Learning with Neural Networks:
Generalisation, Unseen Data and Boolean Measures

E. Cornacchia
École Polytechnique Fédérale de Lausanne (EPFL)

Based on joint works with E. Abbé\textsuperscript{1}, S. Bengio\textsuperscript{2}, J. Hzął\textsuperscript{1}, J. Kleinberg\textsuperscript{3}, A. Lotfi\textsuperscript{1}, C. Marquis\textsuperscript{1}, M. Raghu\textsuperscript{4}, C. Zhang\textsuperscript{4}

\textsuperscript{1}EPFL, \textsuperscript{2}Apple, \textsuperscript{3}Cornell University, \textsuperscript{4}Google Research

30th September 2022
Deep Learning

Neural networks are able to do image recognition pretty well, e.g. CIFAR, MNIST.

![Images of handwritten digits]

5

1

8
What if we make the problem more complex?

E.g. learn the sum-mod$_{10}$ of an array of MNIST digits.

\[
\begin{array}{c}
54890 \\
67128 \\
45237 \\
\end{array}
\]

\[
\begin{array}{c}
\text{sum-mod}_{10} \\
\rightarrow 6 \\
\rightarrow 4 \\
\rightarrow 1 \\
\end{array}
\]
What if we make the problem more complex?

**Pointer-Value-Retrieval (PVR)** [Zhang, Raghu, Kleinberg, Bengio, '21] [15].

E.g. with window size 2, and aggregation function $\text{sum-mod}_{10}$:

\[
\begin{array}{c}
\text{Pointer} \\
\text{Window} \\
\text{Aggregation function} \\
\text{Output}
\end{array}
\]

\[
20 + 95481 \rightarrow 4
\]
Boolean functions

The previous tasks can be modeled by Boolean functions acting on symbolic representations of the data.
Boolean functions

The previous tasks can be modeled by Boolean functions acting on symbolic representations of the data.

We will assume the ‘perception’ is given, and focusing on the ‘reasoning’, e.g. on functions such as $f: \{-1, 1\}^d \rightarrow \mathbb{R}$.
Boolean functions

The previous tasks can be modeled by Boolean functions acting on symbolic representations of the data.

\[ [1, -1, 1, -1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, 1, 1] \]

(perception)

\[ [1, -1, 1, -1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, 1, 1] \]

(reasoning)

We will assume the 'perception' is given, and focus on the 'reasoning', e.g. on Boolean functions:
Boolean functions

The previous tasks can be modeled by Boolean functions acting on symbolic representations of the data.

We will assume the ‘perception’ is given, and focus on the ‘reasoning’, e.g. on Boolean functions:

\[ f : \{\pm 1\}^d \rightarrow \mathbb{R} \]
Setup

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.
Setup

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.

- Observe samples $(x, f(x))$ where $x \in \{\pm 1\}^d, x \sim \mathcal{D}^{\text{train}}$.
Setup

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.
- Observe samples $(x, f(x))$ where $x \in \{\pm 1\}^d, x \sim D^{\text{train}}$
- Take neural network $\text{NN}(x; \theta)$ with parameters $\theta \in \mathbb{R}^E$
Setup

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.

- Observe samples $(x, f(x))$ where $x \in \{\pm 1\}^d$, $x \sim D_{\text{train}}$

- Take neural network $NN(x; \theta)$ with parameters $\theta \in \mathbb{R}^E$

- Use gradient descent methods (GD/SGD):

$$\begin{align*}
\theta^0 &\sim P_0 \\
\theta^{t+1} &= \theta^t - \gamma \frac{1}{b} \sum_{i=1}^{b} \nabla_{\theta^t} L(f(x_i), NN(x_i; \theta^t)) \quad t \in T
\end{align*}$$

$b$: batch size, $\gamma$: learning rate
Two settings

Generalization error:

$$\varepsilon_{\text{gen}} := \mathbb{E}_{x \sim D_{\text{test}}} \left[ L(f(x), \text{NN}(x; \theta^T)) \right]$$
Two settings

Generalization error:

$$\varepsilon_{\text{gen}} := \mathbb{E}_{x \sim D_{\text{test}}} \left[ L(f(x), \text{NN}(x; \theta^T)) \right]$$

We will look at the generalization error in the following two settings:

1. **Matched setting**: $D_{\text{train}} = D_{\text{test}}$
Two settings

Generalization error:

\[ \varepsilon_{\text{gen}} := \mathbb{E}_{x \sim D_{\text{test}}} \left[ L(f(x), \text{NN}(x; \theta^T)) \right] \]

We will look at the generalization error in the following two settings:

1. **Matched setting**: \( D_{\text{train}} = D_{\text{test}} \)

2. **Mismatched setting**: \( D_{\text{train}} \neq D_{\text{test}} \). Specifically, some samples are hidden during training.
This Talk

We will see that the following measures come into play in learning with GD on neural networks:

1. Matched setting:
   - Initial alignment between the neural network and the data
     [Abbe, C, Hązła, Marquis, ICML’22] [4]
   - Noise stability of data

2. Mismatched setting:
   - Boolean influence
Matched Setting

\[ D_{\text{train}} = D_{\text{test}} \]
Given a fully connected neural network with a certain iid initialization, can we understand if a target function is hard to learn?
Given a fully connected neural network with a certain iid initialization, can we understand if a target function is hard to learn?
Initial Alignment (INAL)

For a target function $f : \mathcal{X} \to \mathbb{R}$, input distribution $P_\mathcal{X}$ and a neural network $\text{NN}_\theta : \mathcal{X} \to \mathbb{R}$ randomly initialized with $\theta^0 \sim P_0$

$\text{INAL}(f, \text{NN}) := \max_{v \in \text{neurons}} \mathbb{E}_{\theta^0 \sim P_0} \mathbb{E}_{x \sim P_\mathcal{X}} [f(x) \cdot \text{NN}^v(x; \theta^0)]^2$,
Initial Alignment (INAL)

For a target function \( f : \mathcal{X} \to \mathbb{R} \), input distribution \( P_{\mathcal{X}} \) and a neural network \( \text{NN}_\theta : \mathcal{X} \to \mathbb{R} \) randomly initialized with \( \theta^0 \sim P_0 \)

\[
\text{INAL}(f, \text{NN}) := \max_{\nu \in \text{neurons}} \mathbb{E}_{\theta^0 \sim P_0} \mathbb{E}_{x \sim P_{\mathcal{X}}} [f(x) \cdot \text{NN}^{(\nu)}(x; \theta^0)]^2,
\]

**Question:** Does small INAL imply that GD cannot learn in a reasonable horizon?
Take $f: \{\pm 1\}^d \rightarrow \mathbb{R}$, $x \sim \text{Unif}\{\pm 1\}^d$. 
Experiments

Take $f : \{\pm 1\}^d \to \mathbb{R}$, $x \sim \text{Unif}\{\pm 1\}^d$.

**Figure:** INAL vs. time to escape initialization for Boolean tasks, on a 2-layer ReLU fully connected NN trained with SGD batch size 64, input size $d = 1000$. 
Setting

Data:
- \((x, f(x)),\) with \(x \sim \text{Unif}\{\pm 1\}^d\) and \(f : \{-1,1\}^d \rightarrow \mathbb{R}\)
- Assume \(\mathbb{E}_x[f(x)] = o_d(1)\) and \(\mathbb{E}_x[f(x)^2] = 1\)
Setting

Data:
- $(x, f(x))$, with $x \sim \text{Unif}\{\pm 1\}^d$ and $f : \{\pm 1\}^d \to \mathbb{R}$
- Assume $\mathbb{E}_x[f(x)] = o_d(1)$ and $\mathbb{E}_x[f(x)^2] = 1$

Architecture/algorithm:
- Fully connected neural network of poly(d) size, with iid gaussian initialization
  - $W_j^{(0)} \sim \mathcal{N}(0, 1/n_j)$ and ReLU activation
Setting

Data:
- \((x, f(x)), \text{ with } x \sim \text{Unif}\{\pm 1\}^d \text{ and } f : \{\pm 1\}^d \to \mathbb{R}\)
- Assume \(\mathbb{E}_x[f(x)] = o_d(1)\) and \(\mathbb{E}_x[f(x)^2] = 1\)

Architecture/algorithm:
- Fully connected neural network of poly(d) size, with iid gaussian initialization
  \(W_j^{(0)} \overset{iid}{\sim} \mathcal{N}(0, 1/n_j)\) and ReLU activation
- Noisy GD with full batch and gradient range \(A\) [AS20,AKMS21] [6, 5]

\[
\theta^t = \theta^{t-1} - \gamma \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} \left[ \nabla_{\theta^{t-1}} L(f(x), \text{NN}(x; \theta^{t-1})) \right]_A + Z^t,
\]

where \(Z^t \overset{iid}{\sim} \mathcal{N}(0, \tau^{-2})\) and \(L\) is any differentiable loss.
If INAL small, GD cannot learn

‘Extended’ function: \( \tilde{f}_{d^2}(x_1, \ldots, x_d, x_{d+1}, \ldots, x_{d^2}) = f_d(x_1, \ldots, x_d) \)
If INAL small, GD cannot learn

‘Extended’ function: \( \bar{f}_{d^2}(x_1, \ldots, x_d, x_{d+1}, \ldots, x_{d^2}) = f_d(x_1, \ldots, x_d) \)

**Theorem 1** ([Abbe,C,Hązła,Marquis,’22])

If INAL\((f_d, \text{NN}_d) = O(d^{-c})\), for \(c \geq 1\), then the noisy GD algorithm after \(T\) steps of training on a network of size \(E\), outputs \(\text{NN}_{d^2}(x; \theta^T)\) such that

\[
|E[\bar{f}_{d^2}(x) \cdot \text{NN}_{d^2}(x; \theta^T)]| = O \left( \frac{\gamma T \sqrt{EA}}{\tau} \cdot d^{-\frac{c-1}{8}} \right)
\]
If INAL small, GD cannot learn

‘Extended’ function: \( \bar{f}_d^2(x_1, \ldots, x_d, x_{d+1}, \ldots, x_{d^2}) = f_d(x_1, \ldots, x_d) \)

**Theorem 1** ([Abbe,C,Hzaza,Marquis,’22])

If \( \text{INAL}(f_d, \text{NN}_d) = O(d^{-c}) \), for \( c \geq 1 \), then the noisy GD algorithm after \( T \) steps of training on a network of size \( E \), outputs \( \text{NN}_{d^2}(x; \theta^T) \) such that

\[
E[\bar{f}_d^2(x) \cdot \text{NN}_{d^2}(x; \theta^T)] = O \left( \frac{\gamma T \sqrt{EA}}{\tau} \cdot d^{\frac{c-1}{8}} \right)
\]

- If INAL small, \( \bar{f}_d^2 \) is not weakly learnable on Gaussian networks, i.e. correlation better than guessing is not achievable.
If INAL small, GD cannot learn

‘Extended’ function: \( \tilde{f}_{d^2}(x_1, \ldots, x_d, x_{d+1}, \ldots, x_{d^2}) = f_d(x_1, \ldots, x_d) \)

**Theorem 1** ([Abbe, C, Hązła, Marquis, ’22])

If \( \text{INAL}(f_d, \text{NN}_d) = O(d^{-c}) \), for \( c \geq 1 \), then the noisy GD algorithm after \( T \) steps of training on a network of size \( E \), outputs \( \text{NN}_{d^2}(x; \theta^T) \) such that

\[
| \mathbb{E}[\tilde{f}_{d^2}(x) \cdot \text{NN}_{d^2}(x; \theta^T)] | = O\left( \frac{\gamma T \sqrt{EA}}{\tau} \cdot d^{-\frac{c-1}{8}} \right)
\]

- If INAL small, \( \tilde{f}_{d^2} \) is not weakly learnable on Gaussian networks, i.e. correlation better than guessing is not achievable.
- Hardness holds for *any* permutation-invariant initialization.
If INAL small, GD cannot learn

‘Extended’ function: $\tilde{f}_{d^2}(x_1, \ldots, x_d, x_{d+1}, \ldots, x_{d^2}) = f_d(x_1, \ldots, x_d)$

Theorem 1 ([Abbe, C, Hązła, Marquis, ’22])

If $\text{INAL}(f_d, \text{NN}_d) = O(d^{-c})$, for $c \geq 1$, then the noisy GD algorithm after $T$ steps of training on a network of size $E$, outputs $\text{NN}_{d^2}(x; \theta^T)$ such that

$$|E[\tilde{f}_{d^2}(x) \cdot \text{NN}_{d^2}(x; \theta^T)]| = O \left( \frac{\gamma T \sqrt{EA}}{\tau} \cdot d^{-\frac{c-1}{8}} \right)$$

- If INAL small, $\tilde{f}_{d^2}$ is not weakly learnable on Gaussian networks, i.e. correlation better than guessing is not achievable.
- Hardness holds for any permutation-invariant initialization
- We obtain hardness only for the ‘extension’ of $f_d$
**Theorem 1: Proof Outline**

**Fourier-Walsh transform:** $f: \{\pm 1\}^d \rightarrow \mathbb{R}$ can be expressed as

$$f(x) = \sum_{S \subseteq [d]} \hat{f}(S) \chi_S(x)$$

where $\chi_S(x) := \prod_{i \in S} x_i$ and $\hat{f}(S) := \mathbb{E}_{x \sim P_x}[f(x)\chi_S(x)]$. 
Theorem 1: Proof Outline

Fourier-Walsh transform: \( f : \{ \pm 1 \}^d \to \mathbb{R} \) can be expressed as

\[
f(x) = \sum_{S \subseteq [d]} \hat{f}(S) \chi_S(x)
\]

where \( \chi_S(x) := \prod_{i \in S} x_i \) and \( \hat{f}(S) := \mathbb{E}_{x \sim P_X} [f(x) \chi_S(x)] \).

Step 1: If \( \text{INAL}(f, \text{NN}) \) is small, \( f \) is high-degree.
Theorem 1: Proof Outline

**Fourier-Walsh transform:** $f : \{-1\}^d \to \mathbb{R}$ can be expressed as

$$f(x) = \sum_{S \subseteq [d]} \hat{f}(S) \chi_S(x)$$

where $\chi_S(x) := \prod_{i \in S} x_i$ and $\hat{f}(S) := \mathbb{E}_{x \sim P_X}[f(x) \chi_S(x)]$.

**Step 1:** If $\text{INAL}(f, \text{NN})$ is small, $f$ is high-degree.

- We look at neurons in the first layer:

$$\text{INAL}(f, \text{ReLU}) := \mathbb{E}_{w^0, b^0} \mathbb{E}_x[f(x) \text{ReLU}(w^0 x + b^0)]^2$$
Theorem 1: Proof Outline

Fourier-Walsh transform: \(f : \{\pm 1\}^d \to \mathbb{R}\) can be expressed as

\[
f(x) = \sum_{S \subseteq [d]} \hat{f}(S) \chi_S(x)
\]

where \(\chi_S(x) := \prod_{i \in S} x_i\) and \(\hat{f}(S) := \mathbb{E}_{x \sim P_x} [f(x) \chi_S(x)]\).

Step 1: If \(\text{INAL}(f, \text{NN})\) is small, \(f\) is high-degree.

- We look at neurons in the first layer:
  \[
  \text{INAL}(f, \text{ReLU}) := \mathbb{E}_{w^0, b^0} \mathbb{E}_x [f(x) \text{ReLU}(w^0 x + b^0)]^2
  \]
- \(\text{INAL}(f, \text{ReLU}) = \sum_{S \subseteq [d]} \hat{f}(S)^2 \text{INAL}(\chi_S, \text{ReLU})\)
Theorem 1: Proof Outline

**Fourier-Walsh transform:** \( f : \{\pm 1\}^d \rightarrow \mathbb{R} \) can be expressed as

\[
f(x) = \sum_{S \subseteq [d]} \hat{f}(S) \chi_S(x)
\]

where \( \chi_S(x) := \prod_{i \in S} x_i \) and \( \hat{f}(S) := \mathbb{E}_{x \sim P_x} [f(x) \chi_S(x)] \).

**Step 1:** If \( \text{INAL}(f, \text{NN}) \) is small, \( f \) is high-degree.

- We look at neurons in the first layer:

\[
\text{INAL}(f, \text{ReLU}) := \mathbb{E}_{w^0, b^0} \mathbb{E}_x [f(x) \text{ReLU}(w^0 x + b^0)]^2
\]

- \( \text{INAL}(f, \text{ReLU}) = \sum_{S \subseteq [d]} \hat{f}(S)^2 \text{INAL}(\chi_S, \text{ReLU}) \)

- For \( S \) such that \( |S| = k \), \( \text{INAL}(\chi_S, \text{ReLU}) = \Omega(d^{-(k+1)}) \)
Step 2: *High-degree functions are hard to learn for noisy GD on fully connected neural networks.*
Step 2: High-degree functions are hard to learn for noisy GD on fully connected neural networks.

- if $f$ is high-degree, the orbit of $\bar{f}$, i.e.

  $$ \text{orb}(\bar{f}) = \{ \bar{f} \circ \pi : \pi \text{ permutation on input space} \} $$

  is hard to learn.
Theorem 1: Proof Outline

Step 2: *High-degree functions are hard to learn for noisy GD on fully connected neural networks.*

- if \( f \) is high-degree, the orbit of \( \bar{f} \), i.e.

\[
\text{orb}(\bar{f}) = \{ \bar{f} \circ \pi : \pi \text{ permutation on input space} \}
\]

is hard to learn.

- In fact, \( \text{orb}(\bar{f}) \) is too “rich”, i.e. it has low Cross-Predictability:

\[
\text{CP}(\text{orb}(\bar{f})) = \mathbb{E}_{F,F' \sim \mathcal{U}_{\text{orb}(\bar{f})}} \mathbb{E}_{x \sim \mathcal{P}_X} [F(x)F'(x)]^2 \quad [\text{Abbe,Sandon,'20} \ [6]].
\]
Theorem 1: Proof Outline

Step 2: *High-degree functions are hard to learn for noisy GD on fully connected neural networks.*

- if $f$ is high-degree, the orbit of $\bar{f}$, i.e.
  \[
  \text{orb}(\bar{f}) = \{\bar{f} \circ \pi : \pi \text{ permutation on input space}\}
  \]
  is hard to learn.

- In fact, $\text{orb}(\bar{f})$ is too “rich”, i.e. it has low Cross-Predictability:
  \[
  \text{CP}(\text{orb}(\bar{f})) = \mathbb{E}_{F,F' \sim \text{U}_{\text{orb}(\bar{f})}} \mathbb{E}_{x \sim P_X} [F(x)F'(x)]^2 \quad [\text{Abbe, Sandon, '20}] \quad [6].
  \]
  Requires the input extension.
Theorem 1: Proof Outline

Step 2: High-degree functions are hard to learn for noisy GD on fully connected neural networks.

- if \( f \) is high-degree, the orbit of \( \bar{f} \), i.e.

\[
\text{orb}(\bar{f}) = \{ \bar{f} \circ \pi : \pi \text{ permutation on input space} \}
\]

is hard to learn.

- In fact, \( \text{orb}(\bar{f}) \) is too “rich”, i.e. it has low Cross-Predictability:

\[
\text{CP}(\text{orb}(\bar{f})) = \mathbb{E}_{F,F' \sim \text{U}_{\text{orb}(\bar{f})}} \mathbb{E}_{x \sim P_X} [F(x)F'(x)]^2 \quad \text{[Abbe, Sandon, ’20] [6].}
\]

- Requires the input extension.

- **Novelty:** express the CP condition only in terms of INAL.
Theorem 2 (Informal, in preparation)

Let $\text{NN}_d$ be a 2-layer neural net with iid gaussian init. and ReLU activation. If $\text{INAL}(f_d, \text{NN}_d) = d^{-\omega(1)}$, then after $T$ steps of training of noisy GD on hinge loss, if $\gamma T \sqrt{EA} \tau^{-1} = \text{poly}(d)$, then $|\mathbb{E}[f_d(x) \cdot \text{NN}_d(x; \theta^T)]| = d^{-\omega(1)}$. 

Theorem 2 does not rely on the orbit trick, and covers functions with degenerate orbit. It does not require the input extension. Both theorems rely on the same $\text{INAL}$. 
Removing the input extension

Theorem 2 (Informal, in preparation)
Let \( NN_d \) be a 2-layer neural net with iid gaussian init. and ReLU activation. If \( \text{INAL}(f_d, NN_d) = d^{-\omega(1)} \), then after \( T \) steps of training of noisy GD on hinge loss, if \( \frac{\gamma T \sqrt{EA}}{\tau} = \text{poly}(d) \), then \( |\mathbb{E}[f_d(x) \cdot NN_d(x; \theta^T)]| = d^{-\omega(1)} \).

- Theorem 2 does not rely on the orbit trick, and covers functions with degenerate orbit.
- It does not require the input extension.
Theorem 2 (Informal, in preparation)

Let $\text{NN}_d$ be a 2-layer neural net with iid gaussian init. and ReLU activation. If $\text{INAL}(f_d, \text{NN}_d) = d^{-\omega(1)}$, then after $T$ steps of training of noisy GD on hinge loss, if $\gamma T \sqrt{\text{EA}} = \text{poly}(d)$, then $|\mathbb{E}[f_d(x) \cdot \text{NN}_d(x; \theta^T)]| = d^{-\omega(1)}$.

- Theorem 2 does not rely on the orbit trick, and covers functions with degenerate orbit.
- It does not require the input extension.

Both theorems rely on the same INAL.
Let $f : \{\pm 1\}^d \to \mathbb{R}$ and $\delta \in [0, 1/2]$. Let $x \sim \text{Unif}\{\pm 1\}^d$ and let $y$ be formed from $x$ by flipping each coordinate independently with prob. $\delta$. The Noise Stability of $f$ is defined by [O’D14][11]:

$$\text{Stab}_\delta[f] := \mathbb{E}_{x,y} [f(x) \cdot f(y)].$$
Noise Stability

Let $f : \{\pm 1\}^d \rightarrow \mathbb{R}$ and $\delta \in [0, 1/2]$. Let $x \sim \text{Unif}\{\pm 1\}^d$ and let $y$ be formed from $x$ by flipping each coordinate independently with prob. $\delta$. The Noise Stability of $f$ is defined by [O'D14][11]:

$$\text{Stab}_\delta[f] := \mathbb{E}_{x,y} [f(x) \cdot f(y)].$$

Examples:

- (Parity) $\chi_d(x) = \prod_{i=1}^d x_i$: $\text{Stab}_\delta[\chi_d] = (1 - 2\delta)^d$
- (Majority) $\text{Maj}_d(x) = \text{sgn}(\sum_{i=1}^d x_i)$: $\text{Stab}_\delta[\text{Maj}_d] \sim \frac{2}{\pi} \arcsin(1 - 2\delta)$
Noise Stability

[Zhang, Raghu, Kleinberg, Bengio, ’21]: conjectured that Noise Stability is a measure for complexity of learning Boolean functions with neural networks.
Noise Stability

- [Zhang, Raghu, Kleinberg, Bengio, ’21]: conjectured that Noise Stability is a measure for complexity of learning Boolean functions with neural networks.

**Theorem 3 ([Abbe, Bengio, C, Kleinberg, Lotfi, Raghu, Zhang, ’22])**

The noisy GD algorithm after $T$ steps of training on any fully connected network of size $E$ and any permutation-invariant initialization, outputs a network $NN_{2d}(x; \theta^T)$ such that for $\delta \leq 1/4$

$$|E[\bar{f}_{2d}(x) \cdot NN_{2d}(x; \theta^T)]| \leq \frac{\gamma T \sqrt{EA}}{\tau} \cdot \text{Stab}_\delta[f]^{1/4}.$$
Noise Stability

- [Zhang, Raghu, Kleinberg, Bengio, '21]: conjectured that Noise Stability is a measure for complexity of learning Boolean functions with neural networks.

Theorem 3 ([Abbe, Bengio, C, Kleinberg, Lotfi, Raghu, Zhang, '22])

The noisy GD algorithm after $T$ steps of training on any fully connected network of size $E$ and any permutation-invariant initialization, outputs a network $\text{NN}_{2d}(x; \theta^T)$ such that for $\delta \leq 1/4$

$$|\mathbb{E}[\bar{f}_{2d}(x) \cdot \text{NN}_{2d}(x; \theta^T)]| \leq \frac{\gamma T \sqrt{EA}}{\tau} \cdot \text{Stab}_\delta[f]^{1/4}.$$  

- High-degree functions have low Noise Stability.
Summary: Matched Setting

For fully connected networks:

- A small Initial Alignment is harmful for learning with gradient descent.
- Datasets that are highly noise sensitive are hard to learn.
Summary: Matched Setting

For fully connected networks:
- A small Initial Alignment is harmful for learning with gradient descent.
- Datasets that are highly noise sensitive are hard to learn.

Future work:
- Positive result for INAL and Stab.
- Extension to other architectures, e.g. CNN.
  [Abbe,Boix,NeurIPS’22] [2]: lower bound for orbit classes under general group actions (beyond permutation).
- Extension to real inputs.
Mismatched Setting

\[ D_{\text{train}} \neq D_{\text{test}} \]
Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.
Canonical hold-out

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.

- Observe samples $(x, f(x))$ where $x \sim D_{\text{train}}$, where $D_{\text{train}}$ is such that for some $k \in [d]$:

$$
\begin{cases}
  x_i \overset{iid}{\sim} \text{Unif}\{\pm 1\} & i \neq k \\
  x_k \equiv 1
\end{cases}
$$

I.e., we “freeze” coordinate $k$ during training.
Canonical hold-out

- Unknown target function \( f : \{-1, 1\}^d \rightarrow \mathbb{R} \).

- Observe samples \((x, f(x))\) where \( x \sim \mathcal{D}_{\text{train}} \), where \( \mathcal{D}_{\text{train}} \) is such that for some \( k \in [d] \):

\[
\begin{cases}
  x_i \overset{iid}{\sim} \text{Unif}\{-1, 1\} & i \neq k \\
  x_k \equiv 1
\end{cases}
\]

  I.e., we “freeze” coordinate \( k \) during training.

- Train neural network with parameters \( \theta \in \mathbb{R}^p \) with gradient descent with squared loss.
Canonical hold-out

- Unknown target function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$.

- Observe samples $(x, f(x))$ where $x \sim \mathcal{D}^{\text{train}}$, where $\mathcal{D}^{\text{train}}$ is such that for some $k \in [d]$:

$$
\begin{align*}
  x_i &\overset{iid}{\sim} \text{Unif}\{\pm 1\} & i \neq k \\
  x_k &\equiv 1
\end{align*}
$$

  I.e., we “freeze” coordinate $k$ during training.

- Train neural network with parameters $\theta \in \mathbb{R}^p$ with gradient descent with squared loss.

- Generalization error:

$$
\varepsilon^{\text{gen}} := \frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} \left[ (f(x) - \text{NN}(x; \theta^T))^2 \right]
$$
Boolean influence

For $f : \{\pm 1\}^d \rightarrow \mathbb{R}$, $k \in [d]$, the Boolean influence is defined as [O’D14]:

$$\text{Inf}_k[f] = \sum_{S \subseteq [d] : k \in S} \hat{f}(S)^2,$$

where $\hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$, $\chi_S(x) = \prod_{i \in S} x_i$. 

Examples:
- (Parities) $\chi_S(x) = \prod_{i \in S} x_i$: $\text{Inf}_k(\chi_S) = 1$ if $k \in S$, $0$ otherwise.
- (Majority) $\text{Maj}_d(x) = \text{sgn}(\sum_{i = 1}^d x_i)$ (for $d$ odd): $\text{Inf}_k(\text{Maj}_d) = 2^{d-1} - \frac{d-1}{2^d}$ for all $k$. 


Boolean influence

For $f : \{\pm 1\}^d \rightarrow \mathbb{R}$, $k \in [d]$, the Boolean influence is defined as [O'D14]:

$$\text{Inf}_k[f] = \sum_{S \subseteq [d]: k \in S} \hat{f}(S)^2,$$

where $\hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$, $\chi_S(x) = \prod_{i \in S} x_i$.

For $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$, $k \in [d]$, $x \sim \text{Unif}\{\pm 1\}^d$

$$\text{Inf}_k(f) := \mathbb{P}_x(f(x) \neq f(x \odot (-1)^{e_k})).$$
Boolean influence

- For \( f : \{\pm 1\}^d \rightarrow \mathbb{R} \), \( k \in [d] \), the Boolean influence is defined as [O’D14]:

\[
\text{Inf}_k[f] = \sum_{S \subseteq [d]: k \in S} \hat{f}(S)^2,
\]

where \( \hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)] \), \( \chi_S(x) = \prod_{i \in S} x_i \).

- For \( f : \{\pm 1\}^d \rightarrow \{\pm 1\} \), \( k \in [d] \), \( x \sim \text{Unif}\{\pm 1\}^d \)

\[
\text{Inf}_k(f) := \mathbb{P}_x(f(x) \neq f(x \odot (-1)^{ek})).
\]

Examples:

- (Parities) \( \chi_S(x) = \prod_{i \in S} x_i \): \( \text{Inf}_k(\chi_S) = 1 \) if \( k \in S \), \( \text{Inf}_k = 0 \) otherwise

- (Majority) \( \text{Maj}_d(x) = \text{sgn}(\sum_{i=1}^{d} x_i) \) (d odd): \( \text{Inf}_k(\text{Maj}_d) = 2^{-(d-1)} \left( \frac{d-1}{d-1} \right) \forall k \)
Experiments

PVR function:

[ZRKB,'21]
Experiments

PVR function:
[ZRKB,’21]

Generalization error of Boolean tasks under canonical hold-out is well approximated by the Boolean influence, for Transformers, MLP and MLP-mixer.
Experiments

PVR tasks with varying window size. $x_6 = 1$ frozen during training.
Low-degree implicit bias

Let $x_{-k}$ be such that $(x_{-k})_k = 1$ and $(x_{-k})_i = x_i \forall i \neq k$.

Define $f_{-k}(x) = f(x_{-k})$ (‘frozen’ function).
Low-degree implicit bias

Let $x_{-k}$ be such that $(x_{-k})_k = 1$ and $(x_{-k})_i = x_i \ \forall i \neq k$.

Define $f_{-k}(x) = f(x_{-k})$ (‘frozen’ function).

**Lemma 4**

$$\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} (f(x) - f_{-k}(x))^2 = \text{Inf}_k(x)$$
Low-degree implicit bias

Let $x_{-k}$ be such that $(x_{-k})_k = 1$ and $(x_{-k})_i = x_i \ \forall i \neq k$.

Define $f_{-k}(x) = f(x_{-k})$ (‘frozen’ function).

**Lemma 4**

$$\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d}(f(x) - f_{-k}(x))^2 = \text{Inf}_k(x)$$

**Example:** $f(x) = x_2x_3 + x_3x_4 - x_1x_2x_3$, assume $k = 2$. 
Low-degree implicit bias

Let \( x_{-k} \) be such that \( (x_{-k})_k = 1 \) and \( (x_{-k})_i = x_i \) \( \forall i \neq k \).

Define \( f_{-k}(x) = f(x_{-k}) \) (‘frozen’ function).

**Lemma 4**

\[
\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} (f(x) - f_{-k}(x))^2 = \text{Inf}_k(x)
\]

**Example:** \( f(x) = x_2 x_3 + x_3 x_4 - x_1 x_2 x_3 \), assume \( k = 2 \).

- \( f_{-2}(x) = x_3 + x_3 x_4 - x_1 x_3 \)
Low-degree implicit bias

Let $x_{-k}$ be such that $(x_{-k})_k = 1$ and $(x_{-k})_i = x_i \ \forall i \neq k$.

Define $f_{-k}(x) = f(x_{-k})$ (‘frozen’ function).

**Lemma 4**

$$\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} (f(x) - f_{-k}(x))^2 = \text{Inf}_k(x)$$

**Example:** $f(x) = x_2x_3 + x_3x_4 - x_1x_2x_3$, assume $k = 2$.

- $f_{-2}(x) = x_3 + x_3x_4 - x_1x_3$
- $g(x) = x_2x_3 + x_3x_4 - x_1x_3$
- $h(x) = x_3 + x_3x_4 - x_1x_2x_3$
Low-degree implicit bias

Let $x_{-k}$ be such that $(x_{-k})_k = 1$ and $(x_{-k})_i = x_i \ \forall i \neq k$.

Define $f_{-k}(x) = f(x_{-k})$ (‘frozen’ function).

**Lemma 4**

$$\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} (f(x) - f_{-k}(x))^2 = \lnf_k(x)$$

**Example:** $f(x) = x_2x_3 + x_3x_4 - x_1x_2x_3$, assume $k = 2$.

- $f_{-2}(x) = x_3 + x_3x_4 - x_1x_3$
- $g(x) = x_2x_3 + x_3x_4 - x_1x_3$ \[\rightarrow\] $f$, $f_{-2}$, $g$, $h$ are indistinguishable during training.
- $h(x) = x_3 + x_3x_4 - x_1x_2x_3$
Low-degree implicit bias

Let \( x_{-k} \) be such that \((x_{-k})_k = 1 \) and \((x_{-k})_i = x_i \; \forall i \neq k \).

Define \( f_{-k}(x) = f(x_{-k}) \) (‘frozen’ function).

**Lemma 4**

\[
\frac{1}{2} \mathbb{E}_{x \sim \text{Unif}\{\pm 1\}^d} (f(x) - f_{-k}(x))^2 = \text{Inf}_k(x)
\]

**Example:** \( f(x) = x_2 x_3 + x_3 x_4 - x_1 x_2 x_3 \), assume \( k = 2 \).

- \( f_{-2}(x) = x_3 + x_3 x_4 - x_1 x_3 \)
- \( g(x) = x_2 x_3 + x_3 x_4 - x_1 x_3 \) \( \longrightarrow \) \( f, f_{-2}, g, h \) are indistinguishable during training.
- \( h(x) = x_3 + x_3 x_4 - x_1 x_2 x_3 \)

The network has a preference for \( f_{-2} \) \( \implies \) Low-degree bias

[Neyshabur et al.,'14],[Neyshabur et al,'17],[Soudry et al,'17],[Lyu,Li,'19],[Xi et al,'19],[Rahaman et al,'19]
Low-degree implicit bias

$f$: PVR-task with window size 3 and majority as aggregation. $x_6 = 1$ frozen during training.

The coefficients of the selected monomials in $f$ are:

\[\hat{f}(\{6\}) = 0.188, \quad \hat{f}(\{3, 6, 7, 8\}) = 0.063, \quad \hat{f}(\{6, 7, 8\}) = -0.063, \quad \hat{f}(\{1, 6\}) = -0.188.\]
Low-degree implicit bias

\( f \): PVR-task with window size 3 and majority as aggregation. 
\( x_6 = 1 \) frozen during training.

The coefficients of the selected monomials in \( f \) are:

\[
\hat{f}({1}) = 0.188, \quad \hat{f}({3, 6, 7, 8}) = 0.063, \quad \hat{f}({6, 7, 8}) = -0.063, \quad \hat{f}({1, 6}) = -0.188.
\]
Linear models

\[ f(x) = \hat{f}(\emptyset) + \sum_{i=1}^{d} \hat{f}(\{i\})x_i, \] train under canonical hold-out, where the \( k^{th} \) component is frozen at training.

**Theorem 5** (Very informal, [Abbe, Bengio, C, Kleinberg, Lotfi, Raghu, Zhang, '22])

For

- linear regression models
- diagonal linear networks with bias, with small enough initialization scale

\[ \varepsilon_{\text{gen}} \approx \text{Inf}_k(f) \]
Summary: Mismatched Setting

For the architectures considered,

- experiments suggest that the Boolean influence is a good measure of the generalization error.
- This is due to the implicit bias towards low-degree representations of the data.
Summary: Mismatched Setting

For the architectures considered,

- experiments suggest that the Boolean influence is a good measure of the generalization error.
- This is due to the implicit bias towards low-degree representations of the data.

Future work:
- Proof beyond the linear case
Summary: Mismatched Setting

For the architectures considered,

- experiments suggest that the Boolean influence is a good measure of the generalization error.
- This is due to the implicit bias towards low-degree representations of the data.

Future work:

- Proof beyond the linear case
- Can we get lower generalization error than the Boolean influence?
Summary: Mismatched Setting

For the architectures considered,

- experiments suggest that the Boolean influence is a good measure of the generalization error.
- This is due to the implicit bias towards low-degree representations of the data.

Future work:

- Proof beyond the linear case
- Can we get lower generalization error than the Boolean influence?
- Can we understand the interplay between reasoning and perception?
Summary: Mismatched Setting

For the architectures considered,

- experiments suggest that the Boolean influence is a good measure of the generalization error.
- This is due to the implicit bias towards low-degree representations of the data.

Future work:

- Proof beyond the linear case
- Can we get lower generalization error than the Boolean influence?
- Can we understand the interplay between reasoning and perception?

Thank you.
Learning to reason with neural networks: Generalization, unseen data and boolean measures.  

Emmanuel Abbe and Enric Boix-Adsera. 
On the non-universality of deep learning: quantifying the cost of symmetry.  

Emmanuel Abbe, Enric Boix-Adsera, and Theodor Misiakiewicz. 

Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Christopher Marquis. 
An initial alignment between neural network and target is needed for gradient descent to learn. 


Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks, 2019.


Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro.
The implicit bias of gradient descent on separable data, 2017.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma.

Chiyuan Zhang, Maithra Raghu, Jon M. Kleinberg, and Samy Bengio.
Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization.
ArXiv, abs/2107.12580, 2021.
**Figure:** INAL vs. generalization error for binary classification in the CIFAR100 dataset with a 1-layer CNN trained with SGD batch size 64.
Step 1: If $\text{INAL}(f, \text{NN})$ is small, $f$ is high-degree

Recall: $\text{INAL}(f, \text{NN}) = \max_{\nu \in \text{neurons}} \mathbb{E}_{\theta_0} \mathbb{E}_x[f(x) \text{NN}_{\theta_0}^{(\nu)}(x)]^2$

We look at neurons in the first layer:

$$\underbrace{\mathbb{E}_{w_0} \mathbb{E}_x[f(x) \text{ReLU}(w_0^T x)]^2}_{\text{INAL}(f, \text{ReLU})} \begin{array}{c} \text{small} \implies \text{f high degree} \end{array}$$

Fourier-Walsh expansion of $f$: $f(x) = \sum_{S \in [n]} \hat{f}(S) \chi_S(x)$

**Lemma 6**

$\text{INAL}(f, \text{ReLU}) = \sum_{S \in [n]} \hat{f}(S)^2 \text{INAL}(\chi_S, \text{ReLU}), \quad \chi_S(x) = \prod_{i \in S} x_i$

**Lemma 7 (Key Lemma)**

*For $S$ such that $|S| = k$, $\text{INAL}(\chi_S, \text{ReLU}) = \Omega(n^{-(k+1)})$*