Massachusetts Institute of Technology

On the hardness of learning under symmetries

presented by Thien Le

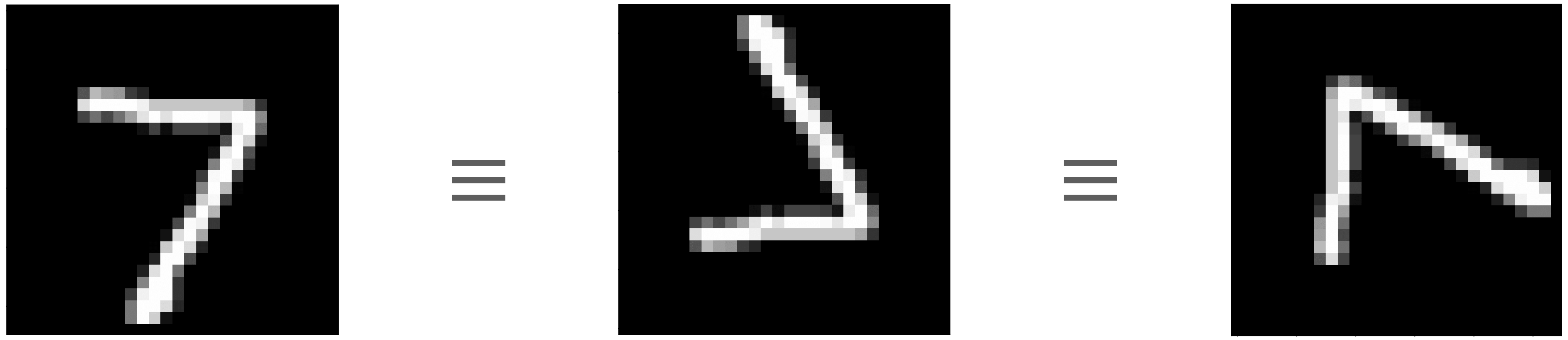
based on the ICLR 2024 paper of the same name
by Bobak T. Kiani*, L*, Hannah Lawrence*, Stefanie Jegelka, Melanie Weber




Input-domain symmetries

Machine learning tasks often specify **symmetries in the input space**

- Object detection in images



- Graphs

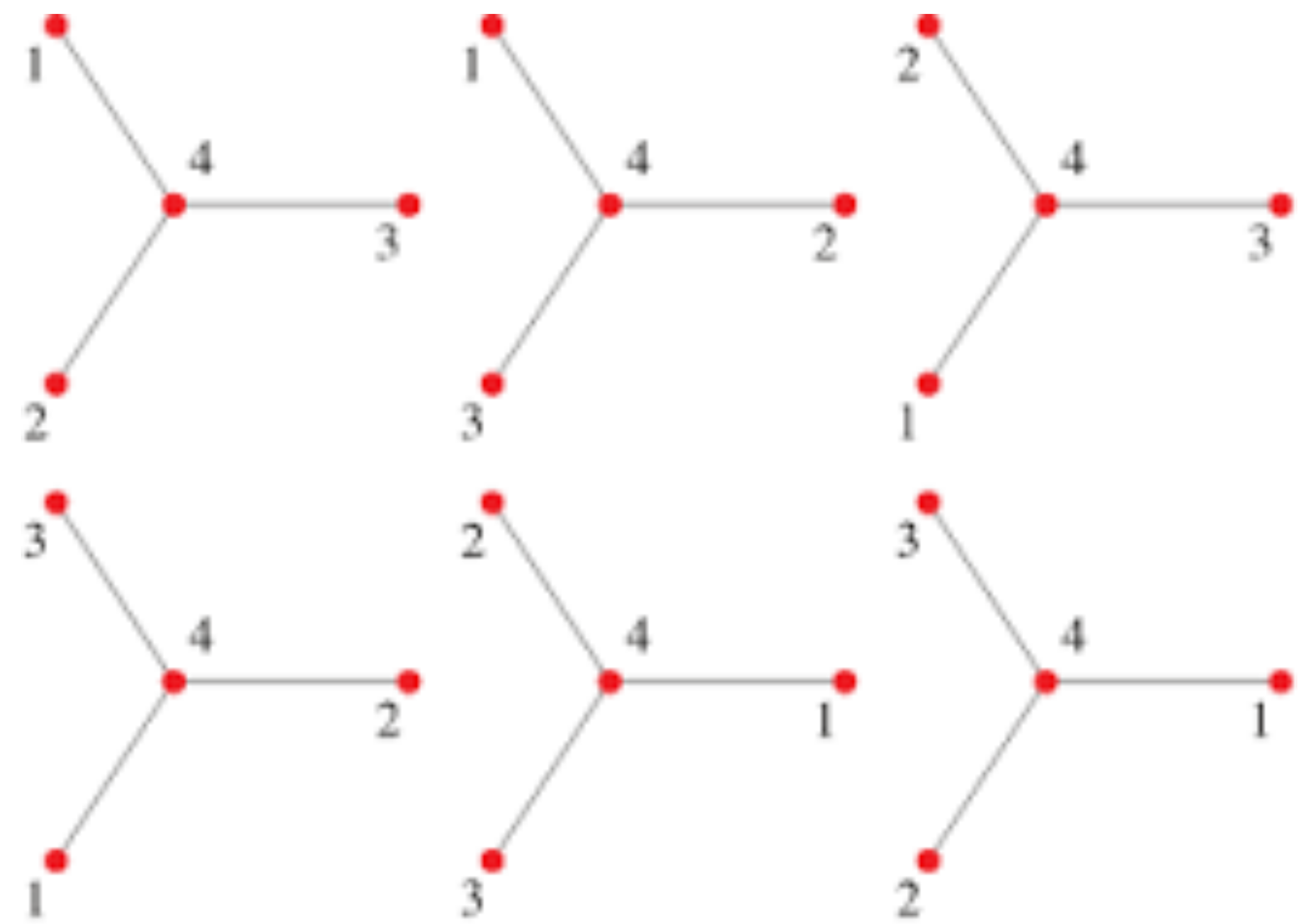
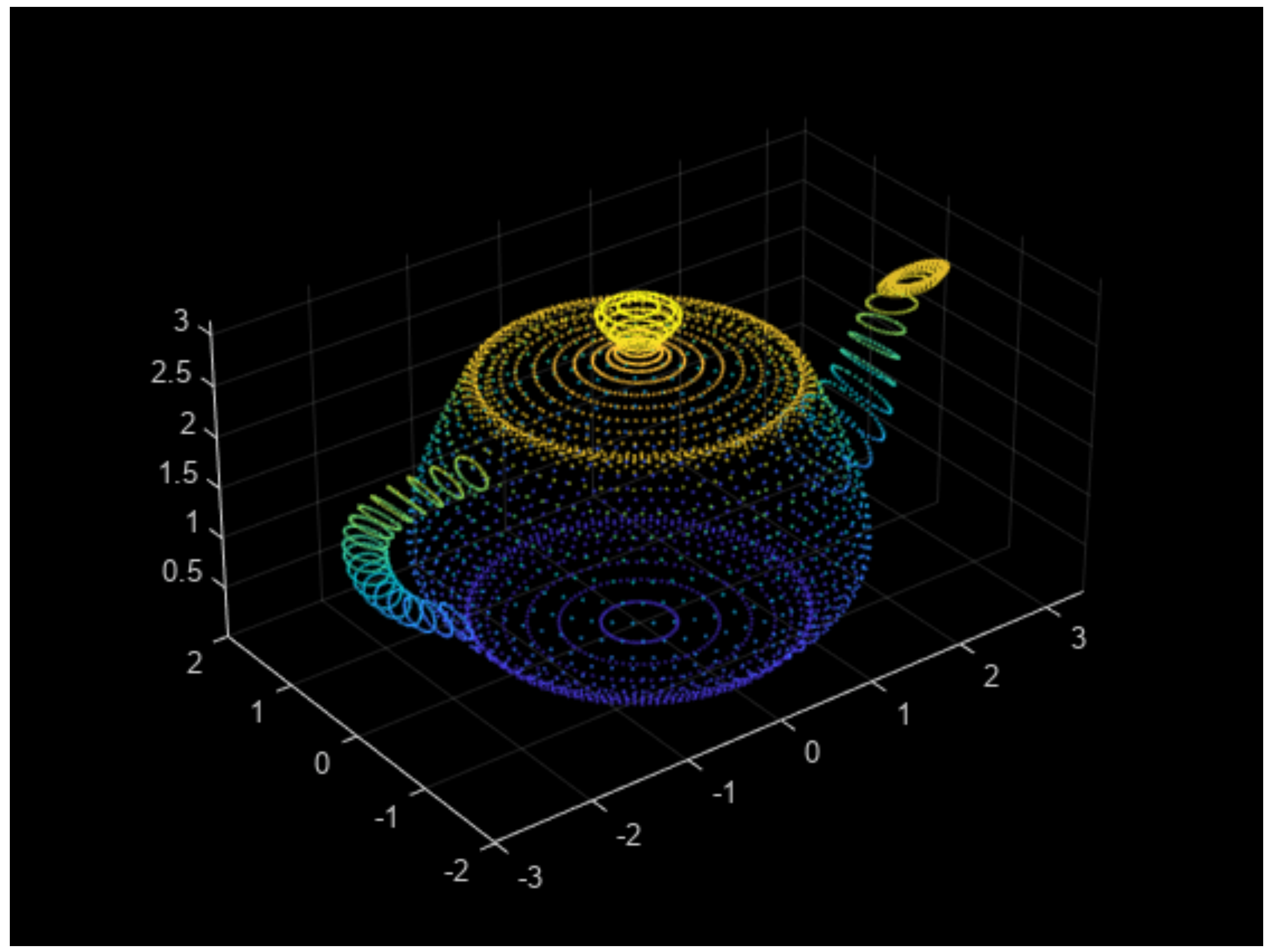


Figure from Wolfram MathWorld ‘Graph automorphism’

- Point clouds

Figure from MathWorks ‘pointCloud’ tutorial



Input-domain symmetries

In general, there is a smaller effective domain

$$\mathscr{X} \to \mathscr{X}/G$$

input to general-purpose functions:
- convenient representation
- compatible with “GPU”-learning

effective input domain:
- smaller, succinct representation
- incorporate known inductive bias

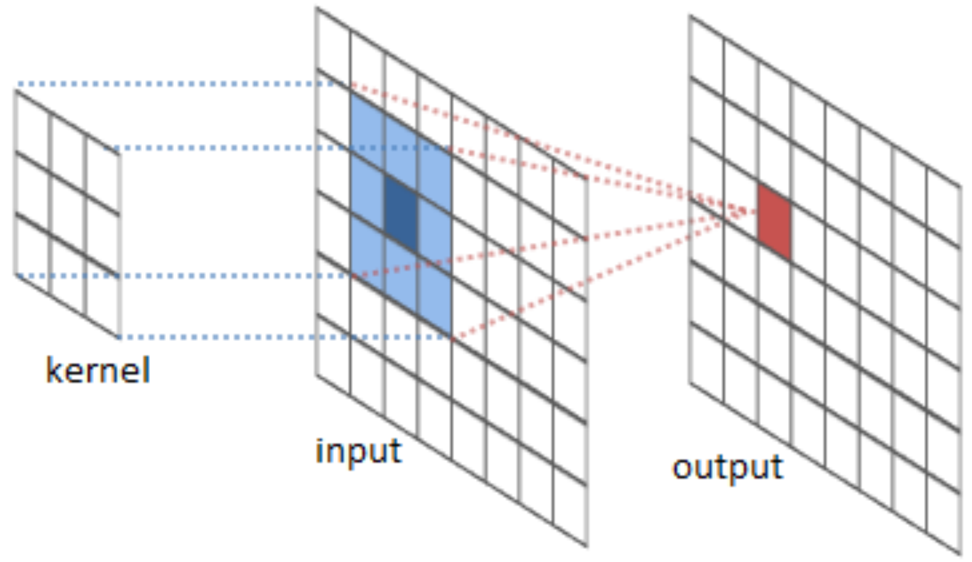
Examples

- pixel RGB values
- adjacency matrix/Laplacian of graphs
- coordinates in 3D space

- equivalence classes of rotated images
- graphs
- object in 3D space

Model symmetries

- Convolutional neural networks (CNN) + looped filter: translation-invariant



- (Invariant) graph neural network: node-permutation-invariant

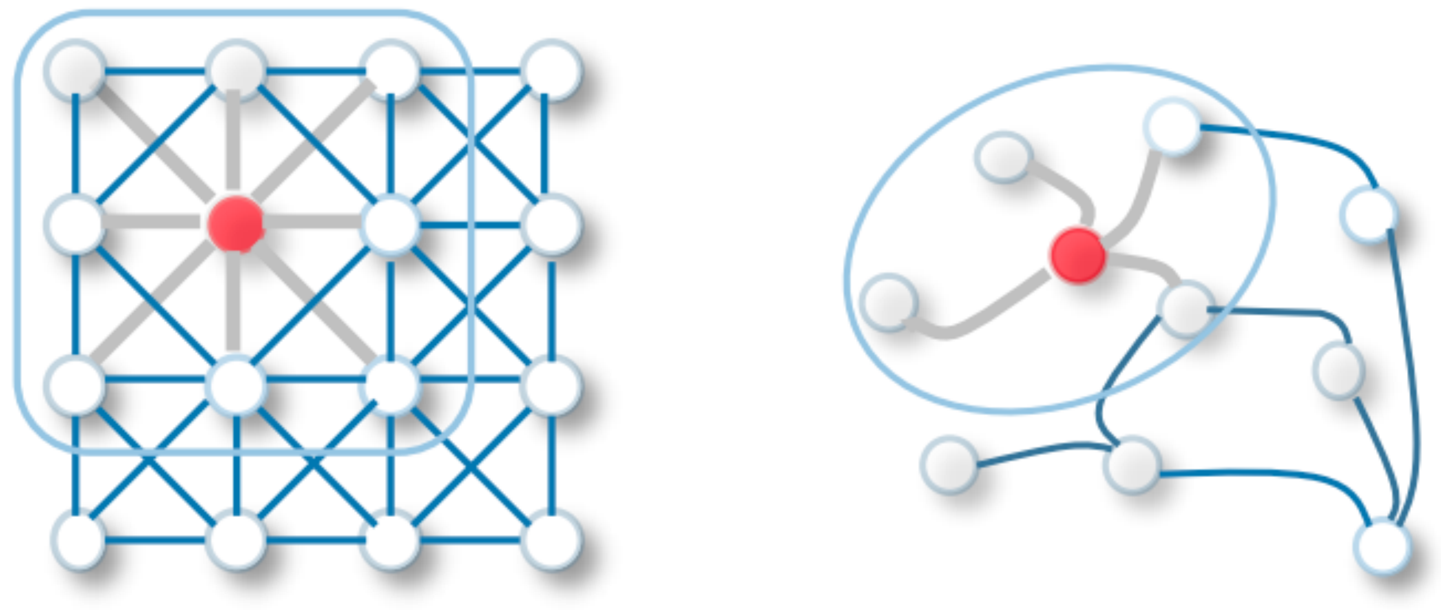
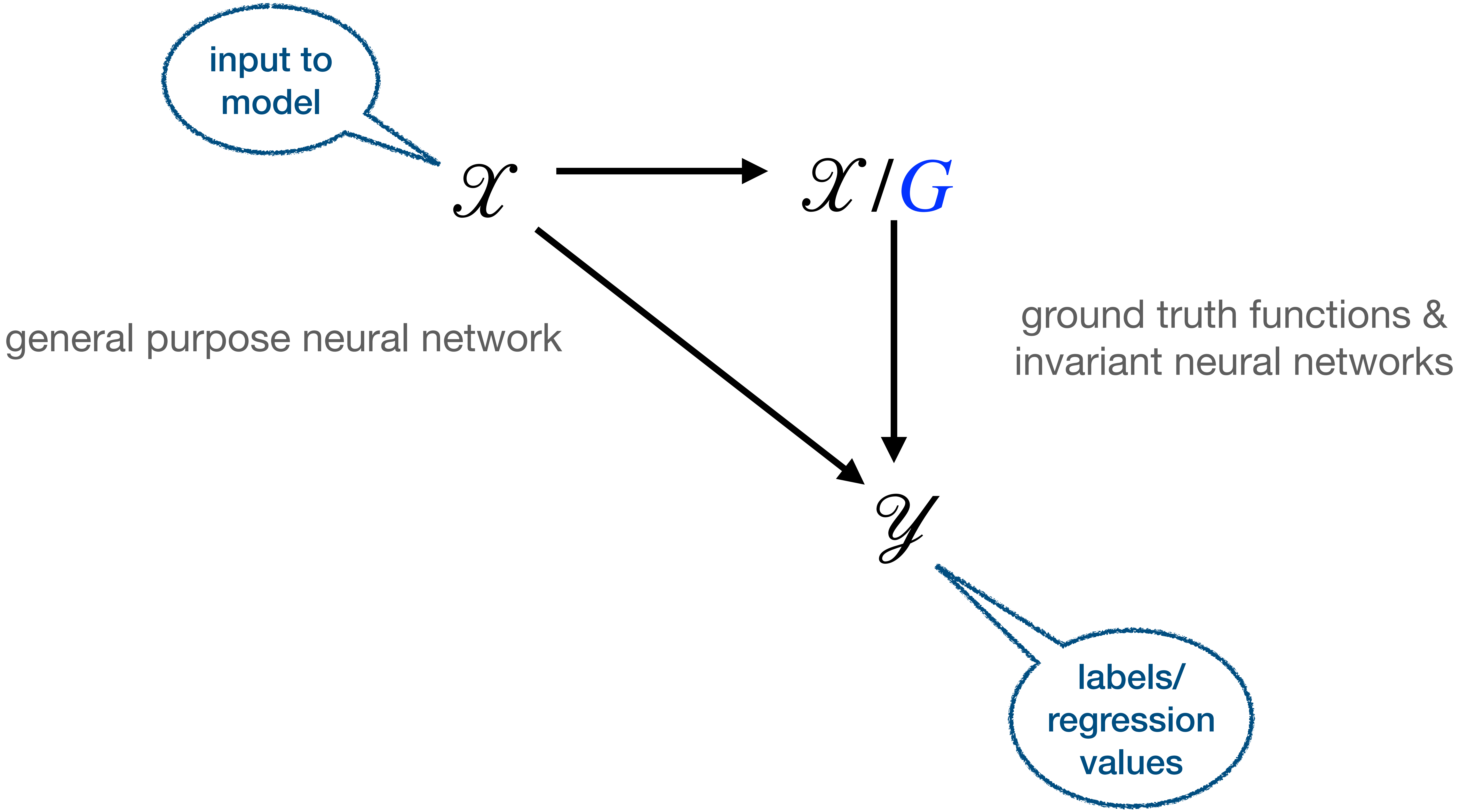


Figure credit: Inneke Mayachita

- Transformer without positional encoding: token-permutation-invariant

Model symmetries

In general, there is a smaller function space containing some ground truth



Does learning become ‘easier’ under symmetric ground truths?

1. How do we prove this formally?
2. Extending existing techniques?

Spoiler:

1. Boolean functions: clear application of our intuition
2. Real-valued functions: messier, but can still show lower bounds!

Learning under symmetries

Learning a smaller function class

- Concept class $\Lambda \subseteq \{f(\cdot, \theta) : \mathscr{X} \to \mathbb{R} \mid \theta \in \Theta, f(X, \theta) = f(gX, \theta), \forall g \in G\}$.
- Ground truth function $\Lambda \ni h^* : \mathscr{X} \to \mathbb{R}$
- E.g. learning algorithm: given n samples $(x_i, y_i = h^*(x_i))_{i=1}^n$, solve ERM

$$\min_{\theta \in \Theta} \sum_{i=1}^{n} \ell(f(x_i, \theta), y_i)$$

Statistical problem: How many samples do we need to learn up to some error? - generalization bounds

Computational problem: Are there efficient algorithms? - NP hardness, PAC learning, SQ learning

PAC learning (L. G. Valiant, 1984)

Set up

- Given a <u>concept class</u> $\mathscr{C} \subseteq 2^{\mathscr{X}}$ (set of Boolean-output functions over $\mathscr{X}$).
- Given a <u>distribution</u> $\mathscr{D}$ over $\mathscr{X}$ and a <u>concept</u> $c \in \mathscr{C}$, samples are drawn from the joint distribution $\mathscr{D}_c$ over $\mathscr{X} \times \{\pm 1\}$.
- Given <u>error parameter</u> $\epsilon \in (0,1)$, <u>confidence parameter</u> $\delta \in (0,1)$.

Examples

input set	0-1 adjacency matrix of graphs
concept	graphs with Eulerian cycles
distribution	Erdős-Rényi

PAC learning (L. G. Valiant, 1984)

- A (distribution-dependent) PAC-learning algorithm is a function $A := A^m_{\epsilon,\delta,\mathscr{C},\mathscr{D}} : (\mathscr{X} \times \{\pm 1\})^m \to 2^{\mathscr{X}}$ such that for any $c \in \mathscr{C}$,

$$\mathbb{P}_{Z\sim\mathscr{D}_c^m}[\text{error}_c(A(Z)) \geq \epsilon] < \delta, \text{ with } \text{error}_c(h) := \mathbb{P}_{X\sim\mathscr{D}}[h(X) \neq c(X)]$$

- It is efficient if m is polynomial in $1/\epsilon, 1/\delta, |c|$ and A can be evaluated in polynomial time in its input.

Very general framework of learning, but hard to give proofs

(Correlational) statistical queries (Kearns, 1998)

A natural restriction of PAC

- Algorithms do not have access to samples but statistics over sample distribution.
- Given concept $c : \mathscr{X} \to \mathscr{Y}$ and sample distribution $\mathscr{D}_c$ over $\mathscr{X} \times \mathscr{Y}$, an SQ query oracle
  - IN: query $g : \mathscr{X} \times \mathscr{Y} \to [-1,1]$ and tolerance parameter τ
  - OUT: $\mathrm{SQ}(g, \tau) \in \mathbb{E}_{(X,Y)\sim\mathscr{D}_c}[g(X, Y)] \pm \tau$
- A CSQ query oracle requires $g(x, y) = f(x) \cdot y$ for some $f : \mathscr{X} \to \mathscr{Y}$
  - $\mathrm{CSQ}(g, \tau) \in \langle f, c\rangle_{L^2(\mathscr{D})} \pm \tau$ returns a correlation value

Hardness of learning in the (C)SQ model

- A class $\mathscr{F}$ of functions $f : \mathscr{X} \to \mathscr{Y}$ is hard to learn under the (C)SQ model if there are no algorithm $A := A^m_{\epsilon,\tau,\mathscr{F},\mathscr{D}}$ such that for all $c \in \mathscr{F}$,
  - A inputs $m = \text{poly}(1/\epsilon, |c|)$ (C)SQ oracle results with tolerance $\tau^{-1} = \text{poly}(1/\epsilon, |c|)$, and
  - outputs a hypothesis f such that $\|f - c\|_{L^2(\mathscr{D})} \leq \epsilon$

Population gradient descent + noise + square loss $\in$ CSQ

Why do we study CSQ model?

- Gradient of the population risk under square loss decomposes as:

$$\frac{1}{2}\nabla_\theta \mathbb{E}_{X,Y}[(f(X,\theta)-Y)^2] = \underbrace{\mathbb{E}_{X,Y}[f(X,\theta)\cdot\nabla_\theta f(X,\theta)]}_{\text{independent of Y}} - \underbrace{\mathbb{E}_{X,Y}[Y\cdot\nabla_\theta f(X,\theta)]}_{\text{CSQ}}$$

- Adding (Gaussian) noise in each gradient step to simulate error in CSQ oracle (controlled by τ)

CSQ ⊂ SQ ⊂ PAC

Relationship between 3 learning models

- There is an exponential separation between SQ and PAC for learning PARITY : $\left\{ f_c : \{\pm 1\}^d \ni z \mapsto \prod_{i \in c} z_i \text{ for } c \in 2^{[d]} \right\}$ over uniform distribution.
- For Boolean-valued functions, CSQ $=$ SQ.
- For real-valued functions, there is an exponential separation between CSQ and SQ for learning sparse polynomial over product distributions.

Andoni, Panigrahy, Valiant, and Zhang. Learning sparse polynomial functions, 2013

A tool to prove lower bound under CSQ

CSQ dimension

- Informally: the maximum number of functions that are pairwise almost orthogonal (in $L^2(\mathscr{D})$ inner-product).

$$\text{CSQdim}(\mathscr{F}) := \sup_{F \subset \mathscr{F}} \{ |F| : \forall f \neq f' \in F, \underbrace{|\langle f, f' \rangle| \leq 1/|F|,}_{\text{almost orthogonal}} \quad \underbrace{\|f\| = \Theta(1)}_{\text{non-vanishing norm}} \quad \}$$

From CSQ dimension to query complexity

- Theorem (Blum, Furst, Jackson, Kearns, Mansour, and Rudich, 1994)

  Any SQ algorithm that uses tolerance parameter lower bounded by τ must make at least $(\mathrm{CSQdim}(\mathscr{F}) \cdot \tau^2 - 1)/2$ queries to learn $\mathscr{F}$ with accuracy at least τ.

- Main proof directions: find a large family of non-vanishing hard functions that are pairwise almost orthogonal

General Boolean functions

Intuitive extension of SQ lower bound techniques leads to a general result

General result

Set up

- Action of a group G on $\mathscr{X} = \{\pm 1\}^n$ partition $\mathscr{X}$ into $\mathscr{O} = \{O_1, \ldots, O_k\}$ orbits
- $p_{\mathscr{O}} \in \mathbb{R}^k$ - vector of *probability a random bit string is in some orbit*
- Concept class $\mathscr{H} = \{f : \{\pm 1\}^n \to \{\pm 1\}$ with $f(g \cdot x) = f(x), \forall g \in G\}$

General result

Main result

- Main result in the section:

  Any SQ algorithm that learns $\mathscr{H}$ to classification error $< \frac{1}{4}$

  with tolerance τ

  requires at least $\tau^2 \| p_{\mathcal{O}} \|_2^{-2} / 2$ queries

Intuition: $\mathcal{O}$ is the effective domain. A uniform distribution over $\mathscr{X}$ induces a distribution $p_{\mathcal{O}}$ over $\mathcal{O}$. Show hardness of learning over $p_{\mathcal{O}}$ instead.

Example of general result for Boolean function

- By Hölder inequality, $\|p_{\mathcal{O}}\|_2^2 \leq 2^{-n} \max_j |O_j|$. If $\tau = \Theta(1)$, then

Group	$2^{-\#bits} \max_{O_k \in \mathcal{O}_\rho} \|O_k\|$	Query Complexity
Symmetric group on n bits	$\frac{\binom{n}{n/2}}{2^n} = \frac{1}{\Theta(\sqrt{n})}$	$\Theta(\sqrt{n})$
Symmetric group on $n \times n$ graphs	$\frac{n!}{2^{n^2}} = 2^{-n^2+n\log n+O(n)}$	$\Omega(2^{O(n^2)})$
Cyclic group on n bits	$\frac{n}{2^n}$	$\Omega(2^n/n)$

Table 1: Query complexity of learning common invariant Boolean function classes.

Summary: symmetric Boolean classes enjoy savings in SQ lower bound!

Proof sketch

- $(1-\eta)$-pairwise independent function class from: *Chen, Gollakota, Klivans, and Meka. Hardness of noise-free learning for two-hidden-layer neural networks. NeurIPS 2022. (traced back to Bogdanov)*

  - Function class $\mathscr{C}$ s.t. $\text{Law}_{f\sim\text{Unif}(\mathscr{C})}((f(x), f(x'))) = \text{Unif}(\mathscr{Y}) \otimes \text{Unif}(\mathscr{Y})$ with probability $1-\eta$ over draw of $x, x' \sim \mathscr{D}$

  - Theorem (informal): If $\mathscr{C}$ is $(1-\eta)$-pairwise independent then any SQ learner capable of distinguishing $\mathscr{D}_c$ from `random label' with tolerance τ requires at least $\tau^2/(2\eta)$ queries.

  - For us, check that $\eta = \|p_{\mathscr{O}}\|_2^2$ for our symmetric function class.

What about even smaller, more practical invariant classes?

Exponential SQ lower bound for Boolean graph neural networks (GNNs)

Even practical, GNN-realizable Boolean functions are hard to learn

Boolean graph neural networks (GNNs)

- Graph-invariant functions $f : \{0,1\}^{n\times n} \to \{0,1\}$ with input adjacency matrix of a graph such that $f(\mathbf{X}) = f(\mathbf{P}\mathbf{X}\mathbf{P}^\top)$ for any permutation matrix $\mathbf{P}$.
- Examples:

Message-passing neural networks

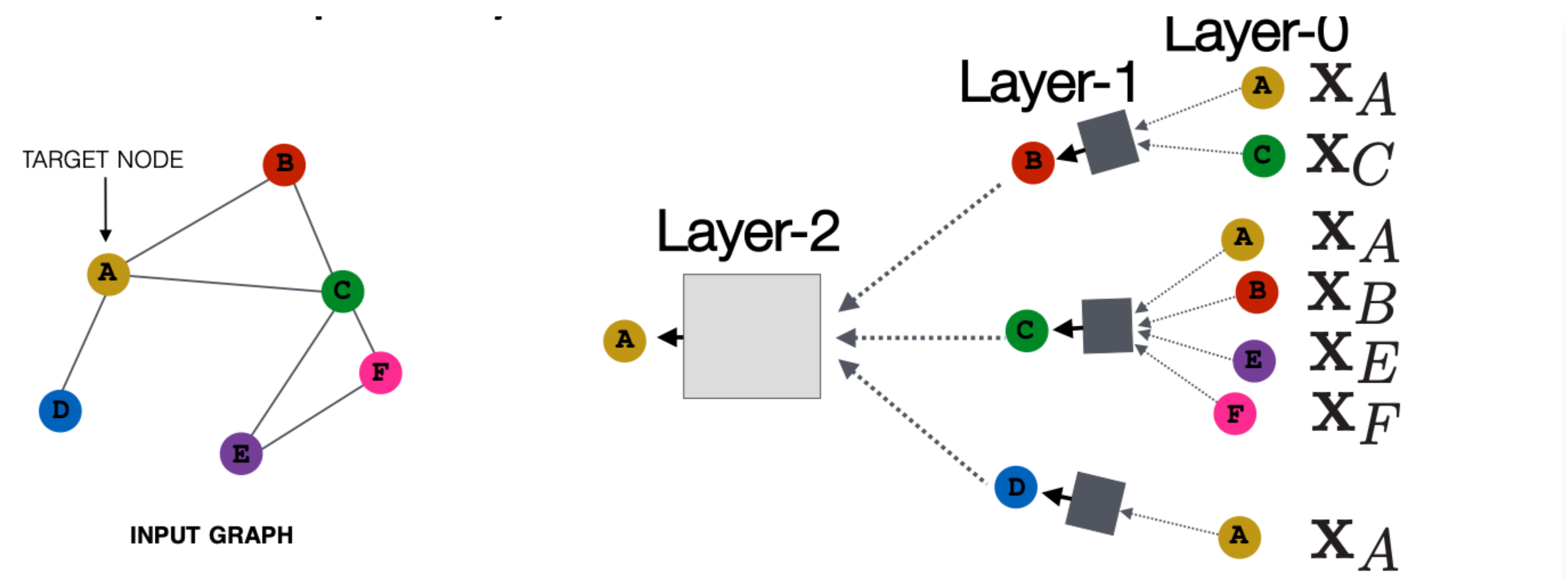


Figure credit: Jure Leskovec Stanford CS224W slide

Graph convolutional networks

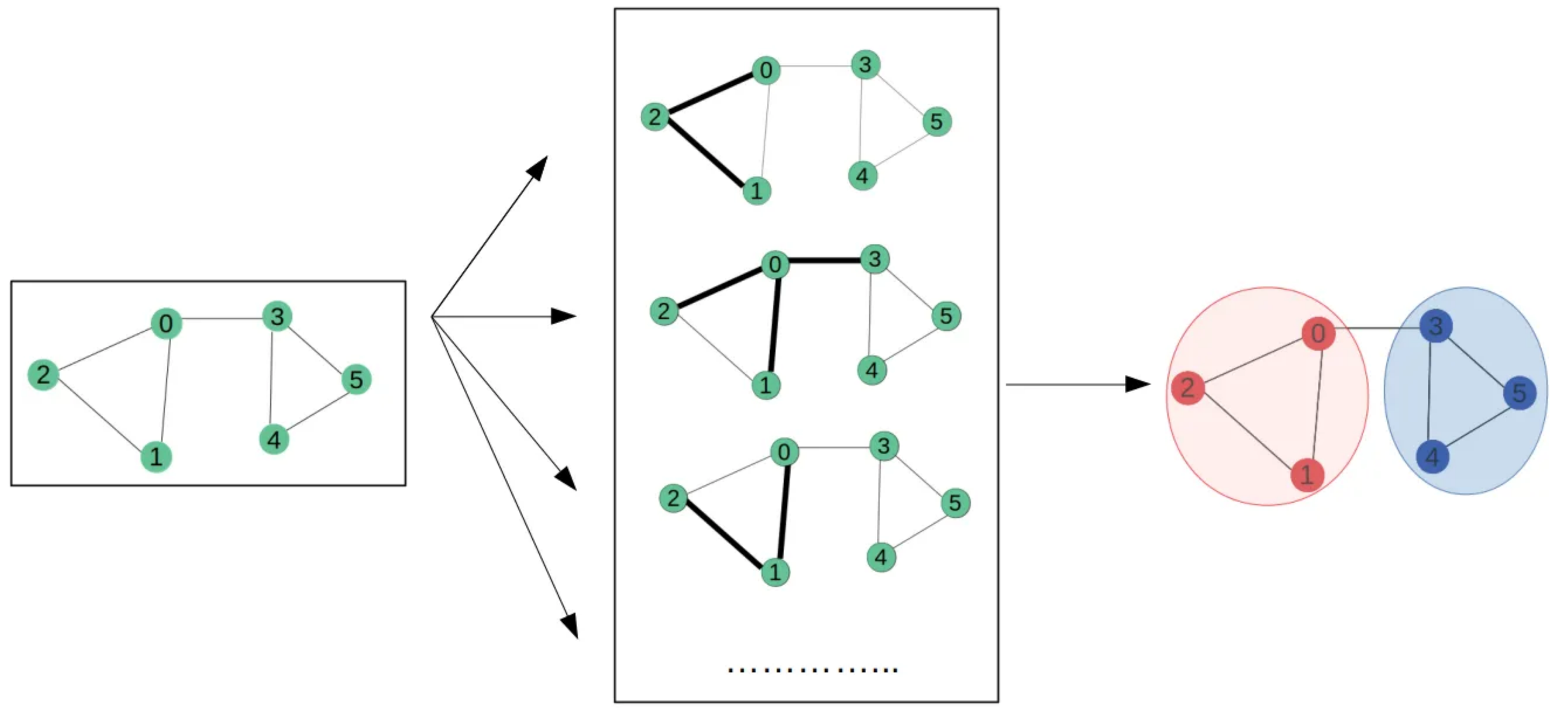


Figure credit: Inneke Mayachita

Hardness of learning GNN in the number of nodes

The concept class

- Concept class: 2-hidden-layer GNNs $f = f^{(2)} \circ f^{(1)}$ with
  - $f^{(1)} : \{0,1\}^{n \times n} \to \mathbb{R}^{k_1}$ message passing $[f^{(1)}(\mathbf{A})]_i = \mathbf{1}_n^\top \sigma(a_i + b_i \mathbf{A} \mathbf{1}_n), i \in [k_1]$
  - $f^{(2)} : \mathbb{R}^{k_1} \to \{0,1\}$ a 1-hidden layer ReLU network with k_2 hidden neurons
- Input distribution: Erdős-Rényi random graphs with edge probability 1/2.
- $k_1, k_2 \in O(n)$

This is an even smaller class than all Boolean graph-invariant functions (since message-passing is non-universal)

Hardness of learning GNN in the number of nodes

Hard family of functions in the concept class

- $c_{\mathbf{A}}(i)$ counts the number of nodes in the graph with outdegree $i \in [n+1]$
- Define a parity-like function indexed by $S \subset [n+1], b \in \{0,1\}$:

$$g_{S,b}(\mathbf{A}) = b + \sum_{i \in S} c_{\mathbf{A}}(i) \mod 2$$

- Define the family of hard function:

$$\mathscr{H}_n = \{g_{S,b} | S \subset [n+1], b \in \{0,1\}\}$$

Hardness of learning GNN in the number of nodes

Main result

- Our result:

Any SQ algorithm that learns $\mathscr{H}_n$ up to classification error $< \frac{1}{4}$

with queries of tolerance τ

requires at least $\Omega\left(\tau^2 \exp(n^{\Omega(1)})\right)$ queries.

This smaller class of realistic Boolean functions are still hard to learn

Exponential CSQ lower bound for real-valued GNNs

Extending exponential lower bound for NN to GNNs

Hardness of learning GNN in feature dimension

- GNNs often has both graph data (adjacency matrix) and node features as input.
- Node features are iid Gaussian $\mathcal{N}$, graph distribution $\mathcal{E}$ is arbitrary but non-degenerate.
- Consider the function class of 1-hidden-layer graph convolutional network:

$$F_n^{d,k} = \left\{ f : \mathbb{R}^{n \times d} \times \mathbb{G}_n \to \mathbb{R}, f(\mathbf{X}, \mathcal{G}) = \mathbf{1_n}^\top \sigma\big(\overbrace{\underbrace{\mathbf{A}(\mathcal{G})\mathbf{X}}_{\text{sum neighbor features}} \qquad \mathbf{W}}^{\text{apply linear map}} \big)\mathbf{a} \mid \mathbf{W} \in \mathbb{R}^{d \times 2k}, \mathbf{a} \in \mathbb{R}^{2k} \right\}$$

graph

ReLU

number of nodes (fixed)

feature dimension

adjacency matrix

learnable weights

Hardness of learning GNN in feature dimension

Hard functions

- Base on low dimensional subspace enumeration in

  Diakonikolas, Kane, Kontonis, and Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. COLT 2020

- Hard functions $g_n^{\mathbf{B}}(\mathbf{X}, \mathcal{G}) = \dfrac{f_n(\mathbf{XB}, \mathcal{G})}{\|f_n\|_{\mathcal{N}\times\mathcal{E}}}$ with $f_n(\mathbf{Z}, \mathcal{G}) = \mathbf{1_n}^\top \sigma(\mathbf{A}(\mathcal{G})\mathbf{Z}\mathbf{W}^*)\mathbf{a}^*$

low dimensional projection $\mathbb{R}^{d\times 2}$

special weights

hard GNN in low dimension : $\mathbb{R}^{n\times 2} \to \mathbb{R}$

Hardness of learning GNN in feature dimension

- Our result:

For any $d, n = \Theta(1), k = \Theta(d)$,

any CSQ algorithm that learns the hard class of function to some small constant error $\|f - h\|_{L^2(\mathcal{N}\times\mathcal{E})} \leq \epsilon$

requires either $2^{d^{\Omega(1)}}$ queries or at least one query with tolerance $d^{-\Omega(k)} + 2^{-d^{\Omega(1)}}$

Main new tool

Graph-invariant Hermite polynomial

- $$H_J^{\mathbf{A}} : \mathbb{R}^{n \times d} \to \mathbb{R} : \mathbf{X} \mapsto \frac{1}{\sqrt{n}} \sum_{v=1}^{n} H_J\left((\mathbf{A}\mathbf{X})_v \right).$$
- Acts as orthogonal basis for 1-hidden-layer GNN w.r.t $L^2(\mathcal{N})$ inner product.

This works out since action of $\mathbf{A}$ is ‘diagonal’ to action of the weight matrix on input $\mathbf{X}$

Other symmetries: frame-averaged functions

Many complications in deriving lower bound for more general real-valued symmetric functions

Group averaging

A naive approach to making symmetric function

- Given any (nice) function $h : \mathscr{X} \to \mathbb{R}$ and (nice) group G, one can symmetrize:

$$R[f](x) := \sum_{g \in G} f(g \cdot x)$$

- Symmetrizing 1-hidden-layer NN:

$$\mathscr{H}_G := \left\{ f : \mathbb{R}^{n \times d} \to \mathbb{R}, f(\mathbf{X}) = \frac{1}{\sqrt{|G|}} \sum_{g \in G} \mathbf{a}^\top \sigma(\mathbf{W}^\top (g^{-1}\mathbf{X}))\mathbf{1}_d \mid \mathbf{W} \in \mathbb{R}^{n \times k}, \mathbf{a} \in \mathbb{R}^k \right\},$$

- E.g. when $\mathscr{X} = \mathbb{R}^n$, G is the cyclic group, this captures convolutional neural nets (with large, looped filters)

Family of hard function for group-averaging

Using subspace enumeration from Diakonikolas, Kane, Kontonis, Zarifis, 2020

number of elements

orthogonal projection to low-dim subspace

$$C_G^{\mathscr{B}} = \left\{ g_{\mathbf{B}} : \mathbb{R}^{n \times d} \to \mathbb{R} \text{ with } g_{\mathbf{B}}(\mathbf{X}) = \frac{\sum_{g \in G} f^*(\mathbf{B}^\top g^{-1} \mathbf{X})}{\sqrt{|G| \cdot \|f^*\|_{\mathcal{N}}}} \mid \mathbf{B} \in \mathscr{B} \subset \mathbb{R}^{n \times 2} \right\}.$$

feature dimension (fixed)

stock hard function in low dimension: $\mathbb{R}^{n \times 2} \to \mathbb{R}$

Exponential CSQ lower bound for group-averaging

- Our result:

  For any $n, d = \Theta(1), k = \Theta(n),$ there exists a set of projections $\mathscr{B}$ of size at least $2^{\Omega(d^{\Omega(1)})}/|G|^2$, such that

  any CSQ algorithm that learns $C_G^{\mathscr{B}}$ to some small constant error $\|f - h\|_{L^2(\mathcal{N}\times\mathscr{E})} \leq \epsilon$

  requires either $2^{n^{\Omega(1)}}/|G|^2$ queries or at least one query with tolerance $\sqrt{|G|}n^{-\Omega(k)} + |G|2^{-n^{\Omega(1)}}$.

- Exponential when $|G| = \text{poly}(n)$. E.g. cyclic group.

Frame-averaging

- Group averaging is expensive
- Canonicalization: e.g. $G = \mathcal{S}_n, \mathcal{X} = \mathbb{R}^n$, symmetrize $h : \mathbb{R}^n \to \mathbb{R}$ by $h \circ$ sort
- A frame is a function $\mathcal{F} : \mathbb{R}^{n \times d} \to 2^G \backslash \emptyset$ such that symmetrize an arbitrary function h by averaging $\frac{1}{|\mathcal{F}(\mathbf{X})|} \sum_{g \in \mathcal{F}(\mathbf{X})} h(g^{-1}\mathbf{X})$ suffices
- E.g. $\mathcal{F}(\mathbf{X}) = G, \forall \mathbf{X}$ is the group-averaging (Reynold operator)

Frame-averaging 1-hidden-layer MLP

$$\mathscr{H}_{\mathscr{F}} := \left\{ f : \mathbb{R}^{n\times d} \to \mathbb{R}, f(\mathbf{X}) = \frac{1}{\sqrt{|\mathscr{F}(\mathbf{X})|}} \sum_{g\in\mathscr{F}(\mathbf{X})} \mathbf{a}^\top \sigma(\mathbf{W}^\top(g^{-1}\mathbf{X}))\mathbf{1}_d \mid \mathbf{W} \in \mathbb{R}^{n\times k}, \mathbf{a} \in \mathbb{R}^k \right\}$$

- E.g. $f : \mathbb{R}^{n\times 1} \to \mathbb{R}, f(\mathbf{X}) = \mathbf{a}^\top \sigma(\mathbf{W}^\top(\text{sort}(\mathbf{X})))$
- If $\mathbf{X} \sim \mathscr{N}$, $\text{sort}(\mathbf{X})$ has complicated distribution
- Can no longer use Diakonikolas, Kane, Kontonis, Zarifis, 2020 hard functions

Solution: assume sign-invariant frame (e.g. sort by absolute values) and use hard functions from *Goel, Gollakota, Jin, Karmalkar, and Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. ICML 2020*

Other results

- SQ vs CSQ separation for learning invariant polynomial
- NP hardness of proper learning of GNN via hardness of learning halfspace with noise
- Lower bound L^2 norm for all our symmetric hard functions (also nontrivial)

Conclusion

- We formalized the intuition that symmetric function classes are smaller and thus easier to learn, by showing:

SQ/CSQ	**Exponential/Super polynomial**	**Boolean/ Real-valued**	**Symmetric function class**
SQ	depends	Boolean	general
SQ	exponential in nodes	Boolean	2-hidden-layer message-passing NN
CSQ	exponential in feature dimension	real	1-hidden-layer GCN
CSQ	exponential in items	real	(polynomial-sized) group-averaged 1-hidden-layer MLP
CSQ	superpolynomial in item	real	sign invariant frame-averaged 1-hidden-layer MLP

- Developed tools may be of independent interest (e.g. invariant Hermite polynomial)

Thank you!

Q&A

- Paper link: https://arxiv.org/abs/2401.01869
  - 'On the hardness of learning under symmetries' - Bobak T. Kiani*, L.*, Hannah Lawrence*, Stefanie Jegelka, Melanie Weber.

Massachusetts Institute of Technology