A Joint Planning and Learning Framework for Human-Aided Decision-Making

Daoming Lyu[1], **Fangkai Yang**[2], **Steven Gustafson**[3], **Bo Liu**[1] *
[1] Auburn University, Auburn, AL, USA
[2] NVIDIA Corporation, Redmond, WA, USA
[3] Maana Inc., Bellevue, WA, USA
daoming.lyu@auburn.edu, fangkaiy@nvidia.com, steven.gustafson@gmail.com, boliu@auburn.edu

Abstract

Conventional reinforcement learning (RL) allows an agent to learn policies via environmental rewards only, with a long and slow learning curve, especially at the beginning stage. On the contrary, human learning is usually much faster because prior and general knowledge and multiple information resources are utilized. In this paper, we propose a **P**lanner-**A**ctor-**C**ritic architecture for hu**MAN**-centered planning and learning (**PACMAN**), where an agent uses prior, high-level, deterministic symbolic knowledge to plan for goal-directed actions. PACMAN integrates Actor-Critic algorithm of RL to fine-tune its behavior towards both environmental rewards and human feedback. To the best our knowledge, This is the first unified framework where knowledge-based planning, RL, and human teaching jointly contribute to the policy learning of an agent. Our experiments demonstrate that PACMAN leads to a significant jump-start at the early stage of learning, converges rapidly and with small variance, and is robust to inconsistent, infrequent, and misleading feedback.

Introduction

A longstanding goal of artificial intelligence is to enable the programming agent to perform tasks intelligently in a complex domain. Recently, reinforcement learning (RL) algorithms, such as DQN, have made a lot of success on training agent to play Atari games from raw pixel images (Mnih et al. 2015). However, this approach is criticized for being "data-hungry" and "time-hungry", although it can learn fine granular policies that surpass human experts. Such drawbacks are heavily related to two phenomena that have not drawn enough attention before. The first phenomenon is that conventional RL algorithms can only learn from the reward signal, thus limiting its capability of utilizing multiple information resources. In general, humans can learn from multiple resources, and supervised learning that can learn from multiple labels simultaneously in the vector space. On the contrary, RL algorithms can only learn from scalar reward signals by interacting with the environment, i.e., one scalar reward signal per iteration. It would be beneficial to learn from multiple resources beyond merely environmental rewards, such as human feedback. The second phenomenon involves the initial learning phase. Existing RL algorithms usually learn from scratch, and thus often lead to an inferior performance at the initial learning phase (Sutton and Barto 2018), with a very long and slow learning curve. It would be helpful if a certain amount of prior knowledge can be incorporated in advance to help improve the initial learning dynamics.

There have been some studies, along with the two aforementioned research directions. The first topic leads to the proposal of "human-centered reinforcement learning" (HCRL) where an agent learns directly from human feedback. These approaches include interpreting human feedback as a shaping reward (Knox and Stone 2009), applying human feedback directly to policy improvement (Thomaz and Breazeal 2008; Knox and Stone 2010; Griffith et al. 2013), or interpreting human feedback as an estimation of the advantage function $A^\pi(s,a)$ [1] (MacGlashan et al. 2016; 2017). However, all the aforementioned work is limited to learning human feedback only, without considering the environmental rewards. The second topic, learning from human's prior knowledge, is investigated within a rather limited scope. To the best of our knowledge, the most relevant research to this topic is learning from demonstrations (LfD) (Argall et al. 2009), where the original policy search problem is reduced to a supervised distribution matching problem by matching the expert's demonstration trajectory's distribution. A typical strategy is to apply LfD first to obtain a good initial policy and then use RL methods for further policy improvement and refinement. The alternative approach, *learning from explicitly represented, formal, symbolic knowledge* is neglected until recently. Symbolic knowledge is used to capture coarse-granular domain dynamics and provide general guidance to exploration, leading to jump-start at the early stage of learning. After that, RL is used to fine-tune further the performance (Leonetti, Iocchi, and Stone 2016; Yang et al. 2018; Lyu et al. 2019; Jiang et al. 2019), which can significantly improve the sample-efficiency.

In this paper, we argue that prior knowledge, learning from environmental rewards, and human teaching should

*Correspondence to: Bo Liu<boliu@auburn.edu>.

[1] An advantage function $A^\pi(s,a)$ is a state-action value roughly corresponding to how much better or worse an action a is compared to the current policy at state s.

jointly contribute to obtaining the optimal behavior. By representing at a sufficiently abstract level rather than specifically tailored towards individual problems, symbolic knowledge can be light-weight and concise to be a useful guideline for data-driven learning. The agent can further learn domain details and uncertainties to refine its behavior simultaneously from both environmental rewards and human feedback. In this way, the prior knowledge is enriched with experience and tailored towards individual problem instances. Based on the motivation above, we propose the **P**lanner–**A**ctor–**C**ritic architecture for hu**MAN** centered planning and learning (**PACMAN**). PACMAN interprets human feedback as the advantage function estimation similar to COACH framework but further incorporates prior, symbolic knowledge. The contributions of this paper are summarized as follows: **(i.)** The framework of PACMAN features symbolic planner-actor-critic trio iteration, where planning and RL mutually benefit each other. In particular, the logical representation of action effects is dynamically generated by sampling a stochastic policy learned from actor-critic (AC) algorithm of RL. PACMAN allows the symbolic knowledge and actor-critic framework to integrate into a unified framework seamlessly. **(ii.)** This framework enables joint learning from both environmental rewards and human feedback, which can accelerate the learning process of interactive RL as well as improve the tolerance of misleading feedback from human users.

To the best of our knowledge, this paper is the first work that learns simultaneously from human feedback, environmental rewards, and prior symbolic knowledge. While our framework can be quite generic, we choose to use ASP-based action language $\mathcal{BC}$ (Lee, Lifschitz, and Yang 2013), answer set solver CLINGO to perform symbolic planning and conduct our experiments. The evaluation of the framework is performed on RL benchmark problems such as Four Rooms and Taxi domains. We consider various scenarios of human feedback, including the cases of ideal, infrequent, inconsistent, and both infrequent and inconsistent with helpful feedback and misleading feedback, and compare our approach with the state-of-the-art methods. Our experiments indicate that PACMAN empirically leads to a significant jump-start at early stages of learning, converges faster and with smaller variance, and is robust to inconsistent, infrequent cases even misleading feedback.

Related Work

There is a long history of work that combines symbolic planning with reinforcement learning (Parr and Russell 1998; Ryan and Pendrith 1998; Ryan 2002; Hogg, Kuter, and Munoz-Avila 2010; Leonetti, Iocchi, and Patrizi 2012; Leonetti, Iocchi, and Stone 2016). These approaches were based on integrating symbolic planning with value iteration methods, and thus planning and learning cannot be mutually beneficial to each other. The latest work in this direction is PEORL framework (Yang et al. 2018) and SDRL (Lyu et al. 2019), where ASP-based planning was integrated with R-learning (Schwartz 1993) into planning–learning loop. PACMAN architecture is a new framework of integrating symbolic planning with RL, in particular, integrating planning with AC algorithm for the first time, and also features bidirectional communication between planning and learning.

Learning from human feedback takes the framework of reinforcement learning, and incorporate human feedback into reward structure(Thomaz, Breazeal, and others 2006; Knox and Stone 2009; 2012), information directly on policy (Thomaz and Breazeal 2008; Knox and Stone 2010; Griffith et al. 2013), or advantage function (MacGlashan et al. 2016; 2017). Learning from both human feedback and environmental rewards were investigated (Thomaz, Breazeal, and others 2006; Knox and Stone 2012; Griffith et al. 2013), mainly integrating the human feedback to reward or value function via reward shaping or Q-value shaping. Such methods do not handle well the samples with missing human feedback, and in reality, human feedback may be infrequent.

Recent work of COACH (MacGlashan et al. 2016; 2017) showed that human feedback seems better to formulate as an estimation of the policy-dependent advantage function, but it does not consider learning simultaneously from environmental rewards and human feedback. Besides, none of these work considers the setting where an agent is equipped with prior knowledge and generates a goal-directed plan that is further to be fine-tuned by reinforcement learning and a human user. By integrating human feedback into PACMAN, our framework allows the integration of logic-based symbolic planning into the data-driven learning process, where environmental rewards and human feedback can be unified into advantage function to shape the agent's behavior in the context of long-term planning.

Preliminaries

Symbolic Planning Symbolic planning concerns on describing preconditions and effects of actions using a formal language and automated plan generation, which has been used for high-level task planning in a variety of robotic applications (Hanheide et al. 2015; Khandelwal et al. 2017). An *action description* D in the language $\mathcal{BC}$ includes two kinds of symbols, *fluent constants* that represent the properties of the world, with the signature denoted as $\sigma_F(D)$, and *action constants*, with the signature denoted as $\sigma_A(D)$. A *fluent atom* is an expression of the form $f = v$, where f is a fluent constant and v is an element of its domain. For Boolean domain, denote $f = \mathbf{t}$ as f and $f = \mathbf{f}$ as $\sim f$. An action description is a finite set of *causal laws* that describe how fluent atoms are related with each other in a single time step, or how their values are changed from one step to another, possibly by executing actions. For instance, A **if** $A_1, \ldots, A_m$ is a *static law* that states at a time step, if $A_1, \ldots, A_m$ holds then A is true. a **causes** A_0 **if** $A_1, \ldots, A_m$ is a *dynamic law*, stating that at any time step, if $A_1, \ldots, A_m$ holds, by executing action a, A_0 holds in the next step.[2] An action description captures a dynamic transition system. Let I and G be states. The triple (I, G, D) is called a planning problem. (I, G, D) has a plan of length $l - 1$ **iff** there exists a transition path of length l such that $I = s_1$ and

[2] In $\mathcal{BC}$, causal laws are defined in a more general form. In this paper, without loss of generality, we assume the above form of causal laws for defining effects of actions.

$G = s_l$. Throughout the paper, we use Π to denote both the plan and the transition path by following the plan. Generating a plan of length l can be achieved by solving the answer set program $PN_l(D)$, consisting of rules translated from D and appending timestamps from 1 to l, via a translating function PN. For instance, PN_l turns the static law to $i : A \leftarrow i : A_1, \ldots, i : A_m$, where $1 \leq i \leq l$ and turns the dynamic law to $i + 1 : A \leftarrow i : a, i : A_1, \ldots, i : A_m$, where $1 \leq i < l$. See (Lee, Lifschitz, and Yang 2013) for details.

Reinforcement Learning and Actor-Critic Method RL problem is usually defined as a Markov Decision Process (MDP), which is a tuple of $(\mathcal{S}, \mathcal{A}, P^a_{ss'}, r, \gamma)$. Specifically, $\mathcal{S}$ and $\mathcal{A}$ denotes state space and action space, the transition kernel $P^a_{ss'}$ specifies the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $r(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function bounded by $r_{\max}$, and $0 \leq \gamma < 1$ is a discount factor. RL concerns learning a near-optimal policy $\pi(a|s)$ (maps a state s to an action a) by executing actions and observing the state transitions and rewards.

An actor-critic (Peters and Schaal 2008; Bhatnagar et al. 2009; Sutton and Barto 2018) approach is a framework of RL which has two components: the actor and the critic. Typically, the actor is a policy function $\pi_\theta(a|s)$ parameterized by θ for action selection, while the critic is a state-value function $V_x(s)$ parameterized by x to criticize the action made by the actor. For example, after each action selection, the critic will evaluate the new state to determine whether things have gone better or worse than expected by computing TD error (Sutton and Barto 2018). If the TD error is positive, it suggests that the tendency to select current action a should be strengthened for the future, whereas if the TD error is negative, it suggests the tendency should be weakened. This TD error is actually an unbiased estimation of advantage function $A^\pi(s, a)$ (Schulman et al. 2015).

PACMAN Architecture

In this section, we will present our PACMAN architecture, which is shown in Figure 1. With the encoded prior knowledge and the policy function (from the actor), the symbolic planner would generate a plan that contains a sequence of actions, and send it to RL (actor-critic) to execute. During the interaction between RL and environment, the estimation of advantage function can be either from TD error computed by the critic or the value of human feedback. The detailed process will be defined formally as follows.

Sample-based Symbolic Planning

We introduce a *sample-based planning problem* as a tuple (I, G, D, π_θ) where I is the initial state condition, G is a goal state condition, D is an action description in $\mathcal{BC}$, and π_θ is a stochastic policy function parameterized by θ, i.e., a mapping $\mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. For D, defines its l-step *sampled action description* $D^l_\pi = D_s \cup D_d \cup \bigcup_{t=1}^{l} P^t_\pi$ with respect to policy π and time stamp $1 \leq t \leq l$, where

- D_s is a set of causal laws consisting of static laws and dynamic laws that does not contains action symbols;

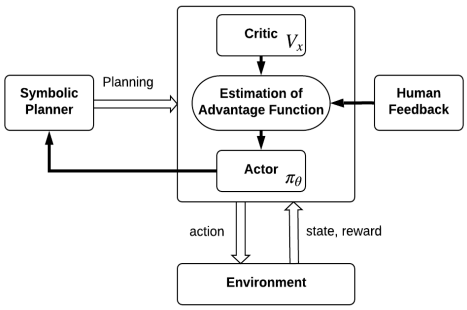


Figure 1: Architecture illustration

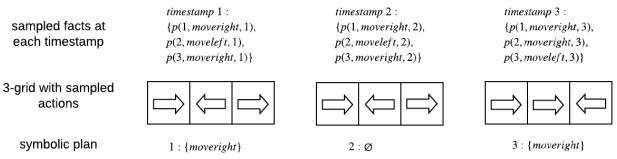


Figure 2: A possible sample-based planning result for 3-grid domain

Algorithm 1 Sample-based Symbolic Planning

Require: a sample based planning problem (I, G, D, π_θ)
1: $\Pi \Leftarrow \emptyset$, calculate D^0_π, $k \Leftarrow 1$
2: **while** $\Pi = \emptyset$ and $k <$ *maxstamp* **do**
3: sample P^k_π over $p(s, a) \sim \pi_\theta(\cdot|s)$ for $s \in \mathcal{S}$, $a \in \mathcal{A}$
4: $D^k_\pi \Leftarrow D^{k-1}_\pi \cup P^k_\pi$
5: $\Pi \leftarrow$ CLINGO.*solve*$(I \cup G \cup \mathcal{T}(D^k_\pi))$
6: $k \leftarrow k + 1$
7: **end while**
8: **return** Π

- D_d is a set of causal laws obtained by turning each dynamic law of the form a **causes** A_0 **if** $A_1, \ldots, A_m$, into rules of the form a **causes** A_0 **if** $A_1, \ldots, A_m, p(s, a)$ where p is a newly introduced fluent symbol and $\{A_1, \ldots, A_m\} \subseteq s$, for $s \in \mathcal{S}$; and
- P^t_π is a set of facts sampled at timestamp t that contains $p(a, s)$ such that $p(s, a) \in P^t_\pi \sim \pi(\cdot|s, \theta)$ where for $s \in \mathcal{S}$, $A \in \mathcal{A}$.

Define translation $\mathcal{T}(D^l_\pi)$ as $PN_l(D_s \cup D_d) \cup \bigcup_{t=1}^{l} \{p(s, a, t), \text{for } p(s, a) \in P^t_\pi\}$ that turns D^l_π into answer set program. A *sample-based plan* up to length l of (I, G, D, π_θ) can be calculated from the answer set of program $\mathcal{T}(D^l_\pi)$ such that I and G are satisfied. The planning algorithm is shown in Algorithm 1.

Example. Consider 3×1 horizontal gridworld where the grids are marked as state 1, 2, 3, horizontally. Initially the agent is located in state 1. The goal is to be located in state 3. The agent can move to left or right. Using action language $\mathcal{BC}$, moving to the left and moving to the right can be for-

mulated as dynamic laws

$$\textit{moveleft} \ \textbf{causes} \ \textit{Loc} = L - 1 \ \textbf{if} \ \textit{Loc} = L.$$
$$\textit{moveright} \ \textbf{causes} \ \textit{Loc} = L + 1 \ \textbf{if} \ \textit{Loc} = L.$$

Turning them into sample-based action description leads to

$$\textit{moveleft} \ \textbf{causes} \ \textit{Loc} = L - 1 \ \textbf{if} \ \textit{Loc} = L, p(L, \textit{moveleft}).$$
$$\textit{moveright} \ \textbf{causes} \ \textit{Loc} = L + 1 \ \textbf{if} \ \textit{Loc} = L, p(L, \textit{moveright}).$$

The policy estimator π_θ accepts an input state and output probability distribution on actions *moveleft* and *moveright*. Sampling π_θ with input s at time stamp i generates a fact of the form $p(s, a, i)$ where $a \in \{\textit{moveleft}, \textit{moveright}\}$ following the probability distribution of $\pi_\theta(\cdot|s)$.

At any timestamp, CLINGO solves answer set program consisting of rules translated from the above causal laws:

```
loc(L-1,k+1):-moveleft(k),loc(L,k), p(L,moveleft,k).
loc(L+1,k+1):-moveright(k),loc(L,k), p(L,moveright,k).
```

for time stamp $1, \ldots, k$, plus a set of facts of the form `p(s,a,i)` sampled from π_θ where for states $s \in \{1, 2, 3\}$ and timestamps $i \in \{1, \ldots, k\}$. Note that the planner can skip time stamps if there is no possible actions to use to generate plan, based on sampled results. Figure 2 shows a possible sampling results over 3 timestamps, and a plan of 2 steps is generated to achieve the goal, where time stamp 2 is skipped with no planned actions. Since sample-based planning calls a policy approximator as an oracle to obtain probability distribution and samples the distribution to obtain available actions, it can be easily applied to other planning techniques such as PDDL planning. For instance, the policy appropriator can be used along with heuristics on relaxed planning graph (Helmert 2006).

Planning and Learning Loop

The planning and learning loop for PACMAN, as shown in Algorithm 2, starts from a random policy (uniform distribution over action space), and then generate a sample-based symbolic plan. After that, it follows the plan to explore and update the policy function π_θ, leading to an improved policy, which is used to generate the next plan.

Algorithm 2 PACMAN

Require: (I, G, D, π_θ) and a value function estimator V_x
1: **for** *episode* $= 0, 1, \ldots, maxepisode$ **do**
2: Generate symbolic plan Π from (I, G, D, π_θ) by Algorithm 1
3: **for** $\langle s_i, a_i, r_i, s_{i+1} \rangle \in \Pi$ **do**
4: Compute TD error as $\delta_i = r_i + \gamma V_{x_{i+1}}(s_{i+1}) - V_{x_{i+1}}(s_i)$.
5: Update V_x via $x_{i+1} = x_i + \alpha \delta_i \nabla V_{x_i}(s_i)$.
6: **if** human feedback f_i is available **then**
7: Replace TD error δ_i with human feedback f_i.
8: **end if**
9: Update π_θ via $\theta_{i+1} = \theta_i + \beta \delta_i \nabla \log \pi_\theta(a_i|s_i)$.
10: **end for**
11: **end for**

For the i-th experience tuple of an episode, (s_i, a_i, r_i, s_{i+1}), the TD error is computed as $\delta_i = r_i + \gamma V_{x_{i+1}}(s_{i+1}) - V_{x_{i+1}}(s_i)$, which is a stochastic estimation of the advantage function. The value function V_x is updated using reinforcement learning approaches, such as TD method (Sutton and Barto 2018): $x_{i+1} = x_i + \alpha \delta_i \nabla V_{x_i}(s_i)$, where α is the learning rate. The policy function π_θ will be updated by $\theta_{i+1} = \theta_i + \beta \delta_i \nabla \log \pi_\theta(a_i|s_i)$, where β is the learning rate. If the human feedback signal f_i is available, then this feedback signal will replace the previous computed TD error and be used to update the policy function; If there is no human feedback signal available at this iteration, TD error will be used to update the policy function directly. For this reason, human feedback here can be interpreted as guiding exploration towards human preferred state-action pairs.

Experiment

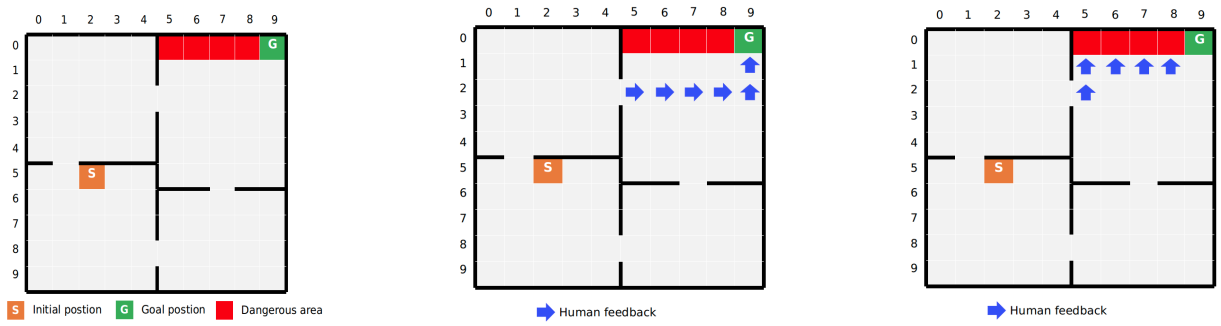
We evaluate our method in two RL-benchmark problems: Four Rooms (Sutton, Precup, and Singh 1999) and Taxi domain (Barto and Mahadevan 2003). For experiments, we consider the discrete value of (positive or negative) feedback with the cases of ideal (feedback is always available without reverting), infrequent (only giving feedback at 50% probability), inconsistent (randomly reverting feedback at 30% probability) and infrequent+inconsistent (only giving feedback at 50% probability, while randomly reverting feedback at 30% probability). We compare the performance of PACMAN with 3 methods: TAMER+RL Reward Shaping from (Knox and Stone 2012), BQL Reward Shaping from (Griffith et al. 2013), and PACMAN without symbolic planner (AC with Human Feedback) as our ablation analysis. All plotting curves are averaged over 10 runs, and the shadow around the curve denotes the variance.

Four Rooms

Four rooms domain is shown in Fig. 3a. In this 10×10 grid, there are 4 rooms and an agent navigating from the initial position (5,2) to the goal position (0,9). If the agent can successfully achieve the task, it would receive a reward of +5. And it may obtain a reward of -10 if the agent steps into the red grids (dangerous area). Each move will cost -1. The human feedback of Four Rooms domain concerns 2 scenarios:

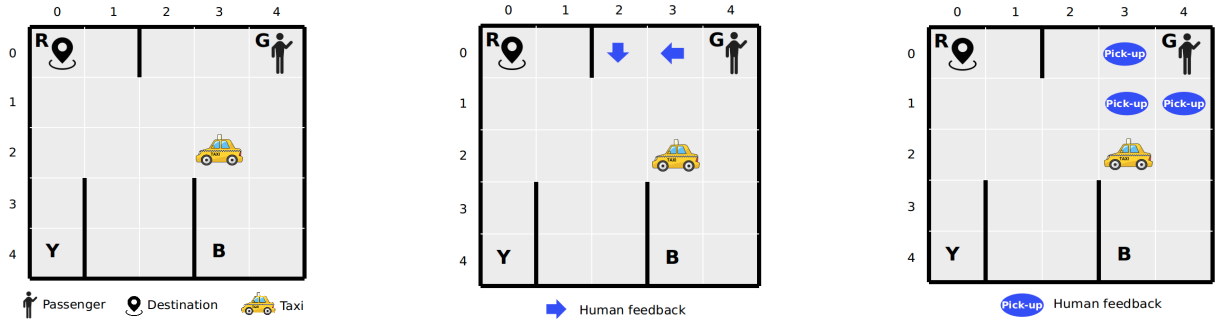
- Helpful feedback: consider an experienced user that wants to help the agent to navigate safer and better, such that the agent can stay away from the dangerous area and reach the goal position with the shortest path. Therefore, human feedback can guide the agent to improve its behavior towards the task, as shown in Fig. 3b.
- Misleading feedback: consider an inexperienced user who doesn't know there is a dangerous area, but wants the agent to step into those red grids (Fig. 3c). In this case, human feedback contradicts with the behavior that the agent learns from an environmental reward.

The results are shown in Fig. 5 and Fig. 6. Obviously, PACMAN has a jump-start and quickly converged with small variance, compared to BQL Reward Shaping, TAMER+RL Reward Shaping, and AC with Human Feedback under four different cases. This is because symbolic



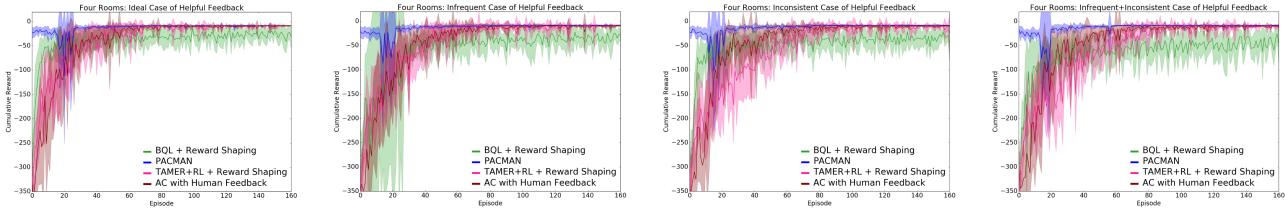
(a) Four rooms domain (b) Helpful feedback (c) Misleading feedback

Figure 3: The snapshot of 2 scenarios on four rooms domain



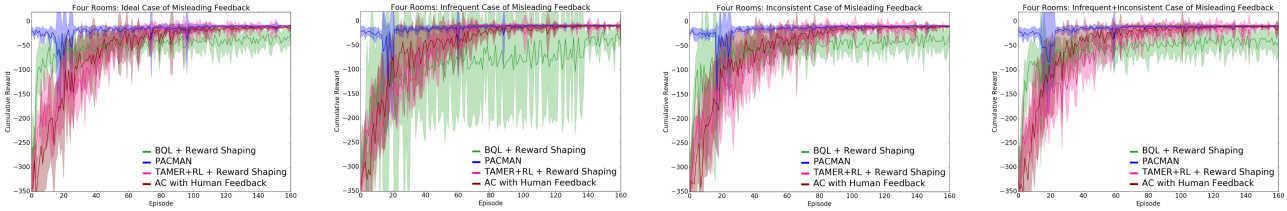
(a) Taxi domain (b) Helpful feedback (c) Misleading feedback

Figure 4: The snapshot of 2 scenarios on taxi domain



(a) Ideal case (b) Infrequent case (c) Inconsistent case (d) Infrequent+Inconsistent case

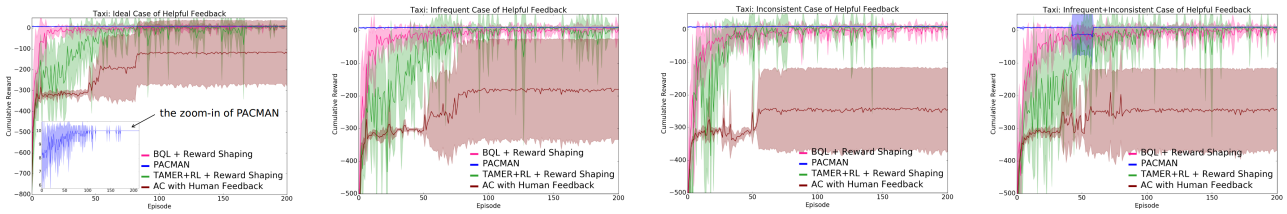
Figure 5: Four rooms with helpful feedback: learning curves



(a) Ideal case (b) Infrequent case (c) Inconsistent case (d) Infrequent+Inconsistent case

Figure 6: Four rooms with misleading feedback: learning curves

planning leading to goal-directed behavior biasing exploration. Though the infrequent case, inconsistent case, and their combination case for both helpful feedback and misleading feedback can lead to more uncertainties, the performance of PACMAN remains unaffected, which means more robust than others. Meticulous readers may find that there is a large variance in the initial stage of PACMAN, especially in Fig. 5, Fig. 6, this is due to the reason that the symbolic planner will first generate a short plan that is reasonably well, then the symbolic planner will perform exploration



(a) Ideal case (b) Infrequent case (c) Inconsistent case (d) Infrequent+Inconsistent case

Figure 7: Taxi with helpful feedback: learning curves

(a) Ideal case (b) Infrequent case (c) Inconsistent case (d) Infrequent+Inconsistent case

Figure 8: Taxi with misleading feedback: learning curves

by generating longer plans. After doing the exploration, the symbolic planner will converge to the short plan with the optimal solution. But the large variance at the initial phase of PACMAN can be partially alleviated by setting the maximal number of actions in a plan to reduce plan space.

Taxi Domain

Taxi domain concerns a 5×5 grid (Fig. 4a) where a taxi needs to navigate to a passenger, pick up the passenger, then navigate to the destination and drop off the passenger. Each move has a reward of -1. Successful drop-off received a reward of +20, while improper pick-up or drop-off would receive a reward of -10. When formulating the domain symbolically, the precondition of performing picking up a passenger is specified that the taxi has to be located in the same place as the passenger. We consider human feedback in the following two scenarios:

- Helpful feedback: consider the rush hour, the passenger can suggest a path that would guide the taxi to detour and avoid the slow traffic, which is shown in Fig. 4b. The agent should learn a more preferred route from human's feedback.
- Misleading feedback: consider a passenger who is not familiar enough with the area and may inaccurately inform the taxi of his location before approaching the passenger (Fig. 4c), which is the wrong action and will mislead the taxi. In this case, the feedback conflicts with symbolic knowledge specified by PACMAN and the agent should learn to ignore such feedback.

The results are shown in Fig. 7 and Fig. 8. In the scenarios of both helpful and misleading feedback, the curve of PACMAN has the smallest variance so that it looks like a straight line, whereas it actually has the learning process (the zoom-in curve shown in the figures of the ideal case). But in the case of Infrequent+Inconsistent, there is a big chattering in the initial stage of PACMAN, that's because the symbolic planner is trying some longer plans to do the exploration. In the misleading feedback scenario, the learning speed of the other methods except for PACMAN is quite slow. That's because the human feedback will misguide the agent to perform the improper action that can result in the penalty, and the agent needs a long time to correct its behavior via learning from the environmental reward. But PACMAN keeps unaffected in this case due to the symbolic knowledge that a taxi can pick up the passenger only when it moves to the passenger's location.

Conclusion

In this paper we propose the PACMAN framework, which takes into consideration of the prior knowledge, learning from environmental rewards and human teaching together and jointly contribute to obtaining the optimal policy. Experiments demonstrate that PACMAN tends to lead to a significant jump-start at early stages of learning, converge faster with reduced variance, and perform robustly to inconsistent, infrequent, and even misleading human feedback. Our future work involves investigation of using PACMAN to perform decision-making from high-dimensional sensory input such as pixel images, autonomous driving where the vehicle can learn human's preference on comfort and driving behaviors, as well as multi-agent systems such as mobile service robots.

Acknowledgment

This research was supported in part by the National Science Foundation (NSF) under grants NSF IIS-1910794 and Amazon Research Award.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5):469–483.

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal* 13:41–77.

Bhatnagar, S.; Sutton, R.; Ghavamzadeh, M.; and Lee, M. 2009. Natural actor-critic algorithms. *Automatica* 45(11):2471–2482.

Griffith, S.; Subramanian, K.; Scholz, J.; Isbell, C. L.; and Thomaz, A. L. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, 2625–2633.

Hanheide, M.; Göbelbecker, M.; Horn, G. S.; et al. 2015. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hogg, C.; Kuter, U.; and Munoz-Avila, H. 2010. Learning methods to generate good plans: Integrating htn learning and reinforcement learning. In *AAAI*.

Jiang, Y.; Yang, F.; Zhang, S.; and Stone, P. 2019. Task-motion planning with reinforcement learning for adaptable mobile service robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), to appear*.

Khandelwal, P.; Zhang, S.; Sinapov, J.; Leonetti, M.; Thomason, J.; Yang, F.; Gori, I.; Svetlik, M.; Khante, P.; and Lifschitz, V. 2017. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *The International Journal of Robotics Research* 36(5-7):635–659.

Knox, W. B., and Stone, P. 2009. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, 9–16. ACM.

Knox, W. B., and Stone, P. 2010. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 5–12.

Knox, W. B., and Stone, P. 2012. Reinforcement learning from simultaneous human and mdp reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 475–482.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action Language $\mathcal{BC}$: A Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Leonetti, M.; Iocchi, L.; and Patrizi, F. 2012. Automatic generation and learning of finite-state controllers. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, 135–144. Springer.

Leonetti, M.; Iocchi, L.; and Stone, P. 2016. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence* 241:103–130.

Lyu, D.; Yang, F.; Liu, B.; and Gustafson, S. 2019. Sdrl: Interpretable and data-efficient deep reinforcement learningleveraging symbolic planning. In *AAAI*.

MacGlashan, J.; Littman, M. L.; Roberts, D. L.; Loftin, R.; Peng, B.; and Taylor, M. E. 2016. Convergent actor critic by humans. In *International Conference on Intelligent Robots and Systems*.

MacGlashan, J.; K Ho, M.; Loftin, R.; Peng, B.; Wang, G.; Roberts, D. L.; Taylor, M. E.; and Littman, M. L. 2017. Interactive learning from policy-dependent human feedback. In *ICML*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Parr, R., and Russell, S. J. 1998. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, 1043–1049.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7):1180–1190.

Ryan, M. R., and Pendrith, M. D. 1998. Rl-tops: An architecture for modularity and re-use in reinforcement learning. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, 481–487. Morgan Kaufmann.

Ryan, M. R. K. 2002. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *In Proceedings of The 19th International Conference on Machine Learning*, 522–529. Morgan Kaufmann.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schwartz, A. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *Proc. 10th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2):181–211.

Thomaz, A. L., and Breazeal, C. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172(6-7):716–737.

Thomaz, A. L.; Breazeal, C.; et al. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, 1000–1005. Boston, MA.

Yang, F.; Lyu, D.; Liu, B.; and Gustafson, S. 2018. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *International Joint Conference of Artificial Intelligence (IJCAI)*.