Seminar: Few-Shot Bayesian Imitation Learning with Logical Program Policies

Yu-Zhe Shi

May 3, 2020

Auther Profiles

- Joshua B. Tenenbaum, Prof, The Computational Cognitive Science Group, MIT.http://web.mit.edu/cocosci/josh.html
- Tom Silver, PhD, The Computational Cognitive Science Group, MIT.http://web.mit.edu/tslvr/www/index.html
- Kesley Allen, PhD, The Computational Cognitive Science Group, MIT.https://web.mit.edu/krallen/www/
- Alex Lew, PhD, The Computational Cognitive Science Group, MIT.http://alexlew.net/
- Leslie Kaelbling, Prof, Computer Science and Engineering, MIT.https://www.csail.mit.edu/person/leslie-kaelbling

Overview

Algorithm 1: `LPP` imitation learning

input: Demos $\mathcal{D}$, ensemble size K, max iters L
Create anti-demos $\overline{\mathcal{D}} = \{(s, a') : (s, a) \in \mathcal{D}, a' \neq a\}$;
Set labels $y[(s, a)] = 1$ if $(s, a) \in \mathcal{D}$ else 0;
Initialize approximate posterior q;
for i *in* $1, ..., L$ **do**
  f_i = generate_next_feature();
  $X = \{(f_1(s, a), ..., f_i(s, a))^T : (s, a) \in \mathcal{D} \cup \overline{\mathcal{D}}\}$
  μ_i, w_i = logical_inference(X, y, p(f), K);
  update_posterior(q, μ_i, w_i);
end
return q;

Dilemma of Imitation Learning

- Behavior Cloning: Overfitting, underconstrained policy class and weak prior.
- Policy logical learning: need hand-crafted predicates, poor scalability.
- Program synthesis: Large search space.

Logical Program Policies

- "Top": Logical structure.
- "Bottom": Domain specific language expressions.
- Logically generate infinite policy classes from small scale DSL and score the candidates with likelihood and prior to prune searching space.

Prerequisites

- Objective: Given demo $\mathcal{D}$, learn policies $p(\pi|\mathcal{D})$
- $\mathcal{D} = (s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$, states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$
- Markov Process: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \mathcal{G})$ where $\mathcal{G} \subset \mathcal{S}$ is goal states, $T(s'|s, a)$ is transition distribution.
- State-conditional distribution over actions:

$$\pi(a|s) \in \Pi \tag{1}$$

  where Π is hypothesis classes.
- We learn π^* which is optimal to $\mathcal{M}$.

Policy classes

- Want to learn State-action classifiers:

$$h : \mathcal{S} \times \mathcal{A} \to \{0, 1\} \quad (2)$$

- $h(s, a) = 0$ action a never takes place when s.
- $h(s, a) = 1$ action a may takes place when s.
- $\pi(a|s) \propto h(s, a)$
- $\pi(a|s) \propto 1$ when $\forall a, h(s, a) = 0$

Bottom Level: Invent Predicates by Domain Specific Language

- Bottom Level: feature detection functions $f \in \mathcal{H} : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$.
- Input: s, a.
- Output: Binary decision of wether a should take place when s.

Top Level: Disjunctive Normal Form

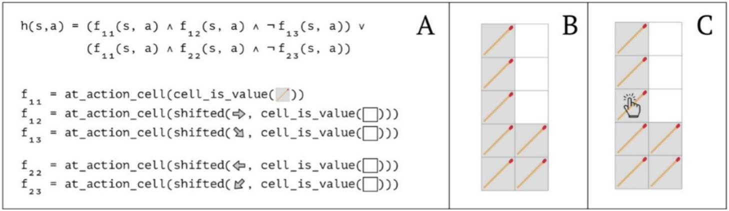
- $$h(s, a) = \vee_{i=1}^{m} (\wedge_{j=1}^{n_i} f_{i,j}(s, a)) \tag{3}$$
- $$h(s, a) = \vee_{i=1}^{m} \left(\wedge_{j=1}^{n_i} f_{i,j}(s, a)^{b_{i,j}} \left(1 - f_{i,j}(s, a)\right)^{1-b_{i,j}} \right) \tag{4}$$

  where $b_{i,j}$ determines whether $f_{i,j}$ is negated.

DSL

Method	Type	Description
cell_is_value	$V \to C$	Check whether the attended cell has a given value
shifted	$O \times C \to C$	Shift attention by an offset, then check a condition
scanning	$O \times C \times C \to C$	Repeatedly shift attention by the given offset, and check which of two conditions is satisfied first
at_action_cell	$C \to P$	Attend to the action cell and check a condition
at_cell_with_value	$V \times C \to P$	Attend to a cell with the value and check condition

Example of LPP



Imitation Learning

- Prior distribution of π over LPP:

$$p(\pi) \propto \prod_{i=1}^{m} \prod_{j=1}^{n_i} p(f_{i,j}) \tag{5}$$

  where $p(f)$ is a probabilistic context-free grammar, indicating how likely different rewritings are. The intuition is that we want to encode the prior with fewer and simpler fs.

- Likelyhood $p(\mathcal{D}|\pi)$ indicates the probabilistic of generating a demo $\mathcal{D}$ from policies π.

$$p(\mathcal{D}|\pi) \propto \prod_{i=1}^{n} \prod_{j=1}^{T_i} \pi(a_{ij}|s_{ij}) \tag{6}$$

p(f)

Production rule	Probability
Programs	
P → `at_cell_with_value` (V, C)	0.5
P → `at_action_cell` (C)	0.5
Conditions	
C → `shifted` (O, B)	0.5
C → B	0.5
Base conditions	
B → `cell_is_value` (V)	0.5
B → `scanning` (O, C, C)	0.5
Offsets	
O → (N, 0)	0.25
O → (0, N)	0.25
O → (N, N)	0.5
Numbers	
N → $\mathbb{N}$	0.5
N → $-\mathbb{N}$	0.5
Natural numbers (for $i = 1, 2, \ldots$)	
$\mathbb{N} \rightarrow i$	$(0.99)(0.01)^{i-1}$
Values (for each value v in this game)	
V → v	1/|V|

Approximate the posterior

- q is a weighted mixture of K policies $\mu_1, \ldots, \mu_K$

$$q(\pi) \approx p(\pi|\mathcal{D}) \tag{7}$$

- Minimize KL divergence $D_{KL}(q(\pi)|p(\pi|\mathcal{D}))$

$$q(\mu_j) = \frac{p(\mu_i|\mathcal{D})}{\sum_{i=1}^{K} p(\mu_i|\mathcal{D})} \tag{8}$$

Training Algorithm

1. Given a set of demos $\mathcal{D}$ where $h(s, a) = 1$.
2. Generate negative samples

$$\overline{\mathcal{D}} = \{(s, a')|(s, a) \in \mathcal{D}, a \neq a'\} \tag{9}$$

3. At iteration i, we use i simplist (i.e. of highest probability under p(f)) feature detectors $f_1, \ldots, f_i$ converting (s, a) into

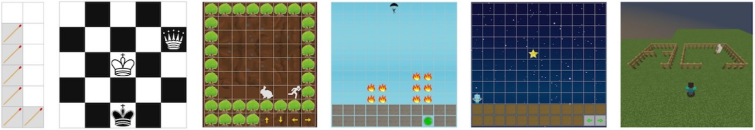
$$\mathbf{x} \in \{0, 1\}^i = (f_1(s, a), \ldots, f_i(s, a))^T \tag{10}$$

4. A stochastic greedy decision-tree learner to learn a binary classifier $h(s, a)$.
5. Induce a candidate policy $\mu_*(a|s) \propto (s, a)$, calculate $p(\mu_*), p(\mathcal{D}|\mu_*)$ to decide whether to include μ_* into the mixture q.

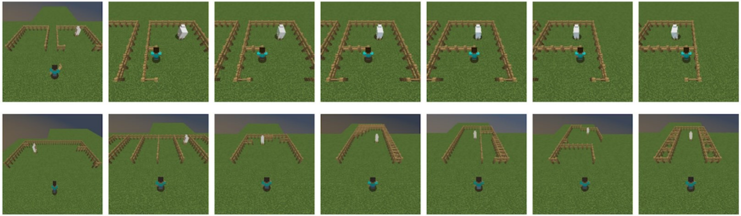
Inference

- $$\pi_*(s) = \arg_{a \in \mathcal{A}} \max \mathbb{E}_q[\pi(a|s)] = \arg_{a \in \mathcal{A}} \max \sum_{\mu \in q} q(\mu)\mu(a|s) \tag{11}$$

Experiments

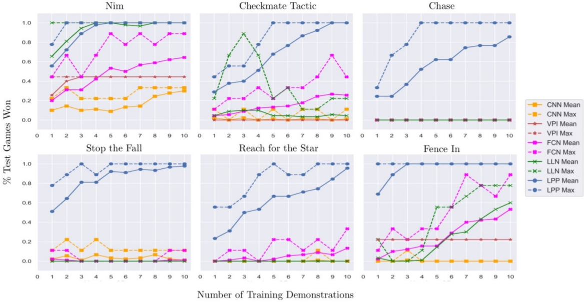


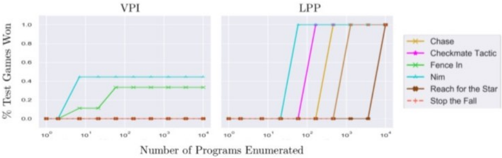
Nim | Checkmate Tactic | Chase | Stop the Fall | Reach for the Star | Fence In



Experiments: Baseline Comparison

- Local Linear Network, Fully Connected Network, CNN: trained to classify whether each cell should be clicked based on 8 surrounding cells. Vanilla Program Induction: Policy Learning with brute force.





Ablation Study

	Nim	CT	Chase	STF	RFTS	Fence
LPP	1.0	1.0	1.0	1.0	1.0	1.0
Features + NN	1.0	0.67	0.0	0.0	0.22	0.67
Features + NN + L_1 Reg	1.0	0.11	0.0	0.0	0.0	0.0
No Prior	1.0	0.44	0.78	1.0	1.0	1.0
Sparsity Prior	1.0	0.78	1.0	0.78	1.0	1.0

Summary and Inspiration

- Logical Program Policies: Reduce predicate invention into binary classification.
- Shared Domain Specific Language serves as meta-feature.
- Bayesian Imitation Learning: exploits probabilistic context-free grammar as priori to approximate posterior.