Why are Convolutional Nets more sample-efficient than Fully-Connected Nets?

Zhiyuan Li, Yi Zhang, Sanjeev Arora

Presented by

Evangelos Chatzipantazis
vaghat@seas.upenn.edu

University of Pennsylvania

April 21, 2022

Or more accurately...

On Equivariant Learning Algorithms and their sample complexity

Table of Contents

Table of Contents

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.
- This design principle typically leads to models with fewer parameters that generalize better from fewer training data points. But this intuition has not been quantified.

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.
- This design principle typically leads to models with fewer parameters that generalize better from fewer training data points. But this intuition has not been quantified.
- Also there is no guarantee that we will indeed end up with fewer parameters for every FCNN. (The CNNs inside the FCNN class are (probably) not all the equivariant maps inside the class).

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.
- This design principle typically leads to models with fewer parameters that generalize better from fewer training data points. But this intuition has not been quantified.
- Also there is no guarantee that we will indeed end up with fewer parameters for every FCNN. (The CNNs inside the FCNN class are (probably) not all the equivariant maps inside the class).
- In principle, for every CNN we can construct an FCNN that can simulate it.

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.
- This design principle typically leads to models with fewer parameters that generalize better from fewer training data points. But this intuition has not been quantified.
- Also there is no guarantee that we will indeed end up with fewer parameters for every FCNN. (The CNNs inside the FCNN class are (probably) not all the equivariant maps inside the class).
- In principle, for every CNN we can construct an FCNN that can simulate it.
- But these FCNNs tend to require many more data points to find the correct CNN inside the class.

Motivation

- Convolutional Networks (CNNs) tend to perform better than Fully-Connected Networks (FCNNs), especially in vision tasks.
- The intuition is that we know beforehand that those tasks satisfy certain symmetries so we build the models to respect this symmetry.
- This design principle typically leads to models with fewer parameters that generalize better from fewer training data points. But this intuition has not been quantified.
- Also there is no guarantee that we will indeed end up with fewer parameters for every FCNN. (The CNNs inside the FCNN class are (probably) not all the equivariant maps inside the class).
- In principle, for every CNN we can construct an FCNN that can simulate it.
- But these FCNNs tend to require many more data points to find the correct CNN inside the class.
- Quantifying it should not rely only on the expressivity of the model, it must be the combination (model, training algorithm).

Main Result

Observe the quantifiers!

Theorem (Informal Theorem)

There exists a distribution, and a labelling function s.t. for all orthogonally equivariant algorithms (including (S)GD on Fully-Connected Nets) we need $\Omega(d^2)$ examples to learn the labelling function, while there is some Convolutional Net that approximates it in $O(1)$ examples (and with GD).

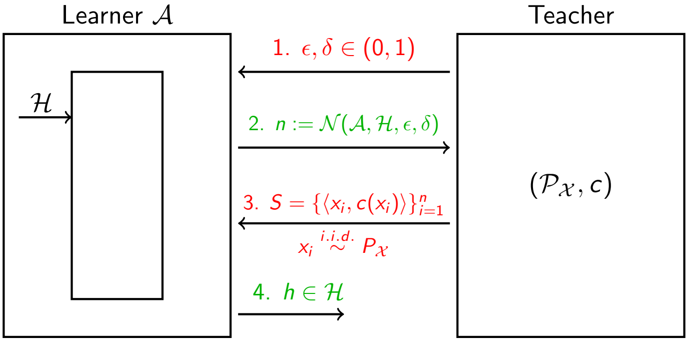
- Show that no "(S)GD + FCNet" can learn some (specific) CNN with good generalization.
- Typical lower bounds that ignore the algorithm do not work. But a lower bound that would take the algorithm into account is hard because of the non-convexity of the problem and the unknown dynamics of (S)GD.
- They unify all Fully-Connected Nets under the concept of orthogonal equivariance. Independently of height, width. With optional weight decay, batchnorm, momentum. Then they prove that orthogonally equivariant algorithms are bad learners.

Table of Contents

Preliminaries on Learning Theory

- Binary classification i.e., $\mathcal{Y} = \{-1, 1\}$. Data domain: $\mathcal{X} = \mathbb{R}^d$ and $P_{\mathcal{X}}$ be a distribution over $\mathcal{X}$.
- We denote a hypothesis by $h : \mathcal{X} \to \mathcal{Y}$ and the hypothesis space by $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$.
- Joint distribution P supported on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$. In this setting $P_{Y|X}$ is a deterministic function, $h^* : \mathbb{R}^d \to \{-1, 1\}$, i.e., $P = P_{\mathcal{X}} \diamond h^*$.
- A problem $\mathcal{P} = \mathcal{P}_{\mathcal{X}} \diamond \mathcal{H} = \{P_{\mathcal{X}} \diamond h | P_{\mathcal{X}} \in \mathcal{P}_{\mathcal{X}}, h \in \mathcal{H}\}$ is a set on joint distributions.
- A learning algorithm $\mathcal{A}$ is a map $\mathcal{A} : \bigcup_{n=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}$. In other words, given a sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ it produces a hypothesis $\mathcal{A}(S) \in \mathcal{H}$. If $\mathcal{A}$ is randomized then it outputs a distribution over hypotheses.
- 0-1 error of a hypothesis $h \in \mathcal{H}$: $\mathrm{err}_P(h) := \mathbb{P}_{(x,y)\sim P}[h(x) \neq y]$

The PAC Learning Protocol


Learner $\mathcal{A}$
Teacher
$\mathcal{H}$
1. $\epsilon, \delta \in (0,1)$
2. $n := \mathcal{N}(\mathcal{A}, \mathcal{H}, \epsilon, \delta)$
3. $S = \{\langle x_i, c(x_i)\rangle\}_{i=1}^n$
$x_i \overset{i.i.d.}{\sim} P_{\mathcal{X}}$
4. $h \in \mathcal{H}$
$(\mathcal{P}_{\mathcal{X}}, c)$

- Teacher: deterministic, infallible. Requests accuracy ϵ, confidence δ.
- Realizable setting: $c \in \mathcal{H}$ (learner has the capacity to learn)
- We say the algorithm $\mathcal{A}$ (ϵ, δ)-learns $\mathcal{H}$ if for all $P_{\mathcal{X}}, c \in \mathcal{H}$:

$$\mathbb{P}_{S \sim P_{\mathcal{X}}^n}[\mathrm{err}_{P_{\mathcal{X}} \diamond c}(\mathcal{A}(S)) < \epsilon] \geq 1 - \delta$$

The PAC Learning Protocol

- If $\mathcal{A}$ is randomized the probability is over its randomization too.
- Sample complexity $\mathcal{N}(\mathcal{A}, \mathcal{H}, \epsilon, \delta)$: smallest $n \in \mathbb{N}$ s.t. $\mathcal{A}$ (ϵ, δ)-learns $\mathcal{H}$.
- One (computer scientist) could argue that this is not an algorithm! Time-complexity requirements are missing. The learner might not even be computable.

The PAC Learning Protocol

- If $\mathcal{A}$ is randomized the probability is over its randomization too.
- Sample complexity $\mathcal{N}(\mathcal{A}, \mathcal{H}, \epsilon, \delta)$: smallest $n \in \mathbb{N}$ s.t. $\mathcal{A}$ (ϵ, δ)-learns $\mathcal{H}$.
- One (computer scientist) could argue that this is not an algorithm! Time-complexity requirements are missing. The learner might not even be computable.

Theorem (Blumer et al. 1989)

If $\mathcal{A}$ is consistent (the output hypothesis agrees with the sample) and ranges in $\mathcal{H}$, then for any $\epsilon, \delta \in (0,1)$

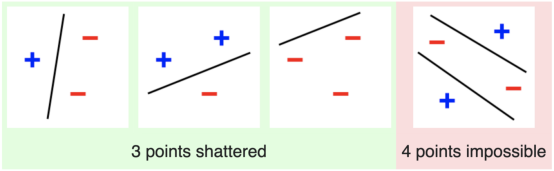
$$\mathcal{N}(\mathcal{A}, \mathcal{H}, \epsilon, \delta) = O\left(\frac{1}{\epsilon}(\mathrm{VCdim}(\mathcal{H}) \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta})\right)$$

For any $\mathcal{A}$ and any $\epsilon, \delta \in (0,1)$

$$\mathcal{N}(\mathcal{A}, \mathcal{H}, \epsilon, \delta) = \Omega\left(\frac{1}{\epsilon}(\mathrm{VCdim}(\mathcal{H}) + \ln \frac{1}{\delta})\right)$$

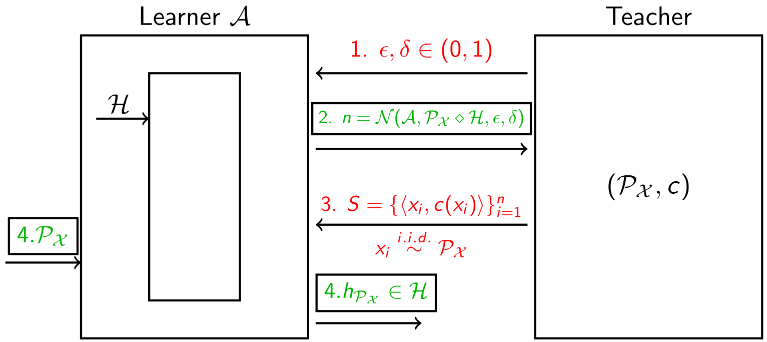
VC-dimension and Growth Function

- Intricate connection between PAC-learnability and VC- dimension.
- *Growth function* $\Pi_{\mathcal{H}}(n) := \sup_{x_1,\cdots,x_n \in \mathcal{X}} |\{(h(x_1),\cdots,h(x_n)) | h \in \mathcal{H}\}|$
- $\mathrm{VCdim}(\mathcal{H}) := \max\{n | \Pi_{\mathcal{H}} = 2^n\}$
- VCdim of 2D linear functions is 3.



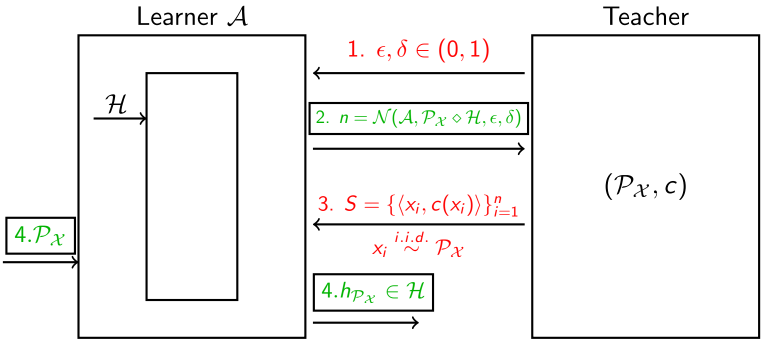
- VCdim:
  - Given $\mathcal{H}$, select a configuration of d points from $\mathcal{X}$.
  - An adversary that also knows $\mathcal{H}$ will label them.
  - You need to select an $h \in \mathcal{H}$ that classifies them correctly.
  - If you can do it, $\mathrm{VCdim} \geq d$, else $\mathrm{VCdim} < d$.
- Let $\mathrm{VCdim}(\mathcal{H}) = d$. For $n < d$, $\Pi_{\mathcal{H}}(n) = 2^n$. For $n \geq d$, $\Pi_{\mathcal{H}}(n) \leq (\frac{en}{d})^d$

[Benedek and Itai 1991] Learning with Fixed Distributions


Learner $\mathcal{A}$
Teacher
1. $\epsilon, \delta \in (0,1)$
$\mathcal{H}$
2. $n = \mathcal{N}(\mathcal{A}, \mathcal{P}_{\mathcal{X}} \diamond \mathcal{H}, \epsilon, \delta)$
$(\mathcal{P}_{\mathcal{X}}, c)$
3. $S = \{\langle x_i, c(x_i)\rangle\}_{i=1}^{n}$
$x_i \overset{i.i.d.}{\sim} \mathcal{P}_{\mathcal{X}}$
4.$\mathcal{P}_{\mathcal{X}}$
4.$h_{\mathcal{P}_{\mathcal{X}}} \in \mathcal{H}$

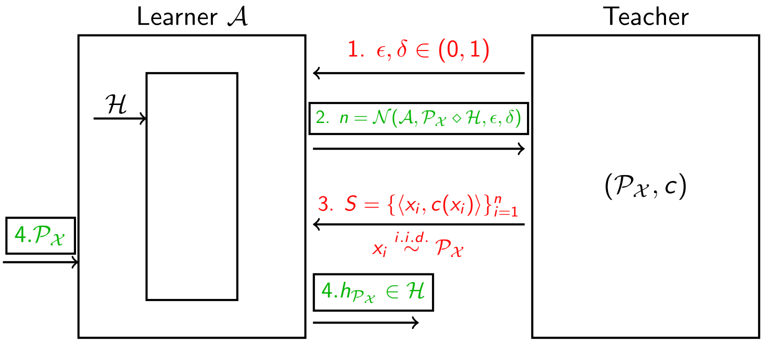
- In this framework learners $\mathcal{A}$ are stronger.

[Benedek and Itai 1991] Learning with Fixed Distributions


Learner $\mathcal{A}$
Teacher
$\mathcal{H}$
1. $\epsilon, \delta \in (0,1)$
2. $n = \mathcal{N}(\mathcal{A}, \mathcal{P}_\mathcal{X} \diamond \mathcal{H}, \epsilon, \delta)$
3. $S = \{\langle x_i, c(x_i)\rangle\}_{i=1}^n$
$x_i \overset{i.i.d.}{\sim} \mathcal{P}_\mathcal{X}$
$4. h_{\mathcal{P}_\mathcal{X}} \in \mathcal{H}$
$4. \mathcal{P}_\mathcal{X}$
$(\mathcal{P}_\mathcal{X}, c)$

- In this framework learners $\mathcal{A}$ are stronger.
- For example, if $\mathcal{P}_\mathcal{X}$ is discrete then every class $\mathcal{H}$ is learnable w.r.t. $\mathcal{P}_\mathcal{X}$ (in realizable setting). How? Forget the ϵ-tail!

[Benedek and Itai 1991] Learning with Fixed Distributions


Learner $\mathcal{A}$
Teacher
1. $\epsilon, \delta \in (0,1)$
$\mathcal{H}$
2. $n = \mathcal{N}(\mathcal{A}, \mathcal{P}_{\mathcal{X}} \diamond \mathcal{H}, \epsilon, \delta)$
$(\mathcal{P}_{\mathcal{X}}, c)$
3. $S = \{\langle x_i, c(x_i)\rangle\}_{i=1}^{n}$
$x_i \overset{i.i.d.}{\sim} \mathcal{P}_{\mathcal{X}}$
4.$\mathcal{P}_{\mathcal{X}}$
4.$h_{\mathcal{P}_{\mathcal{X}}} \in \mathcal{H}$

- In this framework learners $\mathcal{A}$ are stronger.
- For example, if $\mathcal{P}_{\mathcal{X}}$ is discrete then every class $\mathcal{H}$ is learnable w.r.t. $\mathcal{P}_{\mathcal{X}}$ (in realizable setting). How? Forget the ϵ-tail!
- Find the k most frequent points, where k is s.t. $C = \{x_i\}_{i=1}^{k}$ s.t. $\sum_{i=1}^{k} \mathcal{P}_{\mathcal{X}}(x_i) > 1 - \epsilon$. Suppose the k-th is the least probable in C. Ask $n \geq \ln(k/\delta)/\mathcal{P}_{\mathcal{X}}(x_k)$ samples. Each element of C appears in the samples at least once w.p. at least $1 - \delta$. Output any hypothesis consistent with the sample.

[Benedek and Itai 1991] Learning with Fixed Distributions

- This framework opens up the possibility to discuss learning a problem $\mathcal{P}$ instead of just learning a class $\mathcal{H}$.
- Especially for lower bounds on the sample complexity because one needs to account for those algorithms that "hard-code" $\mathcal{P}$.
- Given a problem $\mathcal{P}$ and an algorithm $\mathcal{A}$, we can define the (ϵ, δ)-*sample complexity* $\mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta)$ as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$ w.p. at least $1 - \delta$ over the randomness of $S = \{x_i, y_i\}_{i=1}^n$ and the algorithm, $\mathrm{err}_P(\mathcal{A}(S)) \leq \epsilon$.

[Benedek and Itai 1991] Learning with Fixed Distributions

- This framework opens up the possibility to discuss learning a problem $\mathcal{P}$ instead of just learning a class $\mathcal{H}$.
- Especially for lower bounds on the sample complexity because one needs to account for those algorithms that "hard-code" $\mathcal{P}$.
- Given a problem $\mathcal{P}$ and an algorithm $\mathcal{A}$, we can define the (ϵ, δ)*-sample complexity* $\mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta)$ as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$ w.p. at least $1 - \delta$ over the randomness of $S = \{x_i, y_i\}_{i=1}^n$ and the algorithm, $\mathrm{err}_P(\mathcal{A}(S)) \leq \epsilon$.
- Also, we define the ϵ*-expected sample complexity* $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon)$ as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$, $\mathbb{E}_{S \sim P^n}[\mathrm{err}_P(\mathcal{A}(S))] \leq \epsilon$

[Benedek and Itai 1991] Learning with Fixed Distributions

- This framework opens up the possibility to discuss learning a problem $\mathcal{P}$ instead of just learning a class $\mathcal{H}$.
- Especially for lower bounds on the sample complexity because one needs to account for those algorithms that "hard-code" $\mathcal{P}$.
- Given a problem $\mathcal{P}$ and an algorithm $\mathcal{A}$, we can define the *(ϵ, δ)-sample complexity $\mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta)$* as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$ w.p. at least $1 - \delta$ over the randomness of $S = \{x_i, y_i\}_{i=1}^n$ and the algorithm, $\mathrm{err}_P(\mathcal{A}(S)) \leq \epsilon$.
- Also, we define the *ϵ-expected sample complexity $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon)$* as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$, $\mathbb{E}_{S \sim P^n}[\mathrm{err}_P(\mathcal{A}(S))] \leq \epsilon$
- $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon + \delta) \leq \mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta) \leq \mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon\delta), \forall \epsilon, \delta \in [0, 1]$

[Benedek and Itai 1991] Learning with Fixed Distributions

- This framework opens up the possibility to discuss learning a problem $\mathcal{P}$ instead of just learning a class $\mathcal{H}$.
- Especially for lower bounds on the sample complexity because one needs to account for those algorithms that "hard-code" $\mathcal{P}$.
- Given a problem $\mathcal{P}$ and an algorithm $\mathcal{A}$, we can define the *(ϵ, δ)-sample complexity $\mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta)$* as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$ w.p. at least $1 - \delta$ over the randomness of $S = \{x_i, y_i\}_{i=1}^n$ and the algorithm, $\mathrm{err}_P(\mathcal{A}(S)) \leq \epsilon$.
- Also, we define the *ϵ-expected sample complexity $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon)$* as the smallest $n \in \mathbb{N}$ s.t. $\forall P \in \mathcal{P}$, $\mathbb{E}_{S \sim P^n}[\mathrm{err}_P(\mathcal{A}(S))] \leq \epsilon$
- $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon + \delta) \leq \mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta) \leq \mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon\delta), \forall \epsilon, \delta \in [0, 1]$

Theorem (Benedek and Itai 1991)

For any algorithm $\mathcal{A}$ that $(\epsilon, \delta)-$learns $\mathcal{H}$ with n i.i.d. samples from a fixed distribution $\mathcal{P}_{\mathcal{X}}$ it must hold:

$$\mathcal{N}(\mathcal{A}, \mathcal{P}_{\mathcal{X}}, \epsilon, \delta) \geq \log[(1 - \delta) D(\mathcal{H}, \rho_{\mathcal{X}}, 2\epsilon)]$$

$D(\mathcal{H}, \rho_{\mathcal{X}}, \epsilon)$ ϵ-packing number i.e., the largest number of ϵ-far hypotheses $h \in \mathcal{H}$. Distance is measured by $\rho_{\mathcal{X}}(h, h') := \mathbb{P}_{X \sim P_{\mathcal{X}}}[h(X) \neq h'(X)]$.

Table of Contents

Parametric Models: FC Networks, CNN Networks

Let $\mathcal{W}$ be a parameter space (in our cases Euclidean space). A parametric model is an operator $\mathcal{M} : \mathcal{W} \to \mathcal{Y}^{\mathcal{X}}$ i.e., given $W \in \mathcal{W}, \mathcal{M}[W] : \mathcal{X} \to \mathcal{Y}$.

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

FC-NN$[\mathbf{W}](\mathbf{x}) = W^L\sigma(W^{L-1}...\sigma(W_2\sigma(W_1\mathbf{x}+b_1)+b_2)+b_{L-1})+b_L$,
where $\mathbf{W} = (\{W_i, b_i\}_{i=1}^L), W_i \in \mathbb{R}^{d_{i-1}\times d_i}, b_i \in \mathbb{R}^{d_i}, d_0 = d, d_L = 1$.
Here $\sigma : \mathbb{R} \to \mathbb{R}$ activation, (abusing notation) $[\sigma(x)]_i = \sigma(x_i)$

Convolutional Neural Network: $\mathbb{R}^d \to \mathbb{R}$

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

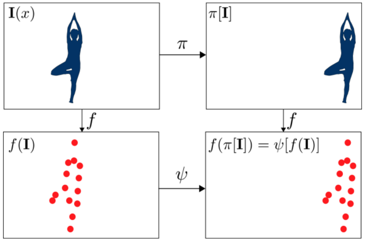
where $\mathbf{W} = (\mathbf{w}, \mathbf{a}, b) \in \mathbb{R}^k \times \mathbb{R}^r \times \mathbb{R}, d = d'r. * : \mathbb{R}^k \times \mathbb{R}^d \to \mathbb{R}^d$ convolution, defined as $[\mathbf{w} * \mathbf{x}]_i = \sum_{j=1}^k w_j x_{[i-j-1 \bmod d]+1}$ and $\sigma : \mathbb{R}^{d'} \to \mathbb{R}$ is a composition of (strided) pooling and element-wise non-linearity.

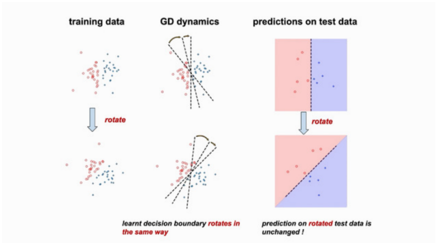
Parametric Models: FC Networks, CNN Networks

- In principle, any CNN can be simulated by a large enough FCNN.
- There might even exist algorithms that, given a finite sample, can always find this CNN among the FCNNs.
- But (stochastic) gradient descent on FCNNs with gaussian initialization and optional weight decay, momentum, batch norm cannot do that for all CNNs.
- The authors prove that there is a CNN such that in the worst case, this algorithm requires samples that scale quadratically with the dimension of the features to achieve that.
- They find an explicit joint distribution on which some CNN is more-sample efficient than every FCNet.

Table of Contents

Equivariance: from Design to Analysis tool





- Usually task unknown, but symmetries (group) known. We exploit that by building parametric models that respect those symmetries (equivariance).

Definition (Equivariance)

Consider a group $\mathcal{G}$ acting on $\mathcal{X}$ via "$\cdot$" and on $\mathcal{X}'$ via "$*$". We call a map $f : \mathcal{X} \to \mathcal{X}'$ $\mathcal{G}$-equivariant (w.r.t. these group actions) if it satisfies:

$$f(g.x) = g * f(x), \forall g \in \mathcal{G}, x \in \mathcal{X}.$$

Equivariant Algorithms

- The algorithm is a function that, given a sample $S = \{x_i, y_i\}_{i=1}^n$, produces a hypothesis $h \in \mathcal{H}$.
- If we define the actions:
  - $g.S = \{(g(x_i), y_i)\}_{i=1}^n$
  - $g * h = h \circ g^{-1}$

  Then the definition above specializes to:

Definition (Equivariant algorithm)

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

$$\mathcal{A}(\{(g(x_i), y_i)\}_{i=1}^n)(g(x)) \overset{d}{=} \mathcal{A}(\{x_i, y_i\}_{i=1}^n)(x)$$

- If $\mathcal{A}$ outputs h on data S and outputs $h \circ g^{-1}$ on data $g.S$, it is equivariant.

Equivariant Algorithms

- (S)GD on FCNNs is (rotational) $\mathcal{O}(d)$-equivariant.
- (S)GD on a network with its first layer fully-connected is (rotational) $\mathcal{O}(d)$-equivariant even if the rest of the layers are e.g. convolutional.

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

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

```
Input: Initial parameter distribution $P_{init}$ , total iterations $T$, training dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$, loss function $\ell$
  Sample $\mathbf{W}^{(0)} \sim P_{init}$.
  for $t = 0$ to $T-1$ do
    $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta\sum_{i=1}^n \nabla\ell(\text{FC-NN}(\mathbf{W}^{(t)})(\mathbf{x}_i), y_i)$
  return $h = \text{sign}\left[\text{FC-NN}[\mathbf{W}^{(T)}]\right]$.
```

Goal: $\text{FC-NN}[\widetilde{\mathbf{W}}^{(t)}](R\mathbf{x}) = \text{FC-NN}[\mathbf{W}^{(t)}](\mathbf{x})$, where $\widetilde{\mathbf{W}}$ trained on $R\mathbf{x}_i$ and $\mathbf{W}$ trained on $\mathbf{x}_i$.

Claim: $\widetilde{W}_1^{(0)} = W_1^{(0)}R^{-1}, \widetilde{\mathbf{W}}_{-1}^{(0)} = \mathbf{W}_{-1}^{(0)} \Longrightarrow \widetilde{W}_1^{(t)} = W_1^{(t)}R^{-1}, \widetilde{\mathbf{W}}_{-1}^{(t)} = \mathbf{W}_{-1}^{(t)}, \forall t$.

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

$$\nabla_{\widetilde{W}_1}\ell(\text{FC-NN}(\widetilde{\mathbf{W}})(R\mathbf{x}_i), y_i) = \nabla_{W_1}\ell(\text{FC-NN}(\mathbf{W})(\mathbf{x}_i), y_i)R^{-1} \quad \text{(chain rule)}$$

$$\nabla_{\widetilde{\mathbf{W}}_{-1}}\ell(\text{FC-NN}(\widetilde{\mathbf{W}})(R\mathbf{x}_i), y_i) = \nabla_{\mathbf{W}_{-1}}\ell(\text{FC-NN}(\mathbf{W})(\mathbf{x}_i), y_i) \quad (\widetilde{W}_1R\mathbf{x}_i = W_1\mathbf{x}_i)$$

- Is (S)GD on CNNs $\mathcal{O}(d)$-equivariant? Not in general! No proof.
- Intuitively before a rotation $x \to Rx$ was absorbed by $W_1 \to W_1R^{-1}$. If we stack the convolutional weights on a matrix W_1, then W_1R^{-1} will not be implementable by a convolution.

Iterative Equivariant Algorithms on Parametric Models

Algorithm 1 Iterative algorithm $\mathcal{A}$

```
Input: Initial parameter distribution $P_{init}$ supported in $\mathcal{W} = \mathbb{R}^m$, total iterations $T$, training dataset
  $\{\mathbf{x}_i, y_i\}_{i=1}^n$, parametric model $\mathcal{M} : \mathcal{W} \to \mathcal{Y}^{\mathcal{X}}$,
  (possibly random) iterative update rule $F(\mathbf{W}, \mathcal{M}, \{\mathbf{x}_i, y_i\}_{i=1}^n)$
Output: Hypothesis $h : \mathcal{X} \to \mathcal{Y}$.
  Sample $\mathbf{W}^{(0)} \sim P_{init}$.
  for $t = 0$ to $T - 1$ do
    $\mathbf{W}^{(t+1)} = F(\mathbf{W}^{(t)}, \mathcal{M}, \{\mathbf{x}_i, y_i\}_{i=1}^n)$.
  return $h = \mathrm{sign}\left[\mathcal{M}[\mathbf{W}^{(T)}]\right]$.
```

Theorem

The iterative algorithm $\mathcal{A}$ is $\mathcal{G}_{\mathcal{X}}$-equivariant if the following conditions are met:

1. *There's a group $\mathcal{G}_{\mathcal{W}}$ acting on $\mathcal{W}$ and a group isomorphism $\tau : \mathcal{G}_{\mathcal{X}} \to \mathcal{G}_{\mathcal{W}}$, such that $\mathcal{M}[\tau(g)(\mathbf{W})](g(\mathbf{x})) = \mathcal{M}[\mathbf{W}](\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}, \mathbf{W} \in \mathcal{W}, g \in \mathcal{G}$.*
2. *The initialization P_{init} is invariant under group $\mathcal{G}_{\mathcal{W}}$, i.e. $\forall g \in \mathcal{G}_{\mathcal{W}},\ P_{init} = P_{init} \circ g^{-1}$.*
3. *Update rule F is invariant under any joint group action $(g, \tau(g))$, $\forall g \in \mathcal{G}$. In other words, $[\tau(g)](F(\mathbf{W}, \mathcal{M}, \{\mathbf{x}_i, y_i\}_{i=1}^n)) = F([\tau(g)](\mathbf{W}), \mathcal{M}, \{g(\mathbf{x}_i), y_i\}_{i=1}^n)$.*

Remark

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

Non-Iterative Equivariant Algorithms

For non-iterative algorithms, the authors also show that:

- Kernel Regression:

$$\mathrm{REG}_K(\{x_i, y_i\}_{i=1}^n)(x) := 1[K(x, X_n) \cdot K(X_n, X_n)^\dagger y \geq 0]$$

  is $\mathcal{G}_\mathcal{X}$-equivariant if the kernel $K(x, y)$ is invariant i.e., $K(gx, gy) = K(x, y), \forall x, y \in \mathcal{X}, g \in \mathcal{G}_\mathcal{X}$.

- Empirical Risk Minimization:

$$\mathrm{ERM}_\mathcal{H}(\{x_i, y_i\}_{i=1}^n) := \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} 1[h(x_i) \neq y_i]$$

  is $\mathcal{G}_\mathcal{X}$-equivariant if the minimum is unique and $\mathcal{H} = \mathcal{H} \circ \mathcal{G}_\mathcal{X}$.

Table of Contents

Learning with algorithmic equivariance: Warm-Up Example

- $\mathcal{X} = \mathbb{R}^d$, P is uniform on the set $\{(\mathbf{e}_i y, y) | i \in [d], y = \pm 1\}$
- Observe that the sign of the one-hot encoded vectors determine their label.
- A simple average pooling would learn this task.
- If $\mathcal{A}$ is orthogonal equivariant then $\forall R \in \mathcal{O}(d)$,

$$\mathcal{A}(\{Rx_i, y_i\}_{i=1}^n)(Rx) = \mathcal{A}(\{x_i, y_i\}_{i=1}^n)(x)$$

- Let $S = \{x_i, y_i\}_{i=1}^n$ be a sample from P.
- There is a function f_S s.t. $\mathcal{A}(S)(x) = f_S(x^T x_1, \cdots, x^T x_n)$
- When $n \leq d/2$, with probability at least $1/2$, $\mathcal{A}(S)(x) = f_S(0, 0, \cdots, 0)$. That's because any unseen point would produce only zero inner products.
- Thus $\mathcal{A}$ outputs the wrong answer with probability at least $1/4$.

Learning with algorithmic equivariance

Notation:

- $g : \mathcal{X} \to \mathcal{X}$ is a 1-1 transformation. We can define $gP_{\mathcal{X}} = P_{\mathcal{X}} \circ g^{-1}$ so that $X \sim P_{\mathcal{X}} \iff gX \sim gP_{\mathcal{X}} = P_{\mathcal{X}} \circ g^{-1}$
- $gP = P \circ g^{-1}$ similarly as $X \sim P_{\mathcal{X}} \iff (gX, Y) \sim P \circ g^{-1}$ where $P = P_{\mathcal{X}} \diamond h$. In other words, $(P_{\mathcal{X}} \diamond h) \circ g = (P_{\mathcal{X}} \circ g) \diamond (h \circ g)$

Observation:

- Clearly, for any $\mathcal{G}_{\mathcal{X}}$-equivariant algorithm we have:

$$\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon) = \mathcal{N}^*(\mathcal{A}, \mathcal{P} \circ g, \epsilon), \forall g \in \mathcal{G}_{\mathcal{X}}.$$

or

$$\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon) = \mathcal{N}^*(\mathcal{A}, \mathcal{P} \circ \mathcal{G}_{\mathcal{X}}, \epsilon)$$

Learning with algorithmic equivariance

Theorem (Reducing Learning using Equivariant Algorithms to Learning an augmented class)

1. *Let $\mathbb{A}$ be the set of all algorithms and $\mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$ be the set of all $\mathcal{G}_{\mathcal{X}}-$equivariant algorithms. Then,*

$$\inf_{\mathcal{A}\in\mathbb{A}_{\mathcal{G}_{\mathcal{X}}}} \mathcal{N}^*(\mathcal{A},\mathcal{P},\epsilon) \geq \inf_{\mathcal{A}\in\mathbb{A}} \mathcal{N}^*(\mathcal{A},\mathcal{P}\circ\mathcal{G}_{\mathcal{X}},\epsilon) \tag{1}$$

2. *Let the set $\mathcal{P}_{\mathcal{X}}$ be invariant under $\mathcal{G}_{\mathcal{X}}$, i.e., $\mathcal{P}_{\mathcal{X}}\circ\mathcal{G}_{\mathcal{X}} = \mathcal{P}_{\mathcal{X}}$. Then,*

$$\inf_{\mathcal{A}\in\mathbb{A}_{\mathcal{G}_{\mathcal{X}}}} \mathcal{N}^*(\mathcal{A},\mathcal{P}_{\mathcal{X}}\diamond\mathcal{H},\epsilon) \geq \inf_{\mathcal{A}\in\mathbb{A}} \mathcal{N}^*(\mathcal{A},\mathcal{P}_{\mathcal{X}}\diamond(\mathcal{H}\circ\mathcal{G}_{\mathcal{X}}),\epsilon) \tag{2}$$

The equalities in both are attained when $\mathcal{G}_{\mathcal{X}}$ is a compact group.

- Take home message: Learning under algorithmic equivariance is at least as hard as learning an augmented function class.

Behind the proofs

Suppose we need to learn a single labelling function h^*:

- What is the best algorithm (in sample complexity) in $\mathbb{A}$?

Behind the proofs

Suppose we need to learn a single labelling function h^*:

- What is the best algorithm (in sample complexity) in $\mathbb{A}$?

  An algorithm whose output is h^*! (but always)

Behind the proofs

Suppose we need to learn a single labelling function h^*:

- What is the best algorithm (in sample complexity) in $\mathbb{A}$?

  An algorithm whose output is h^*! (but always)

- But this is not orthogonal equivariant! If you just rotate the data points it observed (not the distribution or the labels): $\{g(x_i), y_i\}_{i=1}^n$ it will still output h^*, not $h^* \circ g^{-1}$!

Behind the proofs

Suppose we need to learn a single labelling function h^*:

- What is the best algorithm (in sample complexity) in $\mathbb{A}$?

  An algorithm whose output is h^*! (but always)
- But this is not orthogonal equivariant! If you just rotate the data points it observed (not the distribution or the labels): $\{g(x_i), y_i\}_{i=1}^n$ it will still output h^*, not $h^* \circ g^{-1}$!
- If $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_\mathcal{X}}$ then by definition $\mathcal{A}$ has the same performance on $P := P_\mathcal{X} \diamond \{h^*\}$ and the rotated $P \circ g^{-1} := (P_\mathcal{X} \circ g^{-1}) \diamond (h^* \circ g^{-1})$.
- Because now the learner $\mathcal{A}$ observes:
  - the rotated data points with the same probability, i.e., $X \to gX$,
  - At the same time, the label stays fixed i.e. $(h^* \circ g^{-1})(gx_i) = h^*(x_i) = y_i$.

Behind the proofs

Suppose we need to learn a single labelling function h^*:

- What is the best algorithm (in sample complexity) in $\mathbb{A}$?

  An algorithm whose output is h^*! (but always)

- But this is not orthogonal equivariant! If you just rotate the data points it observed (not the distribution or the labels): $\{g(x_i), y_i\}_{i=1}^n$ it will still output h^*, not $h^* \circ g^{-1}$!
- If $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_\mathcal{X}}$ then by definition $\mathcal{A}$ has the same performance on $P := P_\mathcal{X} \diamond \{h^*\}$ and the rotated $P \circ g^{-1} := (P_\mathcal{X} \circ g^{-1}) \diamond (h^* \circ g^{-1})$.
- Because now the learner $\mathcal{A}$ observes:
  - the rotated data points with the same probability, i.e., $X \to gX$,
  - At the same time, the label stays fixed i.e. $(h^* \circ g^{-1})(gx_i) = h^*(x_i) = y_i$.
- Dually $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_\mathcal{X}}$:
  (ϵ, δ)-learn h^* on $\mathcal{P}_\mathcal{X} \circ g \implies (\epsilon, \delta)$-learn $h^* \circ g^{-1}$ on $\mathcal{P}_\mathcal{X}$.
- And thus, $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon) = \mathcal{N}^*(\mathcal{A}, \mathcal{P} \circ g^{-1}, \epsilon)$, for all $g \in \mathcal{G}_\mathcal{X}$. That's an invariant for all orthogonal equivariant learning algorithms.

Behind the proofs

- We can consider two important learning settings where the set $\mathcal{P}_{\mathcal{X}}$ satisfies $\mathcal{P}_{\mathcal{X}} \circ \mathcal{G}_{\mathcal{X}} = \mathcal{P}_{\mathcal{X}}$.
  1. All distributions on $\mathcal{X}$. Then if $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$ it learns not only h^* over all distributions but all $h^* \circ \mathcal{G}_{\mathcal{X}}$ over all distributions. But this is the standard PAC setting and learning is at least as hard for $\mathcal{A}$ as the best algorithm on $h^* \circ \mathcal{G}_{\mathcal{X}}$. Show: $\text{VCdim}(h^* \circ \mathcal{G}_{\mathcal{X}}) = \Omega(d^2)$.

Behind the proofs

- We can consider two important learning settings where the set $\mathcal{P}_\mathcal{X}$ satisfies $\mathcal{P}_\mathcal{X} \circ \mathcal{G}_\mathcal{X} = \mathcal{P}_\mathcal{X}$.
  1. All distributions on $\mathcal{X}$. Then if $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_\mathcal{X}}$ it learns not only h^* over all distributions but all $h^* \circ \mathcal{G}_\mathcal{X}$ over all distributions. But this is the standard PAC setting and learning is at least as hard for $\mathcal{A}$ as the best algorithm on $h^* \circ \mathcal{G}_\mathcal{X}$. Show: $\mathrm{VCdim}(h^* \circ \mathcal{G}_\mathcal{X}) = \Omega(d^2)$.
  2. $\mathcal{P}_\mathcal{X} = \{\mathcal{N}(0, I_d)\}$. Now learning over all rotated copies of the joint is learning all $h^* \circ \mathcal{G}_\mathcal{X}$ on the gaussian. Now a bound on VCdim will not suffice and we need to bound the sample complexity in another way. We are not in the standard PAC learning setting. We are in Benedek-Itai's world!

Behind the proofs

- We can consider two important learning settings where the set $\mathcal{P}_{\mathcal{X}}$ satisfies $\mathcal{P}_{\mathcal{X}} \circ \mathcal{G}_{\mathcal{X}} = \mathcal{P}_{\mathcal{X}}$.
  1. All distributions on $\mathcal{X}$. Then if $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$ it learns not only h^* over all distributions but all $h^* \circ \mathcal{G}_{\mathcal{X}}$ over all distributions. But this is the standard PAC setting and learning is at least as hard for $\mathcal{A}$ as the best algorithm on $h^* \circ \mathcal{G}_{\mathcal{X}}$. Show: $\text{VCdim}(h^* \circ \mathcal{G}_{\mathcal{X}}) = \Omega(d^2)$.
  2. $\mathcal{P}_{\mathcal{X}} = \{\mathcal{N}(0, I_d)\}$. Now learning over all rotated copies of the joint is learning all $h^* \circ \mathcal{G}_{\mathcal{X}}$ on the gaussian. Now a bound on VCdim will not suffice and we need to bound the sample complexity in another way. We are not in the standard PAC learning setting. We are in Benedek-Itai's world!
     a In this setting, the lower bound on the sample complexity holds even for the algorithm that knows $\mathcal{P}_{\mathcal{X}}$ (but this algorithm still has to learn all concepts $h^* \circ \mathcal{G}_{\mathcal{X}}$).

Behind the proofs

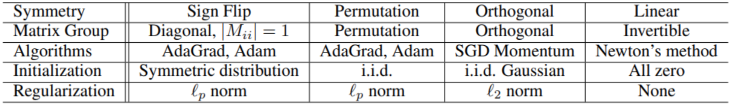
- We can consider two important learning settings where the set $\mathcal{P}_{\mathcal{X}}$ satisfies $\mathcal{P}_{\mathcal{X}} \circ \mathcal{G}_{\mathcal{X}} = \mathcal{P}_{\mathcal{X}}$.
    1. All distributions on $\mathcal{X}$. Then if $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$ it learns not only h^* over all distributions but all $h^* \circ \mathcal{G}_{\mathcal{X}}$ over all distributions. But this is the standard PAC setting and learning is at least as hard for $\mathcal{A}$ as the best algorithm on $h^* \circ \mathcal{G}_{\mathcal{X}}$. Show: VCdim$(h^* \circ \mathcal{G}_{\mathcal{X}}) = \Omega(d^2)$.
    2. $\mathcal{P}_{\mathcal{X}} = \{\mathcal{N}(0, I_d)\}$. Now learning over all rotated copies of the joint is learning all $h^* \circ \mathcal{G}_{\mathcal{X}}$ on the gaussian. Now a bound on VCdim will not suffice and we need to bound the sample complexity in another way. We are not in the standard PAC learning setting. We are in Benedek-Itai's world!
        - a In this setting, the lower bound on the sample complexity holds even for the algorithm that knows $\mathcal{P}_{\mathcal{X}}$ (but this algorithm still has to learn all concepts $h^* \circ \mathcal{G}_{\mathcal{X}}$).
        - b Use Benedek-Itai bound: $\mathcal{N}(\mathcal{A}, \mathcal{P}_{\mathcal{X}}, \epsilon, \delta) \geq \log[(1-\delta)D(\mathcal{H}, \rho_{\mathcal{X}}, 2\epsilon)]$. Show that $D(\mathcal{H}, \rho_{\mathcal{X}}, \epsilon_0) = \Omega(d^2)$
        - c Why is the second case stronger conceptually that the first ? Because it shows that ALL orthogonal equivariant algorithms fail in the same single problem (joint distribution).

Remarks

- The generalization of equivariant algorithms was first studied by [Ng 2004]. The lower bounds there are similar to the first case (but weaker) and cover only the first setting.

Remarks

- The generalization of equivariant algorithms was first studied by [Ng 2004]. The lower bounds there are similar to the first case (but weaker) and cover only the first setting.
- This paper also extends the lower bounds to permutation equivariant algorithms (FCNets+ Adam/Adagrad) as well as l_2-regression.



Symmetry	Sign Flip	Permutation	Orthogonal	Linear
Matrix Group	Diagonal, $\lvert M_{ii}\rvert = 1$	Permutation	Orthogonal	Invertible
Algorithms	AdaGrad, Adam	AdaGrad, Adam	SGD Momentum	Newton's method
Initialization	Symmetric distribution	i.i.d.	i.i.d. Gaussian	All zero
Regularization	ℓ_p norm	ℓ_p norm	ℓ_2 norm	None

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

1. Single function, All Distributions

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

As we discussed, this setup reduces to standard PAC-learning over the augmented class $\{h^* \circ g | g \in \mathcal{O}(d)\}$. Lower bounding the VC-dimension of this class by $\Omega(d^2)$ and using Blumer's lower bound results in:

Theorem (Single function, All Distributions)

Let $\mathcal{P} = \{\textit{all distributions}\} \diamond \{h^*\}$. *Then:*

1. $\mathcal{N}(\mathcal{A}, \mathcal{P}, \epsilon, \delta) = \Omega((d^2 + \ln \delta)/\epsilon)$, *for all* $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$.
2. *There is a 2-layer CNN architecture s.t.* $\mathcal{N}(ERM_{CNN}, \mathcal{P}, \epsilon, \delta) = O(\frac{1}{\epsilon}\left(\log \frac{1}{\epsilon} + \log \frac{1}{\delta}\right))$

This lower bound does not show a separation *for a single task*. While some $\mathcal{P}_{\mathcal{X}}^1 \diamond h^*$ might be hard for some network $FCNN_1$, it might be easy for another. Next result shows that there is a single task that is hard for all FCNNs.

2. Single Distribution

Theorem (Single function, Single Distribution)

Let $\mathcal{P} = \{N(0, I_{2d})\} \diamond h^*$. *There is a constant* $\epsilon_0 > 0$ *(indep. of* d*) s.t.:*

1. $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon_0) = \Omega(d^2)$, *for all* $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$.
2. *There is a 2-layer CNN architecture s.t.* $\mathcal{N}(ERM_{CNN}, \mathcal{P}, \epsilon, \delta) = O(\frac{1}{\epsilon}\left(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right))$.
3. *Moreover,* ERM_{CNN} *could be realized by gradient descent (on the second layer only).*

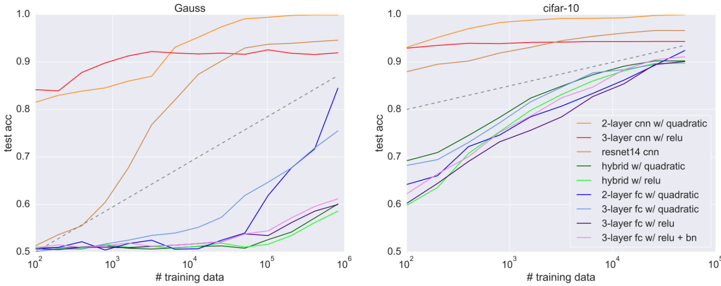
Theorem (Multiple functions, Single Distribution)

Let $\mathcal{P} = \{N(0, I_d)\} \diamond \{\text{sign}\left[\sum_{i=1}^{d} a_i x_i^2\right] | a_i \in \mathbb{R}\}$.

1. $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon) = \Omega(d^2/\epsilon)$, *for all* $\mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}$.
2. *There is a 2-layer CNN architecture s.t.* $\mathcal{N}(ERM_{CNN}, \mathcal{P}, \epsilon, \delta) = O(\frac{1}{\epsilon}\left(d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right))$.
3. *Moreover,* ERM_{CNN} *could be realized by gradient descent (on the second layer only).*

Single Distribution

- The task is to find whether the Red or the Green channel in CIFAR10 images has larger l_2-norm.
- CIFAR10 has $50k$ images of size $32 \times 32 \times 3$. The theory predicts that any FCNN would need around $32^4 \approx 1M$ images to learn this function. (if the distribution is complex enough to be close to i.i.d. gaussian; which isn't)



Single Function, Single Distribution: Proof Sketch

Theorem (Single function, Single Distribution)

Let $\mathcal{P} = \{N(0, I_{2d})\} \diamond h^*$, *s.t.* $h^*(x) = \mathrm{sign}[\sum_{i=1}^{d} x_i^2 - \sum_{i=d+1}^{2d} x_i^2]$.
There is a constant $\epsilon_0 > 0$ *(indep. of* d*) s.t.:*

$$\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon_0) = \Omega(d^2), \quad \textit{for all } \mathcal{A} \in \mathbb{A}_{\mathcal{G}_{\mathcal{X}}}.$$

Sketch of Proof.

1. We have $\mathcal{H}_o = \{h_U := \mathrm{sign}[x_{1:d}^T U x_{d+1:2d}] | U \in \mathcal{O}(d)\} \subseteq h^* \circ \mathcal{O}(2d)$
2. Use Benedek-Itai on $\mathcal{P} = N(0, I_{2d}) \diamond \mathcal{H}_o$. Then, $\mathcal{N}^*(\mathcal{A}, \mathcal{P}, \epsilon_0) \geq \log(1-\delta) + \log D(\mathcal{H}_o, \rho_{\mathcal{X}}, 2\epsilon)$
3. Show $\rho_{\mathcal{X}}(h_U, h_V) = \Omega(\frac{\|U-V\|_F}{\sqrt{d}})$. Larger pairwise distances imply bigger packing.
4. Pack just the elements of $\mathcal{SO}(d)$. Again, supersets imply bigger packing.
5. Groups are hard to handle. But (some) Lie groups can be described by their Lie algebra, which is a vector space.
6. Select a 1-1 set (or smaller) on the Lie algebra $\mathfrak{so}(2d)$ and show that for this set $\|\exp u - \exp v\|_F = \Omega(\|u - v\|_F)$.
7. Inverse Santalo's inequality to show $\geq (\frac{C}{\epsilon})^{\binom{d}{2}}$

Table of Contents

Conclusions and Discussion

- We can unify a lot of algorithms, with practical relevance, under the umbrella of algorithmic equivariance. The authors proved lower bound on their sample complexity.
- From the bounds we get that there are even very small CNNs (or tasks that a CNN can approximate) that no FCNN can find them efficiently from samples.

Conclusions and Discussion

- We can unify a lot of algorithms, with practical relevance, under the umbrella of algorithmic equivariance. The authors proved lower bound on their sample complexity.
- From the bounds we get that there are even very small CNNs (or tasks that a CNN can approximate) that no FCNN can find them efficiently from samples.
- The class of CNNs is kind of unnatural. Especially the strided pooling. Also, the distribution might be overly complex relative e.g. to image datasets.

Conclusions and Discussion

- We can unify a lot of algorithms, with practical relevance, under the umbrella of algorithmic equivariance. The authors proved lower bound on their sample complexity.
- From the bounds we get that there are even very small CNNs (or tasks that a CNN can approximate) that no FCNN can find them efficiently from samples.
- The class of CNNs is kind of unnatural. Especially the strided pooling. Also, the distribution might be overly complex relative e.g. to image datasets.
- The method does not unify CNNs. There might be other CNNs, with more practical relevance, that perform worse than some FCNNs.

Table of Contents

Bibliography

[BI91] Gyora M. Benedek and Alon Itai. "Learnability with respect to fixed distributions". In: *Theoretical Computer Science* 86.2 (1991), pp. 377–389. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(91)90026-X. URL: https://www.sciencedirect.com/science/article/pii/030439759190026X.

[Blu+89] Anselm Blumer et al. "Learnability and the Vapnik-Chervonenkis Dimension". In: *J. ACM* 36.4 (Oct. 1989), pp. 929–965. ISSN: 0004-5411. DOI: 10.1145/76359.76371. URL: https://doi.org/10.1145/76359.76371.

[Ng04] Andrew Ng. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning* (Sept. 2004). DOI: 10.1145/1015330.1015435.