Universality and limitations of deep learning

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

MSR, 08.19
Class 1

Class 2

Apples

Bananas
Class 1

odd

Class 2

even
Class 1

odd

Class 2

even

Class 1

odd

Class 2

even

Class 1

odd

Class 2

even

Class 1

odd

Class 2

even

Class 1

odd

Class 2

even

\[ \psi(X) = 1 \text{ if and only if } \sum_{\varphi \in \Phi} \alpha_{\varphi} \varphi(X) > \theta. \]
u/mattfyles: ResNet-50 neural network from Microsoft Research.
~3 million nodes and ~10 million edges (layout in Gephi)

https://i.reddit.it/55/f8b8h6uz.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 \( \text{poly}(n) \)-time can be expressed by a \( \text{poly}(n) \)-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 $\text{poly}(n)$-time can be expressed by a $\text{poly}(n)$-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 $\text{poly}(n)$-time can be expressed by a $\text{poly}(n)$-size NN [Parberry 94, Sipser 06]

**Estimation.** $\text{poly}(n)$-size NN can be learned with empirical risk minimization (ERM) with $\text{poly}(n)$-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 \( \text{poly}(n) \)-time can be expressed by a \( \text{poly}(n) \)-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 \( \text{poly}(n) \)-time can be expressed by a \( \text{poly}(n) \)-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: \( f_S(x) = \prod_{i \in S} x_i \), \( S \sim U [2^n] \)

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

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

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

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

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

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

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

- $F \sim P_\mathcal{Y}$
- 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 \( \mathcal{F} = \mathcal{Y} \times \mathcal{X} \)

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

**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  
\[ \mathbb{P}(\hat{F}^{(t)}(X_{t+1}) = F(X_{t+1})) = \frac{1}{2} + \Omega_n(1) \]

**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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2 \]

\[(X, F, F') \sim P_X \times P_F \times P_F\]
Theorem 1. GD cannot learn efficiently function distributions having low CP

\[ \text{CP}(P_X, P_{\mathcal{F}}) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2 \]

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

low = super-poly decay in \(n\)
Theorem 1. GD cannot learn efficiently function distributions having low CP

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

$\text{CP}(P_X, P_F) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X) F'(X))^2$

$(X, F, F') \sim P_X \times P_F \times P_F$

low = super-poly decay in $n$
Theorem 1. GD cannot learn efficiently function distributions having low CP

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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X)F'(X))^2 \]

\[ (X, F, F') \sim P_X \times P_F \times P_F \]

low = super-poly decay in n
related to statistical dim. [Kearns 98, Blum et al. 01]
Theorem 1. GD cannot learn efficiently function distributions having low CP

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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X) F'(X))^2 \]

\( X, F, F' \sim P_X \times P_F \times P_F \)

\text{low} = \text{super-poly decay in } n

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

“average-case” v.s. “worst-case” SQ (E. Boix)
Theorem 1. GD cannot learn efficiently function distributions having low CP

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2 \]

\[ (X, F, F') \sim P_X \times P_F \times P_F \]

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

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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X)F'(X))^2 \]
\((X, F, F') \sim P_X \times P_F \times P_F\)

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

\[ CP(P_X, P_F) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X) F'(X))^2 \]

\[ (X, F, F') \sim P_X \times P_F \times P_F \]

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
Theorem 2. SGD can learn efficiently any efficiently learnable distribution \((P_X, P_X)\) that can be learned by some algorithm in poly-time with poly-samples.
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 \((P_X, P_F)\) that can be learned by some algorithm in poly-time with poly-samples
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 \( (P_X, P_F) \) 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 \( P_X, P_F \) 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 $\mathcal{F} = \mathcal{Y} \times \mathcal{X}$

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 $\mathbb{P}(\hat{F}^{(t)}(X_{t+1}) = F(X_{t+1})) = 1/2 + \Omega_n(1)$

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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X) F'(X))^2 \]

\[ (X, F, F') \sim P_X \times P_F \times P_F \]

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

\[ \text{CP}(P_X, P_{\mathcal{F}}) = \mathbb{E}_{F, F'}(\mathbb{E}_X F(X)F'(X))^2 \]

\[ (X, F, F') \sim P_X \times P_{\mathcal{F}} \times P_{\mathcal{F}} \]

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.
Theorem 2. SGD can learn efficiently any efficiently learnable distribution \((P_X, P_F)\) 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

\[ CP(P_X, P_F) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X) F'(X))^2 \]

\((X, F, F') \sim P_X \times P_F \times P_F\)

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

\[ \text{CP}(P_X, P_F) = \mathbb{E}_{F,F'}(\mathbb{E}_X F(X)F'(X))^2 \]
\[ (X, F, F') \sim P_X \times P_F \times P_F \]
\[ \text{low} = \text{super-poly decay in } n \]
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 \((P_X, P_F)\) 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

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

\[ W^{(0)} = W_F^{(0)} = W_\star^{(0)} \quad D_F: \text{true data} \quad D_\star: \text{junk data} \quad \text{(Gradient descent)} \]
CP: information-theoretic argument

\[ W_H^{(t)} = W_H^{(t-1)} - \gamma E_{(X,Y) \sim D_H} \nabla L(W_H^{(t-1)}(X), Y) + Z_{\sigma}^{(t)} \quad H \in \{ F, \ast \} \]

\[ W^{(0)} = W_F^{(0)} = W_{\ast}^{(0)} \quad D_F: \text{true data} \quad D_{\ast}: \text{junk data} \quad \text{(Gradient descent)} \]

\[ P\{W_F^{(T)}(X) = F(X)\} \leq P\{W_{\ast}^{(T)}(X) = F(X)\} + \mathbb{E}_F d(Q_F^{(T)}, Q_{\ast}^{(T)})_{TV} \]

\[ 1/2 \]

(Total-variation bound)
CP: information-theoretic argument

\[ W_H^{(t)} = W_H^{(t-1)} - \gamma\mathbb{E}_{(X,Y) \sim D_H} \nabla L(W_H^{(t-1)}(X), Y) + Z^{(t)}_\sigma \quad H \in \{F, \ast\} \]

\[ W^{(0)} = W_F^{(0)} = W_\ast^{(0)} \quad D_F: \text{true data} \quad D_\ast: \text{junk data} \quad \text{(Gradient descent)} \]

\[
\mathbb{P}\{W_F^{(T)}(X) = F(X)\} \leq \mathbb{P}\{W_\ast^{(T)}(X) = F(X)\} + \mathbb{E}_F d(Q_F^{(T)}, Q_\ast^{(T)})_{TV} \]

\[
\frac{1}{2} \quad \text{(Total-variation bound)}
\]

\[
\mathbb{E}_F d(Q_F^{(T)}, Q_\ast^{(T)})_{TV} \leq T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\ast^{(1)})_{TV} \quad \text{(Data-processing + triangular inequality)}
\]
CP: information-theoretic argument

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

\[ W^{(0)} = W_F^{(0)} = W_*^{(0)} \quad D_F: \text{true data} \quad D_*: \text{junk data} \quad \text{(Gradient descent)} \]

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

\( \text{(Total-variation bound)} \)

\[
\mathbb{E}_F d(Q_F^{(T)}, Q_*^{(T)})_{TV} \leq T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_*^{(1)})_{TV} \quad \text{(Data-processing + triangular inequality)}
\]

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

\( \text{( Pinsker's inequality)} \)
CP: information-theoretic argument

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

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

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

$$\left(\text{Total-variation bound}\right)$$

$$\mathbb{E}_F d(Q_F^{(T)}, Q_\ast^{(T)})_{TV} \leq T \cdot \mathbb{E}_F d(Q_F^{(1)}, Q_\ast^{(1)})_{TV} \quad \text{(Data-processing + triangular inequality)}$$

$$\mathbb{E}_F d(Q_F, Q_\ast)_{TV} \leq \frac{\gamma}{2\sigma} \left(\mathbb{E}_F \|\mathbb{E}_D F \nabla - \mathbb{E}_{D_\ast} \nabla\|_2^2\right)^{1/2} \quad \text{(Pinsker's inequality)}$$

$$\mathbb{E}_F \left(\mathbb{E}_D F \nabla \mathcal{E} - \mathbb{E}_{D_\ast} \nabla \mathcal{E}\right)^2 = \mathbb{E}_F \left(\mathbb{E}_{D_\ast} \nabla \mathcal{E} \left(1 - \frac{D_F}{D_\ast}\right)\right)^2 \quad \text{(Radon-Nikodym)}$$
CP: information-theoretic argument

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

\[ W^{(0)} = W_F^{(0)} = W_\star^{(0)} \]

\[ D_F: \text{true data} \quad D_\star: \text{junk data} \]

\( \mathbb{P}\{W_F^{(T)}(X) = F(X)\} \leq \mathbb{P}\{W_\star^{(T)}(X) = F(X)\} + \mathbb{E}_Fd(Q_F^{(T)}, Q_\star^{(T)})_{TV} \]

(Total-variation bound)

\[ \mathbb{E}_Fd(Q_F^{(T)}, Q_\star^{(T)})_{TV} \leq T \cdot \mathbb{E}_Fd(Q_F^{(1)}, Q_\star^{(1)})_{TV} \]

(Data-processing + triangular inequality)

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

(Pinsker's inequality)

\[ \mathbb{E}_F(\mathbb{E}_{D_F} \nabla e - \mathbb{E}_{D_\star} \nabla e)^2 = \mathbb{E}_F(\mathbb{E}_{D_\star} \nabla e (1 - D_F/D_\star))^2 \]

(Radon-Nikodym)

\[ = \mathbb{E}_F \mathbb{E}_{D_\star} \nabla e^2 (1 - D_F/D_\star)^2 \]

(Tensorization)
CP: information-theoretic argument

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

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

(Gradient descent)

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

(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)

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

(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)

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

(Tensorization)

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

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

\[
\text{JF} := \sum_{t=1}^{T} \gamma_t \| \mathbb{E}_{D_t} \nabla^{(t)} \|_2
\]

\[
\text{CP} := \mathbb{E}_{F,F'}(\mathbb{E}_X F(X) F(X'))^2
\]
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
Al 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

# of parameters $>>$ # training data
Motivation: overparameterization without overfitting

Mystery

# of parameters $\gg$ # training data

overfitting
Motivation: overparameterization without overfitting

- Mystery
  - # of parameters $\gg$ # training data

- Diagram:
  - Overfitting
  - Just right!
Motivation: overparameterization without overfitting

<table>
<thead>
<tr>
<th>Mystery</th>
</tr>
</thead>
<tbody>
<tr>
<td># of parameters $&gt;&gt;$ # training data</td>
</tr>
</tbody>
</table>

![Graph showing overfitting and just right fitting](image)

Challenges:
- **Optimization:** Why can neural nets fit to the training data?
- **Generalization:** Why can neural nets predict?
Experiment 1: 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 1: Overfitting to corruption

Add corruption

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

<table>
<thead>
<tr>
<th>Inputs</th>
<th>5</th>
<th>0</th>
<th>4</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training labels</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>True Labels</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Experiment I: Overfitting to corruption

Add corruption

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

Results after 200 epochs

Accuracy vs. % of label corruption

Inputs: 5 0 4 1
Training labels: 5 8 4 1
True Labels: 5 0 4 1

Train accuracy
Test accuracy
Train acc w.r.t. true 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

- Train accuracy
- Test accuracy
- Train acc w.r.t. true labels

% of label corruption
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

![Diagram showing inputs, training labels, true labels, and error over epochs for train and test cases.](image-url)
Theory for overparameterization without overfitting

- Optimization
- Generalization
- Early stopping
Optimization
One-hidden layer

\( y_i = v^T \phi(Wx_i) \)
Theory for smooth activations

- Data set \( \{(x_i, y_i)\}_{i=1}^{n} \in \mathbb{R}^d \times \mathbb{R} \)

- Loss

\[
\min_{v, W} \mathcal{L}(v, W) := \sum_{i=1}^{n} (v^T \phi(Wx_i) - y_i)^2
\]
Theory for smooth activations

- Data set \( \{ (x_i, y_i) \}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R} \)
- Loss \( \min_{v, W} \mathcal{L}(v, W) := \sum_{i=1}^n (v^T \phi(Wx_i) - y_i)^2 \)
- Run gradient descent
  \( (v_{\tau+1}, W_{\tau+1}) = (v_{\tau+1}, W_{\tau+1}) - \mu_\tau \nabla \mathcal{L}(v_{\tau}, W_{\tau}) \)
Theory for smooth activations

- Data set $\{(x_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$
- Loss
  $$\min_{v, W} \mathcal{L}(v, W) := \sum_{i=1}^n (v^T \phi(Wx_i) - y_i)^2$$
- Run gradient descent
  $$(v_{\tau+1}, W_{\tau+1}) = (v_{\tau+1}, W_{\tau+1}) - \mu \nabla \mathcal{L}(v_{\tau}, W_{\tau})$$

**Theorem**

**Assume**

- Distinct data points ($x_i \neq x_j$)
- Smooth activations e.g. $\phi(z) = \log(1 + e^z)$
- Overparameterization: $\# \text{ training data} \leq 2 \times \# \text{ width} \ (k \geq \frac{n}{2})$
- Initialization $v_0$ at random i.i.d. $\mathcal{N}(0, \nu^2)$ and $W_0$ i.i.d. $\mathcal{N}(0, 1)$ with $\nu >> 1$
Theory for smooth activations

- Data set \( \{ (x_i, y_i) \}_{i=1}^{n} \in \mathbb{R}^d \times \mathbb{R} \)
- Loss
  \[
  \min_{v, W} \mathcal{L}(v, W) := \sum_{i=1}^{n} (v^T \phi(Wx_i) - y_i)^2
  \]
- Run gradient descent
  \[
  (v_{\tau+1}, W_{\tau+1}) = (v_{\tau+1}, W_{\tau+1}) - \mu_{\tau} \nabla \mathcal{L}(v_{\tau}, W_{\tau})
  \]

**Theorem**

**Assume**
- Distinct data points \( x_i \neq x_j \)
- Smooth activations e.g. \( \phi(z) = \log(1 + e^z) \)
- Overparameterization: \( \# \) training data \( \leq 2 \times \# \) width \( (k \geq \frac{n}{2}) \)
- Initialization \( v_0 \) at random i.i.d. \( \mathcal{N}(0, \nu^2) \) and \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \) with \( \nu \gg 1 \)

**Then, with high probability**
- Zero training error: \( \mathcal{L}(v_{\tau}, W_{\tau}) \leq (1 - \rho)^{\tau} \mathcal{L}(v_0, W_0) \)
Theory for smooth activations

- Data set \( \{(x_i, y_i)\}_{i=1}^{n} \in \mathbb{R}^d \times \mathbb{R} \)
- Loss
  \[
  \min_{v, W} \mathcal{L}(v, W) := \sum_{i=1}^{n} (v^T \phi(Wx_i) - y_i)^2
  \]
- Run gradient descent
  \[
  (v_{\tau+1}, W_{\tau+1}) = (v_{\tau+1}, W_{\tau+1}) - \mu_{\tau} \nabla \mathcal{L}(v_{\tau}, W_{\tau})
  \]

**Theorem**

**Assume**
- Distinct data points \( (x_i \neq x_j) \)
- Smooth activations e.g. \( \phi(z) = \log(1 + e^z) \)
- Overparameterization: \( \# \) training data \( \leq 2 \times \# \) width \( (k \geq \frac{n}{2}) \)
- Initialization \( v_0 \) at random i.i.d. \( \mathcal{N}(0, \nu^2) \) and \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \) with \( \nu \gg 1 \)

**Then, with high probability**
- Zero training error: \( \mathcal{L}(v_{\tau}, W_{\tau}) \leq (1 - \rho)^T \mathcal{L}(v_0, W_0) \)

Possible extension: If \( \text{rank}(X) = d \), \( \# \) training data \( \leq 2 \times \# \) parameters*
Theory for smooth activations

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

Theorem

Assume
- Distinct data points \( (x_i \neq x_j) \)
- Smooth activations e.g. \( \phi(z) = \log(1 + e^z) \)
- Overparameterization: \# training data \( \leq 2 \times \# width \ (k \geq \frac{n}{2}) \)
- Initialization \( v_0 \) at random i.i.d. \( \mathcal{N}(0, \nu^2) \) and \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \) with \( \nu \gg 1 \)

Then, with high probability
- Zero training error: \( \mathcal{L}(v_{\tau}, W_{\tau}) \leq (1 - \rho^\tau) \mathcal{L}(v_0, W_0) \)

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

• Data: \((x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}^K\)

• Training Loss:
  \[ \mathcal{L}(W) := \frac{1}{2} \sum_{i=1}^{n} \| V \phi(W x_i) - y_i \|_{\ell_2}^2 \]

  Concatenate label and prediction vectors

  \[ y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad f(W) = \begin{bmatrix} V f(x_1; W) \\ \vdots \\ V f(x_n; W) \end{bmatrix} \in \mathbb{R}^{nK}. \]

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

• Algorithm: gradient descent from random initialization
  \[ W_{t+1} = W_t - \eta \nabla \mathcal{L}(W_t) \]
Model and training

- Data: \((x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}^K\)
- Training Loss:
  \[
  \mathcal{L}(W) := \frac{1}{2} \sum_{i=1}^{n} \| V \phi(Wx_i) - y_i \|_{\ell_2}^2
  \]
- Concatenate label and prediction vectors
  \[
  y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix},
  f(W) = \begin{bmatrix} Vf(x_1; W) \\ \vdots \\ Vf(x_n; W) \end{bmatrix} \in \mathbb{R}^{nK}.
  \]
  \[
  \min_{W \in \mathbb{R}^{k \times d}} \mathcal{L}(W) := \frac{1}{2} \| f(W) - y \|_{\ell_2}^2.
  \]
- Algorithm: gradient descent from random initialization
  \[
  W_{\tau+1} = W_\tau - \eta \nabla \mathcal{L}(W_\tau)
  \]
  \[
  \nabla \mathcal{L}(W) = J^T(W)(f(W) - y) \quad \text{with} \quad J(W) = \frac{\partial f(W)}{\partial \text{vect}(W)}.
  \]
Model and training

- **Data:** $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}^K$

- **Training Loss:**
  \[
  \mathcal{L}(W) := \frac{1}{2} \sum_{i=1}^{n} \| V \phi(W x_i) - y_i \|_2^2
  \]

  Concatenate label and prediction vectors

  \[
  y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad f(W) = \begin{bmatrix} V f(x_1; W) \\ \vdots \\ V f(x_n; W) \end{bmatrix} \in \mathbb{R}^{nK}.
  \]

  \[
  \min_{W \in \mathbb{R}^{k \times d}} \mathcal{L}(W) := \frac{1}{2} \| f(W) - y \|_2^2.
  \]

- **Algorithm:** Gradient descent from random initialization

  \[
  W_{\tau+1} = W_{\tau} - \eta \nabla \mathcal{L}(W_{\tau})
  \]

  \[
  \nabla \mathcal{L}(W) = J^T(W)(f(W) - y) \quad \text{with} \quad J(W) = \frac{\partial f(W)}{\partial \text{vec}(W)}.
  \]

- **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

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

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

Information and nuisance space of the Jacobian

Jacobian \( J \in \mathbb{R}^{nK \times p} \)

\[ J = \sum_{s=1}^{nK} \lambda_s u_s v_s^T = U \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{nK}) V^T \]

- **Information space**: \( \mathcal{I} = \text{span}\{u_s\}_{s=1}^r \)
- **Nuisance space**: \( \mathcal{N} = \text{span}\{u_s\}_{s=r+1}^n \)

Diagram:
- 1. Large singulars → Fast learning → Easy to generalize
- 2. Small singulars → Slow learning → Hard to generalize

Spectrum cut-off level at \( \alpha \).
Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK \log(K)}}$ and initialize $W_0$ 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 $W_0$ i.i.d. $\mathcal{N}(0, 1)$

Theorem

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

- Set $J = \left( \mathbb{E}[J(W_0) J^T(W_0)] \right)^{\frac{1}{2}}$ and $I$ based on cut-off $\alpha = \sqrt{nK \bar{\alpha}}$

- $k \geq \frac{\Gamma^d}{\alpha^5}$
Theory (random initialization)

Fix output layer to i.i.d. \( \frac{1}{\sqrt{kK \log(K)}} \) and initialize \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \)

**Theorem**

Fix number \( \tilde{\alpha} \geq 0 \) and \( \Gamma \geq 1 \).
- Set \( J = \left( \mathbb{E}[J(W_0)J^T(W_0)] \right)^{\frac{1}{2}} \) and \( I \) based on cut-off \( \alpha = \sqrt{nK} \tilde{\alpha} \)
- \( k \gtrsim \frac{\Gamma^4}{\tilde{\alpha}^8} \)

Then running gradient descent with \( T = \frac{\Gamma}{\tilde{\alpha}^8} \) iterations

\[
\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \frac{\Pi_{\mathcal{N}(y)}}{\sqrt{n}} + \frac{\Gamma}{\tilde{\alpha}} \frac{1}{\sqrt{n}}
\]

\( \underbrace{\text{bias}}_{\text{variance}} \)
Theory (random initialization)

Fix output layer to i.i.d. \( \frac{1}{\sqrt{kK \log(K)}} \) and initialize \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \)

**Theorem**

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

- Set \( J = \left( \mathbb{E}[J(W_0)J^T(W_0)] \right)^{1/2} \) and \( \mathcal{I} \) based on cut-off \( \alpha = \sqrt{nK\bar{\alpha}} \)
- \( k \geq \frac{\Gamma^4}{\alpha^8} \)

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

\[
\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \frac{\sum\mathcal{N}(y)}{\sqrt{n}} + \frac{\Gamma}{\bar{\alpha} \sqrt{n}}
\]

Bias \text{ and } Variance

![Graph showing the spectrum and classification of singular values](image-url)
Theory (random initialization)

Fix output layer to i.i.d. $\frac{1}{\sqrt{kK \log(K)}}$ and initialize $W_0$ i.i.d. $\mathcal{N}(0, 1)$

**Theorem**

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

- Set $J = \left( \mathbb{E}[J(W_0)J^T(W_0)] \right)^{\frac{1}{2}}$ and $I$ based on cut-off $\alpha = \sqrt{nK\tilde{\alpha}}$
- $k \succeq \frac{\Gamma^4}{\alpha^8}$

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

$$\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \eta_{\mathcal{N}(y)} + \frac{\Gamma}{\tilde{\alpha} \sqrt{n}}$$

- 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 \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \)

Theorem

Fix number \( \tilde{\alpha} \geq 0 \) and \( \Gamma \geq 1 \).

- Set \( \tilde{J} = \left( \mathbb{E}[\tilde{J}(W_0) \tilde{J}^T(W_0)] \right)^{\frac{1}{2}} \) and \( \tilde{I} \) based on cut-off \( \alpha = \sqrt{nK \tilde{\alpha}} \)
- \( k_i \geq \frac{\Gamma^4}{\tilde{\alpha}^5} \)

Then running gradient descent with \( T = \frac{\Gamma}{\tilde{\alpha}^8} \) iterations:

\[
\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \frac{\Pi_\mathcal{N}(y)}{\sqrt{n}} + \frac{\Gamma}{\tilde{\alpha} \sqrt{n}}
\]

- Structured datasets generalize easier and require smaller networks.
- With constant \( \tilde{\alpha} \), 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 \( W_0 \) i.i.d. \( \mathcal{N}(0, 1) \)

**Theorem**

Fix number \( \tilde{\alpha} \geq 0 \) and \( \Gamma \geq 1 \).

- Set \( J = \left( \mathbb{E}[J(W_0)J^T(W_0)] \right)^{\frac{1}{2}} \) and \( \mathcal{I} \) based on cut-off \( \alpha = \sqrt{nK \tilde{\alpha}} \)
- \( k \geq \frac{\Gamma^4}{\tilde{\alpha}^3} \)

Then running gradient descent with \( T = \frac{\Gamma}{\tilde{\alpha}^3} \) iterations,

\[
\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \frac{\Pi_{\mathcal{N}}(y)}{\sqrt{n}} + \frac{\Gamma}{\tilde{\alpha} \sqrt{n}}
\]

- Structured datasets generalize easier and require smaller networks.
- with constant \( \tilde{\alpha} \), 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.]

\[
\text{missclass error} \lesssim \sqrt{\frac{y^T(JJ^T)^{-1}y}{n}} \quad \text{with} \quad k \geq \frac{n^4 \log n}{\lambda_{\min}(JJ^T)}
\]
Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $W_0$. 
Theory (deterministic)

Fix output layer entries bounded \( \frac{1}{\sqrt{kK}} \) and initialize at a deterministic point \( W_0 \).

**Theorem**

Fix number \( \tilde{\alpha} \geq 0 \) and \( \Gamma \geq 1 \).

- Set \( J = J(W_0) \) and \( I \) based on cut-off \( \alpha = \sqrt{nK \tilde{\alpha}} \)
- \( k \geq \frac{\Gamma^d}{\tilde{\alpha}^8} \)

Then running gradient descent with \( T = \frac{\Gamma}{\tilde{\alpha}^8} \) iterations

\[
\text{missclass}(f(W_T)) \lesssim e^{-T} + \frac{\Pi_N f(W_0) - y}{\sqrt{n}} + \frac{\Gamma}{\tilde{\alpha} \sqrt{n}}
\]

- **bias**
- **variance**
Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $W_0$

Theorem

Fix number $\tilde{\alpha} \geq 0$ and $\Gamma \geq 1$.
- Set $J = J(W_0)$ and $I$ based on cut-off $\alpha = \sqrt{nK\tilde{\alpha}}$
- $k \gtrsim \frac{\Gamma^d}{\alpha^\delta}$

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

$$\text{missclass}(f(W_T)) \lesssim e^{-\Gamma} + \frac{\Pi_N(f(W_0) - y)}{\sqrt{n}} + \frac{\Gamma}{\tilde{\alpha} \sqrt{n}}$$

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

Question: What can we say about pre-trained networks vs random?
Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $W_0$

Theorem

Fix number $\alpha \geq 0$ and $\Gamma \geq 1$.

- Set $J = J(W_0)$ and $I$ based on cut-off $\alpha = \sqrt{nK} \alpha$
- $k \gtrsim \frac{\Gamma^2}{\alpha^2}$

Then running gradient descent with $T = \frac{\Gamma}{\alpha^2}$ iterations

$$\operatorname{missclass}(f(W_T)) \leq e^{-T} + \frac{\Pi_N(f(W_0) - y)}{\sqrt{n}} + \frac{\Gamma}{\alpha \sqrt{n}}$$

\[\text{bias} + \text{variance}\]
Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $W_0$.

**Theorem**

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

- Set $J = J(W_0)$ and $I$ based on cut-off $\alpha = \sqrt{nK\bar{\alpha}}$
- $k \gtrapprox \frac{\Gamma^4}{\bar{\alpha}^2}$

Then running gradient descent with $T = \frac{\Gamma}{\bar{\alpha}^2}$ iterations,

$$\text{misclass}(f(W_T)) \lesssim e^{-T} + \frac{\Pi_N(f(W_0) - y)}{\sqrt{n}} + \frac{\Gamma}{\bar{\alpha}\sqrt{n}}$$

- Applies to any iteration $J = J(W_T)$.
Theory (deterministic)

Fix output layer entries bounded $\frac{1}{\sqrt{kK}}$ and initialize at a deterministic point $W_0$.

**Theorem**

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

- Set $J = J(W_0)$ and $I$ 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,

$$
\text{missclass}(f(W_T)) \lesssim e^{-T} + \frac{\Pi_N(f(W_0) - y)}{\sqrt{n}} + \frac{\Gamma}{\bar{\alpha}} \frac{1}{\sqrt{n}}
$$

- Applies to any iteration $J = J(W_t)$
- A step towards kernel adaptation $K_t = J(W_t)J^T(W_t)$

$K_0 \rightarrow K_1 \rightarrow K_2 \rightarrow \ldots \rightarrow$

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 $\sigma^2$.

**Theorem**

With high probability

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

$$\text{misclass}(f(W_T)) \lesssim \Gamma \sqrt{\frac{K^2C}{n}} + e^{-\Gamma}$$
Concrete example: Gaussian Mixture Model (GMM)

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

Theorem

With high probability

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

$$\text{misclass} (f(W_T)) \lesssim \Gamma \sqrt{\frac{K^2C}{n}} + e^{-\Gamma}$$

[Arora et. al. 2019] $k \to \infty$ as $\sigma \to 0$
Numerical experiments
No label corruption

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

Consider evolution of residual \( r_\tau = f(W_\tau) - y \)

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

|        | \[||\gamma||_{L_2}/||y||_{L_2}\] | \[||\gamma_N||_{L_2}/||y||_{L_2}\] | \[||J_{2,x}r_0||_{L_2}/||r_0||_{L_2}\] | \[||\gamma_N(r_\tau)||_{L_2}/||r_\tau||_{L_2}\] | \[||J_{2,x}r_\tau||_{L_2}/||r_\tau||_{L_2}\] |
|--------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| \(J_{\text{init}}\)       | 0.724                           | 0.690                           | 5.44 \cdot 10^{-3}               | 0.886                           | 0.465                           | 4.10 \cdot 10^{-3}               |
| \(J_{\text{final}}\)      | 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

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

Consider evolution of residual \( r_\tau = f(W_\tau) - y \)
green (total), red (on \( I \)), black (on \( N \))

![Graphs showing residual energy over epochs for initial and final Jacobian](image)

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

<table>
<thead>
<tr>
<th>Residual</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>| J_{yI} (y) |_2 |_2)</td>
<td>0.702</td>
</tr>
<tr>
<td>(</td>
<td>| J_{yN} (y) |_2 |_2)</td>
<td>0.712</td>
</tr>
<tr>
<td>(</td>
<td>| J_{yI} (r_\tau) |_2 |_2)</td>
<td>5.36 \times 10^{-3}</td>
</tr>
<tr>
<td>(</td>
<td>| J_{yN} (r_\tau) |_2 |_2)</td>
<td>0.814</td>
</tr>
<tr>
<td>(</td>
<td>| J_{yI} (r_\tau) |_2 |_2)</td>
<td>0.582</td>
</tr>
<tr>
<td>(</td>
<td>| J_{yN} (r_\tau) |_2 |_2)</td>
<td>4.43 \times 10^{-1}</td>
</tr>
</tbody>
</table>

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 \( r_{\tau} = f(W_{\tau}) - y \)

![Graphs showing residual energy and error over epochs.](image)

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

<table>
<thead>
<tr>
<th></th>
<th>Initial Residual Energy</th>
<th>Initial Gradient</th>
<th>Initial Error</th>
<th>Final Residual Energy</th>
<th>Final Gradient</th>
<th>Final Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>0.587</td>
<td>0.810</td>
<td>1.72 \times 10^{-3}</td>
<td>0.643</td>
<td>0.766</td>
<td>1.98 \times 10^{-3}</td>
</tr>
<tr>
<td>Final</td>
<td>0.751</td>
<td>0.660</td>
<td>1.87 \times 10^{-3}</td>
<td>0.763</td>
<td>0.646</td>
<td>1.20 \times 10^{-3}</td>
</tr>
</tbody>
</table>

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

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

Consider evolution of residual \( r_\tau = f(W_\tau) - y \)

---

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

|      | \(|\Pi_{\mathcal{Z}}(y)|_{L_2}\) | \(|\Pi_{\nu}(y)|_{L_2}\) | \(|J_y|_{L_2}\) | \(|\Pi_{\mathcal{Z}}(r_\tau)|_{L_2}\) | \(|\Pi_{\nu}(r_\tau)|_{L_2}\) | \(|J_{r_\tau}|_{L_2}\) |
|------|----------------------------------|--------------------------|-----------------|---------------------------------|-------------------------------|------------------|
| \(J_{\text{init}}\) | 0.587                           | 0.810                    | 1.72 \times 10^{-3} | 0.643                           | 0.766                         | 1.98 \times 10^{-3} |
| \(J_{\text{final}}\) | 0.751                           | 0.690                    | 1.87 \times 10^{-3} | 0.763                           | 0.646                         | 1.20 \times 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

![Graph showing the relationship between percent of label corruption and accuracy for train and test data. The graph includes a line for train accuracy, another for test accuracy, and a line for train accuracy with respect to true labels. The x-axis represents percent of label corruption ranging from 0 to 90, and the y-axis represents accuracy ranging from 100 to 0.]
**Model (without corruption)**

**clean data:** clusterable data

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

$L$ clusters and $K$ classes

![Diagram showing three classes: Class 1, Class 2, and Class 3. Each class contains points that are clustered together, with some overlap between classes.](image)
Robustness to corruption

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

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

Fit

\[
L(W) := \frac{1}{2} \sum_{i=1}^{n} \| f(W, x_i) - y_i \|_{\ell_2}^2
\]

via gradient descent
Robustness to corruption

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

Fit

\[
\mathcal{L}(W) := \frac{1}{2} \sum_{i=1}^{n} \| f(W, x_i) - y_i \|_{\ell_2}^2
\]

via gradient descent

**Theorem (Oymak and Soltanolkotabi 2019)**

Assume

- Corruption level \( \rho < \frac{1}{16} \)
- Overparameterization: \( \# \text{parameters} \gtrsim L^4 \)
Robustness to corruption

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

Fit

\[
L(W) := \frac{1}{2} \sum_{i=1}^{n} \|f(W, x_i) - y_i\|_{\ell_2}^2
\]

via gradient descent

**Theorem (Oymak and Soltanolkotabi 2019)**

Assume

- Corruption level \( \rho < \frac{1}{16} \)
- Overparameterization: \( \#\text{parameters} \gtrsim L^4 \)

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

closest label to \( f(W_\tau, x_i) = \text{true label } \bar{y}_i \)
Key Intuition
Key Intuition

Jacobian is low-rank

Info $\mathcal{I}$  Nuisance $\mathcal{N}$

Jacobian spectrum

L large singular vals  Remaining are small
Key Intuition

Jacobian is low-rank

Info $I$  Nuisance $N$

Prominent eigenvectors of Jacobian are diffused

Jacobian spectrum

Large singular vals  Remaining are small
Key Intuition

Jacobian is low-rank

Prominent eigenvectors of Jacobian are diffused

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

- Jacobian is low-rank
- Prominent eigenvectors of Jacobian are diffused
- Interaction of Jacobian and residual \( \nabla L(\theta) = J^T(\theta) (f(\theta, X) - y) \)
- Residual can be decomposed into two terms:
  \[
  r(\theta) := f(\theta, X) - y = \underbrace{f(\theta, X) - \bar{y}}_{\text{Residual w.r.t. true labels}} + \underbrace{\bar{y} - y}_{\text{corruption}}
  \]
Key Intuition

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

Residual can be decomposed into two terms

\[ r(\theta) = f(\theta, X) - y = f(\theta, X) - \bar{y} + \bar{y} - y \]

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

Jacobian is low-rank

Prominent eigenvectors of Jacobian are diffused

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

Residual can be decomposed into two terms:

$$r(\theta) := f(\theta, X) - y = f(\theta, X) - \bar{y} + \bar{y} - y$$

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

Prominent eigenvectors 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 $\sigma^2$

**Theorem**

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

\[
\text{misclass}(f(W_T)) \lesssim \Gamma \sqrt{\frac{K^2 C}{n}} + e^{-\Gamma}
\]
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).
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).
[Neyshabur Tomioka S ICLR’15]

The graph shows the error rate against the number of hidden units for both training and test data. The error rate decreases significantly as the number of hidden units increases. The test error rate is lower than the training error rate at convergence.
For valid generalization, the size of the weights is more important than the size of the network.

1997

Peter L. Bartlett
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)?
What is the relevant “complexity measure” (eg norm)?

How is this minimized (or controlled) by the opt algorithm?
What is the relevant “complexity measure” (e.g., norm)?
How is this minimized (or controlled) by the opt algorithm?
How does it change if we change the opt algorithm?
The figure illustrates the performance of Path-SGD compared to SGD for the MNIST and CIFAR-10 datasets. The graphs show the cross-entropy, 0/1 training error, and 0/1 test error over epochs. The Path-SGD approach shows a lower error rate compared to SGD for both training and testing phases, indicating improved performance.
SGD vs ADAM

Results on Penn Treebank using 3-layer LSTM

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:

$$\text{arg} \min_w \frac{1}{m} \sum_{i=1}^{m} \text{loss}(h_w(x); y) \quad (*)$$

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 $\min_w \|Xw - y\|^2$

  $\Rightarrow w \rightarrow \arg \min_{xw=y} \|w\|_2$
Effect of Optimization Geometry

• Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$
  $\Rightarrow w \rightarrow \arg \min_{xw=y} \|w\|_2$

• Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$
  $\Rightarrow w \rightarrow \arg \min_{xw=y} \Psi(w)$
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 separable logistic regression \( \min_w \sum_i g(y_i(w, x_i)) \)
  \[ \frac{w}{\|w\|} \to \arg \min \|w\|_2 \text{ s.t. } y_i(w, x_i) \geq 1 \text{ (Hard Margin SVM)} \]
Effect of Optimization Geometry

- Gradient descent on underspecified linear regression \( \min_w \|Xw - y\|^2 \)
  \( \Rightarrow w \rightarrow \arg \min_{xw=y} \|w\|_2 \)

- Natural Gradient/Mirror Descent w.r.t. \( \Psi(w) \) on \( \min_w \|Xw - y\|^2 \)
  \( \Rightarrow w \rightarrow \arg \min_{xw=y} \Psi(w) \)

- Gradient descent on separable logistic regression \( \min_w \sum_i g(y_i(w, x_i)) \)
  \( \Rightarrow \frac{w}{||w||} \rightarrow \arg \min \|w\|_2 \text{ s.t. } y_i(w, x_i) \geq 1 \) (Hard Margin SVM)

- Coordinate descent on \( \min_w \sum_i g(y_i(w, x_i)) \)
  \( \Rightarrow \frac{w}{||w||} \rightarrow \arg \min \|w\|_1 \text{ s.t. } y_i(w, x_i) \geq 1 \) [Telgarsky 2013]
Effect of Optimization Geometry

- Gradient descent on underspecified linear regression \( \min_w \|Xw - y\|^2 \)
  \[ w \rightarrow \arg \min_{Xw = y} \|w\|_2 \]

- Natural Gradient/Mirror Descent w.r.t. \( \Psi(w) \) on \( \min_w \|Xw - y\|^2 \)
  \[ w \rightarrow \arg \min_{Xw = y} \Psi(w) \]

- Gradient descent on separable logistic regression \( \min_w \sum_i g(y_i \langle w, x_i \rangle) \)
  \[ \frac{w}{\|w\|} \rightarrow \arg \min \|w\|_2 \text{ s.t. } y_i \langle w, x_i \rangle \geq 1 \text{ (Hard Margin SVM)} \]

- Coordinate descent on \( \min_w \sum_i g(y_i \langle w, x_i \rangle) \)
  \[ \frac{w}{\|w\|} \rightarrow \arg \min \|w\|_1 \text{ s.t. } y_i \langle w, x_i \rangle \geq 1 \text{ [Telgarsky 2013]} \]

- Gradient descent on matrix factorization \( \min_{u,v} \sum_i (\langle A_i, UV \rangle - y_i)^2 \)
  \[ UV \rightarrow \arg \min \|W\|_* \text{ s.t. } \langle A_i, W \rangle = y_i \]
  with init \( \rightarrow 0 \) and stepsize \( \rightarrow 0 \)
  (proven in special cases, empirically at least aprox, conjectured in general)
Effect of Parametrization

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

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017][Li Ma Zhang 2018]
Effect of Optimization Geometry

- Gradient descent on underspecified linear regression $\min_w \|Xw - y\|^2$
  \[w \rightarrow \arg \min_{Xw = y} \|w\|_2\]

- Natural Gradient/Mirror Descent w.r.t. $\Psi(w)$ on $\min_w \|Xw - y\|^2$
  \[w \rightarrow \arg \min_{Xw = y} \Psi(w)\]

- Gradient descent on separable logistic regression $\min_w \sum_i g(y_i(w, x_i))$
  \[\frac{w}{||w||} \rightarrow \arg \min \|w\|_2 \quad \text{s.t.} \quad y_i(w, x_i) \geq 1 \quad \text{(Hard Margin SVM)}\]

- Coordinate descent on $\min_w \sum_i g(y_i(w, x_i))$
  \[\frac{w}{||w||} \rightarrow \arg \min \|w\|_1 \quad \text{s.t.} \quad y_i(w, x_i) \geq 1 \quad [\text{Telgarsky 2013}]\]

- Gradient descent on matrix factorization $\min_{U, V} \sum_i (\langle A_i, UV \rangle - y_i)^2$
  \[UV \rightarrow \arg \min \|W\|_* \quad \text{s.t.} \quad \langle A_i, W \rangle = y_i\]
  with init $\rightarrow 0$ and stepsize $\rightarrow 0$
  (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 $U, V \Rightarrow$ implicitly minimize $\|W\|_*$

[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 $U, V \Rightarrow$ implicitly minimize $\|W\|_*$

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

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors $\beta$
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 $U, V \Rightarrow$ implicitly minimize $\|W\|_*$

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

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors $\beta$
  - GD on weight $\Rightarrow$ implicitly minimize $\|DFT(\beta)\|_p$ for $p = \frac{2}{\text{depth}}$ (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 $U, V \Rightarrow$ implicitly minimize $\|W\|_*$

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

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

  [Gunasekar Lee Soudry S 2018]

- Infinite Width ReLU Net with 1-d input:
  - Parametrization of essentially all functions $f : \mathbb{R} \rightarrow \mathbb{R}$
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 $U, V \Rightarrow$ implicitly minimize $\|W\|_*$

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

- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors $\beta$
  - GD on weight $\Rightarrow$ implicitly minimize $\|\text{DFT}(\beta)\|_p$ for $p = \frac{2}{\text{depth}}$.
    (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 $\Rightarrow$ implicitly minimize
    $\max(\int |f''| dx, |f'(-\infty) + f'(+\infty)|)$

  [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: $F(w) = h_w$  
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: $L_S(w) = \frac{1}{m} \sum_i \ell(f(w, x_i), y_i)$
Model: \( F(w) = h_w \)  
Model Class: \( \mathcal{H} = \text{range}(F) \)

\[ f(w, x) = h_w(x) = \text{prediction on } x \text{ with params ("weights") } w \]

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

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: $F(w) = h_w$  
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 $F(w) = w$, $f(w, x) = \langle w, x \rangle$
Model: $F(w) = h_w$  \quad Model Class: $\mathcal{H} = \text{range}(F)$

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

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

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: \( \mathcal{H} = \text{range}(F) \)

\[
f(w, x) = h_w(x) = \text{prediction on } x \text{ with params ("weights") } w \
\]

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 \( L_s(w) = \sum_i g(y_if(w, x_i)) \) (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 \( \to 0 \) (or finite but ensures convergence in direction):

\[
w_\infty \propto \text{first order stationary point of } \quad \arg \min \|w\|_2 \quad \text{s.t. } \forall i y_if(w, x_i) \geq 1
\]
Consider gradient descent w.r.t. logistic loss \( L_s(w) = \sum_i g(y_if(w, x_i)) \) (or other loss exp-tail loss) on a D-homogenous model \( f(w, x) \):

**Theorem** [Nacson Gunasekar Lee S Soudry 2019][1][Lyu Li 2019]:
If \( L_s(w) \to 0 \), with stepsize \( \to 0 \) (or finite but ensures convergence in direction):

\[
w_\infty \propto \text{first order stationary point of } \quad \arg \min \|w\|_2 \quad \text{s.t. } \forall y_if(w, x_i) \geq 1
\]

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

\[
h_\infty = F(w_\infty) \propto \text{first order stationary point of } \quad \arg \min R_F(h) \quad \text{s.t. } y_if(x_i) \geq 1
\]

But need to be careful: f.o.s.p of (*) does not imply f.o.s.p of (**)
Is implicit bias of GD just $\ell_2$ 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 $\ell_2$ 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 $L_s(w) = \sum_i g(y_if(w, x_i))$ (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 $\to 0$ (or finite but ensures convergence in direction):

$$w_\infty \propto \text{first order stationary point of } \arg\min \|w\|_2 \text{ s.t. } \forall y_if(w, x_i) \geq 1$$

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

$$h_\infty = F(w_\infty) \propto \text{first order stationary point of } \arg\min R_F(h) \text{ s.t. } y_if(x_i) \geq 1$$

But need to be careful: f.o.s.p of (**) does not imply f.o.s.p of (***)
Is implicit bias of GD just $\ell_2$ 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 \( w^{(0)} \):
\[
f(w, x) \approx h_{w^{(0)}}(x) + \langle w, \phi_0(x) \rangle
\]
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 $w^{(0)}$:

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

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: $\arg\min \|h - h_{w^{(0)}}\|_K$ s.t. $h(x_i) = y_i$
  (we will focus on “unbiased initialization”: $h_{w^{(0)}} = F(w^{(0)}) = 0$)

- For logistic (or exp) loss: $\propto \arg\min \|h\|_K$ s.t. $y_i h(x_i) \geq 1$
Kernel Regime and Scale of Init

- For $D$-homogenous model, consider gradient flow with:
  \[ \dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \]  
  with unbiased $F(w_0) = 0$
Is it all just a Kernel?

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

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

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: $\text{arg min} \| h - h_{w^{(0)}} \|_K \text{ s.t. } h(x_i) = y_i$

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

- For logistic (or exp) loss: $\propto \text{arg min} \| h \|_K \text{ s.t. } y_i h(x_i) \geq 1$
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 \( w^{(0)} \):

\[
f(w, x) \approx h_{w^{(0)}}(x) + \langle w, \phi_0(x) \rangle
\]

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: \( \arg \min \| h - h_{w^{(0)}} \|_K \) s.t. \( h(x_i) = y_i \)
  
  (we will focus on “unbiased initialization”: \( h_{w^{(0)}} = F(w^{(0)}) = 0 \))

- For logistic (or exp) loss: \( \propto \arg \min \| h \|_K \) s.t. \( y_i h(x_i) \geq 1 \)
Kernel Regime and Scale of Init

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

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

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

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

with unbiased $F(w_0) = 0$

We are interested in $w_\alpha(\infty) = \lim_{t \to \infty} w_\alpha(t)$ and $h_\alpha(\infty) = F(w_\alpha(\infty))$
Kernel Regime and Scale of Init

- For $D$-homogenous model, consider gradient flow with:
  \[ \dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \quad \text{with unbiased} \quad F(w_0) = 0 \]
  We are interested in $w_\alpha(\infty) = \lim_{t \to \infty} w_\alpha(t)$ and $h_\alpha(\infty) = F(w_\alpha(\infty))$

- 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:
  \[
  \dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \quad \text{with unbiased} \quad F(w_0) = 0
  \]
  
  We are interested in $w_\alpha(\infty) = \lim_{t \to \infty} w_\alpha(t)$ and $h_\alpha(\infty) = F(w_\alpha(\infty))$

- 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:
  \[
  \dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \quad \text{with unbiased } F(w_0) = 0
  \]
  We are interested in $w_\alpha(\infty) = \lim_{t \to \infty} w_\alpha(t)$ and $h_\alpha(\infty) = F(w_\alpha(\infty))$

- 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]:
  \[
  \lim_{\alpha \to \infty} \sup_t \left\| \frac{1}{\alpha} w_\alpha \left( \frac{1}{\alpha^{D-1}} t \right) - w_K(t) \right\| = 0
  \]
  \[
  \Rightarrow \frac{1}{\alpha} h_\alpha(\infty) \to \hat{h}_{K_0} = \arg\min \| h \|_{K_0} \quad \text{s. t. } h(x_i) = y_i
  \]
Kernel Regime and Scale of Init

- For $D$-homogenous model, consider gradient flow with:
  \[ \dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0 \]  
  with unbiased $F(w_0) = 0$

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

- When will this behave like the kernel gradient flow:
  \[ \dot{w}_K = -\nabla L_S(x \mapsto \langle w, \phi_0(x) \rangle) \quad \text{and} \quad w(0) = w_0 \]
  \[ \text{where } \phi_0(x) = \nabla f(w_0, x) \text{ with associated kernel } K_0 \]

- For squared loss, under some conditions [Chizat and Bach 18]:
  \[ \lim_{\alpha \to \infty} \sup_t \left\| \frac{1}{\alpha} w_\alpha \left( \frac{1}{\alpha^{D-1}} t \right) - w_K(t) \right\| = 0 \]
  \[ \Rightarrow \frac{1}{\alpha} h_\alpha(\infty) \to \hat{h}_{K_0} = \arg \min_{h} \| h \|_{K_0} \text{ s.t. } h(x_i) = y_i \]

- 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:

\[ f(w, x) = \sum_j (w_+ [j]^2 - w_- [j]^2) x_i \]

i.e. \[ F(w) = \beta = w_+^2 - w_-^2 \]
Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w, x) = \sum_j (w_+ [j]^2 - w_- [j]^2) x_i$$

i.e.

$$F(w) = \beta = w_+^2 - w_-^2$$

And unbiased initialization $w_\alpha (0) = \alpha 1$ (so that $\beta_\alpha (0) = F(w_\alpha (0)) = 0$).
Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

$$f(w, x) = \sum_j (w_+ [j]^2 - w_- [j]^2) x_i$$

i.e. $$F(w) = \beta = w_+^2 - w_-^2$$

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

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

$$\beta_{\alpha}(\infty) = \lim_{t \to \infty} F(w_{\alpha}(t))$$
Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

\[ f(w, x) = \sum_j (w_+ [j]^2 - w_- [j]^2)x_j \]

i.e., \[ F(w) = \beta = w_+^2 - w_-^2 \]

And unbiased initialization \[ w_\alpha(0) = \alpha \mathbf{1} \] (so that \[ \beta_\alpha(0) = F(w_\alpha(0)) = 0 \]).

What’s the implicit bias of grad flow w.r.t square loss \[ L_S(w) = \|X F(w) - y\|_2^2 \]?

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

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

\[ \beta_\alpha(\infty) \to \hat{\beta}_{L2} = \arg \min_{\beta} \|\beta\|_2 \]
Scale of Init: Kernel vs Deep

Consider linear regression with squared parametrization:

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

And unbiased initialization $$w_\alpha(0) = \alpha 1$$ (so that $$\beta_\alpha(0) = F(w_\alpha(0)) = 0$$).

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

$$\beta_\alpha(\infty) \to \hat{\beta}_{L_2} = \arg \min_{x \beta = y} \|\beta\|_2$$

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

$$\beta_\alpha(\infty) \to \hat{\beta}_{L_1} = \arg \min_{x \beta = y} \|\beta\|_1$$
Consider linear regression with squared parametrization:

\[ f(w, x) = \sum_j (w_+ [j]^2 - w_- [j]^2) x_i \]

i.e. \( F(w) = \beta = w_+^2 - w_-^2 \)

And unbiased initialization \( w_\alpha(0) = \alpha 1 \) (so that \( \beta_\alpha(0) = F(w_\alpha(0)) = 0 \)).

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

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

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

\[ \beta_\alpha(\infty) \to \hat{\beta}_{L2} = \arg \min_{\beta \neq 0} \| \beta \|_2 \]

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

\[ \beta_\alpha(\infty) \to \hat{\beta}_{L1} = \arg \min_{\beta \neq 0} \| \beta \|_1 \]

For any \( \alpha \):

\[ \beta_\alpha(\infty) = \arg \min_{x \beta = y} Q_\alpha(\beta) \]

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 $$Q_\alpha(\beta) = \sum_j q \left( \frac{\beta_\alpha^j}{\alpha^2} \right)$$ 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

\[ y_i = \langle \beta^*, x_i \rangle + N(0,0.01) \]

\[ d = 1000, \quad \| \beta^* \|_0 = 5, \quad m = 100 \]
Sparse Learning

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

How small does \( \alpha \) need to be to get \( L(\beta_\alpha(\infty)) < 0.025 \)
Relationship to Explicit Reg

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

\[
\beta^R_\alpha = F \left( \arg \min_{L_S(w) = 0} \| w - \alpha 1 \|_2^2 \right) = \arg \min_{x_\beta = y} R_\alpha (\beta)
\]

Where \( R_\alpha (\beta) = \min_{F(w) = \beta} \| w - \alpha 1 \|_2^2 \)
Relationship to Explicit Reg

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

$$\beta^R_\alpha = F \left( \arg \min_{L_s(w) = 0} \| w - \alpha 1 \|_2^2 \right) = \arg \min_{x_\beta = y} R_\alpha(\beta)$$

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

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

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

![Graph of r(z) and q(z)](image-url)
Kernel and Deep Regimes in Matrix Completion

\[ f((U, V), (i, j)) = (UV^T)_{ij} \]

Tangent kernel: \[ K_{U,V}((i, j), (i', j')) = \delta_{i=i'} \langle V_j, V'_j \rangle + \delta_{j=j'} \langle U_i, U_{i'} \rangle \]

For orthogonal initialization: \[ K_0((i, j), (i', j')) \propto \delta_{(i,j)=(i',j')} \]
Squared Loss vs Exp Loss

- Back to general $D$-homogenous model

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

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

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