Why Are Convolutional Nets More Sample-Efficient than Fully-Connected Nets?

Zhiyuan Li
Joint work with Sanjeev Arora, Yi Zhang
Princeton University
August 19, 2020
@ IJTCS
Table of Contents

1 Introduction

2 Intuition and Warm-up example

3 Identifying Algorithmic Equivariance

4 Lower Bound for Equivariant Algorithms
CNNs (Convolutional neural networks) often perform better than its fully connected counterparts, *FC Nets*, especially on vision tasks.
CNNs (Convolutional neural networks) often perform better than its *fully connected* counterparts, *FC Nets*, especially on vision tasks.

Not an issue of expressiveness — FC nets easily gets full training accuracy, but still generalizes poorly.
CNNs (Convolutional neural networks) often perform better than its fully connected counterparts, FC Nets, especially on vision tasks.

Not an issue of expressiveness — FC nets easily gets full training accuracy, but still generalizes poorly.

Often explained by “better inductive bias”.

Ex: Over-parametrized Linear Regression has multiple solutions, and GD (Gradient Descent) initialized from 0 picks the one with min $\ell_2$ norm.
CNNs (Convolutional neural networks) often perform better than its *fully connected* counterparts, *FC Nets*, especially on vision tasks.

Not an issue of expressiveness — FC nets easily gets full training accuracy, but still generalizes poorly.

Often explained by “better inductive bias”.

Ex: Over-parametrized Linear Regression has multiple solutions, and GD (Gradient Descent) initialized from 0 picks the one with min $\ell_2$ norm.

**Question:** Can we justify this rigorously by showing a sample complexity separation?
Introduction

- CNN often performs better than *FC Nets*, especially on vision tasks.
- Often explained by “better inductive bias”.
  Ex: Over-parametrized Linear Regression has multiple solutions, and GD (Gradient Descent) initialized from 0 picks the one with min $\ell_2$ norm.
- **Question:** Can we justify this rigorously by showing a sample complexity separation?
- Since ultra-wide FC nets can simulate any CNN, the hurdle here is how to show that (S)GD + FC net doesn’t learn those CNN with good generalization.
Introduction

- CNN often performs better than *FC Nets*, especially on vision tasks.
- Often explained by “better inductive bias”.
  Ex: Over-parametrized Linear Regression has multiple solutions, and GD (Gradient Descent) initialized from 0 picks the one with min $\ell_2$ norm.
- **Question:** Can we justify this rigorously by showing a sample complexity separation?
- Since ultra-wide FC nets can simulate any CNN, the hurdle here is how to show that (S)GD + FC net doesn’t learn those CNN with good generalization.

**This Work**

A single distribution + a single target function which can be learnt by CNN with constant samples, but SGD on FC nets of any depth and width require $\Omega(d^2)$ samples.
Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
- Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y} | \mathcal{X}}$ is always a deterministic function, $h^* : \mathbb{R}^d \to \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \diamond h^*$. 
Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$

Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y}|\mathcal{X}}$ is always a deterministic function, $h^*: \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \diamond h^*$.

A Learning Algorithm $A$ maps from a sequence of training data, $\{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $A(\{(x_i, y_i)\}_{i=1}^n) \in \mathcal{Y}^\mathcal{X}$. $A$ could also be random.
Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
- Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y}|\mathcal{X}}$ is always a deterministic function, $h^*: \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \diamond h^*$.
- A Learning Algorithm $\mathcal{A}$ maps from a sequence of training data, $\{x_i, y_i\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $\mathcal{A}(\{x_i, y_i\}_{i=1}^n) \in \mathcal{Y}^\mathcal{X}$. $\mathcal{A}$ could also be random.
Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
- Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y}|\mathcal{X}}$ is always a deterministic function, $h^*: \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \diamond h^*$.
- A Learning Algorithm $A$ maps from a sequence of training data, $\{x_i, y_i\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $A(\{x_i, y_i\}_{i=1}^n) \in \mathcal{Y}^\mathcal{X}$. $A$ could also be random.

Two examples, Kernel Regression and ERM (Empirical Risk Minimization):
Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
- Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y}|\mathcal{X}}$ is always a deterministic function, $h^*: \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \circ h^*$.
- A Learning Algorithm $A$ maps from a sequence of training data, $\{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $A(\{(x_i, y_i)\}_{i=1}^n) \in \mathcal{Y}^\mathcal{X}$. $A$ could also be random.

Two examples, Kernel Regression and ERM (Empirical Risk Minimization):
- $\text{REG}_K(\{(x_i, y_i)\}_{i=1}^n)(x) := 1 \left[ K(x, X_n) \cdot K(X_n, X_n)^\top y \geq 0 \right]$. 

Setting

- Binary classification, $\mathcal{Y} = \{-1, 1\}$. Data domain $\mathcal{X} = \mathbb{R}^d$
- Joint distribution $P$ supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this talk, $P_{\mathcal{Y}|\mathcal{X}}$ is always a deterministic function, $h^*: \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. $P = P_{\mathcal{X}} \diamond h^*$.
- A Learning Algorithm $A$ maps from a sequence of training data, $\{x_i, y_i\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $A(\{x_i, y_i\}_{i=1}^n) \in \mathcal{Y}^\mathcal{X}$. $A$ could also be random.

Two examples, Kernel Regression and ERM (Empirical Risk Minimization):
- $\text{REG}_K(\{x_i, y_i\}_{i=1}^n)(x) := 1 \left[K(x, X_n) \cdot K(X_n, X_n)^\dagger y \geq 0\right]$.
- $\text{ERM}_\mathcal{H}(\{x_i, y_i\}_{i=1}^n) = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^n 1 \left[h(x_i) \neq y_i\right]$. \(^1\)

\(^1\)Strictly speaking, $\text{ERM}_\mathcal{H}$ is not a well-defined algorithm. In this talk, we consider the worst performance of all the empirical minimizers in $\mathcal{H}$. 
Setting

\[ \text{err}_P(h) = \mathbb{P}_{(X,Y) \sim P} [h(X) \neq Y]. \]

Sample Complexity: single joint distribution \( P \)

The \((\varepsilon, \delta)\)-sample complexity, denoted \( \mathcal{N}(\mathcal{A}, P, \varepsilon, \delta) \), is the smallest number \( n \) such that w.p. \( 1 - \delta \) over the randomness of \( \{x_i, y_i\}_{i=1}^n \), \( \text{err}_P(\mathcal{A}(\{x_i, y_i\}_{i=1}^n)) \leq \varepsilon. \)

We also define the \( \varepsilon \)-expected sample complexity, \( \mathcal{N}^*(\mathcal{A}, P, \varepsilon) \), as the smallest number \( n \) such that

\[ \mathbb{E}_{(x_i, y_i) \sim P} \left[ \text{err}_P(\mathcal{A}(\{x_i, y_i\}_{i=1}^n)) \right] \leq \varepsilon. \]
Sample Complexity: single joint distribution $P$

The $(\varepsilon, \delta)$-sample complexity, denoted $\mathcal{N}(A, P, \varepsilon, \delta)$, is the smallest number $n$ such that w.p. $1 - \delta$ over the randomness of $\{x_i, y_i\}_{i=1}^n$, $\text{err}_P(A(\{x_i, y_i\}_{i=1}^n)) \leq \varepsilon$.

We also define the $\varepsilon$-expected sample complexity, $\mathcal{N}^*(A, P, \varepsilon)$, as the smallest number $n$ such that
\[
\mathbb{E}_{(x_i,y_i) \sim P} \left[ \text{err}_P(A(\{x_i, y_i\}_{i=1}^n)) \right] \leq \varepsilon.
\]

Sample Complexity: a family of distributions, $\mathcal{P}$

\[
\mathcal{N}(A, \mathcal{P}, \varepsilon, \delta) = \max_{P \in \mathcal{P}} \mathcal{N}(A, P, \varepsilon, \delta) ; \quad \mathcal{N}^*(A, \mathcal{P}, \varepsilon) = \max_{P \in \mathcal{P}} \mathcal{N}^*(A, P, \varepsilon)
\]

Fact: $\mathcal{N}^*(A, \mathcal{P}, \varepsilon + \delta) \leq \mathcal{N}(A, \mathcal{P}, \varepsilon, \delta) \leq \mathcal{N}^*(A, \mathcal{P}, \varepsilon \delta)$, $\forall \varepsilon, \delta \in [0, 1]$. 
Parametric Models

A parametric model $M : \mathcal{W} \rightarrow \mathcal{Y}^\mathcal{X}$ is a functional mapping from weight $\mathbf{W}$ to a hypothesis $M(\mathbf{W}) : \mathcal{X} \rightarrow \mathcal{Y}$. 
Parametric Models

A parametric model $\mathcal{M} : \mathcal{W} \rightarrow \mathcal{Y}^{\mathcal{X}}$ is a functional mapping from weight $\mathbf{W}$ to a hypothesis $\mathcal{M}(\mathbf{W}) : \mathcal{X} \rightarrow \mathcal{Y}$.

Fully-connected (FC) Neural Networks: $\mathbb{R}^d \rightarrow \mathbb{R}$

$$\text{FC-NN}[\mathbf{W}](\mathbf{x}) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 \mathbf{x} + b_1) + b_2) + b_{L-1}) + b_L,$$

where $\mathbf{W} = (\{W_i\}_{i=1}^L, \{b_i\}_{i=1}^L), W_i \in \mathbb{R}^{d_{i-1} \times d_i}, b_i \in \mathbb{R}^{d_i}, d_0 = d$, and $d_L = 1$. Here, $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function, and we abuse the notation such that $\sigma$ is also defined for vector inputs, i.e. that $[\sigma(\mathbf{x})]_i = \sigma(x_i)$. 

Zhiyuan Li (Princeton University)
A parametric model $\mathcal{M} : \mathcal{W} \rightarrow \mathcal{Y}^\mathcal{X}$ is a functional mapping from weight $\mathcal{W}$ to a hypothesis $\mathcal{M}(\mathcal{W}) : \mathcal{X} \rightarrow \mathcal{Y}$.

**Fully-connected (FC) Neural Networks:** $\mathbb{R}^d \rightarrow \mathbb{R}$

$$\text{FC-NN}[\mathcal{W}](x) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 x + b_1) + b_2) + b_{L-1}) + b_L,$$

where $\mathcal{W} = (\{W_i\}_{i=1}^L, \{b_i\}_{i=1}^L)$, $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$, $b_i \in \mathbb{R}^{d_i}$, $d_0 = d$, and $d_L = 1$. Here, $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function, and we abuse the notation such that $\sigma$ is also defined for vector inputs, i.e. that $[\sigma(x)]_i = \sigma(x_i)$.

**Convolutional Neural Networks (CNN):** $\mathbb{R}^d \rightarrow \mathbb{R}$

$$\text{CNN}[\mathcal{W}](x) = \sum_{i=1}^r a_r \sigma([w \ast x]_{d'(r-1)+1:d'q}) + b,$$

where $\mathcal{W} = (w, a, b) \in \mathbb{R}^k \times \mathbb{R}^r \times \mathbb{R}$, $d = d' r$. $\ast : \mathbb{R}^k \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the convolution operator, defined as $[w \ast x]_i = \sum_{j=1}^k w_j x_{i-j-1 \mod d} + 1$, and $\sigma : \mathbb{R}^{d'} \rightarrow \mathbb{R}$ is the composition of pooling and element-wise non-linearity.
A parametric model $\mathcal{M} : \mathcal{W} \rightarrow \mathcal{Y}^{\mathcal{X}}$ is a functional mapping from weight $\mathbf{W}$ to a hypothesis $\mathcal{M}(\mathbf{W}) : \mathcal{X} \rightarrow \mathcal{Y}$.

$$\text{FC-NN}[\mathbf{W}](\mathbf{x}) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 \mathbf{x} + b_1) + b_2) + b_{L-1}) + b_L,$$

$$\text{CNN}[\mathbf{W}](\mathbf{x}) = \sum_{i=1}^{r} a_r \sigma([\mathbf{w} \ast \mathbf{x}]_{d'(r-1)+1:d'} + b).$$

Not possible to separate every learning algorithm on FC nets from CNN, as FC nets could simulate CNN.

**Question**

What property of SGD prevents it from finding CNNs among FC nets?
# Table of Contents

1. Introduction

2. Intuition and Warm-up example

3. Identifying Algorithmic Equivariance

4. Lower Bound for Equivariant Algorithms
Key Intuition: Equivariance

Definition (Equivariant Algorithms)
A learning algorithm $\mathcal{A}$ is $G_{\mathcal{X}}$-equivariant iff for any dataset $\{x_i, y_i\}_{i=1}^n$ and $\forall g \in G_{\mathcal{X}}, x \in \mathcal{X}$, $\mathcal{A}(\{g(x_i), y_i\}_{i=1}^n)(g(x)) \overset{d}{=} [\mathcal{A}(\{x_i, y_i\}_{i=1}^n)](x)$.

- SGD for FC Nets are $O(d)$-equivariant. (a.k.a. orthogonal/rotation equivariant)
Key Intuition: Equivariance

Definition (Equivariant Algorithms)
A learning algorithm $\mathcal{A}$ is $G_{\mathcal{X}}$-equivariant iff for any dataset $\{x_i, y_i\}_{i=1}^n$ and $\forall g \in G_{\mathcal{X}}, x \in \mathcal{X}$, $\mathcal{A}({\{g(x_i), y_i\}_{i=1}^n})(g(x)) \overset{d}{=} \mathcal{A}({\{x_i, y_i\}_{i=1}^n})(x)$.

- SGD for FC Nets are $O(d)$-equivariant. (a.k.a. orthogonal/rotation equivariant)
- Algorithmic equivariance constraints lead to sample complexity lower bounds.
Key Intuition: Equivariance

Definition (Equivariant Algorithms)
A learning algorithm $A$ is $G_X$-equivariant iff for any dataset $\{x_i, y_i\}_{i=1}^n$ and $\forall g \in G_X, x \in X$, $A(\{g(x_i), y_i\}_{i=1}^n)(g(x)) \overset{d}= [A(\{x_i, y_i\}_{i=1}^n)](x)$.

- SGD for FC Nets are $O(d)$-equivariant. (a.k.a. orthogonal/rotation equivariant)
- Algorithmic equivariance constraints lead to sample complexity lower bounds.
- Convolution and pooling layers in CNN break these constraints.
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_iy, cy) \mid i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_i, y, cy) \mid i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
- w. Global average pooling: only 1 sample required.
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_iy, cy) \mid i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
- w. Global average pooling: only 1 sample required.
- For orthogonal equivariant $A$, $\forall R \in O(d)$,
  $$A(\{(Rx_i, y_i)\}_{i=1}^n)(Rx) = A(\{(x_i, y_i)\}_{i=1}^n)(x)$$

Let $S = \{x_i, y_i\}_{i=1}^n$, $A(S)(x) = f_S(x_1^T, \ldots, x_n^T)$, i.e.

$$(x_1^T, \ldots, x_n^T) = (x_1'^T, \ldots, x_n'^T) \Rightarrow A(S)(x) = A(S)(x')$$

when $n \leq d^2$, w.p. $\frac{1}{2}$, $A(S)(x) = f_S(0, \ldots, 0) \Rightarrow$ at least $\frac{1}{2}$ error w.p. $\frac{1}{2}$.  

Zhiyuan Li (Princeton University)
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_i y, c y) \mid i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
- w. Global average pooling: only 1 sample required.
- For orthogonal equivariant $A$, $\forall R \in O(d)$,
  $$A((R x_i, y_i)_{i=1}^n)(Rx) = A((x_i, y_i)_{i=1}^n)(x)$$
- Let $S = \{x_i, y_i\}_{i=1}^n$, $A(S)(x) = f_S(x^\top x_1, \ldots, x^\top x_n)$, i.e.
  $$(x^\top x_1, \ldots, x^\top x_n) = (x'^\top x_1, \ldots, x'^\top x_n) \implies A(S)(x) = A(S)(x').$$
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_i y, cy) \mid i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
- w. Global average pooling: only 1 sample required.
- For orthogonal equivariant $A$, $\forall R \in O(d)$,
  
  $$A(\{(R x_i, y_i)\}_{i=1}^n)(Rx) = A(\{(x_i, y_i)\}_{i=1}^n)(x)$$

- Let $S = \{(x_i, y_i)\}_{i=1}^n$, $A(S)(x) = f_S(x^\top x_1, \ldots, x^\top x_n)$, i.e.
  
  $$(x^\top x_1, \ldots, x^\top x_n) = (x'^\top x_1, \ldots, x'^\top x_n) \implies A(S)(x) = A(S)(x').$$
- when $n \leq \frac{d}{2}$, w.p. $\frac{1}{2}$, $A(S)(x) = f_S(0, \ldots, 0)$
Warmup: a $\Omega(d)$ lower bound against orthogonal equivariant algorithms

- $\mathcal{X} = \mathbb{R}^d$, $P_c = \text{Unif}\{(e_i y, cy) | i \in [d], y = \pm 1\}$, $c \in \{-1, 1\}$
- w. Global average pooling: only 1 sample required.
- For orthogonal equivariant $\mathcal{A}$, $\forall R \in O(d)$,
  
  $$\mathcal{A}((\mathcal{R}x_i, y_i)_{i=1}^n)(R \mathbf{x}) = \mathcal{A}((\mathbf{x}_i, y_i)_{i=1}^n)(\mathbf{x})$$

- Let $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, $\mathcal{A}(S)(\mathbf{x}) = f_S(\mathbf{x}^\top \mathbf{x}_1, \ldots, \mathbf{x}^\top \mathbf{x}_n)$, i.e.
  $$(\mathbf{x}^\top \mathbf{x}_1, \ldots, \mathbf{x}^\top \mathbf{x}_n) = (\mathbf{x'}^\top \mathbf{x}_1, \ldots, \mathbf{x'}^\top \mathbf{x}_n) \implies \mathcal{A}(S)(\mathbf{x}) = \mathcal{A}(S)(\mathbf{x'})$$
- when $n \leq \frac{d}{2}$, w.p. $\frac{1}{2}$, $\mathcal{A}(S)(\mathbf{x}) = f_S(0, \ldots, 0)$
- $\implies$ at least $\frac{1}{2}$ error w.p. $\frac{1}{2}$. 
Related Work

- [DWZ+18] proved $\Theta(\text{filter size})$ worst-case sample complexity for two-layer CNNs, better than the folklore $\Omega(d)$ lower bound for linear function class. Not a sample complexity separation, as their upper and lower bounds are proved on different classes of tasks.

- [Ng04] showed that every orthogonal equivariant algorithm requires $\Omega(d)$ samples to learn a fixed linear function for all distributions. However, it doesn’t imply a sample complexity separation between FC nets and CNNs on image distributions or other natural distributions.

- Recently, there have been progress in showing lower bounds against learning with kernels. [WLLM19] constructed a single task on which they proved a sample complexity separation between learning with neural networks vs. with neural tangent kernels [JGH18]. Relatedly, [AZL19] showed a sample complexity lower bound against all kernels for a family of tasks, i.e., learning $k$-XOR on the hypercube.
Related Work

- [DWZ\textsuperscript{+}18] proved $\Theta(\#\text{filter size})$ worst-case sample complexity for two-layer CNNs, better than the folklore $\Omega(d)$ lower bound for linear function class. Not a sample complexity separation, as their upper and lower bounds are proved on different classes of tasks.

- [Ng04] showed that every orthogonal equivariant algorithm requires $\Omega(d)$ samples to learn a fixed linear function for \textit{all distributions}. However, it doesn’t imply a sample complexity separation between FC nets and CNNs on image distributions or other natural distributions.
Related Work

- [DWZ⁺18] proved \( \Theta(\#\text{filter size}) \) worst-case sample complexity for two-layer CNNs, better than the folklore \( \Omega(d) \) lower bound for linear function class. Not a sample complexity separation, as their upper and lower bounds are proved on different classes of tasks.

- [Ng04] showed that every orthogonal equivariant algorithm requires \( \Omega(d) \) samples to learn a fixed linear function for all distributions. However, it doesn’t imply a sample complexity separation between FC nets and CNNs on image distributions or other natural distributions.

- Recently, there have been progress in showing lower bounds against learning with kernels. [WLLM19] constructed a single task on which they proved a sample complexity separation between learning with neural networks vs. with neural tangent kernels [JGH18]. Relatedly, [AZL19] showed a sample complexity lower bound against all kernels for a family of tasks, i.e., learning \( k \)-XOR on the hypercube.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction

Ex: CIFAR10 has 50k images of size $32 \times 32 \times 3$. To learn whether the Red channel or Green Channel has larger signal strength (in $\ell_2$ sense), FC nets needs around $32^4 \approx 1M$ images if the image distribution is complex enough, e.g. close to i.i.d. gaussian.

$\Omega(d^2/\epsilon)$ lower bound for $O(d)$-equivariance, all distributions and single quadratic function.

$\Omega(d^2)$ lower bound for $O(d)$-equivariance, single gaussian distribution and single quadratic function.

All above problems can be learnt by simple 2-layer CNN with GD using $O(1)$ samples.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction
  - $\Omega(d^2/\varepsilon)$ lower bound for $O(d)$-equivariance, all distributions and single quadratic function.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction
  - $\Omega(d^2/\epsilon)$ lower bound for $O(d)$-equivariance, all distributions and single quadratic function.
  - $\Omega(d^2)$ lower bound for $O(d)$-equivariance, single gaussian distribution and single quadratic function.
**Our Contributions**

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction
  - $\Omega(d^2/\varepsilon)$ lower bound for $O(d)$-equivariance, all distributions and single quadratic function.
  - $\Omega(d^2)$ lower bound for $O(d)$-equivariance, single gaussian distribution and single quadratic function.

Ex: CIFAR10 has $50k$ images of size $32 \times 32 \times 3$. To learn whether the Red channel or Green Channel has larger signal strength (in $\ell_2$ sense), FC nets needs around $32^4 \approx 1M$ images if the image distribution is complex enough, e.g. close to i.i.d. gaussian.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction
  - $\Omega(d^2/\epsilon)$ lower bound for $\mathcal{O}(d)$-equivariance, all distributions and single quadratic function.
  - $\Omega(d^2)$ lower bound for $\mathcal{O}(d)$-equivariance, single gaussian distribution and single quadratic function. Ex: CIFAR10 has 50k images of size $32 \times 32 \times 3$. To learn whether the Red channel or Green Channel has larger signal strength (in $\ell_2$ sense), FC nets needs around $32^4 \approx 1M$ images if the image distribution is complex enough, e.g. close to i.i.d. gaussian.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check
- Sample complexity lower bounds for equivariant algorithms via reduction
  - $\Omega(d^2/\varepsilon)$ lower bound for $O(d)$-equivariance, all distributions and single quadratic function.
  - $\Omega(d^2)$ lower bound for $O(d)$-equivariance, single gaussian distribution and single quadratic function. Ex: CIFAR10 has 50k images of size $32 \times 32 \times 3$. To learn whether the Red channel or Green Channel has larger signal strength (in $\ell_2$ sense), FC nets needs around $32^4 \approx 1M$ images if the image distribution is complex enough, e.g. close to i.i.d. gaussian.
  - $\Omega(d)$ lower bound for permutation equivariance, single distribution and single function.
Our Contributions

- Prove orthogonal/permutation equivariance for a broad class of gradient based methods for FC nets.
  - Identifying a sufficient condition for general iterative algorithms to be equivariant, which is easy to check

- Sample complexity lower bounds for equivariant algorithms via reduction
  - \( \Omega(d^2/\varepsilon) \) lower bound for \( O(d) \)-equivariance, all distributions and single quadratic function.
  - \( \Omega(d^2) \) lower bound for \( O(d) \)-equivariance, single gaussian distribution and single quadratic function. *Ex: CIFAR10 has 50k images of size 32 \( \times \) 32 \( \times \) 3. To learn whether the Red channel or Green Channel has larger signal strength (in \( \ell_2 \) sense), FC nets needs around \( 32^4 \approx 1M \) images if the image distribution is complex enough, e.g. close to i.i.d. gaussian.*
  - \( \Omega(d) \) lower bound for permutation equivariance, single distribution and single function.
  - All above problems can be learnt by simple 2-layer CNN with GD using \( O(1) \) samples.
Identifying Algorithmic Equivariance

Table of Contents

1 Introduction

2 Intuition and Warm-up example

3 Identifying Algorithmic Equivariance

4 Lower Bound for Equivariant Algorithms
Iterative Algorithms on Parametric Models

**Algorithm 1 Iterative algorithm \( \mathcal{A} \)**

**Input:** Initial parameter distribution \( P_{\text{init}} \) supported in \( \mathcal{W} = \mathbb{R}^m \), total iterations \( T \), training dataset \( \{x_i, y_i\}_{i=1}^n \), parametric model \( \mathcal{M} : \mathcal{W} \to \mathcal{Y}^X \), (possibly random) iterative update rule \( F(\mathcal{W}, \mathcal{M}, \{x_i, y_i\}_{i=1}^n) \)

**Output:** Hypothesis \( h : \mathcal{X} \to \mathcal{Y} \).

1. Sample \( \mathcal{W}^{(0)} \sim P_{\text{init}} \).
2. for \( t = 0 \) to \( T - 1 \) do
   - \( \mathcal{W}^{(t+1)} = F(\mathcal{W}^{(t)}, \mathcal{M}, \{x_i, y_i\}_{i=1}^n) \).
3. return \( h = \text{sign} [\mathcal{M}[\mathcal{W}^{(T)}]] \).

**Examples (Gradient Based Iterative Algorithms)**

- SGD (+ \( \ell_2 \) regularization)(+ BatchNorm)
- SGD + Momentum/ Adam/ AdaGrad \( (\mathcal{W}^{(t+1)} = F(\{\mathcal{W}^{(t')}\}_{t'=1}^t, \mathcal{M}, \{x_i, y_i\}_{i=1}^n)) \)
Gradient Descent for FC Nets

$$\text{FC-NN}[\mathbf{W}](\mathbf{x}) = \mathbf{W}_L\sigma(\mathbf{W}_{L-1} \cdots \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + b_1) + b_2) + b_{L-1}) + b_L.$$
Gradient Descent for FC Nets

\[ \text{FC-NN}[W](x) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 x + b_1) + b_2) + b_{L-1}) + b_L. \]

**Algorithm 3** Gradient Descent for FC-NN (FC networks)

**Input:** Initial parameter distribution \( P_{init} \), total iterations \( T \), training dataset \( \{x_i, y_i\}_{i=1}^n \), loss function \( \ell \)

Sample \( W^{(0)} \sim P_{init}. \)

for \( t = 0 \) to \( T - 1 \) do

\( W^{(t+1)} = W^{(t)} - \eta \sum_{i=1}^n \nabla \ell(\text{FC-NN}(W^{(t)})(x_i), y_i) \)

return \( h = \text{sign} \left[ \text{FC-NN}[W^{(T)}] \right] \).
Gradient Descent for FC Nets

FC-NN[\(W\)](x) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 x + b_1) + b_2) + b_{L-1}) + b_L.

**Algorithm 4** Gradient Descent for FC-NN (FC networks)

**Input:** Initial parameter distribution \(P_{init}\), total iterations \(T\), training dataset \(\{x_i, y_i\}_{i=1}^n\), loss function \(\ell\)

Sample \(W(0) \sim P_{init}\).

for \(t = 0\) to \(T - 1\) do

\[ W^{(t+1)} = W^{(t)} - \eta \sum_{i=1}^n \nabla \ell(\text{FC-NN}(W^{(t)})(x_i), y_i) \]

return \(h = \text{sign} \left[ \text{FC-NN}[W^{(T)}] \right] \).

**Goal:** FC-NN[\(\tilde{W}^{(t)}\)](Rx) = FC-NN[\(W^{(t)}\)](x), where \(\tilde{W}\) trained on \(Rx_i\) and \(W\) trained on \(x_i\).
Gradient Descent for FC Nets

\[ \text{FC-NN}[W](x) = W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 x + b_1) + b_2) + b_{L-1}) + b_L. \]

Algorithm 5 Gradient Descent for FC-NN (FC networks)

**Input:** Initial parameter distribution \( P_{\text{init}} \), total iterations \( T \), training dataset \( \{x_i, y_i\}_{i=1}^n \), loss function \( \ell \)

Sample \( W^{(0)} \sim P_{\text{init}} \).

for \( t = 0 \) to \( T - 1 \) do

\[ W^{(t+1)} = W^{(t)} - \eta \sum_{i=1}^n \nabla \ell(\text{FC-NN}(W^{(t)})(x_i), y_i) \]

return \( h = \text{sign} \left[ \text{FC-NN}[W^{(T)}] \right] \).

**Goal:** \( \text{FC-NN}[\hat{W}^{(t)}](Rx) = \text{FC-NN}[W^{(t)}](x) \), where \( \hat{W} \) trained on \( Rx_i \) and \( W \) trained on \( x_i \).

**Claim:** \( \hat{W}_1^{(0)} = W_1^{(0)} R^{-1}, \hat{W}_{-1}^{(0)} = W_{-1}^{(0)} \implies \hat{W}_1^{(t)} = W_1^{(t)} R^{-1}, \hat{W}_{-1}^{(t)} = W_{-1}^{(t)}, \forall t. \)
Gradient Descent for FC Nets

FC-NN[W](x) = W_Lσ(W_{L-1} \cdots σ(W_2σ(W_1x + b_1) + b_2) + b_{L-1}) + b_L.

Algorithm 6 Gradient Descent for FC-NN (FC networks)

Input: Initial parameter distribution $P_{init}$, total iterations $T$, training dataset $\{x_i, y_i\}_{i=1}^n$, loss function $\ell$.

Sample $W^{(0)} \sim P_{init}$.

for $t = 0$ to $T - 1$ do

$W^{(t+1)} = W^{(t)} - \eta \sum_{i=1}^n \nabla \ell(FC-NN(W^{(t)})(x_i), y_i)$

return $h = \text{sign}[FC-NN[W^{(T)}]]$.

Goal: FC-NN[\tilde{W}^{(t)}](Rx) = FC-NN[W^{(t)}](x)$, where $\tilde{W}$ trained on $Rx_i$ and $W$ trained on $x_i$.

Claim: $\tilde{W}_1^{(0)} = W_1^{(0)}R^{-1}, \tilde{W}_{-1}^{(0)} = W_{-1}^{(0)} \Rightarrow \tilde{W}_1^{(t)} = W_1^{(t)}R^{-1}, \tilde{W}_{-1}^{(t)} = W_{-1}^{(t)}, \forall t$.

Induction: If $\tilde{W} = (\tilde{W}_1, \tilde{W}_{-1}) = (W_1R^{-1}, W_{-1})$, then $\forall R \in O(d)$,

$\nabla_{\tilde{W}_1} \ell(FC-NN(\tilde{W})(Rx_i), y_i) = \nabla_{W_1} \ell(FC-NN(W)(x_i), y_i)R^{-1}$ (chain rule)

$\nabla_{\tilde{W}_{-1}} \ell(FC-NN(\tilde{W})(Rx_i), y_i) = \nabla_{W_{-1}} \ell(FC-NN(W)(x_i), y_i)$ ($\tilde{W}_1Rx_i = W_1x_i$)
Sufficient Conditions for general equivariance

Theorem

The iterative algorithm $A$ is $G_X$-equivariant if the following conditions are met:

1. There's a group $G_W$ acting on $W$ and a group isomorphism $\tau : G_X \rightarrow G_W$, such that $M[\tau(g)(W)](g(x)) = M[W](x)$, $\forall x \in X$, $W \in W$, $g \in G$.

2. The initialization $P_{\text{init}}$ is invariant under group $G_W$, i.e. $\forall g \in G_W$, $P_{\text{init}} = P_{\text{init}} \circ g^{-1}$.

3. Update rule $F$ is invariant under any joint group action $(g, \tau(g))$, $\forall g \in G$. In other words, $[\tau(g)](F(W, M, \{x_i, y_i\}_{i=1}^n)) = F([\tau(g)](W), M, \{g(x_i), y_i\}_{i=1}^n)$.

Remark

(1) is the minimum expressiveness requirement, (2) is the induction basis and (3) is the for induction
Examples of equivariance

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Sign Flip</th>
<th>Permutation</th>
<th>Orthogonal</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Group</td>
<td>Diagonal, $</td>
<td>M_{ii}</td>
<td>= 1$</td>
<td>Permutation</td>
</tr>
<tr>
<td>Algorithms</td>
<td>AdaGrad, Adam</td>
<td>AdaGrad, Adam</td>
<td>GD</td>
<td>Newton’s method</td>
</tr>
<tr>
<td>Initialization</td>
<td>Symmetric distribution</td>
<td>i.i.d.</td>
<td>i.i.d. Gaussian</td>
<td>All zero</td>
</tr>
<tr>
<td>Regularization</td>
<td>$\ell_p$ norm</td>
<td>$\ell_p$ norm</td>
<td>$\ell_2$ norm</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 1: Examples of gradient-based equivariant training algorithms for FC networks. The initialization requirement is only for the first layer of the network.
Examples of equivariance

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Sign Flip</th>
<th>Permutation</th>
<th>Orthogonal</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Group</td>
<td>Diagonal, $</td>
<td>M_{ii}</td>
<td>= 1$</td>
<td>Permutation</td>
</tr>
<tr>
<td>Algorithms</td>
<td>AdaGrad, Adam</td>
<td>AdaGrad, Adam</td>
<td>GD</td>
<td>Newton’s method</td>
</tr>
<tr>
<td>Initialization</td>
<td>Symmetric distribution</td>
<td>i.i.d.</td>
<td>i.i.d. Gaussian</td>
<td>All zero</td>
</tr>
<tr>
<td>Regularization</td>
<td>$\ell_p$ norm</td>
<td>$\ell_p$ norm</td>
<td>$\ell_2$ norm</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 1: Examples of gradient-based equivariant training algorithms for FC networks. The initialization requirement is only for the first layer of the network.

Equivariance for non-iterative algorithms
Examples of equivariance

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Sign Flip</th>
<th>Permutation</th>
<th>Orthogonal</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Group</td>
<td>Diagonal, $</td>
<td>M_{ii}</td>
<td>= 1$</td>
<td>Permutation</td>
</tr>
<tr>
<td>Algorithms</td>
<td>AdaGrad, Adam</td>
<td>AdaGrad, Adam</td>
<td>GD</td>
<td>Newton’s method</td>
</tr>
<tr>
<td>Initialization</td>
<td>Symmetric dist.</td>
<td>i.i.d.</td>
<td>i.i.d. Gaussian</td>
<td>All zero</td>
</tr>
<tr>
<td>Regularization</td>
<td>$\ell_p$ norm</td>
<td>$\ell_p$ norm</td>
<td>$\ell_2$ norm</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 1: Examples of gradient-based equivariant training algorithms for FC networks. The initialization requirement is only for the first layer of the network.

Equivariance for non-iterative algorithms

**Kernel Regression:** If kernel $K$ is $\mathcal{G}_X$-equivariant, i.e., $\forall g \in \mathcal{G}_X, x, y \in X, K(g(x), g(y)) = K(x, y)$, then algorithm $\text{REG}_K$ is $\mathcal{G}_X$-equivariant.

- Inner product kernel, i.e. $K(x, y) = f(\langle x, y \rangle)$, is $O(d)$-equivariant, including NTK.
- CNTK[ADH+19] is translation and flipping equivariant on images. (Acceleration when data aug is on [LWY+19])
## Examples of equivariance

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Sign Flip</th>
<th>Permutation</th>
<th>Orthogonal</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Group</td>
<td>Diagonal, (</td>
<td>M_{ii}</td>
<td>= 1)</td>
<td>Permutation</td>
</tr>
<tr>
<td>Algorithms</td>
<td>AdaGrad, Adam</td>
<td>AdaGrad, Adam</td>
<td>GD</td>
<td>Newton’s method</td>
</tr>
<tr>
<td>Initialization</td>
<td>Symmetric distribution</td>
<td>i.i.d.</td>
<td>i.i.d. Gaussian</td>
<td>All zero</td>
</tr>
<tr>
<td>Regularization</td>
<td>(\ell_p) norm</td>
<td>(\ell_p) norm</td>
<td>(\ell_2) norm</td>
<td>None</td>
</tr>
</tbody>
</table>

**Table 1:** Examples of gradient-based equivariant training algorithms for FC networks. The initialization requirement is only for the first layer of the network.

### Equivariance for non-iterative algorithms

**Kernel Regression:** If kernel \(K\) is \(\mathcal{G}_\mathcal{X}\)-equivariant, i.e., \(\forall g \in \mathcal{G}_\mathcal{X}, x, y \in \mathcal{X}, K(g(x), g(y)) = K(x, y)\), then algorithm \(\text{REG}_K\) is \(\mathcal{G}_\mathcal{X}\)-equivariant.

- Inner product kernel, i.e. \(K(x, y) = f(\langle x, y \rangle)\), is \(O(d)\)-equivariant, including NTK.
- CNTK\([\text{ADH}+19]\) is translation and flipping equivariant on images. (Acceleration when data aug is on \([\text{LWY}+19]\))

**ERM:** If \(\mathcal{F} = \mathcal{F} \circ \mathcal{G}_\mathcal{X}\), and \(\arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq y_i]\) is unique, then \(\text{ERM}_\mathcal{F}\) is \(\mathcal{G}_\mathcal{X}\)-equivariant.
Table of Contents

1. Introduction
2. Intuition and Warm-up example
3. Identifying Algorithmic Equivariance
4. Lower Bound for Equivariant Algorithms
Recap: Upper and lower bounds related to VC dimension

**Growth function** $\Pi_H(n) := \sup_{x_1, \ldots, x_n \in \mathcal{X}} |\{(h(x_1), \ldots, h(x_n)) | h \in H}\|.$

**VC dimension** $\text{VCdim}(\mathcal{H}) := \max\{n | \Pi_H(n) = 2^n\}.$
Recap: Upper and lower bounds related to VC dimension

**Growth function** \( \Pi_\mathcal{H}(n) := \sup_{x_1, \ldots, x_n \in \mathcal{X}} |\{ (h(x_1), \ldots, h(x_n)) | h \in \mathcal{H}\}|. \)

**VC dimension** \( \text{VCdim}(\mathcal{H}) := \max\{ n | \Pi_\mathcal{H}(n) = 2^n\}. \)

**Lemma (Sauer-Shelah)**

\[
\Pi_\mathcal{H}(n) \leq \left( \frac{en}{\text{VCdim}(\mathcal{H})} \right)^{\text{VCdim}(\mathcal{H})} \quad \text{for } n \geq \text{VCdim}(\mathcal{H})
\]
Recap: Upper and lower bounds related to VC dimension

**Growth function** \( \Pi_H(n) := \sup_{x_1, \ldots, x_n \in \mathcal{X}} |\{(h(x_1), \ldots, h(x_n)) | h \in H\}|. \)

**VC dimension** \( \text{VCdim}(H) := \max\{n | \Pi_H(n) = 2^n\}. \)

**Lemma (Sauer-Shelah)**

\[
\Pi_H(n) \leq \left( \frac{en}{\text{VCdim}(H)} \right)^{\text{VCdim}(H)} \text{ for } n \geq \text{VCdim}(H)
\]

**Theorem ([BEHW89])**

*If \( A \) is consistent and ranges in \( H \), then for any distribution \( P_X, \forall 0 < \varepsilon, \delta < 1, \)

\[
N(A, P_X \diamond H, \varepsilon, \delta) = O\left( \frac{\text{VCdim}(H) \ln \frac{1}{\varepsilon} + \ln \frac{1}{\delta}}{\varepsilon} \right).
\]

(1)

Let \( P_X \) be the set of all possible distributions on \( X \), for any \( 0 < \varepsilon, \delta < 1 \) and \( A, \)

\[
N(A, P_X \diamond H, \varepsilon, \delta) = \Omega\left( \frac{\text{VCdim}(H) + \ln \frac{1}{\delta}}{\varepsilon} \right).
\]

(2)
Reduction to learning with algorithmic equivariance

**Notation:** Define $P_X \circ g$ by $X \sim P_X \iff g^{-1}(X) \sim P_X \circ g$
and $P \circ g$ by $(X, Y) \sim P \iff (g^{-1}(X), Y) \sim P \circ g$, where $P = P_X \diamond h$.
That is, $(P_X \diamond h) \circ g = (P_X \circ g) \circ (h \circ g^{-1})$. 
Reduction to learning with algorithmic equivariance

**Notation:** Define $P_X \circ g$ by $X \sim P_X \iff g^{-1}(X) \sim P_X \circ g$
and $P \circ g$ by $(X, Y) \sim P \iff (g^{-1}(X), Y) \sim P \circ g$, where $P = P_X \diamond h$.
That is, $(P_X \diamond h) \circ g = (P_X \circ g) \circ (h \circ g^{-1})$.

Thus $A$ is $G_X$-equivariant $\iff N^*(A, P, \varepsilon) = N^*(A, P \circ g, \varepsilon)$, $\forall g \in G_X$. Consequently, we have

$$N^*(A, P, \varepsilon) = N^*(A, P \circ G_X, \varepsilon).$$

(3)
Reduction to learning with algorithmic equivariance

Notation: Define $P_X \circ g$ by $X \sim P_X \iff g^{-1}(X) \sim P_X \circ g$
and $P \circ g$ by $(X, Y) \sim P \iff (g^{-1}(X), Y) \sim P \circ g$, where $P = P_X \diamond h$.
That is, $(P_X \diamond h) \circ g = (P_X \circ g) \circ (h \circ g^{-1})$.

Thus $A$ is $G_X$-equivariant $\iff \mathcal{N}^*(A, P, \varepsilon) = \mathcal{N}^*(A, P \circ g, \varepsilon)$, $\forall g \in G_X$. Consequently, we have

$$\mathcal{N}^*(A, P, \varepsilon) = \mathcal{N}^*(A, P \circ h, \varepsilon). \quad (3)$$

Lemma: Let $\mathcal{A}$ be the set of all algorithms and $\mathcal{A}_{G_X}$ be the set of all $G_X$-equivariant algorithms, we have

$$\inf_{A \in \mathcal{A}_{G_X}} \mathcal{N}^*(A, P, \varepsilon) \geq \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, P \circ G_X, \varepsilon) \quad (4)$$

The equality is attained when $G_X$ is a compact group.
**Lemma:** Let $\mathbb{A}$ be the set of all algorithms and $\mathbb{A}_{G_X}$ be the set of all $G_X$-equivariant algorithms, then

$$\inf_{\mathbb{A} \in \mathbb{A}_{G_X}} N^*(\mathbb{A}, P, \varepsilon) \geq \inf_{\mathbb{A} \in \mathbb{A}} N^*(\mathbb{A}, P \circ G_X, \varepsilon)$$

The equality is attained when $G_X$ is a compact group.
Lemma: Let $\mathbb{A}$ be the set of all algorithms and $\mathbb{A}_{G_X}$ be the set of all $G_X$-equivariant algorithms, then

$$\inf_{A \in \mathbb{A}_{G_X}} N^*(A, \mathcal{P}, \varepsilon) \geq \inf_{A \in \mathbb{A}} N^*(A, \mathcal{P} \circ G_X, \varepsilon)$$

The equality is attained when $G_X$ is a compact group.

Proof of Equality

Let $\mu$ be Haar measure, i.e. $\forall S \subset G_X, g \in G_X, \mu(S) = \mu(g \circ S)$. We construct

$$A'(\{x_i, y_i\}_{i=1}^n) = A(\{g(x_i), y_i\}_{i=1}^n) \circ g, \text{ where } g \sim \mu.$$ 

By the definition of Haar measure, $A'$ is $G_X$-equivariant.
Reduction to learning with algorithmic equivariance

Lemma:

\[ \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P}, \varepsilon) \geq \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P} \circ G_{\mathcal{X}}, \varepsilon) \]
Reduction to learning with algorithmic equivariance

**Lemma:**
\[
\inf_{\mathcal{A} \in \mathcal{A}_{G_X}} \mathcal{N}^* (\mathcal{A}, \mathcal{P}, \varepsilon) \geq \inf_{\mathcal{A} \in \mathcal{A}} \mathcal{N}^* (\mathcal{A}, \mathcal{P} \circ G_X, \varepsilon)
\]

**Theorem:** Suppose \( \mathcal{P}_X \) is invariant under group \( G_X \), i.e., \( \mathcal{P}_X \circ G_X = \mathcal{P}_X \),
\[
\inf_{\mathcal{A} \in \mathcal{A}_{G_X}} \mathcal{N}^* (\mathcal{A}, \mathcal{P}_X \diamond H, \varepsilon) \geq \inf_{\mathcal{A} \in \mathcal{A}} \mathcal{N}^* (\mathcal{A}, \mathcal{P}_X \diamond (H \circ G_X), \varepsilon)
\] (5)
The equality is attained when \( G_X \) is a compact group.
Reduction to learning with algorithmic equivariance

Lemma:
\[ \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P}_X, \varepsilon) \geq \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P} \circ \mathcal{G}_X, \varepsilon) \]

Theorem: Suppose \( \mathcal{P}_X \) is invariant under group \( \mathcal{G}_X \), i.e., \( \mathcal{P}_X \circ \mathcal{G}_X = \mathcal{P}_X \),
\[ \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P}_X \diamond \mathcal{H}, \varepsilon) \geq \inf_{A \in \mathcal{A}} \mathcal{N}^*(A, \mathcal{P}_X \diamond (\mathcal{H} \circ \mathcal{G}_X), \varepsilon) \]

The equality is attained when \( \mathcal{G}_X \) is a compact group.

Proof
\[ (\mathcal{P}_X \diamond \mathcal{H}) \circ \mathcal{G}_X = \bigcup_{g \in \mathcal{G}_X} (\mathcal{P}_X \circ g) \diamond (\mathcal{H} \circ g^{-1}) = \bigcup_{g \in \mathcal{G}_X} \mathcal{P}_X \diamond (\mathcal{H} \circ g^{-1}) = \mathcal{P}_X \diamond (\mathcal{H} \circ \mathcal{G}_X). \]
Reduction to learning with algorithmic equivariance

**Lemma:**
\[
\inf_{A \in A} \mathcal{N}^*(A, \mathcal{P}, \varepsilon) \geq \inf_{A \in A} \mathcal{N}^*(A, \mathcal{P} \circ G_X, \varepsilon)
\]

**Theorem:** Suppose \( \mathcal{P}_X \) is invariant under group \( G_X \), i.e., \( \mathcal{P}_X \circ G_X = \mathcal{P}_X \),
\[
\inf_{A \in A} \mathcal{N}^*(A, \mathcal{P}_X \circ H, \varepsilon) \geq \inf_{A \in A} \mathcal{N}^*(A, \mathcal{P}_X \circ (H \circ G_X), \varepsilon)
\] (5)

The equality is attained when \( G_X \) is a compact group.

**Proof**
\[
(\mathcal{P}_X \circ H) \circ G_X = \bigcup_{g \in G_X} (\mathcal{P}_X \circ g) \circ (H \circ g^{-1}) = \bigcup_{g \in G_X} \mathcal{P}_X \circ (H \circ g^{-1}) = \mathcal{P}_X \circ (H \circ G_X).
\]

**Take Home Message:** Learning under equivariance constraint is as hard as learning an augmented function class.
Separation on single function + all distributions

Construction: Let $\mathcal{X} = \mathbb{R}^{2d}$, and $h^*(x) = \text{sign} \left[ \sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2 \right]$. 
Separation on single function + all distributions

Construction: Let $\mathcal{X} = \mathbb{R}^{2d}$, and $h^*(x) = \text{sign} \left[ \sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2 \right]$.

Theorem (single function All distributions)

Let $\mathcal{P} = \{\text{all distributions}\} \circ \{h^*\}$, for any orthogonal equivariant algorithm $A$,

$$\mathcal{N}(A, \mathcal{P}, \varepsilon, \delta) = \Omega((d^2 + \ln \frac{1}{\delta})/\varepsilon),$$

while there’s a 2-layer CNN architecture, such that

$$\mathcal{N}(\text{ERM}_{\text{CNN}}, \mathcal{P}, \varepsilon, \delta) = O \left( \frac{1}{\varepsilon} \left( \log \frac{1}{\varepsilon} + \log \frac{1}{\delta} \right) \right).$$
Separation on single function + all distributions

**Construction:** Let $X = \mathbb{R}^{2d}$, and $h^*(x) = \text{sign}\left[\sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2\right]$.

**Theorem (single function All distributions)**

Let $\mathcal{P} = \{\text{all distributions}\} \diamond \{h^*\}$, for any orthogonal equivariant algorithm $\mathcal{A}$,

$$\mathcal{N}(\mathcal{A}, \mathcal{P}, \varepsilon, \delta) = \Omega\left(\left(\frac{d^2 + \ln \frac{1}{\delta}}{\varepsilon}\right)\right),$$

while there’s a 2-layer CNN architecture, such that

$$\mathcal{N}(\text{ERM}_{\text{CNN}}, \mathcal{P}, \varepsilon, \delta) = O\left(\frac{1}{\varepsilon} \left(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}\right)\right).$$

**Proof of Lower Bound:**

$$\begin{pmatrix} I_d & 0 \\ 0 & -I_d \end{pmatrix}$$

is similar to

$$\begin{pmatrix} 0 & U \\ U^\top & 0 \end{pmatrix}.$$ Thus $\mathcal{H} = \{h_U \mid U \in \mathcal{O}(d)\} \subseteq h^* \circ G_X$, where

$h_U(x) = \text{sign}\left[x_{1:d}^\top U x_{d+1:2d}\right]$. It suffices to show $\text{VCdim}(\mathcal{H}) = \Omega(d^2)$.
Separation on single function + all distributions

Proof of Lower Bound (Cont’d):

Now we claim $\mathcal{H}$ shatters $\{\mathbf{e}_i + \mathbf{e}_{d+j}\}_{1 \leq i < j \leq d}$, i.e. $O(d)$ can shatter $\{\mathbf{e}_i \mathbf{e}_j^\top\}_{1 \leq i < j \leq d}$, which implies $\text{VCdim}(\mathcal{H}) \geq \frac{d(d-1)}{2}$.

Let $\mathfrak{s}o(d) = \{M \mid M = -M^\top, M \in \mathbb{R}^{d \times d}\}$, we know

$$\exp(u) = I_d + u + \frac{u^2}{2} + \cdots \in SO(d), \; \forall u \in \mathfrak{s}o(d).$$

Thus for any sign pattern $\{\sigma_{ij}\}_{1 \leq i < j \leq d}$, let $u = \sum_{1 \leq i < j \leq d} \sigma_{ij}(\mathbf{e}_i \mathbf{e}_j^\top - \mathbf{e}_j \mathbf{e}_i^\top)$ and $\lambda \to 0$,

$$\text{sign} \left[ \langle \exp(\lambda u), \mathbf{e}_i \mathbf{e}_j^\top \rangle \right] = \text{sign} \left[ 0 + \lambda \sigma_{ij} + O(\lambda^2) \right] = \text{sign} \left[ \sigma_{ij} + O(\lambda) \right] = \sigma_{ij}$$
Proof of Upper bound: $\mathcal{N}(\text{ERM}_{\text{CNN}}, \mathcal{P}, \varepsilon, \delta) = O\left(\frac{1}{\varepsilon} \left(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}\right)\right)$

It suffices to construct a CNN with constant VC dimension but still able to express the target quadratic function.

Let $\sigma : \mathbb{R}^d \to \mathbb{R}$, $\sigma(x) = \sum_{i=1}^{d} x_i^2$ (square activation + average pooling), we have

$F_{\text{CNN}} = \left\{ \text{sign} \left[ \sum_{i=1}^{2} a_i \left( \sum_{j=1}^{d} x_{(i-1)d+j}^2 w_1^2 \right) + b \right] \mid a_1, a_2, w_1, b \in \mathbb{R} \right\}$.
Separation on single function + all distributions

Proof of Upper bound: $\mathcal{N}(\text{ERM}_{\text{CNN}}, \mathcal{P}, \varepsilon, \delta) = O\left(\frac{1}{\varepsilon} \left(\log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)\right)$

It suffices to construct a CNN with constant VC dimension but still able to express the target quadratic function.

Let $\sigma : \mathbb{R}^d \to \mathbb{R}$, $\sigma(x) = \sum_{i=1}^{d} x_i^2$ (square activation + average pooling), we have

$$\mathcal{F}_{\text{CNN}} = \left\{ \text{sign} \left[ \sum_{i=1}^{2} a_i \left( \sum_{j=1}^{d} x_{(i-1)d+j}^2 w_1^2 \right) + b \right] | a_1, a_2, w_1, b \in \mathbb{R} \right\}.$$

Remark

The upper bound would still work for 2-layer CNNs with constantly larger filter size, channels. The point here is to show how simple the target is and the huge loss in sample efficiency by ignoring the prior knowledge of the task, i.e. to learn with an orthogonal equivariant algorithm.
Separation on single function + single distribution

Construction: Let $\mathcal{X} = \mathbb{R}^{2d}$, $h^*(x) = \text{sign} \left[ \sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2 \right]$, and $P_{\mathcal{X}} = N(0, I_d)$.

Theorem: Let $\mathcal{P} = \{P_{\mathcal{X}} \diamond h^*\}$. There is a constant $\varepsilon_0 > 0$, if $\mathcal{A}$ is $O(d)$-equivariant, then

$$\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \varepsilon_0) = \Omega(d^2).$$ (6)
Separation on single function + single distribution

Construction: Let $X = \mathbb{R}^{2d}$, $h^*(x) = \text{sign} \left[ \sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2 \right]$, and $P_X = N(0, I_d)$.

Theorem: Let $\mathcal{P} = \{P_X \diamond h^*\}$. There is a constant $\varepsilon_0 > 0$, if $A$ is $O(d)$-equivariant, then

$$\mathcal{N}^*(A, \mathcal{P}, \varepsilon_0) = \Omega(d^2).$$

(6)

Proof Sketch

Define $h_U = \text{sign} \left[ x_{1:d}^\top U x_{d+1:2d} \right]$, $\forall U \in \mathbb{R}^{d \times d}$, we have $\mathcal{H} = \{h_U \mid U \in O(d)\} \subseteq h^* \circ O(2d)$. Thus it suffices to show $\mathcal{N}^*(A, N(0, I_{2d}) \diamond \mathcal{H}, \varepsilon_0) = \Omega(d^2)$ for any algorithm $A$. 

Zhiyuan Li (Princeton University)
Separation on single function + single distribution

**Construction:** Let \( \mathcal{X} = \mathbb{R}^{2d} \), \( h^*(x) = \text{sign}\left[\sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2\right] \), and \( P_{\mathcal{X}} = N(0, I_d) \).

**Theorem:** Let \( \mathcal{P} = \{P_{\mathcal{X}} \odot h^*\} \). There is a constant \( \varepsilon_0 > 0 \), if \( \mathcal{A} \) is \( \mathcal{O}(d) \)-equivariant, then

\[
N^*(\mathcal{A}, \mathcal{P}, \varepsilon_0) = \Omega(d^2). \tag{6}
\]

**Proof Sketch**

Define \( h_U = \text{sign}\left[\sum_{i=1}^{d} U^\top x_{i:d+1:2d}\right] \), \( \forall U \in \mathbb{R}^{d \times d} \), we have \( \mathcal{H} = \{h_U \mid U \in \mathcal{O}(d)\} \subseteq h^* \circ \mathcal{O}(2d) \). Thus it suffices to show \( N^*(\mathcal{A}, N(0, I_{2d}) \odot \mathcal{H}, \varepsilon_0) = \Omega(d^2) \) for any algorithm \( \mathcal{A} \).

**Theorem (Benedek-Itai’s lower bound[BI91])**

For any algorithm \( \mathcal{A} (\varepsilon, \delta) \)-learns \( \mathcal{H} \) with \( n \) i.i.d. samples from a fixed \( P_{\mathcal{X}} \), it must hold that

\[
\Pi_{\mathcal{G}}(n) \geq (1 - \delta)D(\mathcal{H}, \rho_{\mathcal{X}}, 2\varepsilon) \tag{7}
\]

Since \( \Pi_{\mathcal{G}}(n) \leq 2^n \), we have \( \mathcal{N}(\mathcal{A}, P_{\mathcal{X}} \odot \mathcal{H}, \varepsilon, \delta) \geq \log_2 D(\mathcal{H}, \rho_{\mathcal{X}}, 2\varepsilon) + \log_2(1 - \delta). \) Here \( D(\mathcal{H}, \rho_{\mathcal{X}}, 2\varepsilon) \) is the packing number w.r.t. \( \rho_{\mathcal{X}} \), where \( \rho_{\mathcal{X}}(h, h') = \mathbb{P}_{X \sim P_{\mathcal{X}}} [h(X) \neq h'(X)] \).
Separation on single function + single distribution

Define \( h_U = \text{sign} \left[ x_{1:d}^T U x_{d+1:2d} \right] \), then \( \mathcal{H} = \{ h_U \mid U \in \mathcal{O}(d) \} \subseteq h^* \circ \mathcal{O}(2d) \).

Theorem (Benedek-Itai’s lower bound[BI91])

For any algorithm \( A(\varepsilon, \delta) \)-learns \( \mathcal{H} \) with \( n \) i.i.d. samples from a fixed \( P_X \),

\[
\mathcal{N}(A, P_X \circ \mathcal{H}, \varepsilon, \delta) \geq \log_2 D(\mathcal{H}, \rho_X, 2\varepsilon) + \log_2(1 - \delta).
\]

Here \( D(\mathcal{H}, \rho_X, 2\varepsilon) \) is the packing number w.r.t. \( \rho_X \), where \( \rho_X(h, h') = \mathbb{P}_{X \sim P_X} [h(X) \neq h'(X)] \).
Separation on single function + single distribution

Define $h_U = \text{sign} \left[ x_{1:d}^T U x_{d+1:2d} \right]$, then $\mathcal{H} = \{ h_U \mid U \in \mathcal{O}(d) \} \subseteq h^* \circ \mathcal{O}(2d)$.

Theorem (Benedek-Itai’s lower bound[BI91])

For any algorithm $A (\varepsilon, \delta)$-learns $\mathcal{H}$ with $n$ i.i.d. samples from a fixed $P_X$,

$$\mathcal{N}(A, P_X \circ \mathcal{H}, \varepsilon, \delta) \geq \log_2 D(\mathcal{H}, \rho_X, 2\varepsilon) + \log_2(1 - \delta).$$

Here $D(\mathcal{H}, \rho_X, 2\varepsilon)$ is the packing number w.r.t. $\rho_X$, where $\rho_X(h, h') = P_X \sim P_X [h(X) \neq h'(X)]$.

Proof Sketch of $\log_2 D(\mathcal{H}, \rho_X, 2\varepsilon) = \Omega(d^2)$.

1. $\rho_X(h_U, h_V) = \Omega(\frac{\|U-V\|_F}{\sqrt{d}})$.  

---

Zhiyuan Li (Princeton University)  
Fully-Connected Nets vs Conv Nets  
August 19, 2020 @ IJTCS  
Page 28 / 30
Separation on single function + single distribution

Define $h_U = \sign \left[ x_{1:d}^T U x_{d+1:2d} \right]$, then $\mathcal{H} = \{ h_U \mid U \in \mathcal{O}(d) \} \subseteq h^* \circ \mathcal{O}(2d)$.

**Theorem (Benedek-Itai’s lower bound [BI91])**

For any algorithm $A(\varepsilon, \delta)$-learns $\mathcal{H}$ with $n$ i.i.d. samples from a fixed $P_X$,

$$\mathcal{N}(A, P_X \circ \mathcal{H}, \varepsilon, \delta) \geq \log_2 D(\mathcal{H}, \rho_X, 2\varepsilon) + \log_2 (1 - \delta).$$

Here $D(\mathcal{H}, \rho_X, 2\varepsilon)$ is the packing number w.r.t. $\rho_X$, where $\rho_X(h, h') = \mathbb{P}_{X \sim P_X} [h(X) \neq h'(X)]$.

**Proof Sketch of** $\log_2 D(\mathcal{H}, \rho_X, 2\varepsilon) = \Omega(d^2)$.

1. $\rho_X(h_U, h_V) = \Omega\left( \frac{\|U-V\|_F}{\sqrt{d}} \right)$.

2. $\forall u, v \in so(d), \|u\|_\infty, \|v\|_\infty \leq \frac{\pi}{4}, \|\exp(u) - \exp(v)\|_F = \Omega(\|u - v\|_F)$. [Sza97]
Separation on single function + single distribution

Define $h_U = \text{sign} [x_{1:d}^T U x_{d+1:2d}]$, then $\mathcal{H} = \{ h_U \mid U \in \mathcal{O}(d) \} \subseteq h^* \circ \mathcal{O}(2d)$.

Theorem (Benedek-Itai’s lower bound[BI91])

For any algorithm $A (\varepsilon, \delta)$-learns $\mathcal{H}$ with $n$ i.i.d. samples from a fixed $P_X$,

$$\mathcal{N}(A, P_X \diamond \mathcal{H}, \varepsilon, \delta) \geq \log_2 D(\mathcal{H}, \rho_\mathcal{X}, 2\varepsilon) + \log_2 (1 - \delta).$$

Here $D(\mathcal{H}, \rho_\mathcal{X}, 2\varepsilon)$ is the packing number w.r.t. $\rho_\mathcal{X}$, where $\rho_\mathcal{X}(h, h') = \mathbb{P}_{X \sim P_X} [h(X) \neq h'(X)]$.

Proof Sketch of $\log_2 D(\mathcal{H}, \rho_\mathcal{X}, 2\varepsilon) = \Omega(d^2)$.

1. $\rho_\mathcal{X}(h_U, h_V) = \Omega(\|U - V\|_F / \sqrt{d})$.

2. $\forall u, v \in so(d), \|u\|_\infty, \|v\|_\infty \leq \frac{\pi}{4}, \|\exp(u) - \exp(v)\|_F = \Omega(\|u - v\|_F)$. [Sza97]

3. Covering the spectral norm ball in the tangent space of $SO(d)$ at $I_d$ via volume argument.

$$D(\mathcal{H}, \rho_\mathcal{X}, \varepsilon_0) \geq D(so(d) \cap \frac{\pi}{4} B_\infty^2, \|\cdot\|_F / \sqrt{d}, O(\varepsilon_0)) \geq \left( \frac{\text{vol}(so(d) \cap \varepsilon_0 B_\infty^2)}{\text{vol}(so(d) \cap \varepsilon_0 B_2^2)} \right)^{\frac{2}{d(d-1)}} \geq \left( \frac{C}{\varepsilon_0} \right)^{\frac{d(d-1)}{2}}.$$
Sufficient conditions for iterative algorithms to be equivariant
Conclusions

- Sufficient conditions for iterative algorithms to be equivariant
  - SGD + 1st layer is FC + i.i.d. gaussian initialization
    (+ momentum) (+BatchNorm) is orthogonal equivariant.
Conclusions

- Sufficient conditions for iterative algorithms to be equivariant
  - SGD + 1st layer is FC + i.i.d. gaussian initialization
    (+ momentum) (+BatchNorm) is orthogonal equivariant.
- Worst-case sample complexity under equivariance constraint is equal to that of the augmented function class.
Conclusions

- Sufficient conditions for iterative algorithms to be equivariant
  - SGD + 1st layer is FC + i.i.d. gaussian initialization
    (+ momentum) (+BatchNorm) is orthogonal equivariant.
- Worst-case sample complexity under equivariance constraint is equal to that of the augmented function class.
  - There’s a quadratic function which can be learnt by CNN with constant samples for any distribution, but learning it on $d$ dimensional gaussian distribution requires $\Omega(d^2)$ samples.
Thank You!


[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel:


