Swarat Chaudhuri
The University of Texas at Austin

Neurosymbolic Programming for Science

* Based on work with Jennifer Sun, Atharva Sehgal, Ameesh Shah, Eric Zhan, Ann Kennedy, Megan Tjandrasuwaita, and Yisong Yue
Neurosymbolic Programming

Throughout history, science has required (i) data, and (ii) human insights to make sense of the data.
AI for science

Machine Learning

Hypotheses

Data

Experiments

NEWS | 20 February 2020

Powerful antibiotics discovered using AI

A.I. Predicts the Shape of Nearly Every Protein Known to Science

AI AND UNIVERSE

The AI behind getting the first-ever picture of a ‘black hole’

A celebrated AI has learned a new trick: How to do chemistry

by Marc Zimmer, The Conversation
AI for behavioral neuroscience

Mouse Action Recognition System [Segalin et al., 2021]

Data: videos depicting animal behavior
Challenges

1. Interpretability rather than black-box prediction

How is gait stable vs. unstable?
Challenges

2. Labels can be hard to get

> 200 million image-text pairs

DALL-E2

ChatGPT

> 300 billion words
Challenges

2. Labels can be hard to get

10^4 \sim 10^5 \text{ of frames for training!}

100 \text{ expert hours to annotate one day of recording}
Challenges

3. Labels can even be unknown

Lab A

Lab B

Sniff       Other

Sniff Face,
Sniff Body

Mount

???

Attack

Chase    Bite    ???

Sniff Face,
Sniff Body

???
Challenges

4. Distribution shifts

Lab A: Anderson Lab at Caltech; Lab B,C: Mazmanian Lab at Caltech
Science needs systematic mechanisms for...

(i) interpreting discovered insights
(ii) incorporating domain knowledge to reduce need for data
(iii) reusing code and data across labs
Data-driven discovery as programming

(i) Neurosymbolic programs
(ii) Neurosymbolic learning algorithms

A. Neurosymbolic Programs
Neurosymbolic Program: Example

IF \( (\text{distance between noses}) < A \) AND \( (\text{facing angle}) < B \)

THEN \text{investigation} IF \( (\text{acceleration of mouse 1}) > C \)

ELSE \text{investigation} IF \( (\text{distance from nose 1 to centroid 2}) < D \)
Neurosymbolic Program: Example

IF (distance between noses) < A AND (facing angle) < B

THEN investigation IF (acceleration of mouse 1) > C

ELSE investigation IF (distance from nose 1 to centroid 2) < D

Features defined by experts
Neurosymbolic Program: Example

IF (distance between noses) < A AND (facing angle) < B

THEN investigation IF (acceleration of mouse 1) > C

ELSE investigation IF (distance from nose 1 to centroid 2) < D

Structure & parameters learned from data
Neurosymbolic Program: Example

IF (distance between noses) < A AND (facing angle) < B
THEN investigation
IF (acceleration of mouse 1) > C
ELSE investigation
IF (distance from nose 1 to centroid 2) < D

Filter weight
Time
B. Neurosymbolic Learning Algorithms
Domain-Specific Language (DSL): “A Family of Programs”

Program syntax defined as a grammar:

\[
\alpha ::= x \mid c \\
\mid \oplus(\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta(\alpha_1, \ldots, \alpha_k) \\
\mid \text{sel}_S x \mid \text{map}(\lambda x_1.\alpha_1) \ x \mid \text{fold}(\lambda x_1.\alpha_1) \ c \ x \\
\mid \text{if} \ \alpha_0 \ \text{then} \ \alpha_1 \ \text{else} \ \alpha_2
\]

Type system tracking, for example, vector and matrix dimensions

DSL is differentiable, so you can train an NN in the context of a larger program
• For example, differentiable interpretation of if-then-else statements
Neurosymbolic Program Synthesis

\[\alpha ::= x \mid c \]
\[\quad \mid \oplus(\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta(\alpha_1, \ldots, \alpha_k) \]
\[\quad \mid \text{sel}_S x \mid \text{map}(\lambda x_1.\alpha_1) x \mid \text{fold}(\lambda x_1.\alpha_1) c x \]
\[\quad \mid \text{if } \alpha_0 \text{ then } \alpha_1 \text{ else } \alpha_2\]

Domain Specific Language (DSL)

Learning Objective (Loss Function) \rightarrow Learning Algorithm (program synthesis) \rightarrow Neurosymbolic Program \((\alpha, \theta)\)
Learning as Bilevel Optimization

\[
\min_{\alpha} \left( \min_{\theta} \text{Loss}(\alpha, \theta) + s(\alpha) \right)
\]

- \( \text{Loss}(\alpha, \theta) \) quantifies fit to the dataset
- The **structural cost** \( s(\alpha) \) penalizes complex program structures.
• Setting $\alpha$ as a neural network $\rightarrow$ standard deep learning
• Finding $\alpha$ is analogous to neural architecture search
  • Sometimes call $\alpha$ the “program architecture”
• Classic program synthesis focuses on $\alpha$, with $\theta$ being very simple
Enumerating programs

Program enumeration is really a graph search problem.

- Choices on how to extend the program
- Partially completed program
- More complete program
Enumerating programs

Program enumeration is really a graph search problem
Estimating the “Cost to Go”

- \( P^* \) = partial program (non-terminal nodes)
- \( \mathbb{C}(P^*) \) = completions of \( P^* \) (reachable terminal nodes)

Heuristic Estimate:
\[
d(P^*) \approx \min_{P \in \mathbb{C}(P^*)} \left[ \Delta s(P, P^*) + \min_{\theta} \text{Loss}(\alpha_p, \theta_p) \right]
\]

- Additional Structure Cost
- Training Loss

If \( d(P^*) \) is a lower bound it becomes an “admissible heuristic”
Guiding program search

**Problem:** You only get ground truth on the leaves of the search tree
  - Value for an intermediate node is only an estimate

Can we get a better estimate with deep learning?

\[ s(P) + \min_{\theta} \text{Loss}(\alpha_P, \theta_P) \]
Motivating Observation/Assumption:
Functional Representational Power

“Neural Relaxation”:
Every DSL program can be (approximately) represented by some “large” neural model.
Implication
(abstract form)

\[ \forall P, \exists f \in F \text{ s.t. } d(f) \leq d(P) + \epsilon \]

“Neural Relaxation” Every DSL program can be (approximately) represented by some “large” neural model.
Informed Search (e.g., A*)

• Use $d(P^*)$ to prune the search

Suppose:

$\text{Structural Cost: } \text{Training Loss:}$

\[ s(\lambda x. \text{map} \ x \ g^*) + d(\lambda x. \text{map} \ x \ g^*) > s(\lambda x. x) + \text{Loss}(\lambda x. x) \]

“Cost to Go” Heuristic

\[ \lambda x. f^* \]

\[ \lambda x. x \]

\[ \lambda . \text{foldl} \ x (\lambda z y. h^*)[\ ] \]

\[ \lambda . [ ] \]

\[ \lambda . \text{map} \ x \ g \]

Can Prune This Branch!
**A* Search**

- Priority queue of current leaf nodes:
  - Sorted by $s(P^*) + d(P^*)$

- Pop off top program $P^*$
  - If $P^*$ is complete, terminate
  - Else, expand $P^*$, add child nodes to priority queue

---

**Guarantee:** if $d(P^*)$ is admissible, A* will return optimal $P$

- Tighter $d(P^*)$ prunes more aggressively
- Uninformed $d(P^*)$ (e.g., always 0) → uninformed search
NEAR: Neural Admissible Relaxations

If a large neural network cannot fit this hole, then a completion from the DSL also cannot.

Fill hole with NN
Train parameters
Use training loss as admissible heuristic
Stop any time!

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020
NEAR: Results

Order of magnitude speedup

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020
Other uses of relaxations

\[ f_\theta(x) = \sigma(x) + \nu_\theta(x) \]

**Relax:** Add a parameterized neural component to a program

**Update:** Gradient-based update to neural component
- Approximation to gradient in program space

**Distill:** Synthesize symbolic program closest to current neurosymbolic program

**Distillation:** \( \sigma = \arg\min_{\sigma} \text{BregmanDist}(\sigma, f) \)

Back to behavior analysis

How to describe “attack” behavior?

IF (mouse 1 & 2 acceleration) > A AND (mouse 1 & 2 velocity) < B THEN attack, ELSE not attack

1D Conv Net
F1: 0.86

Learned Program
F1: 0.84

Interpreting Expert Differences in Annotation Behavior. Tjandrasuwita, Sun, Kennedy, Chaudhuri, Yue. CV4Animals 2021.
Handling raw inputs

Use a complementary method (e.g., keypoints) to abstract images into symbolically interpretable features [Sun, Ryuou, et al., CVPR 2022]
Integration into existing tool (Bento)

Segalin, et al., eLife 2021
Extension to unsupervised learning

Variational autoencoders (VAEs)

Latent representations capture semantics of inputs

**In behavior analysis:**
- Cluster the representations
- Create new labels that capture the clusters

During training, maximize $\text{ELBO} := \mathbb{E}_{q_{\phi}(z|x)} \left[ \log p_{\theta}(x|z) \right] - D_{KL}(q_{\phi}(z|x) \| p(z)) \leq \log p(x)$
Neurosymbolic encoders

Cluster 0: The mice are further apart
• Second term is positive, negative product is less than the threshold.

Cluster 1: The mice are close together
• Second term is negative, product is positive.

Unsupervised Learning of Neurosymbolic Encoders. Zhan, Sun, Kennedy, Yue, & Chaudhuri. TMLR 2022
VAEs with neurosymbolic encoders
**Results** (on human-annotated behavior data)

More well-structured latent spaces

Comparable performance to expert-written programs in downstream tasks
What’s ahead?

Full-stack AI-aided science through neurosymbolic programming
Challenge: Scalability

Searching for program structures is fundamentally expensive.

Possible recipes:
- Large Language Models
- Parallelism
- ...

Challenge: Vocabulary discovery

Where does the DSL come from?

Possible recipe: Library learning

Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian learning. Ellis et al., 2021.
Challenge: Vocabulary Discovery

Possible recipe: Symbol discovery through vision-language models

Neurosymbolic Programming Everywhere!

Understanding the World Through Code
Funded through the NSF Expeditions in Computing Program

*LMQL*
[Vechev et al., 2023]

Scallop
[Naik et al., 2022]

[OpenAI Plugins, 2023]
December 5, 2022, 10 am Chicago time

Neurosymbolic Programming

Tutorial at Neural Information Processing Systems, 2022

Speakers

Swarat Chaudhuri
UT Austin

Armando Solar-Lezama
MIT

Jennifer J. Sun
Caltech

Panelists

Jeeyana Inala
Microsoft Research, Redmond

Ann Kennedy
Northwestern University

Pushmeet Kohli
Deepmind

Sriram Rajamani
Microsoft Research, India

Yisong Yue
Caltech
(Moderator)

https://neurips.cc/virtual/2022/tutorial/55804
Acknowledgements
Notebooks on neurosymbolic programming for science


Automated Programming & Reasoning

High-assurance Autonomy

Neurosymbolic Programming

Al for Scientific Discovery