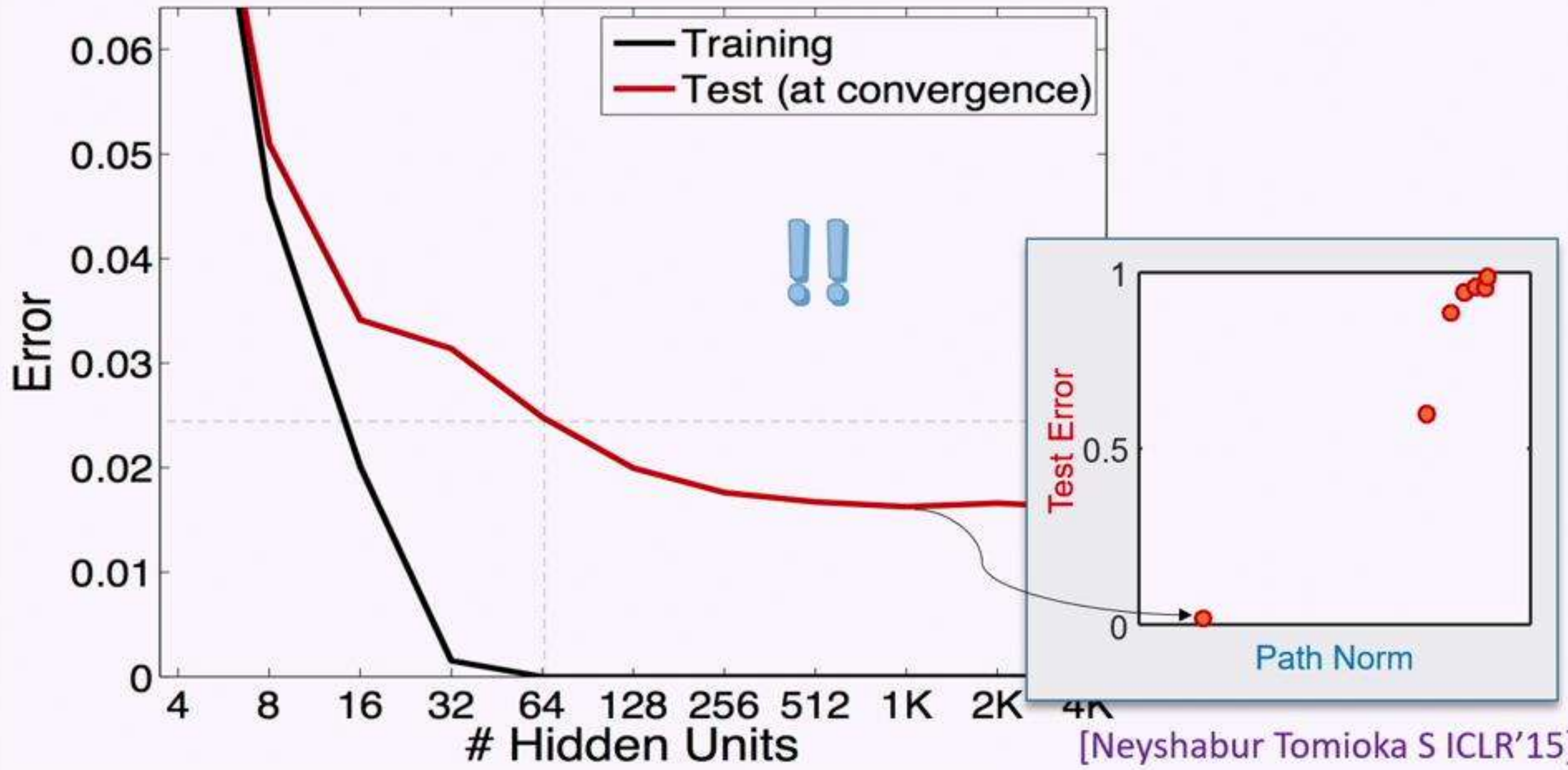
- What is the relevant “complexity measure” (eg norm)?



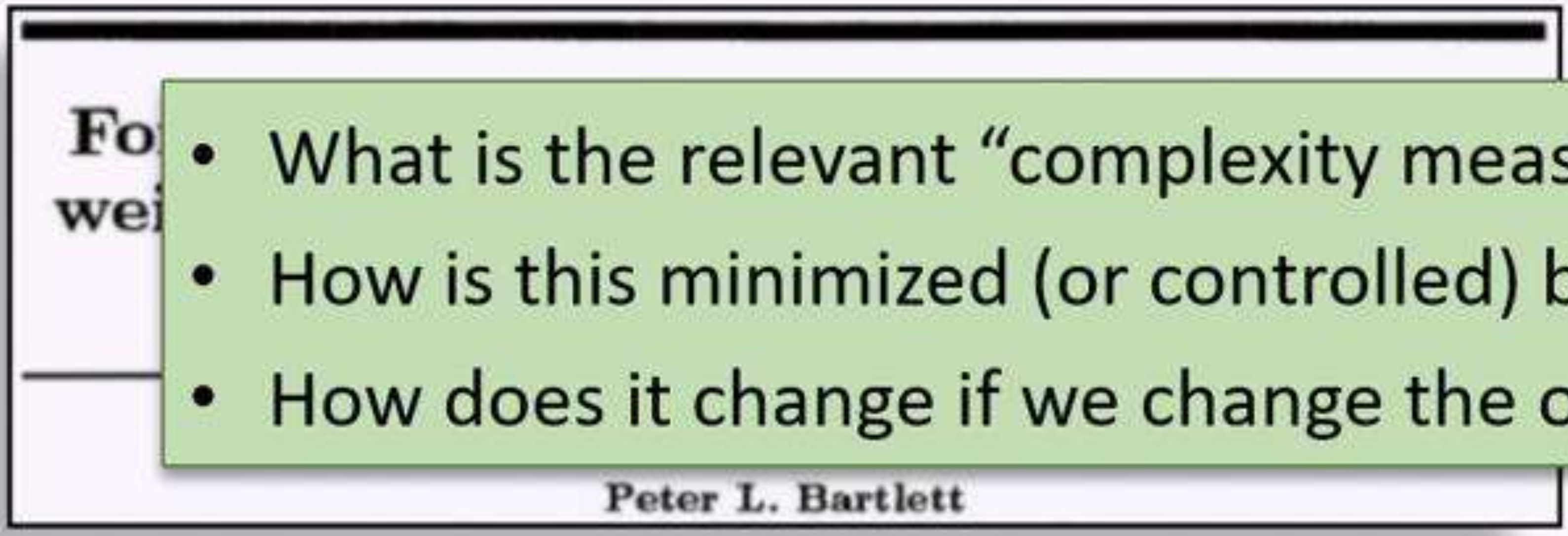
[Neyshabur Tomioka S ICLR'15]



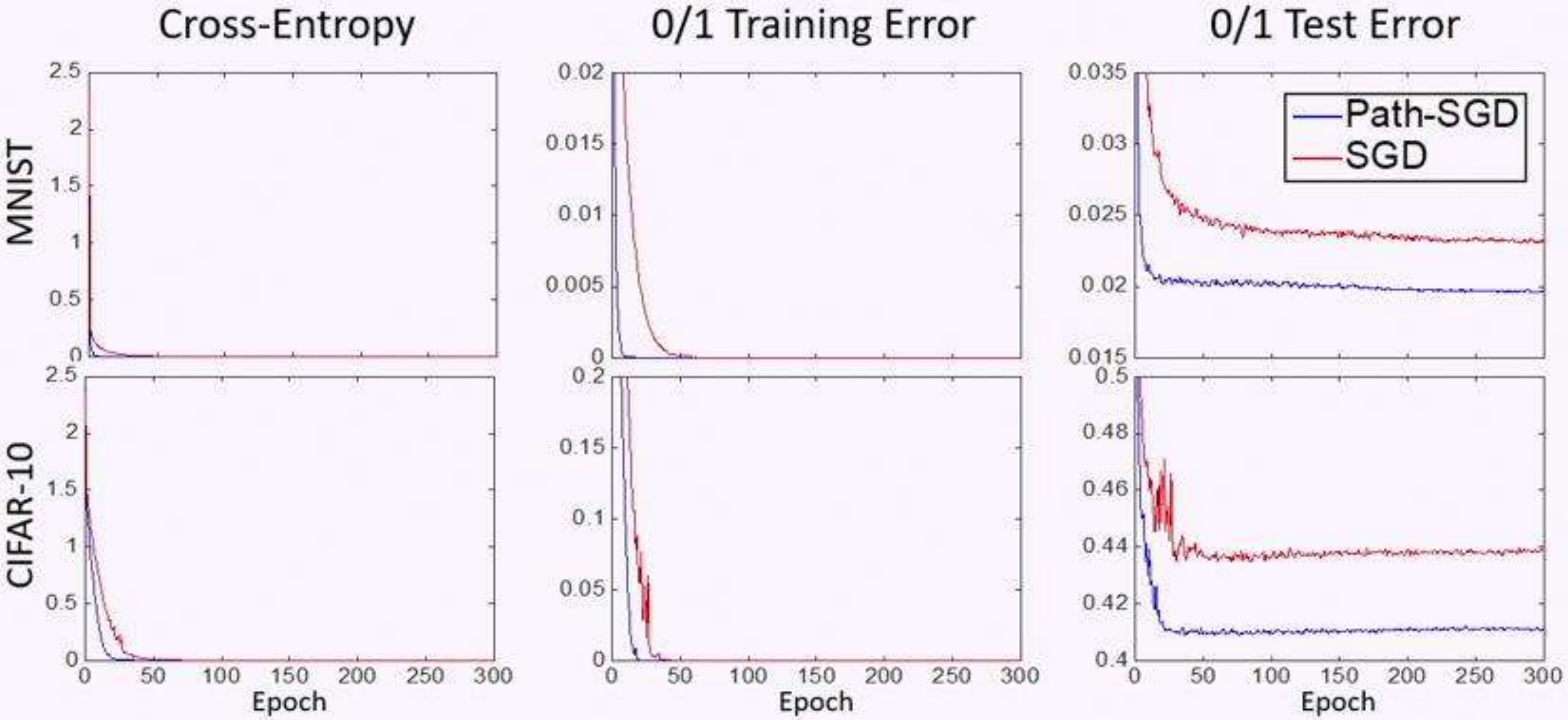
- What is the relevant "complexity measure" (eg norm)?
- How is this minimized (or controlled) by the opt algorithm?



[Neyshabur Tomioka S ICLR'15]

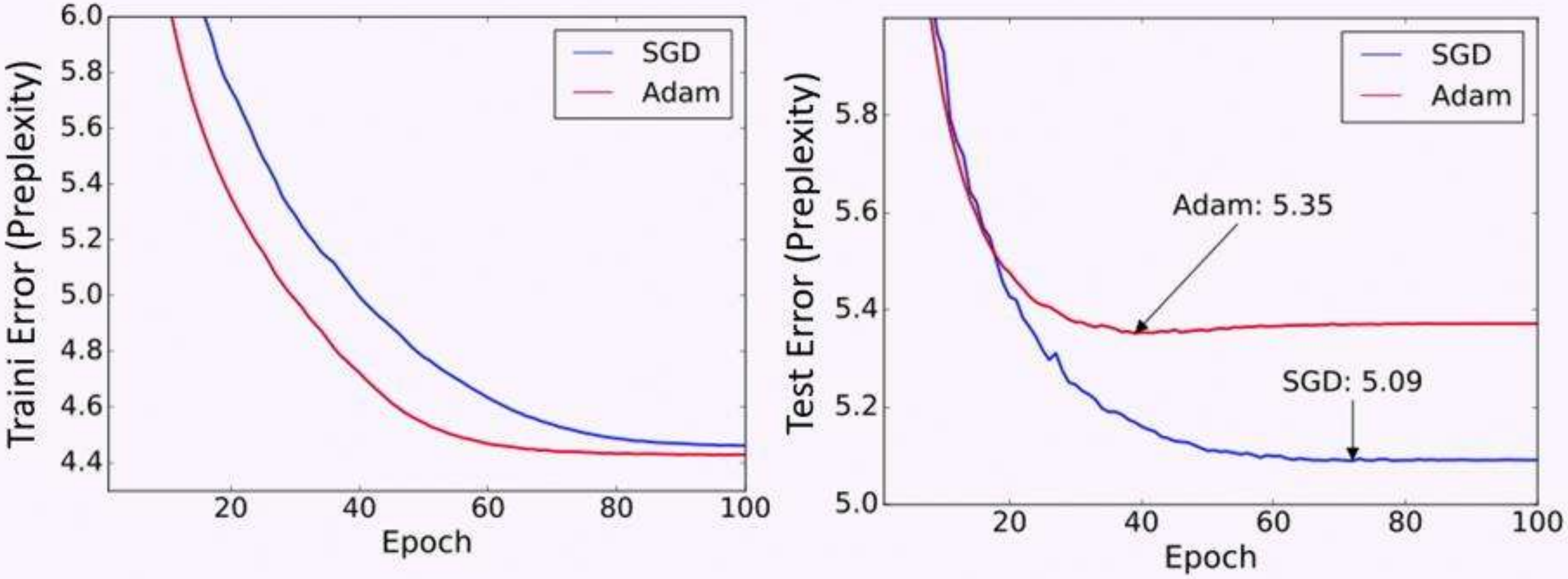

Fo
wei
Peter L. Bartlett

- What is the relevant "complexity measure" (eg norm)?
- How is this minimized (or controlled) by the opt algorithm?
- How does it change if we change the opt algorithm?



[Neyshabur Salakhudtinov S NIPS'15]

SGD vs ADAM



Results on Penn Treebank using 3-layer LSTM

[Wilson Roelofs Stern S Recht, “The Marginal Value of Adaptive Gradient Methods in Machine Learning”, NIPS’17]

The Deep Recurrent Residual Boosting Machine

Joe Flow, DeepFace Labs

Section 1: Introduction

We suggest a new amazing architecture and loss function that is great for learning. All you have to do to learn is fit the model on your training data

Section 2: Learning Contribution: our model

The model class h_w is amazing. **Our learning method is:**

$$\mathbf{arg\,min}_{\boldsymbol{w}} \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{loss}(\boldsymbol{h}_{\boldsymbol{w}}(\boldsymbol{x}); \boldsymbol{y}) \qquad (*)$$

Section 3: Optimization

This is how we solve the optimization problem (*): [...]

Section 4: Experiments

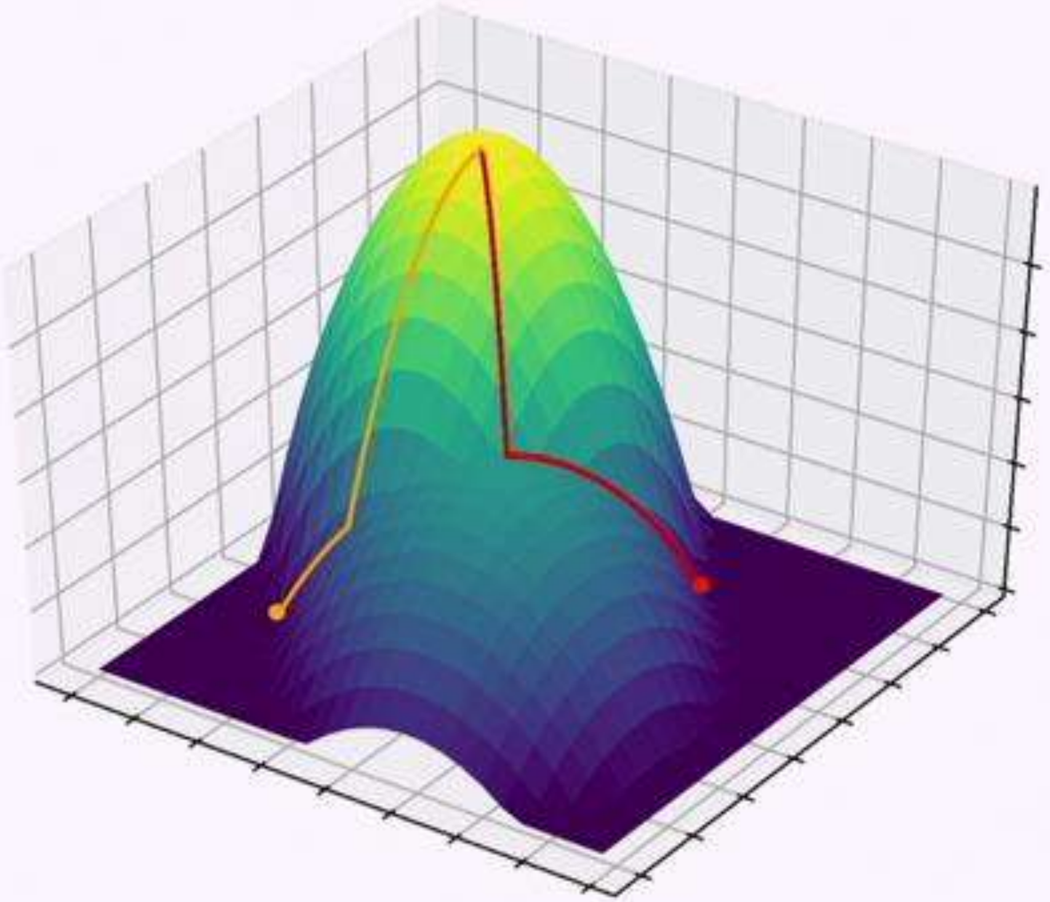
It works!

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties



Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \Psi(w)$

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \Psi(w)$

- Gradient descent on seperable logistic regression $\min_w \sum_i g(y_i\langle w, x_i\rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_2$ s.t. $y_i\langle w, x_i\rangle \geq 1$ **(Hard Margin SVM)**

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \Psi(w)$

- Gradient descent on seperable logistic regression $\min_w \sum_i g(y_i \langle w, x_i \rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_2$ s.t. $y_i \langle w, x_i \rangle \geq 1$ **(Hard Margin SVM)**

- Coordinate descent on $\min_w \sum_i g(y_i \langle w, x_i \rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_1$ s.t. $y_i \langle w, x_i \rangle \geq 1$ [Telgarsky 2013]

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \Psi(w)$

- Gradient descent on seperable logistic regression $\min_w \sum_i g(y_i\langle w, x_i\rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_2$ s.t. $y_i\langle w, x_i\rangle \geq 1$ **(Hard Margin SVM)**

- Coordinate descent on $\min_w \sum_i g(y_i\langle w, x_i\rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_1$ s.t. $y_i\langle w, x_i\rangle \geq 1$ [Telgarsky 2013]

- Gradient descent on matrix factorization $\min_{U,V} \sum_i (\langle A_i, UV\rangle - y_i)^2$

  ➔ $UV \to \arg\min \|W\|_*$ s.t. $\langle A_i, W\rangle = y_i$
  with init→0 and stepsize→0
  (proven in special cases, empirically at least aprox, conjectured in general)

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V ➔ implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \|w\|_2$

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$

  ➔ $w \to \arg\min_{Xw=y} \Psi(w)$

- Gradient descent on seperable logistic regression $\min_w \sum_i g(y_i\langle w, x_i\rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_2$ s.t. $y_i\langle w, x_i\rangle \geq 1$ **(Hard Margin SVM)**

- Coordinate descent on $\min_w \sum_i g(y_i\langle w, x_i\rangle)$

  ➔ $\frac{w}{\|w\|} \to \arg\min \|w\|_1$ s.t. $y_i\langle w, x_i\rangle \geq 1$ [Telgarsky 2013]

- Gradient descent on matrix factorization $\min_{U,V} \sum_i (\langle A_i, UV\rangle - y_i)^2$

  ➔ $UV \to \arg\min \|W\|_*$ s.t. $\langle A_i, W\rangle = y_i$

  with init→0 and stepsize→0

  (proven in special cases, empirically at least aprox, conjectured in general)

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V ➔ implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V → implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors β

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V ➔ implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors β
  - GD on weight ➔ implicitly minimize $\|\boldsymbol{DFT}(\boldsymbol{\beta})\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$.
    (sparsity in frequency domain)

[Gunasekar Lee Soudry S 2018]

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V ➔ implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors β
  - GD on weight ➔ implicitly minimize $\|\boldsymbol{DFT}(\boldsymbol{\beta})\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$.
    (sparsity in frequency domain)

[Gunasekar Lee Soudry S 2018]

- Infinite Width ReLU Net with 1-d input:
  - Parametrization of essentially all functions $f: \mathbb{R} \to \mathbb{R}$

Effect of Parametrization

- Matrix completion (also: reconstruction from linear measurements)
  - $W = UV$ is over-parametrization of all matrices $W \in \mathbb{R}^{n \times m}$
  - GD on U, V ➔ implicitly minimize $\|\boldsymbol{W}\|_*$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]

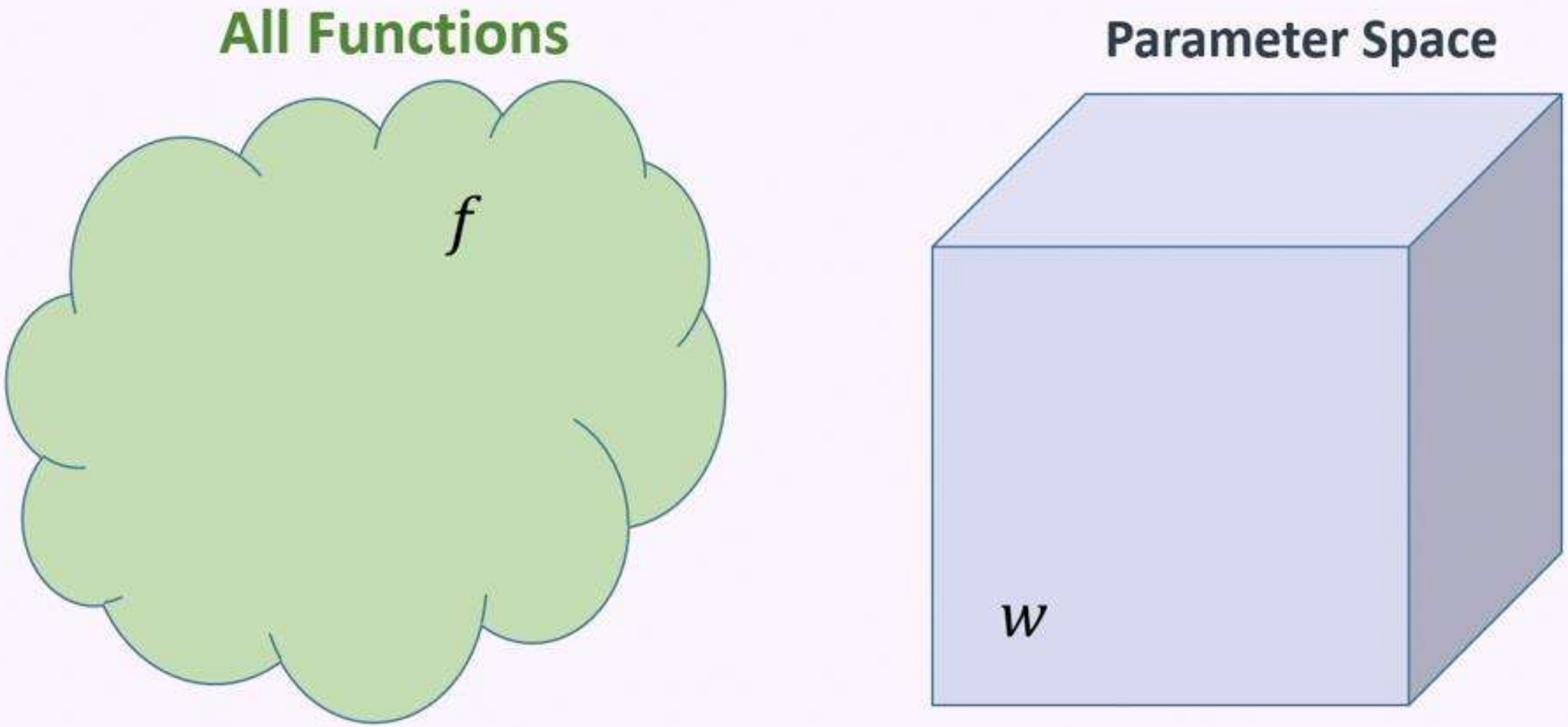
- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors β
  - GD on weight ➔ implicitly minimize $\|\boldsymbol{DFT(\beta)}\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$. (sparsity in frequency domain)

[Gunasekar Lee Soudry S 2018]

- Infinite Width ReLU Net with 1-d input:
  - Parametrization of essentially all functions $f: \mathbb{R} \to \mathbb{R}$
  - Weight decay ➔ implicitly minimize

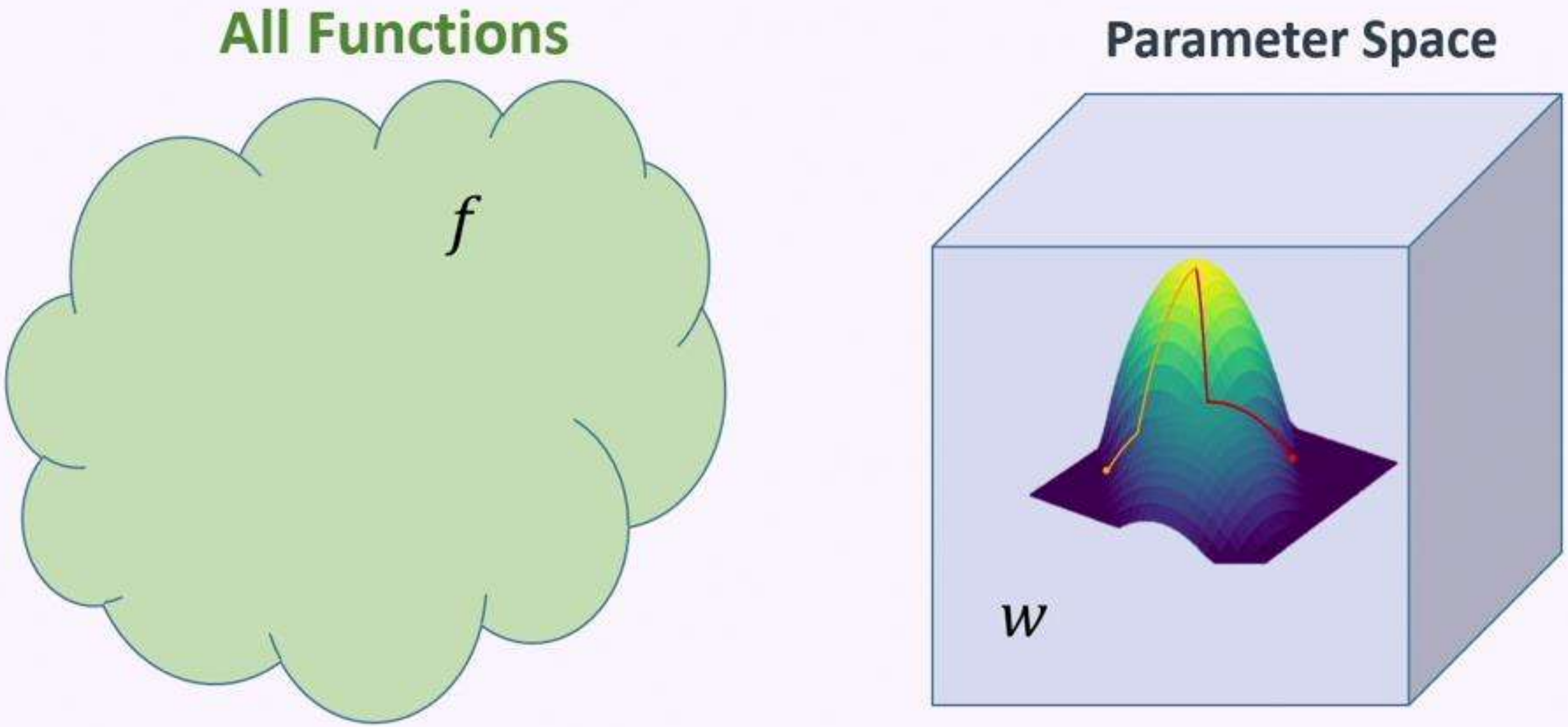
$$\max\left(\int |\boldsymbol{f''}| \boldsymbol{dx} \ , |f'(-\infty) + f'(+\infty)| \right)$$

[Savarese Evron Soudry S 2019]



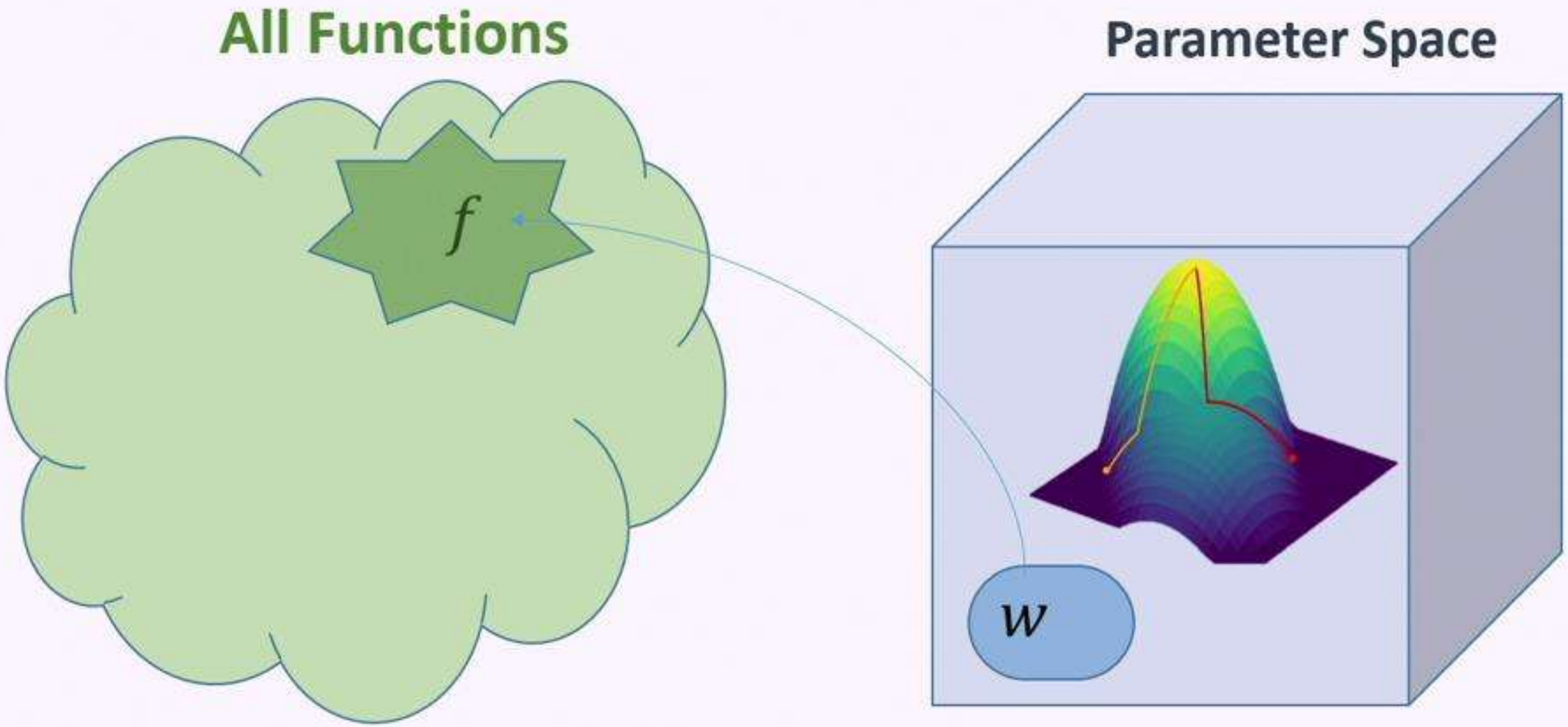
Optimization Geometry and hence Inductive Bias effected by:

- Geometry of local search in parameter space
- Choice of parameterization



Optimization Geometry and hence Inductive Bias effected by:

- Geometry of local search in parameter space
- Choice of parameterization



Optimization Geometry and hence Inductive Bias effected by:

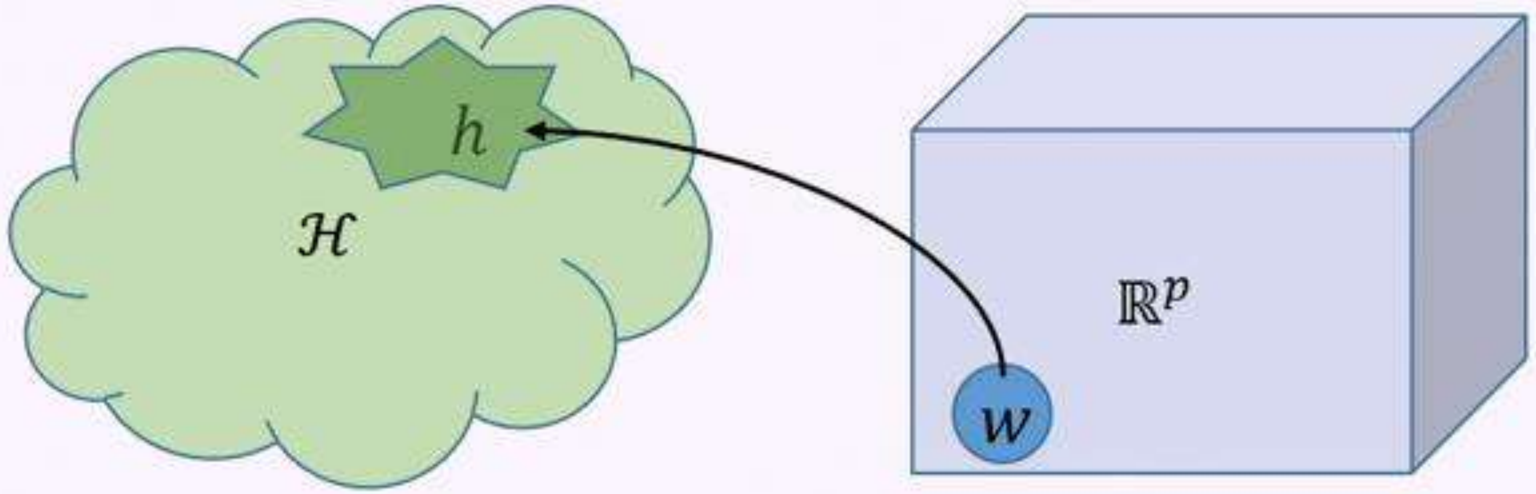
- Geometry of local search in parameter space
- Choice of parameterization

Model: $\boldsymbol{F(w) = h_w}$ **Model Class:** $\boldsymbol{\mathcal{H} = \mathrm{range}(F)}$

$f(w, x) = h_w(x) =$ prediction on x with params ("weights") w

Linear models: $f(w, x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

Loss: $L_S(w) = \frac{1}{m} \sum_i \ell(f(w, x_i), y_i)$



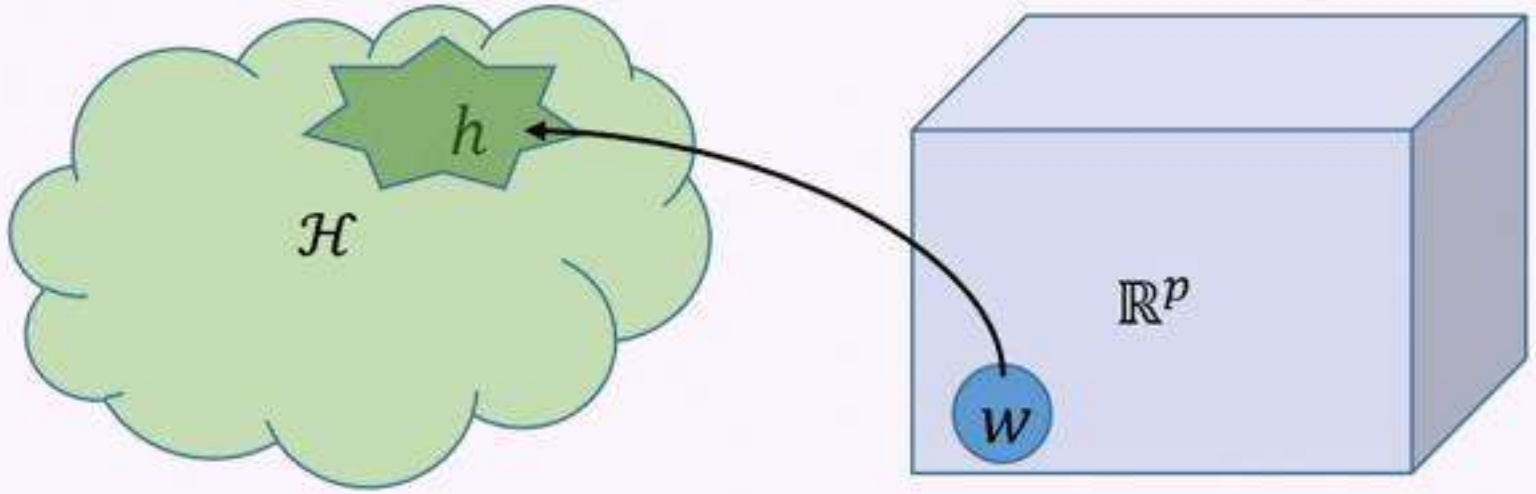
Model: $F(w) = h_w$ Model Class: $\mathcal{H} = \text{range}(F)$

$f(w,x) = h_w(x)$ = prediction on x with params ("weights") w

Linear models: $f(w,x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

Loss: $L_S(w) = \frac{1}{m}\sum_i \ell(f(w,x_i),y_i)$

D-homogenous: $F(cw) = c^D F(w)$, i.e. $f(cw,x) = c^D f(w,x)$



Model: $\boldsymbol{F(w) = h_w}$ **Model Class:** $\boldsymbol{\mathcal{H} = \mathrm{range}(F)}$

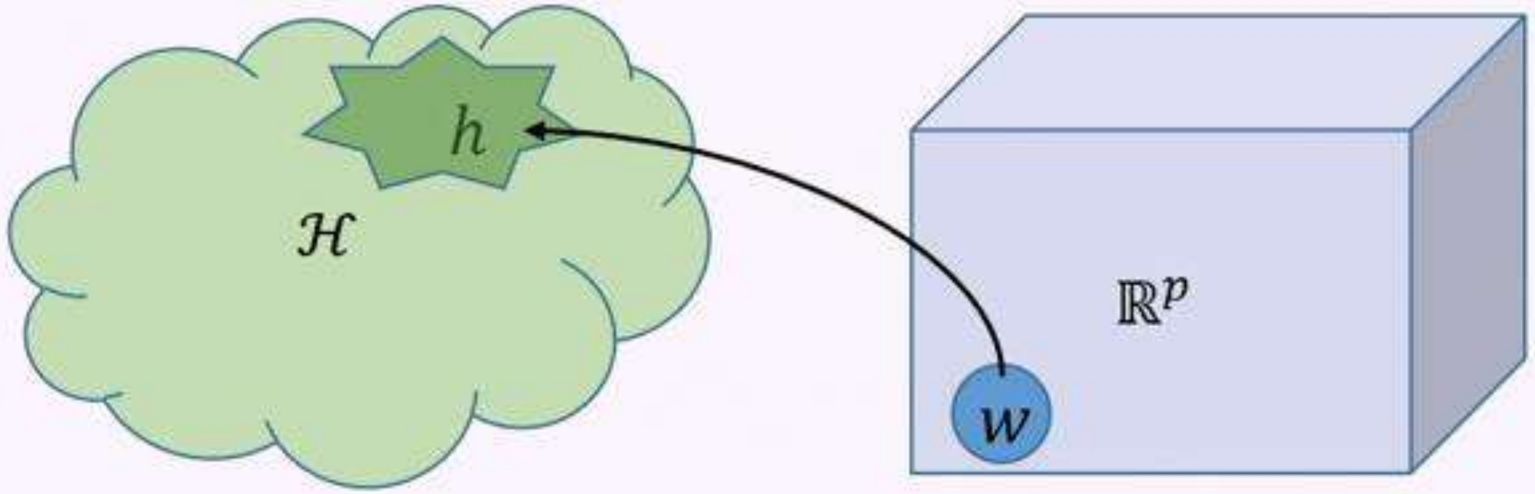
$f(w,x) = h_w(x)$ = prediction on x with params ("weights") w

Linear models: $f(w,x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

Loss: $L_S(w) = \frac{1}{m}\sum_i \ell(f(w,x_i), y_i)$

D-homogenous: $F(cw) = c^D F(w)$, i.e. $f(cw,x) = c^D f(w,x)$

- 1-homogenous: standard linear $F(w) = w$, $f(w,x) = \langle w, x \rangle$



Model: $F(w) = h_w$ Model Class: $\mathcal{H} = \text{range}(F)$

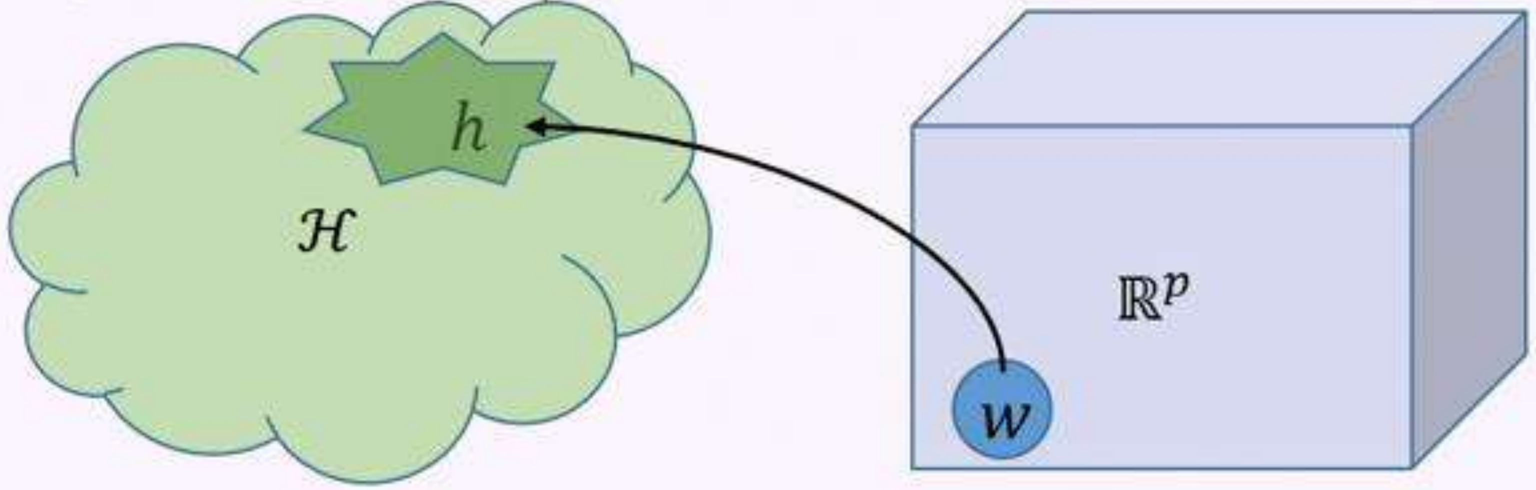
$f(w, x) = h_w(x) =$ prediction on x with params ("weights") w

Linear models: $f(w, x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

Loss: $L_S(w) = \frac{1}{m} \sum_i \ell(f(w, x_i), y_i)$

D-homogenous: $F(cw) = c^D F(w)$, i.e. $f(cw, x) = c^D f(w, x)$

- 1-homogenous: standard linear $F(w) = w$, $f(w, x) = \langle w, x \rangle$
- 2-homogenous:
  - Matrix factorization $F(U, V) = UV$
  - 2-Layer ReLU: $f(W, x) = \sum_j w_{2,j} [\langle w_{1,j}, x \rangle]_+$



Model: $F(w) = h_w$ Model Class: $\mathcal{H} = \mathbf{range}(F)$

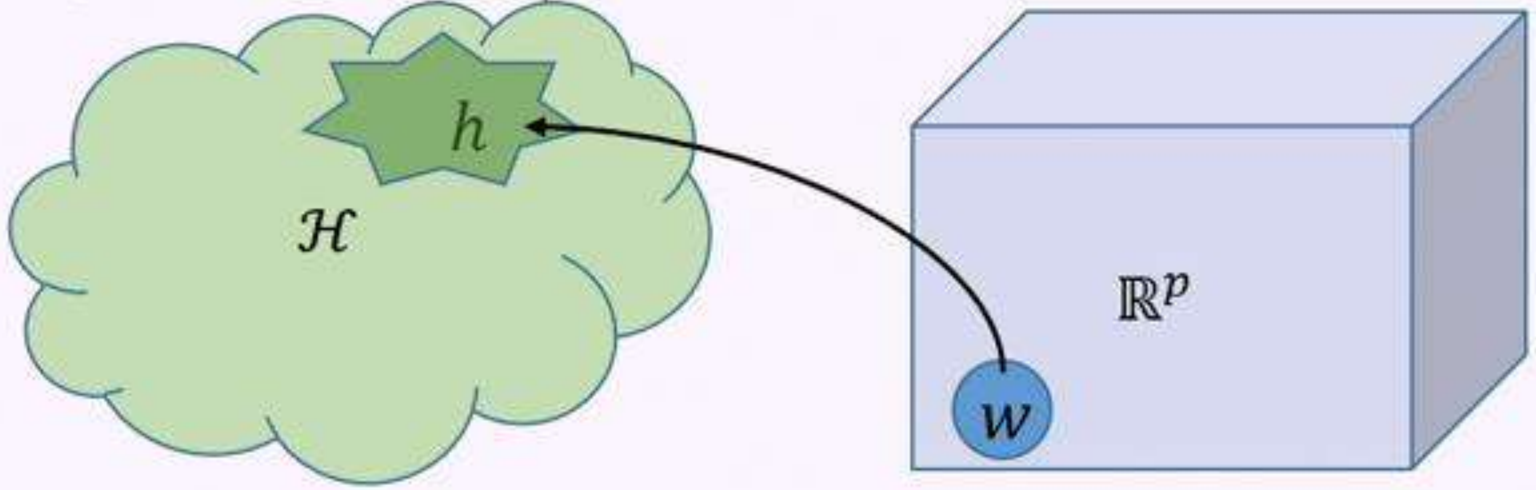
$f(w, x) = h_w(x) =$ prediction on x with params ("weights") w

Linear models: $f(w, x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

Loss: $L_S(w) = \frac{1}{m} \sum_i \ell(f(w, x_i), y_i)$

D-homogenous: $F(cw) = c^D F(w)$, i.e. $f(cw, x) = c^D f(w, x)$

- 1-homogenous: standard linear $F(w) = w$, $f(w, x) = \langle w, x \rangle$
- 2-homogenous:
  - Matrix factorization $F(U, V) = UV$
  - 2-Layer ReLU: $f(W, x) = \sum_j w_{2,j} [\langle w_{1,j}, x \rangle]_+$
- D-homogenous:
  - D layer linear network
  - D layer linear conv net
  - D layer ReLU net



Consider gradient descent w.r.t. logistic loss $L_s(w) = \sum_i g\big(y_i f(w, x_i)\big)$ (or other loss exp-tail loss) on a D-homogenous model $f(w, x)$:

Theorem [Nacson Gunasekar Lee S Soudry 2019][Lyu Li 2019]:
If $L_s(w) \to 0$, with stepsize→0 (or finite but ensures convergence in direction):

$\boldsymbol{w_\infty} \propto$ **first order stationary point of**

$$\arg\min \|w\|_2 \;\; s.t. \forall_i y_i f(w, x_i) \geq 1$$

Consider gradient descent w.r.t. logistic loss $L_s(w) = \sum_i g\big(y_i f(w, x_i)\big)$

(or other loss exp-tail loss) on a D-homogenous model $f(w, x)$:

Theorem [Nacson Gunasekar Lee S Soudry 2019][Lyu Li 2019]:

If $L_s(w) \to 0$, with stepsize→0 (or finite but ensures convergence in direction):

$\boldsymbol{w_\infty} \propto$ **first order stationary point of**

$$\arg\min \|\boldsymbol{w}\|_2 \;\; s.t. \forall_i y_i f(\boldsymbol{w}, \boldsymbol{x_i}) \geq 1$$

Suggests implicit bias defined by $R_F(h) = \arg\min_{F(w)=h} \|w\|_2$ and **(*)**

$h_\infty = F(w_\infty) \propto$ **first order stationary point of**

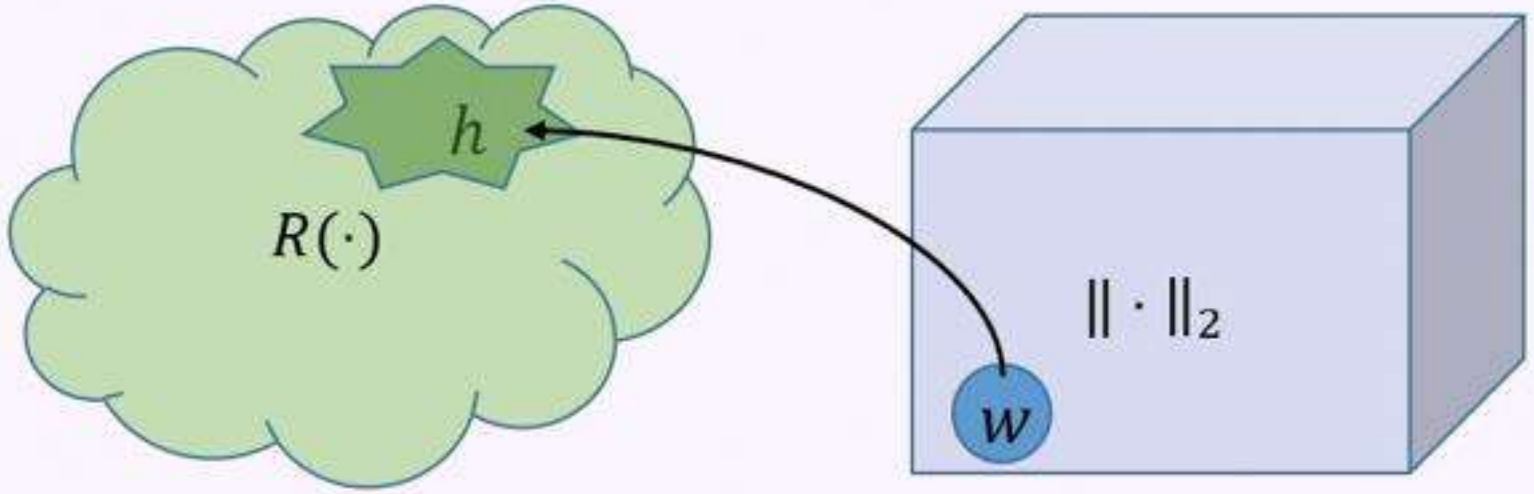
$$\arg\min R_F(h) \; s.t. \; y_i f(x_i) \geq 1$$

(**)

But need to be careful: f.o.s.p of **(*)** does ***not*** imply f.o.s.p of **(**)**

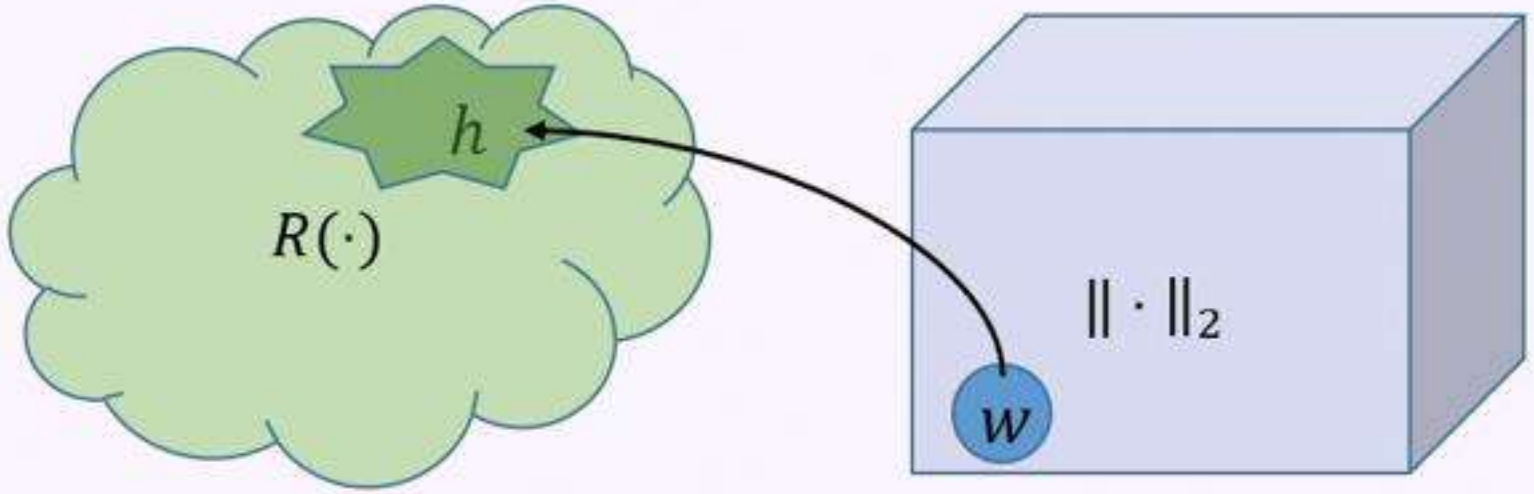
Is implicit bias of GD just ℓ_2 in param space + mapping to func space?

- Maybe yes? Implicit bias given by $R(h) = \arg\min_{F(w)=h} \|w\|_2$?
  - Matrix factorization, $R(\beta)^2 \propto \|\beta\|_{tr}$
  - Linear conv nets, diagonal nets with exp loss $R(\beta) \propto \|\beta\|_{2/D}$
  - Generic result for homogenous



Is implicit bias of GD just ℓ_2 in param space + mapping to func space?

- Maybe yes? Implicit bias given by $R(h) = \arg\min_{F(w)=h} \|w\|_2$?
  - Matrix factorization, $R(\beta)^2 \propto \|\beta\|_{tr}$
  - Linear conv nets, diagonal nets with exp loss $R(\beta) \propto \|\beta\|_{2/D}$
  - Generic result for homogenous
- But…
  - Do those conditions actually hold?
  - Transition from f.o.s.p in w to f.o.s.p/local opt of $R(h)$
  - [Arora Cohen Hu Luo 2019]: with squared loss, implicit reg of diagonal nets $\propto \|\beta\|_1$ for any depth $D \geq 2$!



Consider gradient descent w.r.t. logistic loss $L_s(w) = \sum_i g\big(y_i f(w, x_i)\big)$

(or other loss exp-tail loss) on a D-homogenous model $f(w, x)$:

Theorem [Nacson Gunasekar Lee S Soudry 2019][Lyu Li 2019]:

If $L_s(w) \to 0$, with stepsize→0 (or finite but ensures convergence in direction):

$$\boldsymbol{w_\infty} \propto \textbf{first order stationary point of}$$

$$\arg\min \|\boldsymbol{w}\|_2 \;\; s.t.\, \forall_i y_i f(\boldsymbol{w}, x_i) \geq 1$$

Suggests implicit bias defined by $R_F(h) = \arg\min_{F(w)=h} \|w\|_2$ and **(*)**

$$h_\infty = F(w_\infty) \propto \textbf{first order stationary point of}$$

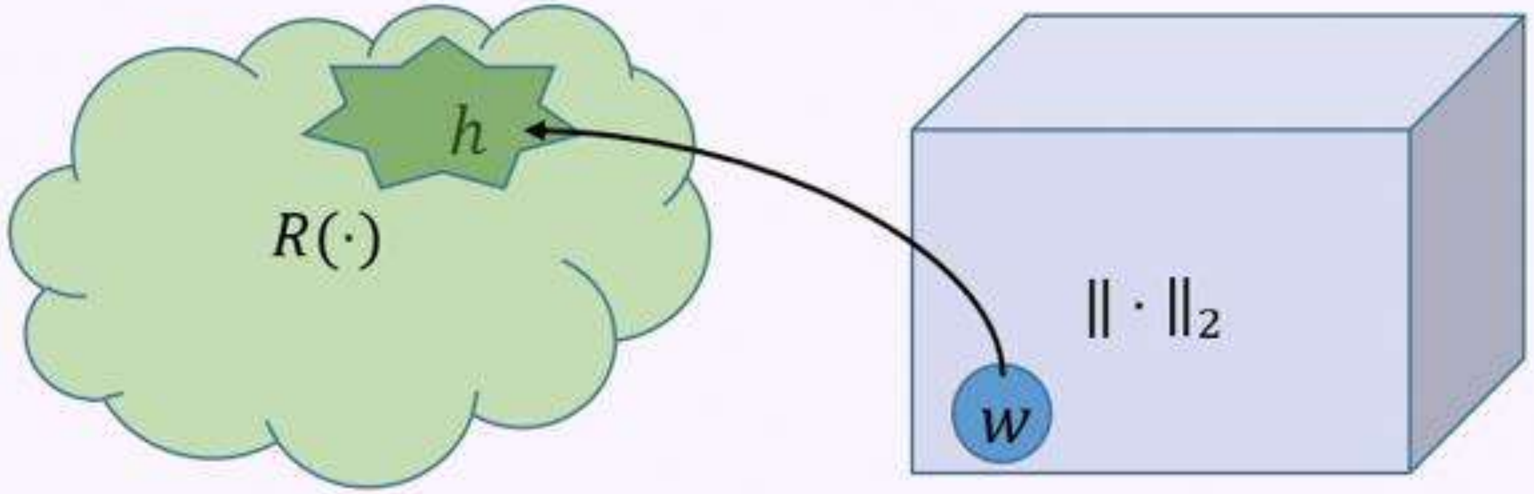
$$\arg\min R_F(h)\; s.t.\, y_i f(x_i) \geq 1$$

(**)

But need to be careful: f.o.s.p of **(*)** does ***not*** imply f.o.s.p of **(**)**

Is implicit bias of GD just ℓ_2 in param space + mapping to func space?

- Maybe yes? Implicit bias given by $R(h) = \arg\min_{F(w)=h} \|w\|_2$?
  - Matrix factorization, $R(\beta)^2 \propto \|\beta\|_{tr}$
  - Linear conv nets, diagonal nets with exp loss $R(\beta) \propto \|\beta\|_{2/D}$
  - Generic result for homogenous
- But…
  - Do those conditions actually hold?
  - Transition from f.o.s.p in w to f.o.s.p/local opt of $R(h)$
  - [Arora Cohen Hu Luo 2019]: with squared loss, implicit reg of diagonal nets $\propto \|\beta\|_1$ for any depth $D \geq 2$!



Doesn’t it all boil down to the NTK?

Is it all just a Kernel?

In kernel regime, training behaves according to 1st order approximation about $w^{(0)}$:

$$\boldsymbol{f}(\boldsymbol{w}, \boldsymbol{x}) \approx \boldsymbol{h}_{\boldsymbol{w}^{(0)}}(\boldsymbol{x}) + \langle \boldsymbol{w}, \boldsymbol{\phi_0}(\boldsymbol{x}) \rangle$$

where: $\phi_w = \nabla_w f(w, x)$ corresponding to

$$K_w(x, x') = \langle \nabla_w f(w, x), \nabla_w f(w, x') \rangle$$

Is it all just a Kernel?

In kernel regime, training behaves according to 1st order approximation about $w^{(0)}$:

$$\boldsymbol{f}(\boldsymbol{w},\boldsymbol{x}) \approx \boldsymbol{h}_{\boldsymbol{w}^{(0)}}(\boldsymbol{x}) + \langle \boldsymbol{w}, \boldsymbol{\phi_0}(\boldsymbol{x})\rangle$$

where: $\phi_w = \nabla_w f(w,x)$ corresponding to

$$K_w(x,x') = \langle \nabla_w f(w,x), \nabla_w f(w,x')\rangle$$

This suggests GD converges to:

- For squared loss: $\mathbf{arg\,min}\left\|\boldsymbol{h} - h_{w^{(0)}}\right\|_{\boldsymbol{K}}\ \boldsymbol{s.t.}\ \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{y_i}$

(we will focus on "unbiased initialization": $h_{w^{(0)}} = F\left(w^{(0)}\right) = 0$)

- For logistic (or exp) loss: $\propto \mathbf{arg\,min}\|\boldsymbol{h}\|_{\boldsymbol{K}}\ \boldsymbol{s.t.}\ \boldsymbol{y_i}\boldsymbol{h}(\boldsymbol{x_i}) \geq \mathbf{1}$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

Is it all just a Kernel?

In kernel regime, training behaves according to 1st order approximation about $w^{(0)}$:

$$\boldsymbol{f}(\boldsymbol{w}, \boldsymbol{x}) \approx \boldsymbol{h}_{\boldsymbol{w}^{(0)}}(\boldsymbol{x}) + \langle \boldsymbol{w}, \boldsymbol{\phi_0}(\boldsymbol{x}) \rangle$$

where: $\phi_w = \nabla_w f(w, x)$ corresponding to

$$K_w(x, x') = \langle \nabla_w f(w, x), \nabla_w f(w, x') \rangle$$

This suggests GD converges to:

- For squared loss: $\mathbf{arg\,min} \left\| \boldsymbol{h} - h_{w^{(0)}} \right\|_{\boldsymbol{K}} \; \boldsymbol{s.t.}\, \boldsymbol{h}(\boldsymbol{x}_i) = \boldsymbol{y}_i$

(we will focus on “unbiased initialization”: $h_{w^{(0)}} = F\left(w^{(0)}\right) = 0$)

- For logistic (or exp) loss: $\propto \mathbf{arg\,min} \|\boldsymbol{h}\|_{\boldsymbol{K}} \; \boldsymbol{s.t.}\, \boldsymbol{y}_i \boldsymbol{h}(\boldsymbol{x}_i) \geq \mathbf{1}$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

Is it all just a Kernel?

In kernel regime, training behaves according to 1st order approximation about $w^{(0)}$:

$$\boldsymbol{f}(\boldsymbol{w}, \boldsymbol{x}) \approx \boldsymbol{h}_{\boldsymbol{w}^{(0)}}(\boldsymbol{x}) + \langle \boldsymbol{w}, \boldsymbol{\phi}_{\mathbf{0}}(\boldsymbol{x}) \rangle$$

where: $\phi_w = \nabla_w f(w, x)$ corresponding to

$$K_w(x, x') = \langle \nabla_w f(w, x), \nabla_w f(w, x') \rangle$$

This suggests GD converges to:

- For squared loss: $\mathbf{arg\,min} \left\| \boldsymbol{h} - h_{w^{(0)}} \right\|_{\boldsymbol{K}} \; \boldsymbol{s.t.}\, \boldsymbol{h}(\boldsymbol{x}_i) = \boldsymbol{y}_i$

(we will focus on "unbiased initialization": $h_{w^{(0)}} = F\left(w^{(0)}\right) = 0$)

- For logistic (or exp) loss: $\propto \mathbf{arg\,min} \|\boldsymbol{h}\|_{\boldsymbol{K}} \; \boldsymbol{s.t.}\, \boldsymbol{y}_i \boldsymbol{h}(\boldsymbol{x}_i) \geq \mathbf{1}$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

We are interested in $w_\alpha(\infty) = \lim_{t\to\infty} w_\alpha(t)$ and $h_\alpha(\infty) = F\big(w_\alpha(\infty)\big)$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

We are interested in $w_\alpha(\infty) = \lim_{t\to\infty} w_\alpha(t)$ and $h_\alpha(\infty) = F\big(w_\alpha(\infty)\big)$

- When will this behave like the kernel gradient flow:

$$\dot{w}_K = -\nabla L_S(x \mapsto \langle w, \phi_0(x)\rangle) \quad \text{and} \quad w(0) = w_0$$

where $\phi_0(x) = \nabla f(w_0, x)$ with associated kernel K_0

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

We are interested in $w_\alpha(\infty) = \lim_{t\to\infty} w_\alpha(t)$ and $h_\alpha(\infty) = F\big(w_\alpha(\infty)\big)$

- When will this behave like the kernel gradient flow:

$$\dot{w}_K = -\nabla L_S(x \mapsto \langle w, \phi_0(x)\rangle) \quad \text{and} \quad w(0) = w_0$$

where $\phi_0(x) = \nabla f(w_0, x)$ with associated kernel K_0

- For squared loss, under some conditions [Chizat and Bach 18]:

$$\lim_{\alpha\to\infty} \sup_t \left\| \frac{1}{\alpha} w_\alpha\left(\frac{1}{\alpha^{D-1}} t\right) - w_K(t) \right\| = 0$$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

We are interested in $w_\alpha(\infty) = \lim_{t\to\infty} w_\alpha(t)$ and $h_\alpha(\infty) = F\big(w_\alpha(\infty)\big)$

- When will this behave like the kernel gradient flow:

$$\dot{w}_K = -\nabla L_S(x \mapsto \langle w, \phi_0(x)\rangle) \quad \text{and} \quad w(0) = w_0$$

where $\phi_0(x) = \nabla f(w_0, x)$ with associated kernel K_0

- For squared loss, under some conditions [Chizat and Bach 18]:

$$\lim_{\alpha\to\infty} \sup_t \left\| \frac{1}{\alpha} w_\alpha\left(\frac{1}{\alpha^{D-1}} t\right) - w_K(t) \right\| = 0$$

$$\Rightarrow \frac{1}{\alpha} h_\alpha(\infty) \to \hat{h}_{K_0} = \arg\min \|h\|_{K_0} \text{ s.t. } h(x_i) = y_i$$

Kernel Regime and Scale of Init

- For D-homogenous model, consider gradient flow with:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \text{ with unbiased } F(w_0) = 0$$

We are interested in $w_\alpha(\infty) = \lim_{t\to\infty} w_\alpha(t)$ and $h_\alpha(\infty) = F\big(w_\alpha(\infty)\big)$

- When will this behave like the kernel gradient flow:

$$\dot{w}_K = -\nabla L_S(x \mapsto \langle w, \phi_0(x)\rangle) \quad \text{and} \quad w(0) = w_0$$

where $\phi_0(x) = \nabla f(w_0, x)$ with associated kernel K_0

- For squared loss, under some conditions [Chizat and Bach 18]:

$$\lim_{\alpha\to\infty} \sup_t \left\| \frac{1}{\alpha} w_\alpha\left(\frac{1}{\alpha^{D-1}} t\right) - w_K(t) \right\| = 0$$

➔ $\frac{1}{\alpha} h_\alpha(\infty) \to \hat{h}_{K_0} = \arg\min \|h\|_{K_0} \text{ s.t. } h(x_i) = y_i$

- But: when $\alpha \to 0$, we got interesting, non-RKHS inductive bias (e.g. nuclear norm, sparsity)

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w, x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right) x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w,x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right) x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha(0) = \alpha \mathbf{1}$ (so that $\beta_\alpha(0) = F\left(w_\alpha(0)\right) = 0$).

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w,x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right) x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha(0) = \alpha \mathbf{1}$ (so that $\beta_\alpha(0) = F\left(w_\alpha(0)\right) = 0$).

What's the implicit bias of grad flow w.r.t square loss $L_S(w) = \|XF(w) - y\|_2^2$?

$$\beta_\alpha(\infty) = \lim_{t \to \infty} F(w_\alpha(t))$$

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w,x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right)x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha(0) = \alpha\mathbf{1}$ (so that $\beta_\alpha(0) = F\left(w_\alpha(0)\right) = 0$).

What's the implicit bias of grad flow w.r.t square loss $L_S(w) = \|XF(w) - y\|_2^2$?

$$\beta_\alpha(\infty) = \lim_{t\to\infty} F(w_\alpha(t))$$

In Kernel Regime $\alpha \to \infty$: $K_0(x,x') = 4\langle x,x'\rangle$ and so

$$\beta_\alpha(\infty) \to \hat{\beta}_{L2} = \arg\min_{X\beta=y} \|\beta\|_2$$

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w,x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right) x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha(0) = \alpha \mathbf{1}$ (so that $\beta_\alpha(0) = F\left(w_\alpha(0)\right) = 0$).

What's the implicit bias of grad flow w.r.t square loss $L_S(w) = \|XF(w) - y\|_2^2$?

$$\beta_\alpha(\infty) = \lim_{t \to \infty} F(w_\alpha(t))$$

In Kernel Regime $\alpha \to \infty$: $K_0(x, x') = 4\langle x, x' \rangle$ and so

$$\beta_\alpha(\infty) \to \hat{\beta}_{L2} = \arg\min_{X\beta = y} \|\beta\|_\mathbf{2}$$

In Deep Regime $\alpha \to 0$: special case of MF with commutative measurements

$$\beta_\alpha(\infty) \to \hat{\beta}_{L1} = \arg\min_{X\beta = y} \|\beta\|_\mathbf{1}$$

Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w,x) = \sum_j \left(w_+[j]^2 - w_-[j]^2\right) x_i \quad \text{i.e.} \quad F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha(0) = \alpha\mathbf{1}$ (so that $\beta_\alpha(0) = F\left(w_\alpha(0)\right) = 0$).

What's the implicit bias of grad flow w.r.t square loss $L_S(w) = \|XF(w) - y\|_2^2$?

$$\beta_\alpha(\infty) = \lim_{t\to\infty} F(w_\alpha(t))$$

In Kernel Regime $\alpha \to \infty$: $K_0(x,x') = 4\langle x, x'\rangle$ and so

$$\beta_\alpha(\infty) \to \hat{\beta}_{L2} = \arg\min_{X\beta=y} \|\beta\|_{\mathbf{2}}$$

In Deep Regime $\alpha \to 0$: special case of MF with commutative measurements

$$\beta_\alpha(\infty) \to \hat{\beta}_{L1} = \arg\min_{X\beta=y} \|\beta\|_{\mathbf{1}}$$

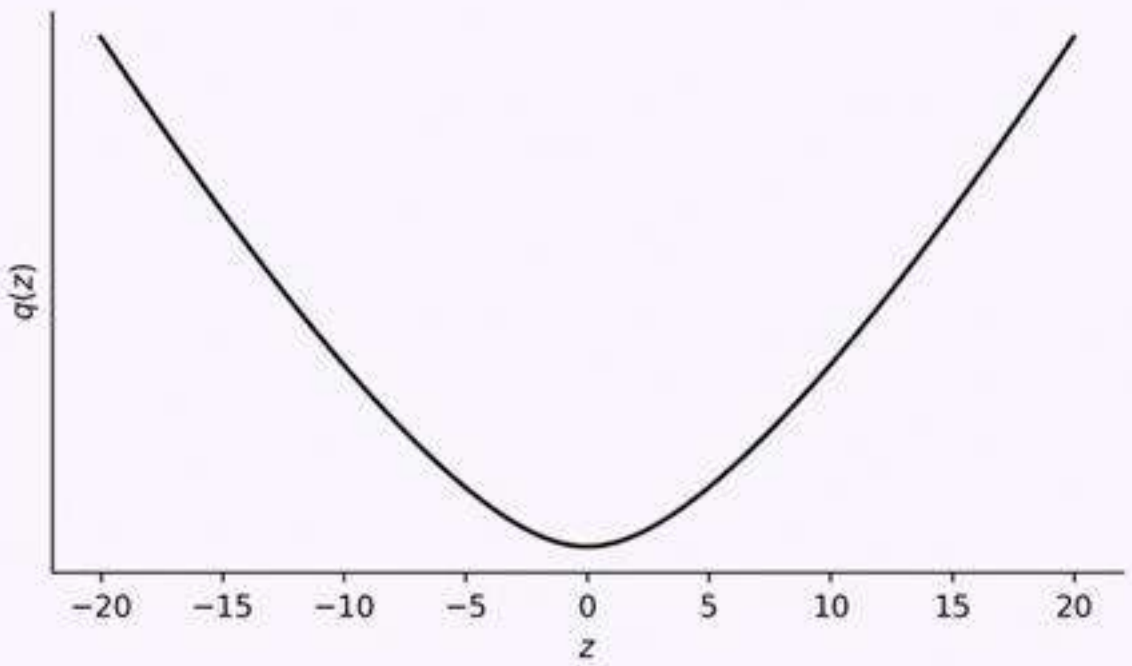
For any α: $\quad \beta_\alpha(\infty) = \arg\min_{X\beta=y} Q_\alpha(\beta)$

where $Q_\alpha(\beta) = \sum_j q\left(\frac{\beta[j]}{\alpha^2}\right)$ and $q(b) = 2 - \sqrt{4-b^2} + b\sinh^{-1}\left(\frac{b}{2}\right)$

Interpolating between Kernel and Deep

$$\beta_\alpha(\infty) = \arg\min_{X\beta=y} Q_\alpha(\beta)$$

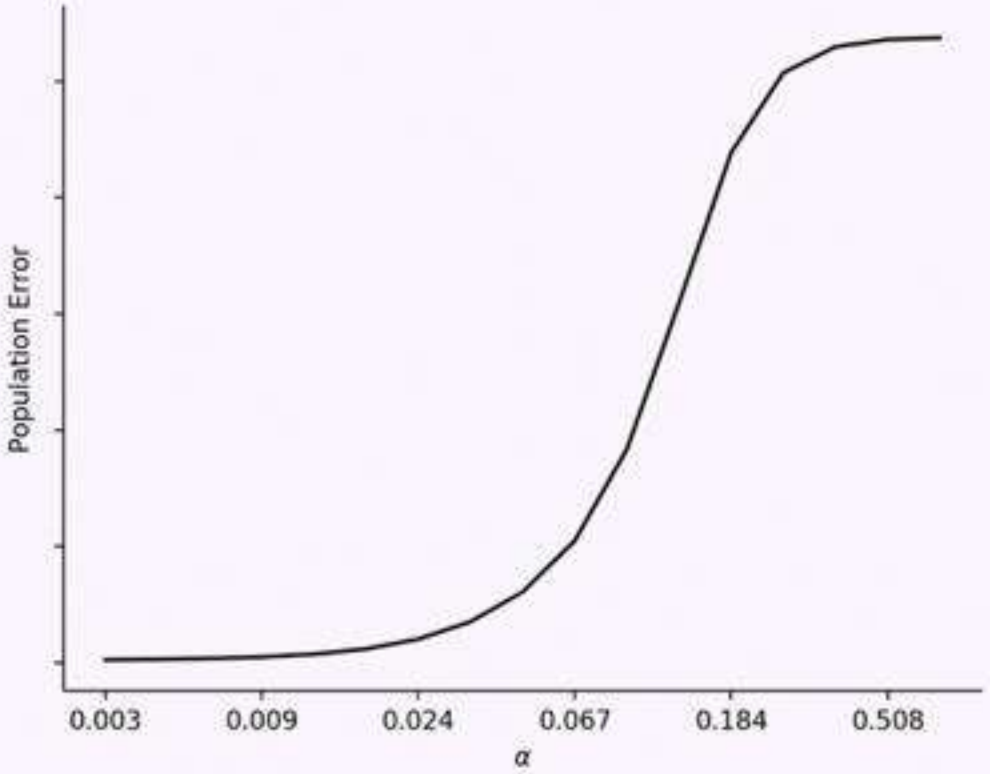
where $Q_\alpha(\beta) = \sum_j q\left(\frac{\beta[j]}{\alpha^2}\right)$ and $q(b) = 2 - \sqrt{4 - b^2} + b\sinh^{-1}\left(\frac{b}{2}\right)$

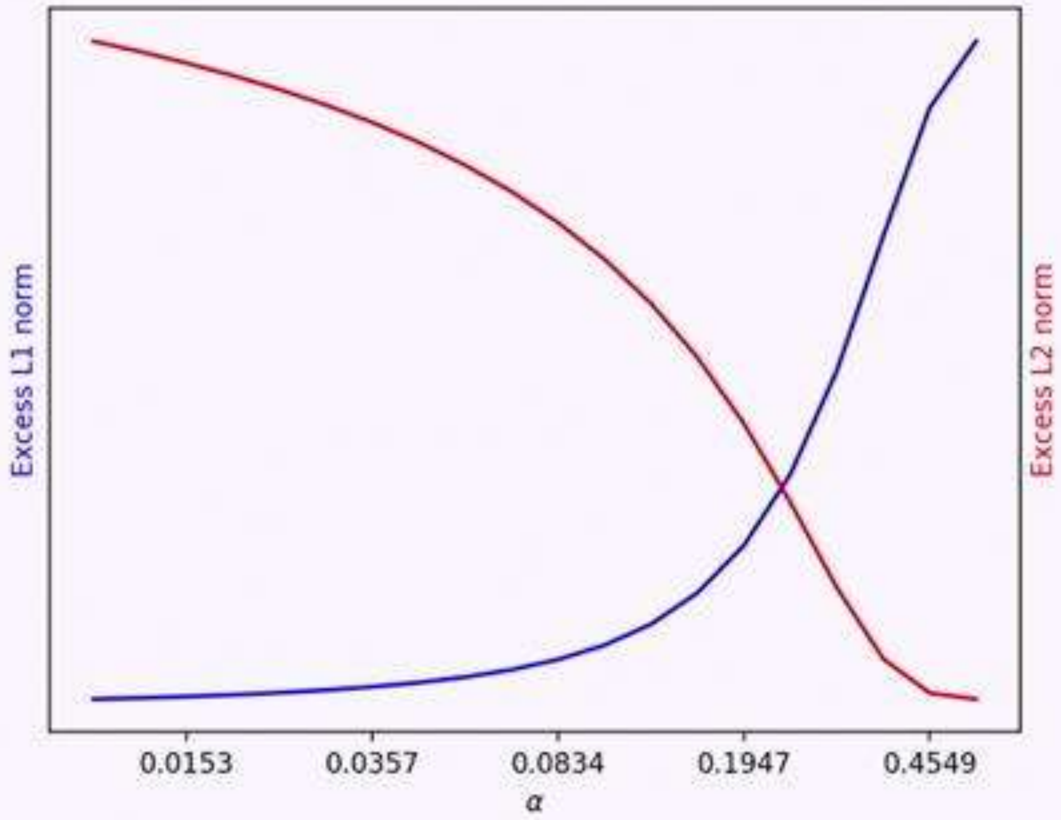


Induced dynamics: $\dot{\beta}_\alpha = -\sqrt{\beta_\alpha^2 + 4\alpha^4} \odot \nabla L_s(\beta_\alpha)$

Sparse Learning

$$y_i = \langle \beta^*, x_i \rangle + N(0,0.01)$$
$$d = 1000, \qquad \|\beta^*\|_0 = 5, \qquad m = 100$$

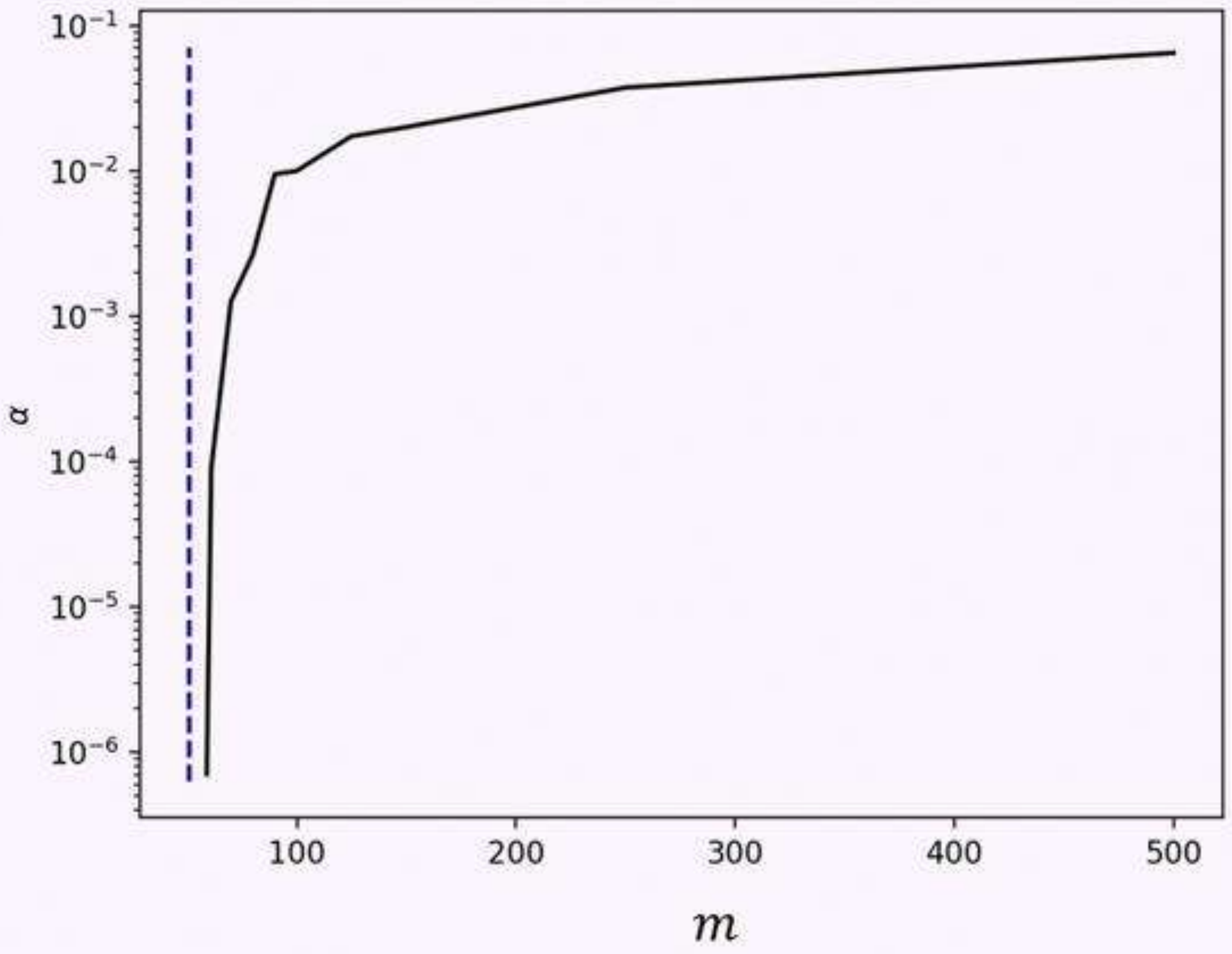




Sparse Learning

$$y_i = \langle \beta^*, x_i \rangle + N(0,0.01)$$
$$d = 1000, \qquad \|\beta^*\|_0 = k$$

How small does α need to be to get $L\big(\beta_\alpha(\infty)\big) < 0.025$



Relationship to Explicit Reg

Is initializing to $w(0) = \alpha\mathbf{1}$ the same as regularizing distance to $\alpha\mathbf{1}$?

$$\beta_\alpha^R = F\left(\arg\min_{L_S(w)=0} \|w - \alpha\mathbf{1}\|_2^2\right) = \arg\min_{X\beta=y} R_\alpha(\beta)$$

Where $R_\alpha(\beta) = \min_{F(w)=\beta} \|w - \alpha\mathbf{1}\|_2^2$

Relationship to Explicit Reg

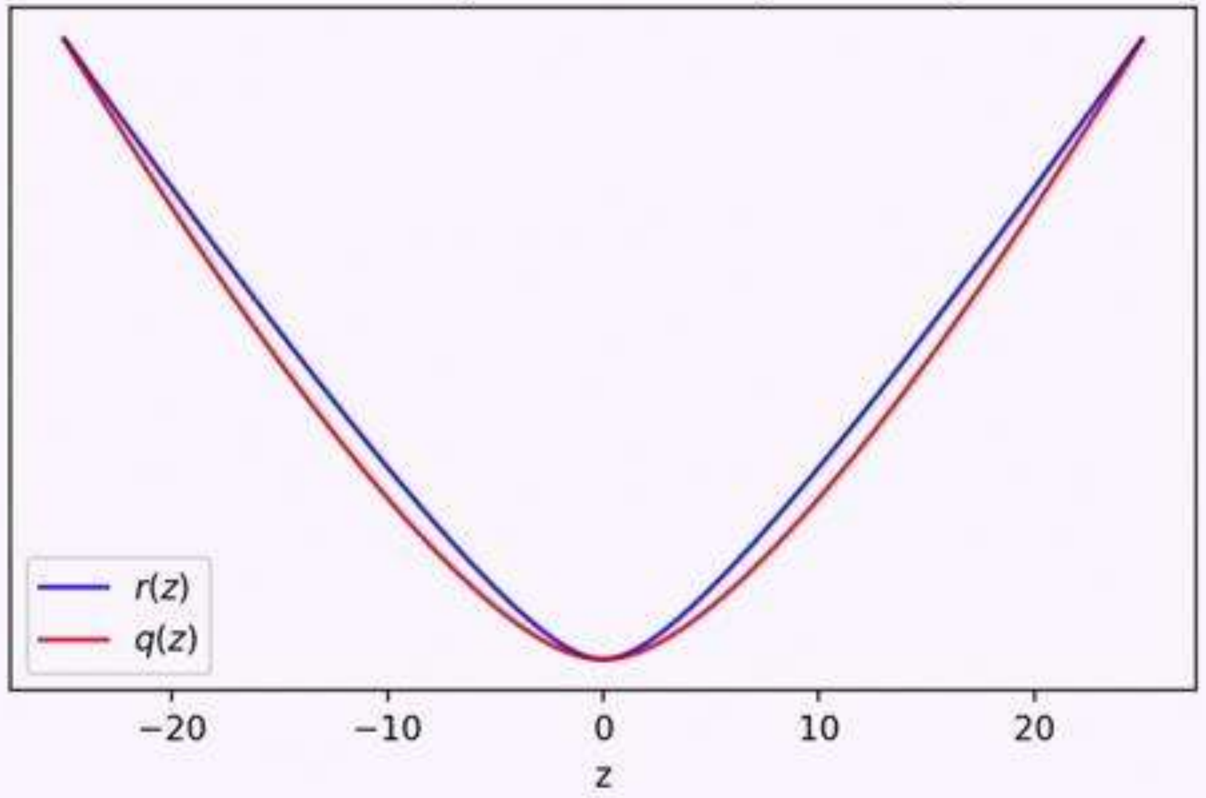
Is initializing to $w(0) = \alpha\mathbf{1}$ the same as regularizing distance to $\alpha\mathbf{1}$?

$$\beta_\alpha^R = F\left(\arg\min_{L_S(w)=0} \|w - \alpha\mathbf{1}\|_2^2\right) = \arg\min_{X\beta=y} R_\alpha(\beta)$$

Where $R_\alpha(\beta) = \min_{F(w)=\beta} \|w - \alpha\mathbf{1}\|_2^2$

$R_\alpha(\beta) = \sum_j r\left(\frac{\beta[j]}{\alpha^2}\right)$ where $r(b)$ is solution of quartic equation:

$$r^4 - 6r^3 + (12 - 2b^2)r^2 - (8 + 10b^2)r + b^2 + b^4 = 0$$

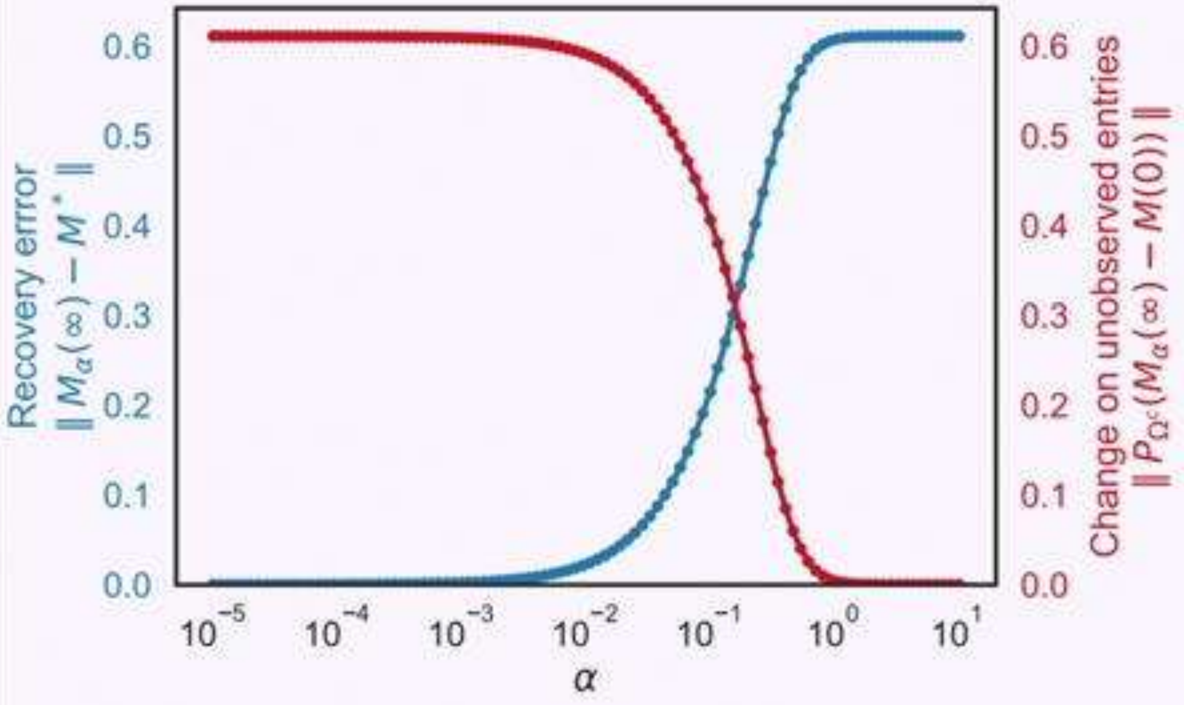


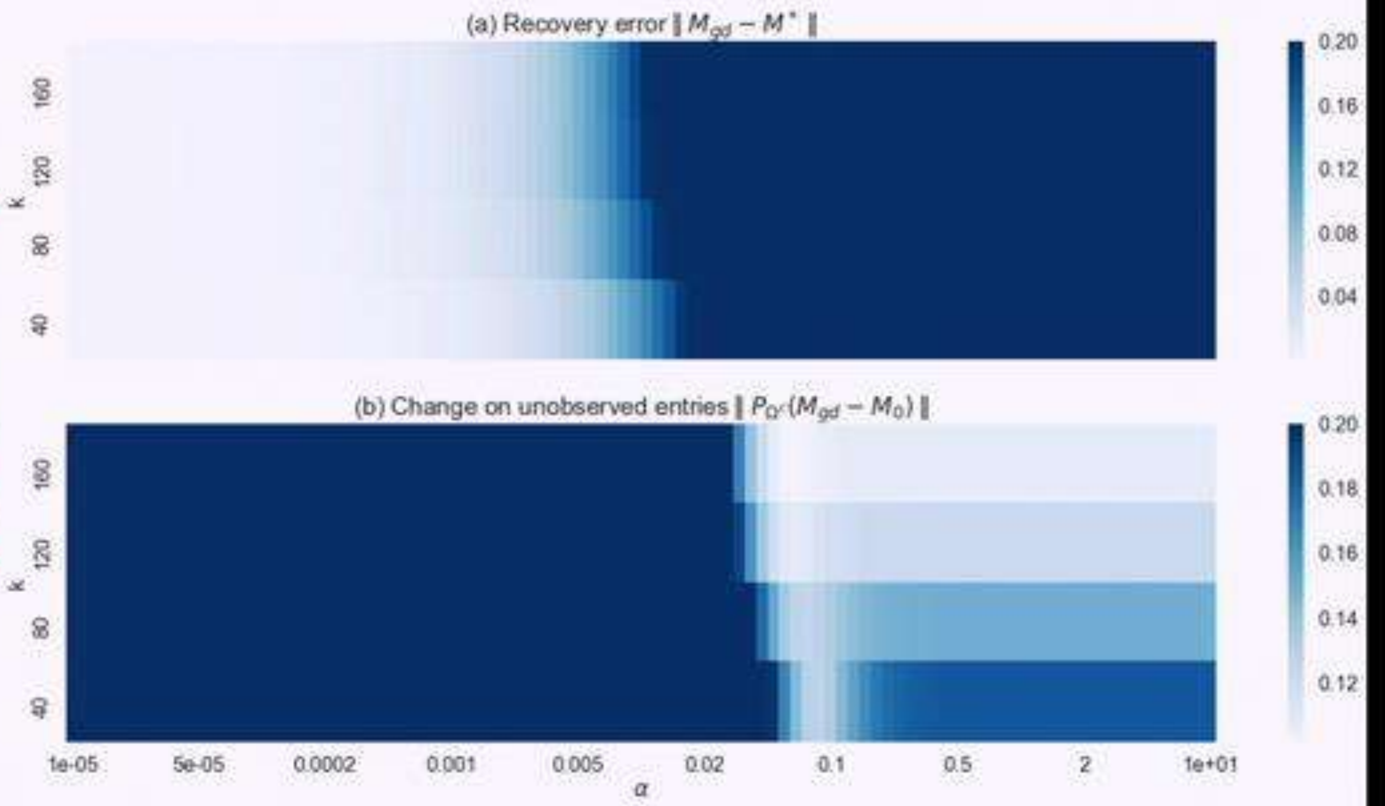
Kernel and Deep Regimes in Matrix Completion

$$f\big((U,V),(i,j)\big) = \big(UV^\top\big)_{ij}$$

Tangent kernel: $K_{U,V}\big((i,j),(i',j')\big) = \delta_{i=i'}\left\langle V_j, V_{j'}\right\rangle + \delta_{j=j'}\langle U_i, U_{i'}\rangle$

For orthogonal initialization: $K_0\big((i,j),(i',j')\big) \propto \delta_{(i,j)=(i',j')}$





Squared Loss vs Exp Loss

- Back to general D-homogenous model

- For squared loss, under some conditions [Chizat and Bach 18]:

$$\lim_{\alpha\to\infty}\sup_t \left\| \frac{1}{\alpha} w_\alpha\left(\frac{1}{\alpha^{D-1}} t\right) - w_K(t) \right\| = 0$$

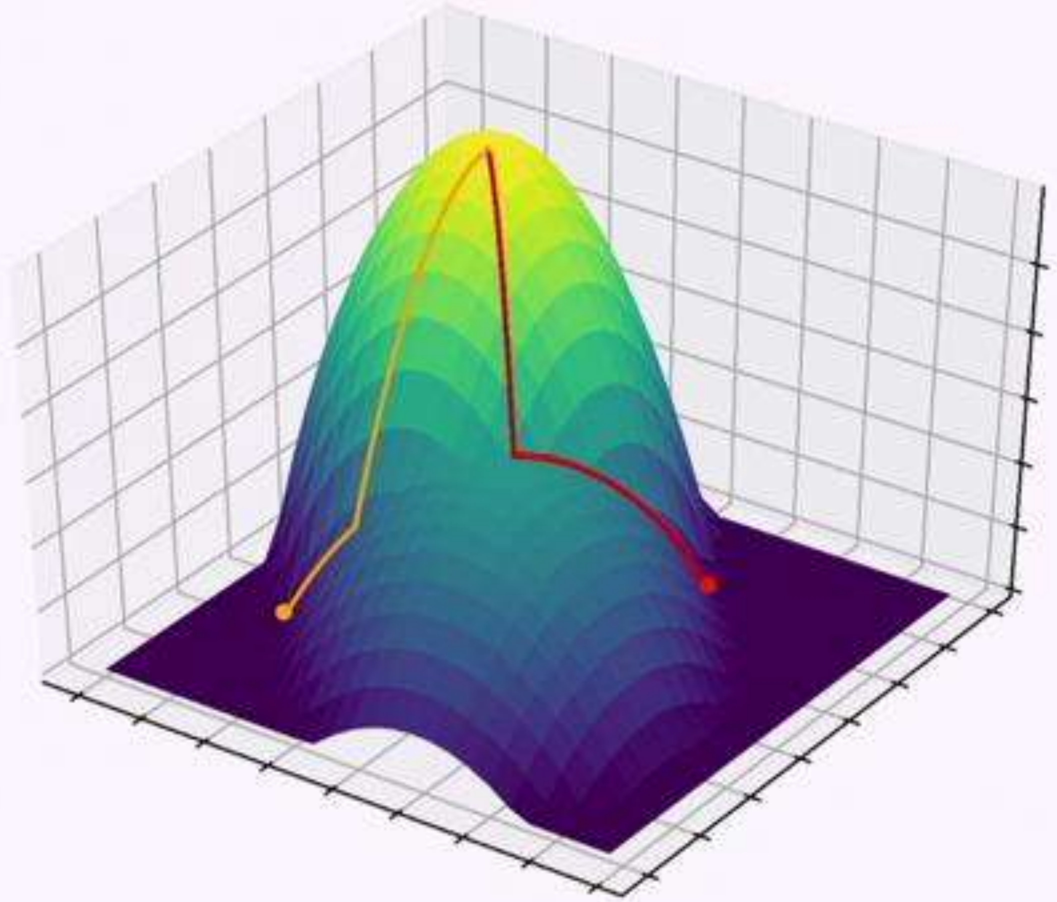
$$\Rightarrow \frac{1}{\alpha} h_\alpha(\infty) \to \hat{h}_K = \arg\min \|h\|_K \text{ s.t. } h(x_i) = y_i$$

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties



Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum