Improving Policy Learning via Programmatic Domain Knowledge

Yisong Yue
Policy/Controller Learning (Reinforcement & Imitation)

**Goal:** Find “Optimal” Policy

**Imitation Learning:**
Optimize imitation loss

**Reinforcement Learning:**
Optimize environmental reward

**Learning-based Approach for Sequential Decision Making**

Non-learning approaches include: optimal control, robust control, adaptive control, etc.
Many Exciting Success Stories

AlphaFold

Microsoft Azure Personalizer
Data-Driven Ghosting

Data-Driven Ghosting using Deep Imitation Learning
Hoang Le et al. SSAC 2017

English Premier League 2012-2013

Match date: 04/05/2013

Blue: Defense
Red: Attack
White: Learning Policies

https://www.youtube.com/watch?v=WI-WL2cj0CA
Imitation Learning Tutorial
https://sites.google.com/view/icml2018-imitation-learning/

Yisong Yue
yyue@caltech.edu
@YisongYue
visongyue.com

Hoang M. Le
Hoang.Le@Microsoft.com
@HoangMinhLe
hoangle.info
Challenges with “Black Box” Policy Learning

Interpretability

“Formal” Interpretability

Sample Efficiency
https://arxiv.org/abs/1709.06560

Controllability

https://arxiv.org/abs/1611.00094

TOUCH

WING THREAT

TUSSLE

WING EXTENSION
Data vs Priors


Environmental Reward

Data Augmentation


Inductive Biases (“Priors”)

https://alexpolozov.com/blog/program-synthesis-2018/
Programmatic Domain Knowledge
aka Symbolic

• Domain Specific Language (DSL)

\[
\begin{align*}
\pi(s) & := a \mid Op(\pi_1(s), \ldots, \pi_k(s)) \mid \text{if } b \text{ then } \pi_1(s) \text{ else } \pi_2(s) \mid \oplus_\theta(\pi_1(s), \ldots, \pi_k(s)) \\
b & := \phi(s) \mid BOp(b_1, \ldots, b_k)
\end{align*}
\]

\[
\begin{align*}
\text{if } (s[\text{TrackPos}] < 0.011 \text{ and } s[\text{TrackPos}] > -0.011) \\
\quad \text{then } \text{PID}\{\text{rpm},0.45,3.54,0.03,53.39\}(s) \text{ else } \text{PID}\{\text{rpm},0.39,3.54,0.03,53.39\}(s)
\end{align*}
\]

\[
\alpha := x \mid c \mid \oplus(\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta(\alpha_1, \ldots, \alpha_k) \mid \text{if } \alpha_1 \text{ then } \alpha_2 \text{ else } \alpha_3 \mid \text{sel}_S x \\
\text{map}(\lambda x_1.\alpha_1) x \mid \text{fold}(\lambda x_1.\alpha_1) c x \mid \text{mapprefix}(\lambda x_1.\alpha_1) x
\]

\[
\text{map(multiply(add(OffenseAffine(x), BallAffine(x)), add(OffenseAffine(x), BallAffine(x))))}
\]

https://arxiv.org/abs/1907.05431
Programmatic Domain Knowledge

- Programmatic Priors
  - "Neurosymbolic" learning

- Programmatic Supervision
  - "Data Programming" for sequential behavioral data
Program Learning

\[ \pi^* = \arg\min_{\pi \in \Pi} J(\pi) \]

**Loss Function**

**Programmatic Policy Class**

**Challenges**

- Policy class has discrete structures
- Fully differentiable learning may not be possible

***Similar challenges in Neural Architecture Search***
Program “Architecture”

```plaintext
map(multiply(add(OffenseAffine(x), BallAffine(x)), add(OffenseAffine(x), BallAffine(x))))
```
Discrete Search for Program Synthesis

• A* Search
• Branch and Bound
• Local Pruning
• Transformations (e.g., simplify programs, identify isomorphisms, etc.)
• Oracle Calls to Satisfiability Sub-Problems
• Reinforcement Learning
• Etc...

Our Philosophy:
• Compatible with all of the above
• Maximally leverages deep learning
• General framing of problem setting
Motivating Observation/Assumption:
Functional Representational Power

“Neural Relaxation” Every programmatic policy can be (approximately) represented by some neural policy.
Implication
(abstract form)

Programmatic $\Pi$

Neural $H$

Slack due to approximation error or training ability

\[ \forall \pi \in \Pi, \exists h \in H \text{ s.t. } J(h) \leq J(\pi) + \epsilon \]

Every programmatic policy can be (approximately) represented by some neural policy.

“Neural Relaxation”
Setting 1: Top-Down Program Induction

- **Nodes**: (Partial) Programs
  - Goal nodes (sinks): complete programs

- **Edges**: Single expansion of program

- **Edge costs**: $s(v_{child}) - s(v_{parent})$
  - Cost of expansion
  - I.e., complexity measure
Setting 1: Top-Down Program Induction

- **Training Loss** included in edge costs to sink nodes (complete programs)

- Optimal search requires a complete traversal of the graph

  Often exponentially large!
  Popular approaches (e.g., A*) require admissible heuristic
NEAR: Neural Admissible Relaxations

• Neural completion of partial program
  • Sub-routine is a “black-box” neural net
  • “Neural Relaxation”

• Can evaluate complete programs
  • Lower loss: $J(\text{best neural}) \leq J(\text{best program}) + \epsilon$
  • Exactly an $\epsilon$-Admissible Heuristic!

• A* is guaranteed to find $\epsilon$-optimal program!
  • Also applies to Branch & Bound, etc.
Story So Far

• Programmatic Policy Class (Inductive Bias)

• Neural Relaxations of Programs

• Admissible Heuristic for Top-Down Induction

• What’s Next?
Recall: Challenges with “Black Box” Policy Learning

Interpretability

https://arxiv.org/abs/1611.00094

“Formal” Interpretability

https://arxiv.org/abs/1709.06560

Controllability

Sample-Efficient Policy Learning

• Deep Reinforcement Learning is notoriously data hungry...

  See: https://arxiv.org/abs/1709.06560

• ...but has been successful (in large part) due to combining stochastic policies with differentiable learning.

• How to enable this generally w/ programs?
Setting 2: Program Learning as Constrained Mirror Descent

\[ \pi^* = \operatorname{argmin}_{\pi \in \Pi} J(\pi) \]

\( \Pi \) is a constrained (approximate) subset of \( H \)!
Constrained Mirror Descent Recipe

- **Update**: gradient-based updates in unconstrained space $H$
  - “Trust Region” defined by divergence function (e.g., L2, KL, etc.)

- **Project**: project onto constrained space $\Pi$

- Repeat

![Diagram of constrained mirror descent process](image)
Useful Notation

• **Unconstrained space:** $H = \Pi \oplus F$

• **Unconstrained policy:** $h(s) = \pi(s) + f(s)$

---

PROPEL: Imitation-Projected Programmatic Reinforcement Learning

- **Unconstrained policy:** $h(s) = \pi(s) + f(s)$

- **Update:** hold $\pi$ fixed, update $f$
  - Can use any deep RL approach

- **Project:** train new $\pi$ to imitate $h$
  - Can use any program learning approach

- Theoretical analysis in paper

Aside: Model/Policy Distillation

• Examples:
  • Distill bigger network into smaller network
  • Distill neural net into program

• Most work focus on 1-shot distillation
  • Train more complex model/policy, then distill

• PROPEL suggests iterative distillation
  • Viewed through the lens of constrained mirror descent
Zero-Shot Generalization

<table>
<thead>
<tr>
<th>Training Track</th>
<th>Test Track</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>G-Track</strong></td>
<td><strong>E-Road</strong></td>
</tr>
<tr>
<td>G-Track</td>
<td>-</td>
</tr>
<tr>
<td>E-Road</td>
<td>102 / 92</td>
</tr>
<tr>
<td>Aalborg</td>
<td>201 / 91</td>
</tr>
<tr>
<td>Ruukskogen</td>
<td>131 / Cr</td>
</tr>
<tr>
<td>Alpine-2</td>
<td>222 / Cr</td>
</tr>
</tbody>
</table>

Reporting: PROPEL / DDPG in time to completion

**Table 1:**

<table>
<thead>
<tr>
<th></th>
<th>G-Track</th>
<th>E-Road</th>
<th>Aalborg</th>
<th>Ruukskogen</th>
<th>Alpine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-Track</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-Road</td>
<td>102 / 92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aalborg</td>
<td>201 / 91</td>
<td>228 / Cr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ruukskogen</td>
<td>131 / Cr</td>
<td>135 / Cr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpine-2</td>
<td>222 / Cr</td>
<td>231 / Cr</td>
<td>184 / Cr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2:**

<table>
<thead>
<tr>
<th></th>
<th>G-Track</th>
<th>E-Road</th>
<th>Aalborg</th>
<th>Ruukskogen</th>
<th>Alpine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-Track</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-Road</td>
<td>102 / 92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aalborg</td>
<td>201 / 91</td>
<td>228 / Cr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ruukskogen</td>
<td>131 / Cr</td>
<td>135 / Cr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpine-2</td>
<td>222 / Cr</td>
<td>231 / Cr</td>
<td>184 / Cr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Imitation-Projected Programmatic Reinforcement Learning,** Abhinav Verma, Hoang Le, et al., NeurIPS 2019
Recall: Challenges with “Black Box” Policy Learning

Interpretability

https://arxiv.org/abs/1611.00094

“Formal” Interpretability

https://arxiv.org/abs/1709.06560

Controllability

Certifiable Policy Learning

- Unconstrained space: \( H = \Pi \oplus F \)

- Partially constrained policy: \( h(s) = \pi(s) + \lambda f(s) \) \( \lambda \) small

- Suppose all \( \pi \in \Pi \) are “certified”
  - Then \( h \) is approximately certified!
Realtime Player Detection and Tracking

Human Operated Camera

FEATURES

FEATUR

TRAIN

PREDICT

Learned Regressor

Disney Research

Autonomous Robotic Camera
Naïve Approach

• Supervised learning of demonstration data
  • Train predictor per frame
  • Predict per frame

In practice, 2-step smoothing:
Provably Smooth Policies

Smooth Imitation Learning for Online Sequence Prediction
Hoang Le, Andrew Kang, Yisong Yue, Peter Carr. ICML 2016
Our Results

Smooth Imitation Learning for Online Sequence Prediction
Hoang Le, Andrew Kang, Yisong Yue, Peter Carr. ICML 2016
Qualitative Comparison

Our Approach

Learning Online Smooth Predictors for Real-time Camera Planning using Recurrent Decision Trees

Jianhui Chen, Hoang Le, Peter Carr, Yisong Yue, Jim Little. CVPR 2016

Disney using human operators to train automatic cameras for broadcasts
Control-Theoretic Guarantees

• (Relaxed) Lyapunov stability bounds:

Control Regularization for Reduced Variance Reinforcement Learning
Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, Joel Burdick. ICML 2019
Summary: Programmatic Inductive Bias

• Represent Policies as Programs
• Generalization / Sample-Efficiency
• Certifiable Guarantees
• Neural Relaxation Perspective
• New Learning Approaches!
Understanding the World Through Code
Funded through the NSF Expeditions in Computing Program
http://www.neurosymbolic.org/

PIs

Armando Solar-Lezama
Associate Professor, MIT

Swarat Chaudhuri
Associate Professor, UT Austin

Yisong Yue
Professor, Caltech

Regina Barzilay
Professor, MIT

Isil Dillig
Associate Professor, UT Austin

Osbert Bastani
Research Assistant Professor, University of Pennsylvania

Michael Carbin
Assistant Professor, MIT

Martin Rinard
Professor, MIT

Phillip Sharp
Institute Professor and Professor of Biology, MIT

Tommi Jaakkola
Professor, MIT

Noah Goodman
Associate Professor, Stanford

Chris Jermaine
Professor, Rice

Advisory Board

Josh Tenenbaum
Professor, MIT

Pushmeet Kohli
Principal scientist and research team leader, Deep Mind

Rishabh Singh
Research scientist, Google Brain

Justin Gottschlich
Head of Machine Programming Research, Intel

Satish Chandra
Engineering Manager, Facebook

Susmit Jha
Principal Computer Scientist, SRI International
Programmatic Domain Knowledge

- Programmatic Priors
  - "Neurosymbolic" learning
  - Program
  - Neural

- Programmatic Supervision
  - Small Distance
  - Large Distance
  - "Data Programming" for sequential behavioral data
Recall: Challenges with “Black Box” Policy Learning

Interpretability

https://arxiv.org/abs/1611.00094

“Formal” Interpretability

Sample Efficiency
https://arxiv.org/abs/1709.06560

Safe Set

Controllability
Style-Calibrated Generation

**Goal**: controllable generation of behaviors

Both trajectories move towards the basket
- Blue style is more aggressive
- Green style is more passive

Learning Calibratable Policies using Programmatic Style-Consistency, Eric Zhan et al., ICML 2020
Many Possible Styles

(a) Expert demonstrations (b) Style: SPEED (c) Style: DESTINATION (d) Both styles

Learning Calibratable Policies using Programmatic Style-Consistency
Eric Zhan, Albert Tseng, Yisong Yue, Adith Swaminathan, Matthew Hausknecht, ICML 2020
Key Challenges

**Challenge #1:** expert labels for behavior styles are hard to obtain

**Challenge #2:** joint style space scales combinatorially wrt. # styles

Our solution:

- Expert-written style-labeling functions (programmatic supervision)
- Style-consistency learning objective
Labeling Functions for Behavior Styles

Behavior styles via programmatic labeling functions: \( \lambda(\tau) = y \)

- e.g. small/large distance to basket

\[
\lambda(\tau) = 1\{\|s_{T+1} - s_{\text{basket}}\|_2 > b\}
\]

Advantages:
- Fast to compute
- Compactly encodes domain knowledge
- Readily available

“Data Programming” for sequential behavioral data
Style-Consistency

Want to directly optimize for style-consistency!
Style-Consistency Objective
(can be extended to multiple styles)

\[ \pi^* = \arg \min_\pi \mathbb{E}_{(\tau,y) \sim \mathcal{D}_\lambda} \left[ \mathcal{L}_{\text{imitation}}(\tau, \pi(\cdot | y)) \right] + \mathbb{E}_{y \sim p(y), \tau \sim \pi(\cdot | y)} \left[ \mathcal{L}_{\text{style}}(\lambda(\tau), y) \right] \]

Standard Imitation Learning

Style Consistency

Trajectory generated by \( \pi(\cdot | y) \) should be consistent with \( y \)

Example:
\[ \mathcal{L}_{\text{style}}(\lambda(\tau), y) = 1 \{ \lambda(\tau) \neq y \} \]

See paper for learning algorithm

Learning Calibratable Policies using Programmatic Style-Consistency
Eric Zhan, Albert Tseng, Yisong Yue, Adith Swaminathan, Matthew Hausknecht, ICML 2020
Experimental Results for Basketball

Style #1: [medium] distance to basket

Style #2: [slow, medium, fast] speed
Experimental Results for Basketball

<table>
<thead>
<tr>
<th>Model</th>
<th>2 styles</th>
<th>3 styles</th>
<th>4 styles</th>
<th>5 styles</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTVAE</td>
<td>0.71</td>
<td>0.58</td>
<td>0.50</td>
<td>0.37</td>
</tr>
<tr>
<td>CTVAE-info</td>
<td>0.69</td>
<td>0.58</td>
<td>0.51</td>
<td>0.32</td>
</tr>
<tr>
<td>CTVAE-mi</td>
<td>0.72</td>
<td>0.56</td>
<td>0.51</td>
<td>0.30</td>
</tr>
<tr>
<td>CTVAE-style</td>
<td>0.93</td>
<td>0.88</td>
<td>0.88</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Joint style space of \(3^5 = 243\) joint styles!

Median style-consistency over 5 seeds, 4000 trajectories each (can interpret as accuracy)

- Using CTVAE as base imitation learner
- Baselines use “indirect” learning objectives
- Evaluate on matching all styles simultaneously

Learning Calibratable Policies using Programmatic Style-Consistency
Eric Zhan, Albert Tseng, Yisong Yue, Adith Swaminathan, Matthew Hausknecht, ICML 2020
Another Motivation: Strategy vs Tactics

• Long-term Goal:
  • Curl around basket

• Tactics
  • Drive left w/ ball
  • Pass ball
  • Cut towards basket
Generative + Hierarchical

- Generative Imitation Learning
  - No single “correct” action

- Hierarchical
  - Make predictions at multiple resolutions

Generating Long-term Trajectories using Deep Hierarchical Networks

Generating Multi-agent Trajectories using Programmatic Weak Supervision
Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, Patrick Lucey. ICLR 2019

NAOMI: Non-Autoregressive Multiresolution Sequence Imputation
Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, Yisong Yue. NeurIPS 2019
Simulation Results

Generating Multi-agent Trajectories using Programmatic Weak Supervision
Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, Patrick Lucey. ICLR 2019

https://www.youtube.com/watch?v=Oq1j22yMipY
Another Project: Programs as Auxiliary Supervision

- Biggest bottleneck in behavior modeling is annotation

https://github.com/annkennedy/bento
Task Programming

Can lead to 10x annotation efficiency!

Task Programming: Learning Data Efficient Behavior Representations,
Jennifer J. Sun, Ann Kennedy, Eric Zhan, David J. Anderson, Yisong Yue, Pietro Perona, CVPR 2021
Summary: Programmatic Supervision

• Expert-written labeling functions
  • (Ongoing work: automatically discovered)

• Programmatic supervision

• New learning objectives

• Controllable generation

• Annotation-efficient downstream tasks
Programmatic Domain Knowledge

Many cool applications!

Data
- Supervised
- Unsupervised
- Self-Supervised

Priors
- Bayesian
- Model Architecture
- Physics

Sequential Decision Making

Programmatic
References

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

Imitation-Projected Programmatic Reinforcement Learning, Abhinav Verma, Hoang Le, et al., NeurIPS 2019
• https://bitbucket.org/averma8053/propel/src/master/

Smooth Imitation Learning for Online Sequence Prediction, Hoang Le, et al., ICML 2016
• https://sites.google.com/view/smooth-imitation-learning


Control Regularization for Reduced Variance Reinforcement Learning, Richard Cheng et al., ICML 2019
• https://github.com/rcheng805/CORE-RL

Learning Calibratable Policies using Programmatic Style-Consistency, Eric Zhan et al., ICML 2020
• https://github.com/ezhan94/calibratable-style-consistency
• http://basketball-ai.com/


Generating Multi-agent Trajectories using Programmatic Weak Supervision, Eric Zhan et al., ICLR 2019
• https://github.com/ezhan94/multiagent-programmatic-supervision

NAOMI: Non-autoregressive Multiresolution Sequence Imputation, Yukai Liu et al., NeurIPS 2019
• https://github.com/felixykliu/NAOMI

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