What Can Neural Networks Reason About?

Keyulu Xu , Jingling Li , Mozhi Zhang , Simon S. Du, Ken-ichi Kawarabayashi, Stefanie Jegelka

ICLR, 2020 [Paper]

Anushree Hede (anuhede@seas.upenn.edu)
April 6, 2020

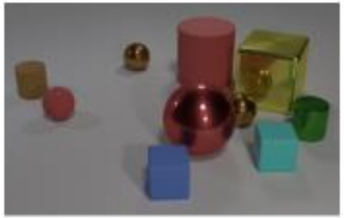
Penn Engineering

Motivation

<u>Observation</u> - Neural nets that succeed on reasoning tasks possess specific structures

- Reasoning processes resemble algorithms

<u>VQA</u>: "Starting at the green cylinder, if each time we jump to the closest object, which object is K jumps away?"



Bellman-Ford algorithm

```
for k = 1 ... |S| - 1:
  for u in S:
    d[k][u] = min_v d[k-1][v] + cost (v, u)
```

- Shortest path task: dynamic programming
- Reasoning process

Graph Neural Network

```
for k = 1 ... GNN iter:
  for u in S:
    h_u^(k) = Σ_v MLP(h_v^(k-1), h_u^(k-1))
```

- Model pairwise relations
- $h_u^{(k)}$ of each node u (in iteration k) recursive updates by aggregation

$$h_S = MLP_2(\sum_{u \in S} h_u^{(K)})$$

Penn Engineering

Problem

- Neural networks that succeed in reasoning tasks usually possess specific structures that generalize better
- <u>Hypothesis</u> Strong alignment of network structure w/ algorithmic structure explains success in reasoning tasks
- <u>Intuition</u> strong alignment → network learns simple algorithm to simulate reasoning process → better sample efficiency
  - Formalize: define a numeric measure for algorithmic alignment
  - Develop theoretical framework to characterize what a neural network can learn about
  - Show experimental support for hypothesis: algorithmic alignment facilitates learning
    - GNNs align with dynamic programming
- What tasks can a neural network (sample efficiently) learn to reason about?

Presentation Map

1. Related work and previous approaches
2. Preliminary concepts and definitions
3. Theoretical framework for **algorithmic alignment**
4. Experiments: Demonstrate generalizability on reasoning tasks
5. Conclusion and takeaways

Previous work

- GNNs suitable for relational reasoning because they have relational inductive biases
  - Battaglia et al. (2018) [1]
- Here, formally introduce algorithmic alignment
  - Quantify relation between network and algorithm structure
  - Derive implications for learning
  - Basis for what reasoning tasks a network can learn well
- Differs from structural assumptions common in learning theory:
  - Bartlett et al., 2017; [2] : norms of network parameters to measure capacity of NNs
  - Golowich et al., 2018 [3] : sample complexity of deep NNs independent of depth and width under additional assumptions
- Aligns with reasoning

Notation

- S: universe, i.e., a configuration of objects to reason about. Each object $s \in S$ is represented by a vector X
  - Given a set of universes $\{S_1 \ldots S_M\}$, answer labels $\{y_1 \ldots y_M\} \subseteq Y$
  - Aim to learn function *g* that can answer questions about unseen universes, $y = g(S)$
- For example:
  - Task: shortest path problem in graphs
    - Universe → graph; objects → vertices, edges; y → shortest path lengths
  - Task: visual question answering
    - Universe → image; objects → questions, context; y → answer

Penn Engineering

Network Structures

MLP

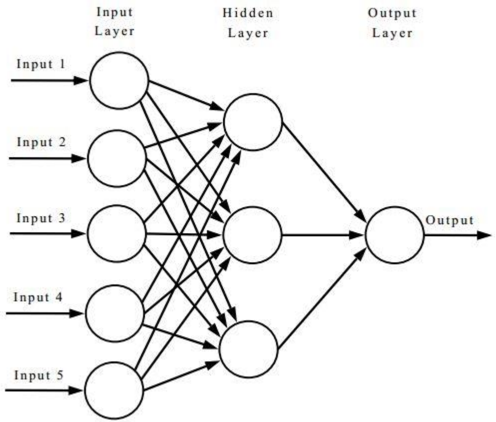
- Works well on single-object universe
- Poor generalizability otherwise
- Eg: simple classifier of objects as vectors

Deep Sets

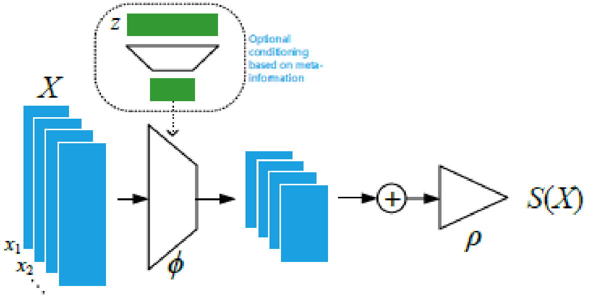
- Induces permutation invariance in neural network
- $y = MLP_2(\sum_{s \in S} MLP_1(X_s))$
- Eg: Compute sum of feature over all objects

GNN

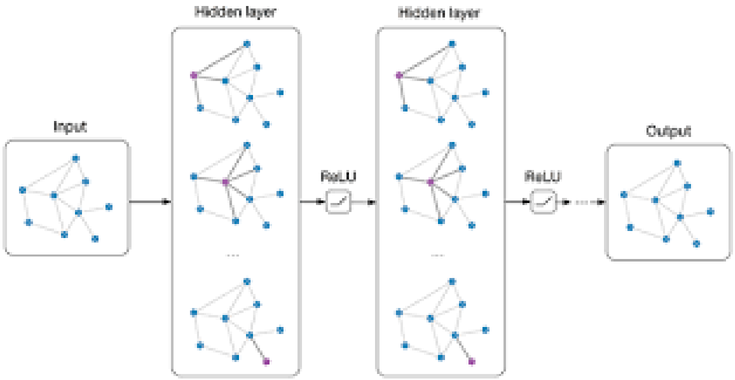
- Models pairwise relations between objects
- Recursive updates by aggregating neighboring nodes
- Eg: shortest path



[4]



[5]



[6]

Penn Engineering

Theory: Sample Complexity

- Given
  - $\{x_i, y_i\}_{i=1}^{M} \sim \mathcal{D}$
  - Data satisfies $y_i = g(x_i)$ for some g
  - $f = \mathcal{A}(\{x_i, y_i\}_{i=1}^{M})$ is function learned by algorithm $\mathcal{A}$
  - Error parameter $\mathcal{E} > 0$ and failure probability δ
- PAC learning theory: analyzes whether and under what conditions a learner $\mathcal{A}$ will probably output an approximately correct classifier
  - Hypothesis h is *approximately correct* if its error over the input distribution is bounded by some $\mathcal{E}$
  - If $\mathcal{A}$ outputs classifier using h with probability $1 - \delta$, classifier is *probably approximately correct*
- Using above, g is $(M, \mathcal{E}, \delta)$-learnable with $\mathcal{A}$ if
  - $\mathbb{P}_{x \sim \mathcal{D}}[\|f(x) - g(x)\| \leq \mathcal{E}] \geq 1 - \delta$
- Sample complexity $\mathcal{C}_{\mathcal{A}}(g, \mathcal{E}, \delta)$
  - Minimum M for which g is $(M, \mathcal{E}, \delta)$-learnable with $\mathcal{A}$

Theory: Algorithmic Alignment

- g: reasoning function, $\mathcal{N}$: neural network with modules $\{\mathcal{N}_i\}_{i=1}^{n}$
- $\mathcal{N}$ $(M, \mathcal{E}, \delta)$- algorithmically aligns with g if
  - $\mathcal{N}$ simulates g using finite number of modules (n)
  - Each module f_i has low sample complexity
    - $\exists \mathcal{A}_i$ for the $\mathcal{N}_i$'s such that $n \cdot \max_i \mathcal{C}_{\mathcal{A}_i}(f_i, \mathcal{E}, \delta) \leq M$
- Alignment value m = $\sum_i \mathcal{C}_{\mathcal{A}_i}\,(f_i, \mathcal{E}, \delta)$
- Small m → all steps f_i to simulate g are *easy to learn*
- Sample complexity of MLP
  - "Simple" functions → polynomial → sample efficiently learnable by MLP
  - Binary classifier → $\sigma(W^T X)$ represented as a polynomial
  - "For loop" is complex algorithm step → not a polynomial

Framework

1 Algorithm steps → 2 Network modules → 3 Compute algorithmic alignment → 4 Sample complexity bound → 5 Generalization

- Theoretical result: sample complexity bound increases with algorithmic alignment value m
  - Simplified setting, sequential training, auxiliary labels
- Generalization ability verified experimentally

Algorithmic alignment improves sample complexity

Theorem 3.6. *(Algorithmic alignment improves sample complexity).* *Fix ϵ and δ. Suppose $\{S_i, y_i\}_{i=1}^M \sim \mathcal{D}$, where $|S_i| < N$, and $y_i = g(S_i)$ for some g. Suppose $\mathcal{N}_1, ..., \mathcal{N}_n$ are network $\mathcal{N}$'s MLP modules in sequential order. Suppose $\mathcal{N}$ and g (M, ϵ, δ)-algorithmically align via functions $f_1, ..., f_n$. Under the following assumptions, g is $(M, O(\epsilon), O(\delta))$-learnable by $\mathcal{N}$.*

a) ***Algorithm stability.*** *Let $\mathcal{A}$ be the learning algorithm for the $\mathcal{N}_i$'s. Suppose $f = \mathcal{A}(\{x_i, y_i\}_{i=1}^M)$, and $\hat{f} = \mathcal{A}(\{\hat{x}_i, y_i\}_{i=1}^M)$. For any x, $\|f(x) - \hat{f}(x)\| \le L_0 \cdot \max_i \|x_i - \hat{x}_i\|$, for some L_0.*
b) ***Sequential learning.*** *We train $\mathcal{N}_i$'s sequentially: $\mathcal{N}_1$ has input samples $\{\hat{x}_i^{(1)}, f_1(\hat{x}_i^{(1)})\}_{i=1}^N$, with $\hat{x}_i^{(1)}$ obtained from S_i. For $j > 1$, the input $\hat{x}_i^{(j)}$ for $\mathcal{N}_j$ are the outputs from the previous modules, but labels are generated by the correct functions $f_{j-1}, ..., f_1$ on $\hat{x}_i^{(1)}$.*
c) ***Lipschitzness.*** *The learned functions $\hat{f}_j$ satisfy $\|\hat{f}_j(x) - \hat{f}_j(\hat{x})\| \le L_1 \|x - \hat{x}\|$, for some L_1.*

Corollary 3.7. *Suppose universe S has ℓ objects $X_1, ..., X_\ell$, and $g(S) = \sum_{i,j}(X_i - X_j)^2$. In the setting of Theorem 3.6, the sample complexity bound for MLP is $O(\ell^2)$ times larger than for GNN.*

Experiments

- Apply algorithmic alignment framework to analyze
  - MLP
  - Deep Sets
  - GNNs

  to explain generalizability
- Reasoning tasks:
  - Summary statistics
  - Relational argmax
  - Dynamic programming
  - NP-hard problem
- Empirical comparison of sample complexity models
  - Extensive hyperparameter tuning to ensure all models perfectly fit training sets
  - Test accuracy reflects generalizability

Summary Statistics

X_s	loc	val	col

- Task: Maximum value difference
  - Object X = [h_1; h_2; h_3] with location h_1, value h_2, and color h_3.
  - Predict the difference in value between the most and the least valuable objects
  - $y(S) = \max_{s \in S} \mathrm{h}_2 (X_s) - \min_{s \in S} \mathrm{h}_2 (X_s)$
- MLP
  - High sample complexity
  - Sorting objects by value reduces →subtraction: $y(S) = h_2(X_{|S|}) - h_2(X_1)$
- Deep Sets
  - Better sample complexity, strong generalization
- GNN
  - Special case of relational argmax; which GNNs can learn

Test Accuracy in %

MLP	Sorted MLP	Deep Sets	GNN1	GNN3
9	100	96	95	100

Relational Argmax

X_s	loc	val	col

- Task: Furthest pair among a set of objects
  - Object X = [h_1; h_2; h_3] with location h_1, value h_2, and color h_3.
  - Find the colors of the two objects with the largest distance
- Deep Sets
  - "Most pairwise relations cannot be encoded as sum of individual objects"
  - MLP learns complex "for loop" → poor sample complexity
- GNN
  - GNN1 sums over all pairs of objects, compares pairwise information
  - Aligns well without learning "for loops"

Test Accuracy in %

MLP	Deep Sets	GNN1	GNN3
9	21	92	95

Dynamic Programming

- General recursive form of DP:
  - Answer[k][i] = DP-Update({Answer[k - 1][j]}; j = 1…n)
  - $Answer[k][i]$ in DP $\leftrightarrow h_i^{(k)}$ in GNN
  - GNN with enough iterations can sample efficiently learn any DP algorithm with a simple DP-update function
- Task: shortest path problem
  - distance[1][u] = cost(s; u); s is source vertex
  - distance[k][u] = $\min_v$ distance[k -1][v] + cost(v; u)
- GNN
  - With at least four iterations generalize well
  - Other networks have high sample complexity
- VQA can be formulated as DP → solved by GNN

Test Accuracy in %

MLP	Deep Sets	GNN1	GNN2	GNN3	GNN4	GNN7
8	11	27	62	91	94	96

NP-hard Problem: Subset sum

- NP-hard problems → cannot be solved by DP → GNN cannot sample-efficiently learn these
  - *Framework: If structure of reasoning algorithm is known, a network with a similar structure can be designed to learn it*
- Task: Subset sum as zero
  - Approach 1: Exhaustive search
    - Enumerate and check whether subset from possible $2^{|S|}$ subsets has zero-sum
  - Approach 2: Neural Exhaustive Search (NES)
    - Each subset → LSTM → MLP_1 → max-pooling layer → MLP_2
    - LSTM + MLP_1 perform simple step: to check zero-sum

Test Accuracy in %

MLP	Deep Sets	GNN1	GNN6	NES
60	61	69	72	98

Results and Conclusion

- Results explain success of current neural architectures on four popular reasoning tasks
  - GNNs generalize because underlying reasoning processes aligns with DP
  - Expected to learn sample efficiently
- Introduce an algorithmic alignment framework to formalize the relation b/w structure of a neural network and a reasoning process
  - Provide preliminary results on sample complexity
- Algorithmic alignment perspective may inspire neural network design for new reasoning tasks
- Future: use algorithmic alignment to learn reasoning paradigms beyond DP

Takeaways

- Many times in deep learning we accept network structures that perform well on a task without too many questions
  - This framework gives a strong intuition for choosing a network that generalizes well for a specific kind of task
- Intuition developed from algorithm
  - Deductive reasoning?
- Can be applied to more reasoning tasks?
  - Preference Learning
  - Logical Induction
- Minor critique: What have we learned after all?
  - Successful empirical results, results not supported by framework?

References

- [1] Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).
- [2] Relational inductive biases, deep learning, and graph networks. Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems* (pp. 6240-6249).
- [3] Golowich, N., Rakhlin, A., & Shamir, O. (2017). Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*.
- [4] https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network
- [5] Deep Sets - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Architecture-of-DeepSets-Equivariant_fig5_315383633 [accessed 2 Apr, 2020]
- [6] https://towardsdatascience.com/https-medium-com-aishwaryajadhav-applications-of-graph-neural-networks-1420576be574