PAC Analysis of Deep Learning Algorithms (?)
Deep Learning

PAC Learning

PAC Analysis of Deep Learning Algorithms
  Positive Results
  Negative Results
Want to learn $h^* : \{\pm 1\}^n \rightarrow \{\pm 1\}$.

Data Source: Examples $(x, h^*(x)) \ x \sim \mathcal{D}$

$\mathcal{D}$ and $h^*$ are unknown
Want to learn $h^* : \{\pm 1\}^n \rightarrow \{\pm 1\}$.

Data Source: Examples $(x, h^*(x))$ $x \sim \mathcal{D}$

$\mathcal{D}$ and $h^*$ are unknown

Goal: Find $h : \{\pm 1\}^n \rightarrow \{\pm 1\}$ with small 0-1-loss

$$L_0^{-1}(h) = \Pr_{x \sim \mathcal{D}}(h(x) \neq h^*(x))$$
Neural Networks Learning

- A network $\mathcal{N}$ is a DAG with
  - $n$ inputs, 1 output
  - Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \rightarrow \mathbb{R}$
Neural Networks Learning

- A network $\mathcal{N}$ is a DAG with
  - $n$ inputs, 1 output
  - Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \to \mathbb{R}$
- For weights $w = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N},w} : \{\pm 1\}^n \to \mathbb{R}$
  (in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
A network $\mathcal{N}$ is a DAG with
- $n$ inputs, 1 output
- Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \to \mathbb{R}$

For weights $\mathbf{w} = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N}, \mathbf{w}} : \{\pm 1\}^n \to \mathbb{R}$

(in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
Neural Networks Learning

- A network $\mathcal{N}$ is a DAG with
  - $n$ inputs, 1 output
  - Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \rightarrow \mathbb{R}$
- For weights $w = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N}, w} : \{\pm 1\}^n \rightarrow \mathbb{R}$
  (in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
Neural Networks Learning

- A network $\mathcal{N}$ is a DAG with
  - $n$ inputs, 1 output
  - Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \rightarrow \mathbb{R}$
- For weights $w = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N},w} : \{\pm 1\}^n \rightarrow \mathbb{R}$
  (in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
A network $\mathcal{N}$ is a DAG with
- $n$ inputs, 1 output
- Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \to \mathbb{R}$
- For weights $w = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N},w} : \{\pm 1\}^n \to \mathbb{R}$
  (in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
A network $\mathcal{N}$ is a DAG with
- $n$ inputs, 1 output
- Each non-input node $v$ labelled by activation $\sigma_v : \mathbb{R} \to \mathbb{R}$

For weights $\mathbf{w} = \{w_{uv}\}_{uv \in E}$, net computes $h_{\mathcal{N}, \mathbf{w}} : \{\pm 1\}^n \to \mathbb{R}$

(in the following example $\sigma(x) = \text{ReLU}(x) = \max(0, x)$)
Neural Networks Learning

- Sample weights $w_{uv} \sim \mathcal{N}(0, \frac{1}{\text{deg}^+(v)})$
Neural Networks Learning

- Sample weights $w_{uv} \sim \mathcal{N}\left(0, \frac{1}{\text{deg}^+(v)}\right)$
- Define $L_D(w) = \mathbb{E}_{x \sim D} \ell(h^*(x)h_w(x))$ for $\ell(z) = \ln(1 + e^{-z})$
Neural Networks Learning

- Sample weights $w_{uv} \sim \mathcal{N}\left(0, \frac{1}{\text{deg}^+(v)}\right)$
- Define $L_D(w) = \mathbb{E}_{x \sim D} \ell(h^*(x)h_w(x))$ for $\ell(z) = \ln(1 + e^{-z})$

Minimize $L_D(w)$ using SGD:

At step $t$, sample mini-batch $S = \{(x_i, y_i)\}_{i=1}^m$ and update

$$w_{t+1} = w_t - \eta \nabla L_S(w_t), \quad L_S(w) = \frac{1}{m} \sum_{i=1}^m \ell(y_i h_w(x_i))$$

(note that $\mathbb{E}_S \nabla L_S(w_t) = \nabla L_D(w_t)$)
This talk: *Fully Connected Networks*

Free Parameters: **Width** and **Depth**
Deep Learning

PAC Learning

PAC Analysis of Deep Learning Algorithms
  Positive Results
  Negative Results
How to analyze learning algorithms?

- Learning general \( h^* : \{\pm 1\}^n \rightarrow \{\pm 1\} \) takes \( \exp(\Omega(n)) \) time.
- Some \( h^* : \{\pm 1\}^n \rightarrow \{\pm 1\} \) might still be learnable in \( \text{poly}(n) \) time.
How to analyze learning algorithms?

- Learning general $h^* : \{-1\}^n \rightarrow \{-1\}$ takes $\exp(\Omega(n))$ time.
- Some $h^* : \{-1\}^n \rightarrow \{-1\}$ might still be learnable in $\text{poly}(n)$ time.
- PAC learning:
PAC learning (Valiant 84)

An algorithm $A$ learns the hypothesis class $\mathcal{H} \subset \{\pm1\}^n$ if:

- Its input is:
  - $\epsilon > 0$ and access to examples $(x, y) = (x, h^*(x)) \sim D$ where $h^* \in \mathcal{H}$.
  - $D$ and $h^*$ are unknown.

- Its output is:
  - $h : \{\pm1\}^n \rightarrow \{\pm1\}$ with $L_0(D)(h) < \epsilon$.
  - The algorithm may return $h / \in \mathcal{H}$.

- It runs in time $\text{poly}(n, 1/\epsilon)$. 

9/28
PAC learning (Valiant 84)

An algorithm $A$ learns the hypothesis class $\mathcal{H} \subset \{\pm 1\}^n$ if:

- Its input is:
  - $\epsilon > 0$ and access to examples $(x, y) = (x, h^*(x)) \sim D$ where $h^* \in \mathcal{H}$.
  - $D$ and $h^*$ are unknown.

- Its output is:
  - $h : \{\pm 1\}^n \to \{\pm 1\}$ with $L_D^0(h) < \epsilon$
  - The algorithm may return $h \notin \mathcal{H}$. 
PAC learning (Valiant 84)

An algorithm $A$ learns the hypothesis class $\mathcal{H} \subset \{-1\}^n$ if:

- Its **input** is:
  - $\epsilon > 0$ and access to examples $(x, y) = (x, h^*(x)) \sim \mathcal{D}$ where $h^* \in \mathcal{H}$.
  - $\mathcal{D}$ and $h^*$ are unknown.

- Its **output** is:
  - $h : \{-1\}^n \to \{-1\}$ with $L_D^{-1}(h) < \epsilon$
  - The algorithm may return $h \notin \mathcal{H}$.

- It runs in time $\text{poly}(n, \frac{1}{\epsilon})$. 
Halfspaces

Halfspaces = \{\text{sign} \circ h_{\mathbf{w}} \mid h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle \text{ for } \mathbf{w} \in \mathbb{R}^n\}
Linear Classes

For $\Psi : \{\pm 1\}^n \to \mathbb{R}^k$

$$\text{LINEAR}(\Psi) = \{\text{sign} \circ h_w \circ \Psi \mid h_w(x) = \langle w, x \rangle \text{ for } w \in \mathbb{R}^k\}$$

E.g., homogenous degree-2 polynomial threshold functions

$\Psi : \{\pm 1\}^n \to \{\pm 1\}^{n^2}$, $\Psi(x) = x \otimes x$
Linear Classes

For $\Psi : \{\pm 1\}^n \rightarrow \mathbb{R}^k$

$$\text{LINEAR}(\Psi) = \{\text{sign} \circ h_w \circ \Psi \mid h_w(x) = \langle w, x \rangle \text{ for } w \in \mathbb{R}^k\}$$

- E.g., homogenous degree-2 polynomial threshold functions

$$\Psi : \{\pm 1\}^n \rightarrow \{\pm 1\}^{n^2}, \quad \Psi(x) = x \otimes x$$
For $\Psi : \{\pm 1\}^n \to B^k$, $M > 0$

\[ M\text{ARGIN}(\Psi, M) = \{h = \text{sign} \circ h_w \circ \Psi : \|w\| \leq M\} \]

\[ \text{sign}(z) = \begin{cases} 
1 & z \geq 1 \\
\ast & -1 < z < 1 \\
-1 & z \leq -1
\end{cases} \]
Intersections of halfspaces

\[ \text{INTER}(q) = \{ h : \{ \pm 1 \}^n \to \{ \pm 1 \} \mid h^{-1}(1) \text{ is an intersection of } q(n) \text{ halfspaces} \} \]
CIRC(k) – Hypotheses realized by depth $k$ circuit of size $\text{poly}(n)$
DNFs

\[ f(x) = (\neg x_1 \land x_{19}) \lor (x_5 \lor x_4 \land x_{33} \land \neg x_{12}) \lor (x_{11} \land x_{19} \land \neg x_2) \]

DNF – Hypotheses realized by a DNF with \( \text{poly}(n) \) terms
Additional Problems

- Neural Networks
- Threshold Circuits
- Polynomial Threshold Functions
- Decision Trees
- Parities
- Juntas
- Automata
- Boolean Formulae
- ...
PAC theory in five words: *(Only)* linear classes are learnable
Deep Learning

PAC Learning

PAC Analysis of Deep Learning Algorithms

Positive Results

Negative Results
This talk: NN algorithms
- What function classes are learnable by deep learning algorithms?
▶ *This talk*: NN algorithms
  ▶ What function classes are learnable by deep learning algorithms?
▶ *Not in this talk*: NN function classes
  ▶ Expressive power (Ohad’s talk)
  ▶ Sample complexity (Sanjeev, Nati, and Tengyu talk’s)
  ▶ Computational complexity (It is hard)
Deep Learning

PAC Learning

PAC Analysis of Deep Learning Algorithms
  Positive Results
  Negative Results
\( \mathcal{P}_d \) – all functions \( h : \{\pm1\}^n \rightarrow \{\pm1\} \) for which there is \( p \) such that

\[ \forall x, \ h(x)p(x) \geq 1 \] for degree \( d \) polynomial \( p \)

\[ p \text{’s largest coefficient is } \leq n^d \]
\( \mathcal{P}_d \) – all functions \( h : \{\pm 1\}^n \to \{\pm 1\} \) for which there is \( p \) such that

- \( \forall x, \ h(x)p(x) \geq 1 \) for degree \( d \) polynomial \( p \)
- \( p \)'s largest coefficient is \( \leq n^d \)

**Theorem (APVZ 14, DFS 16,17)**

For width \( \geq \text{poly}(n, \frac{1}{\epsilon}) \) and depth \( \leq \log(n) \), SGD efficiently learns \( \mathcal{P}_d \)

**Corollary**

For width \( \geq \exp(n, \frac{1}{\epsilon}) \), SGD learns any function

Complements expressivity results from the 80's and 90's
$\mathcal{P}_d$ – all functions $h : \{\pm 1\}^n \rightarrow \{\pm 1\}$ for which there is $p$ such that

- $\forall x, h(x)p(x) \geq 1$ for degree $d$ polynomial $p$
- $p$’s largest coefficient is $\leq n^d$

**Theorem (APVZ 14, DFS 16,17)**

*For width $\geq \text{poly}\left(n, \frac{1}{\epsilon}\right)$ and depth $\leq \log(n)$, SGD efficiently learns $\mathcal{P}_d$*

**Corollary**

*For width $\geq \text{poly}\left(n, \frac{1}{\epsilon}\right)$ and depth $\leq \log(n)$, SGD efficiently learns: Conjunctions, Constant length DNFs, Constant width DNFs, . . .*
\( \mathcal{P}_d \) – all functions \( h : \{\pm 1\}^n \to \{\pm 1\} \) for which there is \( p \) such that

- \( \forall x, \ h(x)p(x) \geq 1 \) for degree \( d \) polynomial \( p \)
- \( p \)'s largest coefficient is \( \leq n^d \)

**Theorem (APVZ 14, DFS 16,17)**

For width \( \geq \text{poly} \left(n, \frac{1}{\epsilon} \right) \) and depth \( \leq \log(n) \), SGD efficiently learns \( \mathcal{P}_d \)

**Corollary**

For width \( \geq \text{poly} \left(n, \frac{1}{\epsilon} \right) \) and depth \( \leq \log(n) \), SGD efficiently learns:
Conjunctions, Constant length DNFs, Constant width DNFs, …

**Corollary**

For width \( \geq \exp \left(n, \frac{1}{\epsilon} \right) \), SGD learns any function

- Complements expressivity results from the 80’s and 90’s
Proof Strategy

- Let $h^* \in \mathcal{P}_d$ be the target function with corresponding polynomial $p$
Proof Strategy

- Let $h^* \in \mathcal{P}_d$ be the target function with corresponding polynomial $p$
- Define $\Psi_w : \{\pm 1\}^n \rightarrow \mathbb{R}^q$:
Proof Strategy

- Let \( h^* \in \mathcal{P}_d \) be the target function with corresponding polynomial \( p \)
- Define \( \Psi_w : \{\pm 1\}^n \rightarrow \mathbb{R}^q : \)

\[
\Psi_w(x) = \left< v, \frac{\Psi_w(x)}{\sqrt{r}} \right> : \|v\| \leq \text{poly}(n, \frac{1}{\epsilon})
\]

- Let \( \mathcal{H}_w := \{ x \mapsto \left< v, \frac{\Psi_w(x)}{\sqrt{r}} \right> : \|v\| \leq \text{poly}(n, \frac{1}{\epsilon}) \} \)
Proof Strategy

- Let $h^* \in \mathcal{P}_d$ be the target function with corresponding polynomial $p$
- Define $\Psi_w : \{-1,1\}^n \to \mathbb{R}^q$:

\[
\Psi_w(x) = \sqrt{r} \langle v, \frac{\Psi_w(x)}{\sqrt{r}} \rangle
\]

\[\|v\| \leq \text{poly}(n, \frac{1}{\epsilon})\]

- Let $\mathcal{H}_w := \{ x \mapsto \langle v, \frac{\Psi_w(x)}{\sqrt{r}} \rangle : \|v\| \leq \text{poly}(n, \frac{1}{\epsilon}) \}$
- W.h.p. over initial $w$, $\exists f \in \mathcal{H}_w$ s.t. $\mathbb{E}_{x \sim D}(f(x) - p(x))^2 \ll \epsilon$
Proof Strategy

- Let $h^* \in \mathcal{P}_d$ be the target function with corresponding polynomial $p$

- Define $\Psi_w : \{\pm 1\}^n \to \mathbb{R}^q$:

- Let $\mathcal{H}_w := \{x \mapsto \langle v, \frac{\Psi_w(x)}{\sqrt{r}} \rangle : \|v\| \leq \text{poly} \left( n, \frac{1}{\epsilon} \right) \}$

- W.h.p. over initial $w$, $\exists f \in \mathcal{H}_w$ s.t. $\mathbb{E}_{x \sim \mathcal{D}} (f(x) - p(x))^2 \ll \epsilon$

- $\Rightarrow$ SGD on the last layer's weights works (due to convexity)
Proof Strategy

- Let $h^* \in \mathcal{P}_d$ be the target function with corresponding polynomial $p$
- Define $\Psi_w : \{\pm 1\}^n \rightarrow \mathbb{R}^q$:

$$\Psi_w := \{ x \mapsto \left\langle v, \frac{\Psi_w(x)}{\sqrt{r}} \right\rangle : \|v\| \leq \text{poly} \left( n, \frac{1}{\epsilon} \right) \}$$

- W.h.p. over initial $w$, $\exists f \in \mathcal{H}_w$ s.t. $\mathbb{E}_{x \sim \mathcal{D}} (f(x) - p(x))^2 \ll \epsilon$
- $\Rightarrow$ SGD on the last layer's weights works (due to convexity)
- $\Psi_{w_t}$ changes slowly $\Rightarrow$ general SGD works
Why \( p \approx f \) for \( f \in \mathcal{H}_w \)?

- \( \mathcal{H}_w \) depends only on the kernel \( k_w(x, x') = \langle \Psi_w(x), \Psi_w(x') \rangle \)

\[ q \]

For random weights \( w_i \sim N(0, \frac{1}{2}n) \), \( b_i \sim N(0, \frac{1}{2}) \),

\[ \langle w_i, x \rangle + b_i, \langle w_i, x' \rangle \] are \((\langle x, x' \rangle n + \frac{1}{2})\)-correlated standard Gaussians.

By Hermite expansion, \( k(x, x') = \sum_{k=0}^{\infty} a_k (\langle x, x' \rangle n)^k \) with \( a_k > 0 \).

By kernel theory, if \( k = k_w \), then \( p \in \mathcal{H}_w \).

\[ k \approx k_w \Rightarrow p \approx f \text{ for } f \in \mathcal{H}_w \]
Why $p \approx f$ for $f \in \mathcal{H}_w$?

$\mathcal{H}_w$ depends only on the kernel $k_w(x, x') = \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$

We have

\[
\frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} = \frac{1}{r} \sum_{i=1}^{q} \sigma \left( \langle w^i, x \rangle + b_i \right) \sigma \left( \langle w^i, x' \rangle + b_i \right)
\approx \mathbb{E}_{w,b} \sigma \left( \langle w, x \rangle + b \right) \sigma \left( \langle w, x' \rangle + b \right)
=: k(x, x')
\]

For random weights $w^i_j \sim \mathcal{N} \left( 0, \frac{1}{2n} \right)$, $b_i \sim \mathcal{N} \left( 0, \frac{1}{2} \right)$
Why $p \approx f$ for $f \in H_w$?

- $H_w$ depends only on the kernel $k_w(x, x') = \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$
- We have
  \[
  \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} = \frac{1}{r} \sum_{i=1}^{q} \sigma \left( \langle w_i, x \rangle + b_i \right) \sigma \left( \langle w_i, x' \rangle + b_i \right)
  \approx \mathbb{E}_{w,b} \sigma \left( \langle w, x \rangle + b \right) \sigma \left( \langle w, x' \rangle + b \right)
  =: \ k(x, x')
  \]

For random weights $w_i \sim \mathcal{N} \left( 0, \frac{1}{2n} \right)$, $b_i \sim \mathcal{N} \left( 0, \frac{1}{2} \right)$

- $\langle w_i, x \rangle + b_i, \langle w_i, x' \rangle + b_i$ are $\left( \frac{\langle x, x' \rangle}{2n} + \frac{1}{2} \right)$-correlated standard Gaussians
Why $p \approx f$ for $f \in \mathcal{H}_w$?

- $\mathcal{H}_w$ depends only on the kernel $k_w(x, x') = \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$

- We have

$$\frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} = \frac{1}{r} \sum_{i=1}^{q} \sigma \left( \langle w^i, x \rangle + b_i \right) \sigma \left( \langle w^i, x' \rangle + b_i \right)$$

$$\approx \mathbb{E}_{w,b} \sigma \left( \langle w, x \rangle + b \right) \sigma \left( \langle w, x' \rangle + b \right)$$

$$= : k(x, x')$$

For random weights $w^i_j \sim \mathcal{N} \left( 0, \frac{1}{2n} \right), \ b_i \sim \mathcal{N} \left( 0, \frac{1}{2} \right)$

- $\langle w^i, x \rangle + b_i, \langle w^i, x' \rangle + b_i$ are $\left( \frac{\langle x, x' \rangle}{2n} + \frac{1}{2} \right)$-correlated standard Gaussians

- By Hermite expansion, $k(x, x') = \sum_{k=0}^{\infty} a_k \left( \frac{\langle x, x' \rangle}{n} \right)^k$ with $a_k > 0$. 
Why $p \approx f$ for $f \in \mathcal{H}_w$?

- $\mathcal{H}_w$ depends only on the kernel $k_w(x, x') = \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$

- We have
  \[
  \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} = \frac{1}{r} \sum_{i=1}^{q} \sigma \left( \langle w^i, x \rangle + b_i \right) \sigma \left( \langle w^i, x' \rangle + b_i \right)
  \approx \mathbb{E}_{w, b, \sigma} \sigma (\langle w, x \rangle + b) \sigma (\langle w, x' \rangle + b)
  =: k(x, x')
  \]

For random weights $w^i_j \sim \mathcal{N} \left( 0, \frac{1}{2n} \right)$, $b_i \sim \mathcal{N} \left( 0, \frac{1}{2} \right)$

- $\langle w^i, x \rangle + b_i, \langle w^i, x' \rangle + b_i$ are \( \left( \frac{\langle x, x' \rangle}{2n} + \frac{1}{2} \right) \)-correlated standard Gaussians

- By Hermite expansion, $k(x, x') = \sum_{k=0}^{\infty} a_k \left( \frac{\langle x, x' \rangle}{n} \right)^k$ with $a_k > 0$.

- By kernel theory, if $k = k_w$, then $p \in \mathcal{H}_w$. 
Why $p \approx f$ for $f \in \mathcal{H}_w$?

- $\mathcal{H}_w$ depends only on the kernel $k_w(x, x') = \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$

- We have

$$\frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} = \frac{1}{r} \sum_{i=1}^{q} \sigma \left( \langle w^i, x \rangle + b_i \right) \sigma \left( \langle w^i, x' \rangle + b_i \right)$$

$$\approx \mathbb{E}_{w,b} \sigma \left( \langle w, x \rangle + b \right) \sigma \left( \langle w, x' \rangle + b \right)$$

$$= : k(x, x')$$

For random weights $w^i_j \sim \mathcal{N} \left( 0, \frac{1}{2n} \right)$, $b_i \sim \mathcal{N} \left( 0, \frac{1}{2} \right)$

- $\langle w^i, x \rangle + b_i, \langle w^i, x' \rangle + b_i$ are \(\left( \frac{\langle x, x' \rangle}{2n} + \frac{1}{2} \right)\)-correlated standard Gaussians

- By Hermite expansion, $k(x, x') = \sum_{k=0}^{\infty} a_k \left( \frac{\langle x, x' \rangle}{n} \right)^k$ with $a_k > 0$.

- By kernel theory, if $k = k_w$, then $p \in \mathcal{H}_w$.

- $k \approx k_w \Rightarrow p \approx f$ for $f \in \mathcal{H}_w$
Deep Learning

PAC Learning

PAC Analysis of Deep Learning Algorithms
  Positive Results
  Negative Results
\[ \| f \| ^2 = \sum_{A \in \mathbb{C}[n]} \binom{n}{|A|} |\hat{f}(A)|^2 ; \quad f(x) = \sum_{A \in \mathbb{C}[n]} \hat{f}(A)x^A \]
\[ \|f\|^2 = \sum_{A \in \mathbb{C}[n]} \binom{n}{|A|} |\hat{f}(A)|^2 \quad ; \quad f(x) = \sum_{A \in \mathbb{C}[n]} \hat{f}(A)x^A \]

\[ \mathcal{F}_M = \{ h : \forall x, \ h(x)f(x) \geq 1 \ \text{for} \ f \ \text{with} \ \|f\| \leq M \} \]
\[ \|f\|^2 = \sum_{A \in \mathcal{C}[n]} \left( \binom{n}{|A|} \right) |\hat{f}(A)|^2 \quad ; \quad f(x) = \sum_{A \in \mathcal{C}[n]} \hat{f}(A)x^A \]

\[ \mathcal{F}_M = \{ h : \forall x, h(x)f(x) \geq 1 \text{ for } f \text{ with } \|f\| \leq M \} \]

**Theorem**

*For width \( \geq \text{poly} \left( n, \frac{1}{\epsilon} \right) \) and depth \( \leq \log(n) \), SGD efficiently learns \( \mathcal{F}_{\text{poly}(n)} \)*

**Theorem (D, Feldman 18)**

*If SGD efficiently learns \( \mathcal{H} \), then \( \mathcal{H} \subset \mathcal{F}_{\text{poly}(n)} \)*
Proof Strategy – Kernes and Symmetry

- If SGD learns $h$, it also learns $h \circ T$ for any isometry $T$ on $\{\pm 1\}^n$
Proof Strategy – Kernes and Symmetry

- If SGD learns \( h \), it also learns \( h \circ T \) for any isometry \( T \) on \( \{\pm 1\}^n \)
- We can assume that \( \mathcal{H} \) is invariant to isometries.
Proof Strategy – Kernes and Symmetry

- If SGD learns $h$, it also learns $h \circ T$ for any isometry $T$ on $\{\pm 1\}^n$
- We can assume that $\mathcal{H}$ is invariant to isometries.
- Define $\Psi_w : \{\pm 1\}^n \rightarrow \mathbb{R}^q$:

\[
\Psi_w (x) = \begin{cases} 
1 & \text{if } x_1, \ldots, x_n \text{ are all even} \\
-1 & \text{otherwise}
\end{cases}
\]

- Let $k(x, x') = \mathbb{E}_w \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$
Proof Strategy – Kernes and Symmetry

- If SGD learns $h$, it also learns $h \circ T$ for any isometry $T$ on $\{\pm 1\}^n$.
- We can assume that $\mathcal{H}$ is invariant to isometries.
- Define $\Psi_w : \{\pm 1\}^n \rightarrow \mathbb{R}^q$.

Let $k(x, x') = \mathbb{E}_w \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q}$.

$\mathcal{H}$ has large margin w.r.t. $k$. 
Proof Strategy – Kernes and Symmetry

- If SGD learns $h$, it also learns $h \circ T$ for any isometry $T$ on $\{\pm 1\}^n$.
- We can assume that $\mathcal{H}$ is invariant to isometries.
- Define $\Psi_w : \{\pm 1\}^n \to \mathbb{R}^q$:

\[ k(x, x') = \mathbb{E}_w \frac{\langle \Psi_w(x), \Psi_w(x') \rangle}{q} \]

- $\mathcal{H}$ has large margin w.r.t. $k$.
- $k$ is invariant to isometries $\Rightarrow \mathcal{H} \subset \mathcal{F}_{\text{poly}(n)}$
Learnable by NN: Large Margin Linear Classes
Learnable by some algorithm but not by NN: Linear Classes
Hard to learnable: Non-Linear Classes (under assumptions / conjectured)
What’s captured by PAC analysis on NN? What’s missing?
What’s captured by PAC analysis on NN? What’s missing?

- Something 😊
What’s captured by PAC analysis on NN? What’s missing?

- Something 😊
- Structure 😊
  - Theory adapts nicely to network’s architecture
What’s captured by PAC analysis on NN? What’s missing?

- **Something 😊**
- **Structure 😊**
  - Theory adapts nicely to network’s architecture
- **Quantitative Bounds 😞**
What’s captured by PAC analysis on NN? What’s missing?

- Something 😊
- Structure 😊
  - Theory adapts nicely to network’s architecture
- Quantitative Bounds 😞
- Hierarchical Learning 😞
  - Inherent limitation of analysis based on hypothesis classes: SGD on NN PAC learns only linear threshold over fixed representation
What’s captured by PAC analysis on NN? What’s missing?

- Something 😊
- Structure 😊
  - Theory adapts nicely to network’s architecture
- Quantitative Bounds 😞
- Hierarchical Learning 😞
  - Inherent limitation of analysis based on hypothesis classes: SGD on NN PAC learns only linear threshold over fixed representation

Thank You!