Neurosymbolic Programming

*** Includes materials from: Armando Solar-Lezama, Osbert Bastani, Swarat Chaudhuri, Ann Kennedy, David Anderson

Yisong Yue

Caltech
Machine learning is transforming science

Halicin: structurally new antibiotic


Personalized Exoskeletons

http://roams.caltech.edu/

https://www.microsoft.com/en-us/research/project/crispr/

But something is missing...

Interpretability

Domain Knowledge

Correlation vs Causation
A revolution in formal methods
Program Synthesis

Behavioral Constraints
- Input/Output examples
- Safety properties
- Distribution over Behaviors

Structural Constraints
- Program components
- Program skeletons
- Prior over program structure

Program Synthesizer

Code
Scientific knowledge is code

\[ E = mc^2 \]
Scientific knowledge is code

Understanding Morpho-phonology

/word/ → [work]

Underlying form → Surface form

https://dspace.mit.edu/handle/1721.1/113870

Synthesis of biological models

https://dl.acm.org/doi/10.1145/2480359.2429125
Neurosymbolic Programs

Symbolic Programs
- Interpretable
- Verifiable
- Structured domain knowledge
- Data efficient

Neural Networks
- Scalable algorithms
- Flexible
- Handles messy data
- Easy to get started
Example in Behavior Analysis

**Goal:** Classify “sniff” action between two mice

\[
\text{map } (\text{fun } x_t. \\
\quad \text{if } \text{DistAffine}_{[.0217;-.2785]}(x_t) \\\n\quad \quad \text{then } \text{AccAffine}_{[-.0007,.0055,.0051,-.0025];3.7426}(x_t) \text{ else } \text{DistAffine}_{[-.2143;1.822]}(x_t) ) x
\]

learned in conjunction with program
Neurosymbolic learning isn’t new...

...but it’s a good time to push on it!

• Respective revolutions in both fields
  • Rapidly maturing tools

• New algorithms that can scale
  • Computation (e.g., neural-guided search)
  • Data (e.g., programmatic weak supervision)

• Demands by the domain experts & science applications
Neurosymbolic Models

Query model
Extract insights
Inject domain knowledge

Check model insights against data

Fit model to data
Suggest experiments

Closing the loop between data and insight

Domain Expert

sniff_genitali

other
sniff_face
sniff_genital
sniff_body

λ x.
The Basic Recipe

Program Structure

Inputs  Algebraic Operators  Parameterized Operators

\( \alpha ::= x \mid \oplus(\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta(\alpha_1, \ldots, \alpha_k) \)

if \( \alpha_1 \) then \( \alpha_2 \) else \( \alpha_3 \) | \( \text{sel}_S x \) | mapaverage (fun \( x_1.\alpha_1 \)) \( x \)

Domain Specific Language (DSL) -- “Family of programs”

Recall Earlier Example:

map (fun \( x_t. \))

if \( \text{DistAffine}_{[.0217];-.2785}(x_t) \)

then \( \text{AccAffine}_{[-.0007,.0055,.0051,-.0025];3.7426}(x_t) \) else \( \text{DistAffine}_{[-.2143];1.822}(x_t) \) \( x \)
The Basic Recipe

Domain Specific Language (DSL) -- “Family of programs”

Program Structure

\[
\alpha ::= \begin{aligned}
\alpha & \mid x \mid \oplus (\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta (\alpha_1, \ldots, \alpha_k) \\
\text{if } & \alpha_1 \text{ then } \alpha_2 \text{ else } \alpha_3 \mid \text{sel}_S x \mid \text{mapaverage} \ (\text{fun } x_1.\alpha_1) \ x
\end{aligned}
\]

Learning Objective (“Loss Function”)

Learning Algorithm (aka synthesis)

Neurosymbolic Program (\(\alpha, \theta\))

Downstream Analyses
Observations:

• Fixed program structure $\alpha \rightarrow \text{train } \theta$ via gradient descent

• 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

Example Program:

```
map (fun x_t.
  if DistAffine_{0.0217; -0.2785}(x_t)
  then AccAffine_{-0.0007, 0.055, 0.051, -0.0025; 3.7426}(x_t)
  else DistAffine_{-0.2143; 1.822}(x_t)) x
```
Remainder of Talk

Algorithm Vignette (Computational Scalability)

User Study Vignette (Interpretability)

Data Augmentation Vignette (Data Efficiency)
Top-Down Induction

Exponentially large search space!

Popular approaches (e.g., A*) require admissible heuristic
Motivating Observation/Assumption:
Functional Representational Power

“Neural Relaxation” Every neurosymbolic model can be (approximately) represented by some “large” neural model.
If a large neural network cannot fit this hole, then a program also cannot
NEAR: Neural Admissible Relaxations

Neural Relaxation as Admissible Heuristic!
Usable in any informed search (e.g., A*)
NEAR: Results

Order of magnitude speedup!

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020
Remainder of Talk

Algorithm Vignette
(Computational Scalability)

User Study Vignette (Interpretability)

Data Augmentation Vignette (Data Efficiency)
Behavior categorization & definitions are ambiguous!

[Diagram showing various behaviors and their categorization]

[Link to the original source: https://www.sciencedirect.com/science/article/pii/S0896627319308414]
Understanding annotator differences

https://arxiv.org/abs/2106.06114

Megan Tjandrasuwita, Jennifer J. Sun, Ann Kennedy, Swarat Chaudhuri, Yisong Yue
Remainder of Talk

Algorithm Vignette (Computational Scalability)

User Study Vignette (Interpretability)

Data Augmentation Vignette (Data Efficiency)
Data Augmentation, Self Supervision, Weak Supervision, etc...

Example: image transformations that preserve “meaning”

- (a) Original
- (b) Crop and resize
- (c) Crop, resize (and flip)
- (d) Color distort. (drop)
- (e) Color distort. (jitter)
- (f) Rotate {90°, 180°, 270°}
- (g) Cutout
- (h) Gaussian noise
- (i) Gaussian blur
- (j) Sobel filtering

Labeled data is expensive

Use augmentations to reduce label burden

Auxiliary Supervision via Programmed Decoding Tasks

"Task Programming"

Task Programming: Learning Data Efficient Behavior Representations,
Jennifer J. Sun, Ann Kennedy, Eric Zhan, David J. Anderson, Yisong Yue, Pietro Perona, CVPR 2021 **Best Student Paper Award**
Task Programming: Learning Data Efficient Behavior Representations,
Jennifer J. Sun, Ann Kennedy, Eric Zhan, David J. Anderson, Yisong Yue, Pietro Perona, CVPR 2021

*Follow-up Work:* Automatically Synthesizing Decoding Tasks via Unsupervised Program Learning

Can lead to 10x annotation efficiency!
Close Collaboration with Domain Experts

David Anderson (Caltech)

Ann Kennedy (Northwestern)
Neurosymbolic Programming

Swarat Chaudhuri  
UT Austin  
swarat@cs.utexas.edu

Kevin Ellis  
Cornell University  
kellis@cornell.edu

Rishabh Singh  
Google  
rising@google.com

Armando Solar-Lezama  
MIT  
asolar@csail.mit.edu

Oleksandr Polozov  
Microsoft Research  
polozov@microsoft.com

Yisong Yue  
Caltech  
yyue@caltech.edu

**ABSTRACT**

*Neurosymbolic programming* is an emerging research area at the interface of deep learning and program synthesis. Like in classic machine learning, the goal here is to learn functions from data. However, these functions are represented as programs that use symbolic primitives, often in conjunction with neural network components, and must, in some cases, satisfy certain additional behavioral constraints. The programs are induced using a combination of symbolic search and gradient-based optimization.

Neurosymbolic programming can offer multiple advantages over end-to-end deep learning. Programs can sometimes naturally represent long-horizon, procedural tasks that are difficult to perform using deep networks. Neurosymbolic representations are also, commonly, easier to interpret, analyze, and trust than neural networks.

The key idea here is to represent ML models as programs of the sort humans would write. Sometimes these programs are built entirely from symbolic primitives. Sometimes, they use a mix of symbolic code and neural modules. The learning problem in this area is to *simultaneously induce* a program’s symbolic and neural components, and this problem is solved using a mix of symbolic and statistical techniques. We call this literature *neurosymbolic programming (NsP)*, and this article is an introduction to this area.

Mathematically, a program $q_0$ in NsP consists of a program architecture $a$ that defines the way in which a program’s symbolic and neural modules are composed, and a vector $\theta$ of parameters of these modules. The program architectures are required to follow the syntax of a domain-specific language (DSL). The learning problem is to induce $a$ and $\theta$ so as to minimize an empirical loss function $L(a, \theta)$. ***Coming out soon!***
Neurosymbolic Programs
http://www.neurosymbolic.org/

Symbolic Programs
- Interpretable
- Verifiable
- Structured domain knowledge
- Data efficient

Neural Networks
- Scalable algorithms
- Flexible
- Handles messy data
- Easy to get started
References:

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020
• https://github.com/trishullab/near

Interpreting Expert Annotation Differences in Animal Behavior, Megan Tjandrasuwita et al., arXiv

Task Programming: Learning Data Efficient Behavior Representations, Jennifer J. Sun, et al., CVPR 2021 ***Best Student Paper Award
• https://sites.google.com/view/task-programming

Unsupervised Learning of Neurosymbolic Encoders, Eric Zhan*, Jennifer J. Sun*, et al., arXiv