Imitation-Projected Programmatic Reinforcement Learning



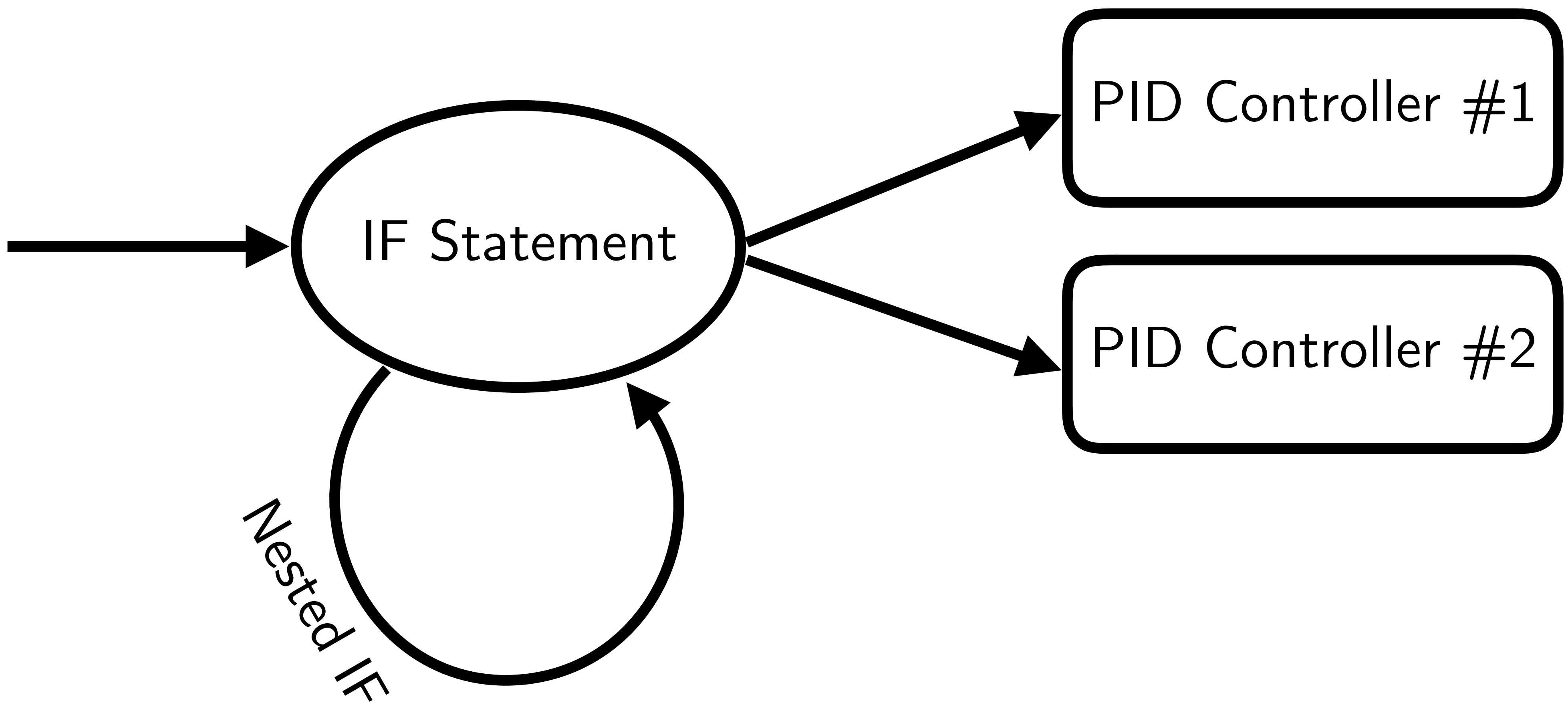

Abhinav Verma* *Rice University*
Hoang M. Le* *Caltech*
Yisong Yue *Caltech*
Swarat Chaudhuri *Rice University*

(equal contribution)*

33rd Conference on Neural Information Processing Systems (NeuRIPS 2019)

Why learning programmatic policies

- Natural way to encode structure into policy class
- Benefits: policy-based guarantee, symbolic verification of programs (& interpretable)
- Example: domain-specific language over a family of parameterized simple controllers



Programmatic reinforcement learning

- The program space Π
  - language (arithmetic, boolean, relational) over simple policies

- **Goal:** find the best program

$$\pi^* = \text{argmin}_{\pi \in \Pi} J(\pi)$$

- Learning programmatic policies is challneging:
  - Program Synthesis: highly structured nature of policy space
  - Deep RL: programmatic policy representation incompatible with deep policy search

- **Approach Overview:**

Building program structure into policy search via "lift-and-project"

Making use of hybrid representation

- Neural policy class F : deep RL representation
  - flexible, but unstable and does not satisfy desired property

- Programmatic policy class Π
  - less flexible, but certifiable

- Hybrid representation (functional regularization)

$$\mathrm{H} \equiv \Pi \oplus \mathrm{F}$$

$h \equiv \pi + \lambda f$ defined as $h(x) = \pi(x) + \lambda f(x)$

Imitation-**pro**jected **p**rogrammatic **re**inforcement **l**earning (PROPEL)

hybrid class: $\mathrm{H} \equiv \Pi \oplus \mathrm{F}$

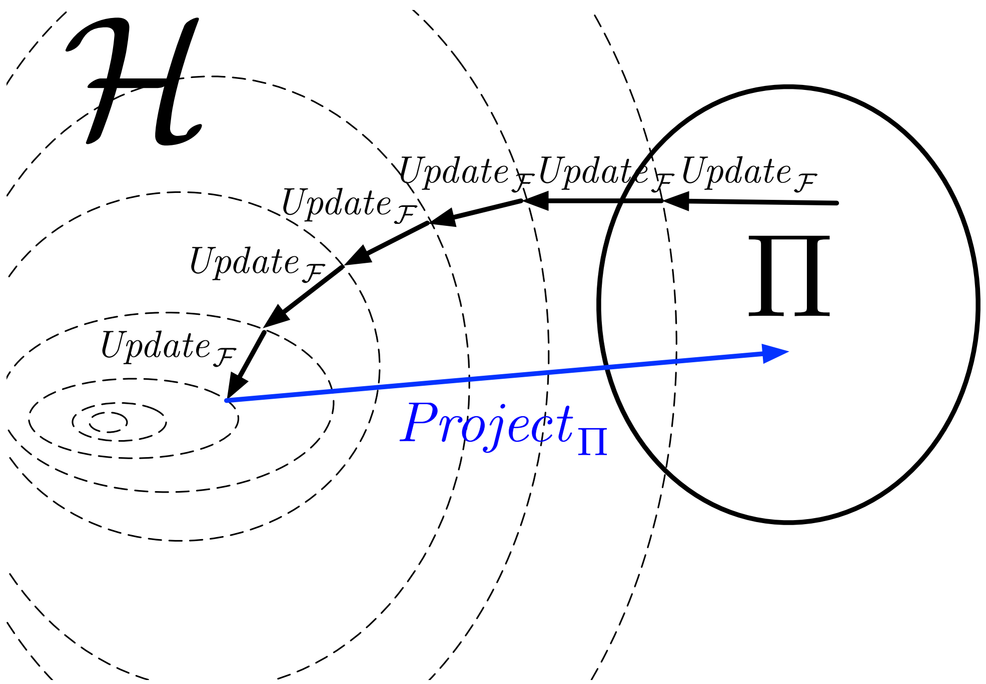
each iteration: $h_t \leftarrow \text{UPDATE}_{\mathrm{F}}(\pi_{t-1})$

$\pi_t \leftarrow \text{PROJECT}_{\Pi}(h_t)$

UPDATE: $f \leftarrow f - \eta\lambda\nabla_{\mathrm{F}} J(\pi + \lambda f)$

$h \leftarrow \pi + \lambda f$

PROJECT: imitation learning



Analysis: approximate (functional) mirror descent perspective

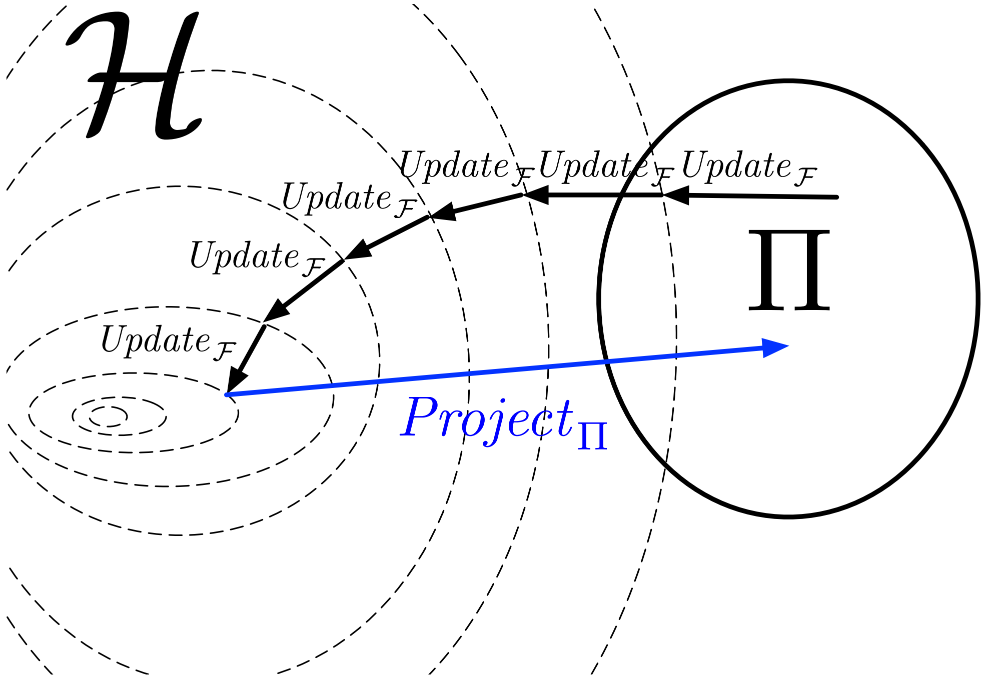
hybrid class: $\mathrm{H} \equiv \Pi \oplus \mathrm{F}$

each iteration: $h_t \leftarrow \text{UPDATE}_{\mathrm{F}}(\pi_{t-1}) \approx \text{UPDATE}_{\mathrm{H}}(\pi_{t-1})$

$\pi_t \leftarrow \text{PROJECT}_{\Pi}(h_t) \approx \operatorname{argmin}_{\pi \in \Pi} ||\pi - h_t||^2$

UPDATE: $f \leftarrow f - \eta\lambda \nabla_{\mathrm{F}} J(\pi + \lambda f)$

$h \leftarrow \pi + \lambda f$



$\text{UPDATE}_{\mathrm{H}}(\pi_{t-1}) = \pi_{t-1} - \nabla_{\mathrm{H}} J(\pi_{t-1})$

PROJECT_{Π} : finite sample error

Experiment in TORCS

Program space: decision tree with simple, parameterized PID controllers at leaf nodes

"if the car is aligned with the axis of the track..."

$$\begin{aligned}&\textbf{if } (\mathbf{obs}_{\texttt{TrackPos}}(0) < 0.001 \textbf{ and } \mathbf{obs}_{\texttt{TrackPos}}(0) > -0.001)\\ &\quad \textbf{then } PID_{\texttt{rpm}}(0.44, 4.92, 0.89, 49.79)\\ &\quad \textbf{else } PID_{\texttt{rpm}}(0.40, 4.92, 0.89, 49.79)\end{aligned}$$

"then accelerate, otherwise slow down"

Experiment in TORCS - Results

Performance: Lap time in seconds / Crash-ratio (over 25 seeds, lower is better)

Length	G-Track 3186m	E-Road 3260m	Aalborg 2588m	Ruudskogen 3274m	Alpine-2 3774m
Prior	312.92 / 0.0	322.59 / 0.0	244.19 / 0.0	340.29 / 0.0	402.89 / 0.0
Ddpg	78.82 / 0.24	89.71 / 0.28	101.06 / 0.4	Cr / 0.68	Cr / 0.92
Ndps	108.25 / 0.24	126.80 / 0.28	163.25 / 0.4	Cr / 0.68	Cr / 0.92
Viper	83.60 / 0.24	87.53 / 0.28	110.57 / 0.4	Cr / 0.68	Cr / 0.92
PropelProg	93.67 / 0.4	119.17 / 0.4	147.28 / 0.12	124.58 / 0.16	256.59 / 0.16
PropelTree	78.33 / 0.4	79.39 / 0.4	109.83 / 0.16	118.80 / 0.24	236.01 / 0.36

Generalization: PROPEL completed 10/20 unseen tracks, DDPG completed 2/20

	G-Track	E-Road	Aalborg	Ruudskogen	Alpine-2
G-Track	-	124 / Cr	Cr / Cr	Cr / Cr	Cr / Cr
E-Road	102 / 92	-	Cr / Cr	Cr / Cr	Cr / Cr
Aalborg	201 / 91	228 / Cr	-	217 / Cr	Cr / Cr
Ruudskogen	131 / Cr	135 / Cr	Cr / Cr	-	Cr / Cr
Alpine-2	222 / Cr	231 / Cr	184/ Cr	Cr / Cr	-

Code available at: https://bitbucket.org/averma8053/propel