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

Protein Sequence
SQTTRKCTEMKFFKNCVRCDESNHCVYRCSDTKYTLC

Neural Network
Databases

Distance Predictions
Angle Predictions

Score
Gradient Descent

Structure

Microsoft Azure Personalizer

Your App

Great Videos!

User & Context Info

Reward Score

Video 1 info

Video 2 info

Video 3 info
Data-Driven Ghosting using Deep Imitation Learning
Hoang Le et al.  SSAC 2017

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  
yisongyue.com

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

Interpretability

https://arxiv.org/abs/1611.00094

Sample Efficiency

https://arxiv.org/abs/1709.06560

Controllability


“Formal” Interpretability
Data vs Priors


![Data Augmentation](https://arxiv.org/abs/2002.05709)

Environmental Reward

![Inductive Biases (“Priors”)](https://alexpolozov.com/blog/program-synthesis-2018/)

---

***Modulo Optimization Concerns***
Programmatic Domain Knowledge
aka Symbolic

• Domain Specific Language (DSL)

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

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

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

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

https://arxiv.org/abs/1907.05431

Programmatic Domain Knowledge

- Programmatic Priors

- Programmatic Supervision

“Neurosymbolic” learning

“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***
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

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

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

Programmatic

Neural

Slack due to approximation error or training ability

Programmatic II

Neural H

Motivating Observation/Assumption:
Functional Representational Power

Loss Function
(Classification/Regression/Imitation/Reinforcement/etc.)
Setting 1: Top-Down Program Induction

• Nodes: (Partial) Programs
  • Goal nodes (sinks): complete programs

• Edges: Single expansion of program

• Edge costs: $s(\nu_{child}) - s(\nu_{parent})$
  • 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.

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah, Eric Zhan, et al., arXiv
Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah, Eric Zhan, et al., arXiv
Story So Far

• Programmatic Policy Class (Inductive Bias)

• Neural Relaxations of Programs

• Admissible Heuristic for Top-Down Induction

• What’s Next?

NEAR: Neural Admissible Relaxations Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah, Eric Zhan, et al., arXiv

“Neural Relaxation”

Can evaluate complete programs

Lower loss: $! \leq !, < \%

Exactly an Admissible Heuristic!

A* is guaranteed to find $!$-optimal program!

Also applies to Branch & Bound, etc.
Recall: Challenges with “Black Box” Policy Learning

Interpretability

https://arxiv.org/abs/1611.00094

“Formal” Interpretability

Sample Efficiency

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

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

\( \Pi \) is a constrained (approximate) subset of \( H \)!

Constrained Mirror Descent Recipe

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

• **Project**: project onto constrained space $\Pi$
Useful Notation

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

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

_Purely Neural Representation_

Motivating Observation/Assumption: Functional Representational Power

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

_Imitation-Projected Programmatic Reinforcement Learning_, Abhinav Verma, Hoang Le, et al., NeurIPS 2019
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>G-Track</th>
<th>E-Road</th>
<th>Aalborg</th>
<th>Ruudskogen</th>
<th>Alpine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-Track</td>
<td>-</td>
<td>124 / Cr</td>
<td>Cr / Cr</td>
<td>Cr / Cr</td>
<td>Cr / Cr</td>
</tr>
<tr>
<td>E-Road</td>
<td>102 / 92</td>
<td>-</td>
<td>Cr / Cr</td>
<td>Cr / Cr</td>
<td>Cr / Cr</td>
</tr>
<tr>
<td>Aalborg</td>
<td>201 / 91</td>
<td>228 / Cr</td>
<td>-</td>
<td>217 / Cr</td>
<td>Cr / Cr</td>
</tr>
<tr>
<td>Ruudskogen</td>
<td>131 / Cr</td>
<td>135 / Cr</td>
<td>Cr / Cr</td>
<td>-</td>
<td>Cr / Cr</td>
</tr>
<tr>
<td>Alpine-2</td>
<td>222 / Cr</td>
<td>231 / Cr</td>
<td>184 / Cr</td>
<td>Cr / Cr</td>
<td>-</td>
</tr>
</tbody>
</table>

Reporting: PROPEL / DDPG in time to completion

---

**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: $h(s) = \pi(s) + \lambda f(s)$ \quad $\lambda$ small

• Suppose all $\pi \in \Pi$ are "certified"
  • Then $h$ is approximately certified!
Realtime Player Detection and Tracking

Human Operated Camera

FEATURES

TRAIN

PREDICT

Learned Regressor

Autonomous Robotic Camera

Disney Research
Naïve Approach

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

In practice, 2-step smoothing:
Provably Smooth Policies

Smooth Complex Predictors \( \Pi \)

Complex Predictors \( H \)

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

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

Our Approach

Disney using human operators to train automatic cameras for broadcasts

2-Step Baseline

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

Our Approach

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

- (Relaxed) Lyapunov stability bounds:

![Diagram showing high and low regularization]

**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

People

PIs

Armando Solar-Lezama
Associate Professor, MIT

Swarat Chaudhuri
Associate Professor, UT Austin

Yisong Yue
Assistant 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

http://www.neurosymbolic.org/
Programmatic Domain Knowledge

• Programmatic Priors

“Neurosymbolic” learning

• Programmatic Supervision

“Data Programming” for sequential behavioral data

Recall: Challenges with “Black Box” Policy Learning

Interpretability

https://arxiv.org/abs/1611.00094

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

Controllability

"Formal" Interpretability
Style-Calibrated Generation

**Goal:** controllable generation of behaviors

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

![Diagram](image)

*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-Conditioned Imitation Learning

Programmatically labeled dataset:

\[ \mathcal{D}_\lambda = \{ (\tau_i, y_i) \}_{i=1}^N, \text{ where } y_i = \lambda(\tau_i) \]

Standard imitation learning:

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

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

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

Standard Imitation Learning  Style Consistency

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

See paper for learning algorithm

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

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><strong>0.93</strong></td>
<td><strong>0.88</strong></td>
<td><strong>0.88</strong></td>
<td><strong>0.75</strong></td>
</tr>
</tbody>
</table>

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

Joint style space of $3^{5} = 243$ joint styles!
Demo

http://www.basketball-ai.com
Another Motivation: **Strategy vs Tactics**

- **Long-term Goal:**
  - Curl around basket

- **Tactics**
  - Drive left w/ ball
  - Pass ball
  - Cut towards basket

---

**Abstract**

We study the problem of modeling spatiotemporal trajectories over long time horizons using expert demonstrations. For instance, in sports, agents often choose action sequences with long-term goals in mind, such as achieving a certain strategic position. Conventional policy learning approaches, such as those based on Markov decision processes, generally fail at learning cohesive long-term behavior in such high-dimensional state spaces, and are only effective when fairly myopic decision-making yields the desired behavior. The key difficulty is that conventional models are "single-scale" and only learn a single state-action policy. We instead propose a hierarchical policy class that automatically reasons about both long-term and short-term goals, which we instantiate as a hierarchical neural network. We showcase our approach in a case study on learning to imitate demonstrated basketball trajectories, and show that it generates significantly more realistic trajectories compared to non-hierarchical baselines as judged by professional sports analysts.

---

**1 Introduction**

Figure 1: The player (green) has two macro-goals: 1) pass the ball (orange) and 2) move to the basket.

Modeling long-term behavior is a key challenge in many learning problems that require complex decision-making. Consider a sports player determining a movement trajectory to achieve a certain strategic position. The space of such trajectories is prohibitively large, and precludes conventional approaches, such as those based on simple Markovian dynamics. Many decision problems can be naturally modeled as requiring high-level, long-term macro-goals, which span time horizons much longer than the timescale of low-level micro-actions (cf. He et al. [8], Hausknecht and Stone [7]). A natural example for such macro-micro behavior occurs in spatiotemporal games, such as basketball where players execute complex trajectories. The micro-actions of each agent are to move around the court and, if they have the ball, dribble, pass or shoot the ball. These micro-actions operate at the centisecond scale, whereas their macro-goals, such as "maneuver behind these 2 defenders towards the basket", span multiple seconds. Figure 1 depicts an example from a professional basketball game, where the player must make a sequence of movements (micro-actions) in order to reach a specific location on the basketball court (macro-goal).

Intuitively, agents need to trade-off between short-term and long-term behavior: often sequences of individually reasonable micro-actions do not form a cohesive trajectory towards a macro-goal. For instance, in Figure 1 the player (green) takes a highly non-linear trajectory towards his macro-goal of positioning near the basket. As such, conventional approaches are not well suited for these settings, as they generally use a single (low-level) state-action policy, which is only successful when myopic or short-term decision-making leads to the desired behavior.
• **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=0q1j22yMipY
Summary: Programmatic Supervision

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

• Programmatic supervision

• New learning objectives

• Controllable generation
Programmatic Domain Knowledge

Many cool applications!
References

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

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


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