Parameterized Modular Inverse Reinforcement Learning

APPROVED BY

SUPERVISING COMMITTEE:

Dana Ballard, Supervisor

Peter Stone

Parameterized Modular Inverse Reinforcement Learning

by

Shun Zhang, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2015

Acknowledgments

I would appreciate multiple people who helped me with this thesis. Prof. Dana Ballard. Prof. Peter Stone. Prof Mary Hayhoe, Matthew Tong. This work wouldn't be possible without Matthew Tong's work on human subjects experiments. Ruohan Zhang. I appreciate the suggestions when I presented this work at the Embodied Cognition Lab meetings.

Parameterized Modular Inverse Reinforcement Learning

Shun Zhang, M.S.
The University of Texas at Austin, 2015

Supervisor: Dana Ballard

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Humans are able to learn and accomplish some complex tasks more efficiently than reinforcement learning agents. The possibilities are humans have some prior knowledge on how to accomplish smaller and easier tasks. When they tackle a complex task, they only need to learn which basic skills they need to use, and how to combine them.

This thesis is organized as follows. Chapter 2 introduces preliminary concepts of reinforcement learning, inverse reinforcement learning, modular reinforcement learing and describes the proposed algorithm. Chapter 3 reviews the literature on task decomposition in reinforcement learning. We report our experimental results in Chapter 4 and conclude in Chapter 5.

Chapter 2

Modular Inverse Reinforcement Learning

2.1 Preliminaries

2.1.1 Markov Decision Process

In this paper, we represent Markov Decision Process (MDP) as a tuple of five elements, (S, A, P, R, γ), where S is the set of states; A is the set of actions; $P : S \times A \times S \to \mathfrak{R}$ denotes the probability of a state-action-state transition; $R : S \to \mathfrak{R}$ represents the reward upon reaching a state; γ is the discount factor in the range of $[0, 1]$.

A policy is a mapping $\pi : S \to A$. A value of a state, given a policy, denoted as V^π, is the accumulated discounted rewards by following π.

$$V^\pi(s) = R(s) + \mathrm{E}[\gamma R(s_1), \gamma^2 R(s_2) + \cdots |\pi]$$

, where $s_1, s_2, \cdots$ are the states by following policy π. Or recursively,

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P(s, \pi(s), s') V^\pi(s')$$

, which is known as Bellman Equation [**?**]. The goal is to find the optimal policy π^* so that $V^{\pi^*}(s) \geq V^\pi(s)$ for all s and for all π. We denote V^{π^*} as V^*.

2.1.2 Factored Markov Decision Process

One common issue in reinforcement learning is the curse of dimensionality. When the size of state space increases, most algorithms are slow to converge to converge to the optimal policies. One promising probability to solve this problem is to decompose the state space.

We will discuss later in the literature review section on state decomposition methods. In this section, we consider a factored approach. We represent S to be $S^{(1)} \times S^{(2)} \times \cdots \times S^{(m)}$, where $S^{(i)}$ is the i-th state component. The transition function can be represented as $P(S_t^{(i)} | S_{t-1}^{(1)}, \cdots, S_{t-1}^{(m)}, a_t)$, where S_t is the state at the time step t. The correlation between the state components are supposed to be sparse, so $S_t^{(i)}$ is independent from most of the state components [**?**].

For example, consider a domain that an agent navigates in a room with targets, penalties and a path []. The agent tries to collect the targets, avoid the penalties and following the path in the meantime. The state can be the location of the agent in the room. However, we can also represent the state by the relative position to the targets, penalties and path. So we can use three state components S^T, S^O, S^P, for these three classes of objects, respectively.

We can further observe that the reward function can also be decomposed, and so does the value function. For example, the reward is given upon reaching a target, independent of relative positions to obstalces and the path. So we can define

2.1.3 Modular Reinforcement Learning

Once the state space and the reward function can be decomposed, the MDP can be decomposed as well. A state component corresponds to a modular MDP $M^{(n)} = \langle S^{(n)}, A, P^{(n)}, R^{(n)}, \gamma^{(n)} \rangle$, where n is the index of this module. All the modules share the same action space. Each module has its own state representation and reward function.

Each module has its own transition function, since the state representation is decomposed. However, the modular transition function should be consistent with the global one. That is, $P^{(n)}(s'^{(n)}|s^{(n)}, a) = P(s'|s, a)$ where $s^{(n)} \in s$ and $s'^{(n)} \in s'$. This actually requires the transition in each modules not to interfere each other.

For example, in the navigation domain, we have a target module, an penalty module and a path module. Each is an MDP for one task. Take the target module for instance. The state of the target module, S^T, is decomposed from the state of the global MDP. S^T can be defined as the distances and angles to the target objects. The reward is given upon reaching an target. $R^T(S^T = \{distance = 0\}) = 1$.

In this setting, the transtion functions in all the modules are consistent with the global one. However, if we involve, say, an obstacle module, an action that taking the action into an obstacle object would make it get stuck, while transitions in other modules get it through. This causes discrepancy between local and global MDPs. Therefore, such modularization is not valid.

The modular RL algorithms assume global Q values can be obtained by summing up module Q values [31, 33]:

$$Q(s_t, a_t) = \sum_{n=1}^{N} Q^{(n)}(s_t^{(n)}, a_t) \tag{2.1}$$

2.1.4 Inverse Reinforcement Learning

While reinforcement learning tries to find the optimal policy given the states, transtions and rewards, inverse reinforcement learning does the opposite. Assuming the states and transitions are known, inverse reinforcement learning recovers the underlying rewards given the policies. The polices could come from the ground truth or an expert.

A common solution to inverse reinforcement learning is the maximum likelihood method []. A popular follow-up work is Bayesian inverse reinforcement learning, which assumes some prior knowledge of rewards. For example, in most of the problems, rewards are generally sparse. So the reward of a state is very likely to be 0. A Gaussian or Laplacian prior can be applied. In planning problems, on the contrary, rewards are generally small but positive for most of the states. Then a Beta prior can be applied.

Most of work in the literature aims at recovering all the rewards in the domain. This may not be realistic due to a sample efficiency issue. Moreover, given a limited number of samples, the observed policy can be optimal for many reward functions. In most real world problems, however, we are not completely ignorant about what the expert aims at. Instead, we propose some hypothesis on what the expert could be doing, and find out which subset of hypothesis is consistent with the expert's behavior. This motivates employing the modular approach that we defined above.

2.2 Modular Inverse Reinforcement Learning

In this section, we describe an inverse reinforcement learning algorithm that assumes a modular representation of the underlying MDP. This is revised from an earlier work on modular IRL algorithm [30].

The input of this learning algorithm is the modular Q functions, and samples from the expert. The samples are represented as (s_t, a_t), which is the action that the expert takes at a state. This approach assumes that the higher the Q-value for an action a_t in state s_t, the more likely state-action pair (s_t, a_t) is observed. There could be some inconsistencies in the samples. For example, different actions are taken at a same state. This can be caused by the noises in expert's decision model or in the state presentation. Let η denote the confidence level in optimality (the extent to which an agent follows the optimal policy).

Same as modular reinforcement learning, we can assume that the Q function is the weighted sum of the Q functions of some fixed modular MDPs. Then, naturally, we only need to find the weight of each modular MDP. However, we can observe two limitations of this method. First, even though one of the motivation is that searching in the raw reward space is inefficient, the weight space is too small and may not contain the optimal solution (that could be represented by a raw reward vector). Second, it is unlikely that we know exactly what the modular MDPs are, and can define the Q functions consistent with the expert's. This motiviates us to make the modular MDPs flexible.

We replace the notation of the i-th module M_i with $M_i(p_i)$, where p_i is a vector that configures the i-th module. The configuration parameter makes the modules flexible, but does not affect the foundamental behaviors of the modules. Take the target mdoule for instance, Let M_1 be the module of target collection. Its configuration parameter can be the reward of the target, and the discount factor ($p^L = (r^L, \gamma^L)$). This captures the significance of the target module compared to other modules, and the agent's sensitivity to this module.

$$\max_p \prod_t \frac{e^{\eta Q(s^{(t)}, a^{(t)})}}{\sum_b e^{\eta Q(s^{(t)}, b)}} \tag{2.2}$$

, which is equivalent to

$$\begin{aligned} &\max_p \log \prod_t \frac{e^{\eta Q(s^{(t)}, a^{(t)})}}{\sum_b e^{\eta Q(s^{(t)}, b)}} && (2.3)\\ =&\max_p \sum_t (\eta Q(s^{(t)}, a^{(t)}) - \log \sum_b e^{\eta Q(s^{(t)}, b)}) && (2.4)\\ =&\max_p \sum_t (\eta \sum_i Q_i(s^{(t)}, a^{(t)}) - \log \sum_b e^{\eta \sum_i Q_i(s^{(t)}, b)}) && (2.5)\end{aligned}$$

Compared to an earlier version of this work [30], we use an general presentation of modular MDPs, instead of using weights on fixed sub-modules. We extends it to handle multiple instances of each module, learning the discount factors, and deriving a different objective function.

Chapter 3

Literature Review

3.1 Overview

In this section, we will discuss some major approaches of MDP decomposition as solutions to the curse of dimensionality. This is summerized in Table **??**.

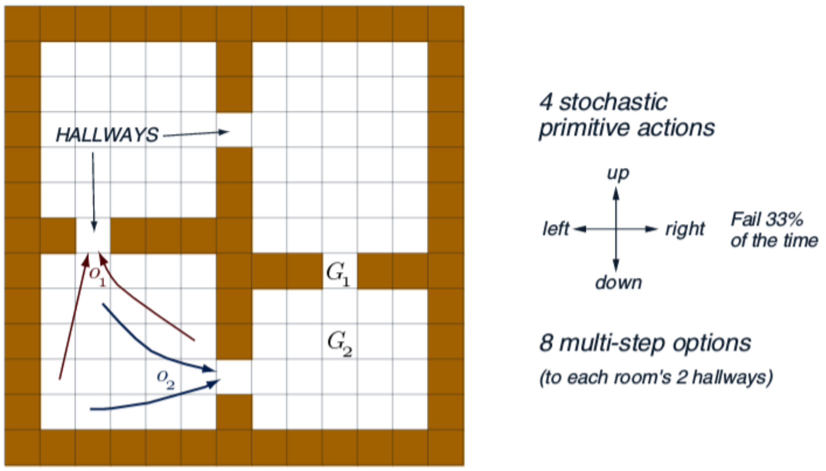
Abstraction on MDP

- Aggregate states: feature extraction.
- Aggregate actions: **option**.
- Decompose transition: factored MDP.
- Decompose value (abstract MDP): **HAM, hierarchical RL, modular RL**.

Approaches	State	Action	Transition	Reward
MDP with Option		Aggregated actions		
Factored MDP	Decomposed		Decomposed	Decomposed or not
HAM	sub-MDP			
Hierarchical RL		Recursive options		Value of options
Modular RL	Decomposed			Decomposed

Table 3.1: Overview of decomposition or aggregation of the components of MDP.

MDP with Option



- Option: (start state, policy, termination condition).

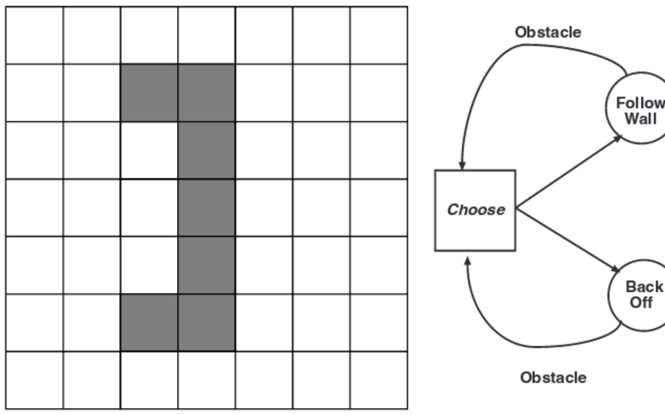
- State: S.

- Action: A, O.

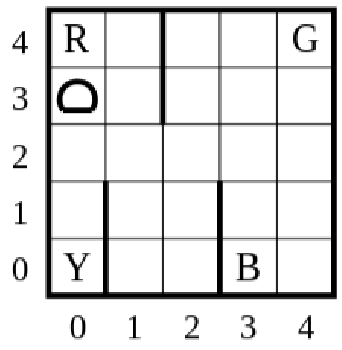
- Transition: $P : S \times \{A, O\} \times S \to \mathcal{R}$.

- Reward: $R : S \times \{A, O\} \times S \to \mathcal{R}$.

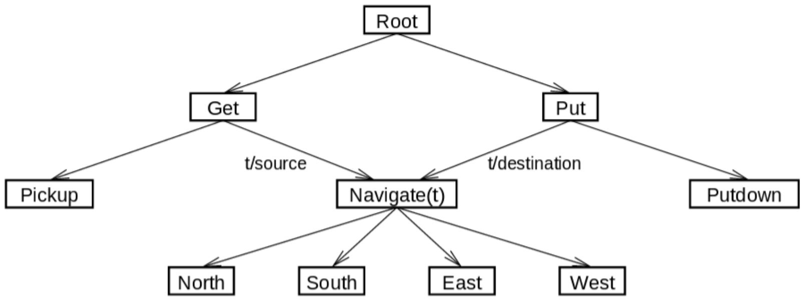
Hierarchies of Abstract Machines (HAM)



- State machine of MDPs.



Hierarchical RL



Hierarchical RL MDP:

- State: $\mathcal{S}$.
- Action: $\mathcal{A}$.
- Transition: $\mathcal{T}$.
- Reward: $\mathcal{R}$.

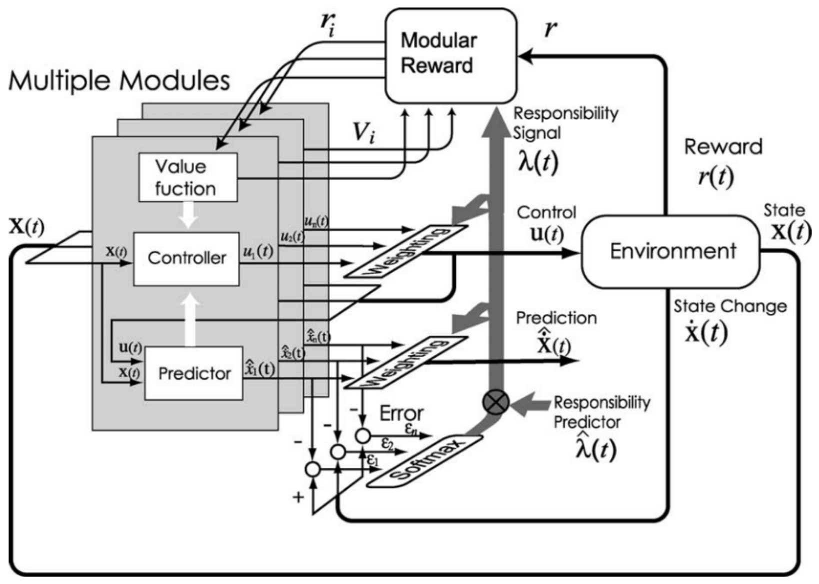


Fig. 1. MMRL.

Modular RL

MDP:

- State: $S_1 \times S_2 \cdots \times S_M$.
- Action: A.
- Transition: $P_1 \times P_2 \cdots \times P_M$.
- Reward: $R_1 \times R_2 \cdots \times R_M$.

3.2 Advances in Recent Work

Not fix a model.

Learn the components,

Dynamic.

Chapter 4

Evaluations and Applications

4.1 Preliminary Evaluation

In this section, we evaluate the inverse modular reinforcement learning algorithm in a navigation domain. We have access to the ground truth. This helps us to empirically verify the correctness and efficiency of the algorithm.

4.2 Sanity Check

First, we test the algorithm in a grid world domain. We define two classes of objects — targets and penalties. We set the rewards and discounters differently for both classes. We use the modular Q function and observed policies on all the grids as input to our algorithm, and run our algorithm to recover the underlying rewards and discounters for all the modules.

4.3 Comparison with Non-Modular Inverse Reinforcement Learning

We compare our algorithm with non-modular Bayesian inverse reinforcement learning [26] to demonstrate the sample efficiency advantage of the modular approach. We use a Laplacian prior in Bayesian IRL since the rewards are sparse. In Figure 4.1, we report the sample efficiency of modular IRL versus Bayesian IRL. There are 4 modular classes and each has 4 instances. We run both algorithms with different number of samples (state-policy pairs). We then compare the policies generated using the learned rewards. Policy agreement is defined as the proportion of the states that have the same policy as the ground truth. We use the metric of policy agreement in our comparison since the outputs of these two algorithms are weights and rewards, which can not be directly compared. Our observation is that modular IRL obtained nearly 100% policy agreement with far fewer samples compared to the non-modular approach.

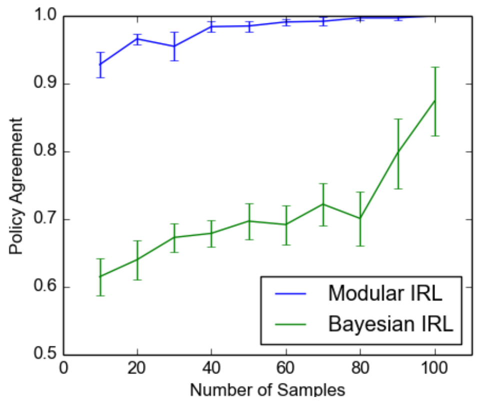


Figure 4.1: Modular IRL vs Bayesian IRL on sample efficiency, measured by policy agreement.

4.4 Human Experiment Results

In this section, we report results from the human virtual navigation experiment. We hypothesize that behavior data can be modeled by our maximum likelihood modular IRL framework and test against baseline models. Figure 4.2 shows the experimental setup. The human subjects wore a binocular head-mounted display. The subjects' eye, head, and body motion were tracked while walking through a virtual room. The subjects were asked to collect the targets (blue spheres) by intercepting them, follow the path (the gray line), and avoid the obstacles (red cubes). Thus this domain has three module classes: following the path, collecting targets, and avoiding obstacles. This general paradigm has been used to evaluate modular IRL algorithms [30] and to study human navigation and gaze behavior [40].

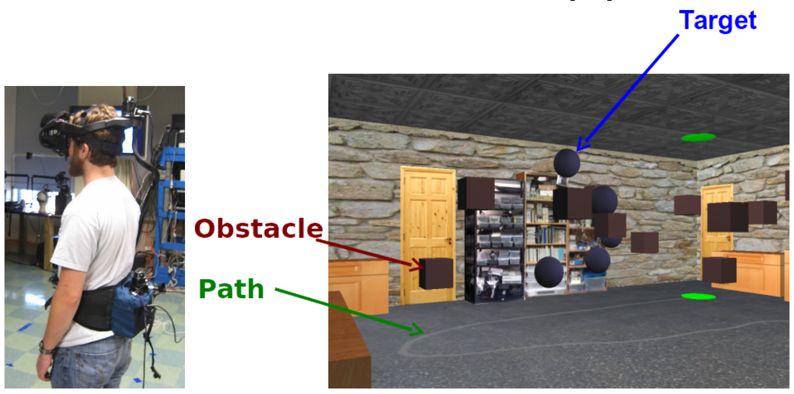


Figure 4.2: The second test domain. (Left) A human subject wears a head mounted display (HMD) and trackers for eyes, head, and body. (Right) The virtual environment as seen through the HMD. The red cubes are obstacles and the blue spheres are targets. There is also a gray path on the ground which the human subject were told to follow.

We gave subjects four types of task instructions, resulting in 4 experimental conditions:

- **Task 1**: Follow the path only and ignore objects
- **Task 2**: Follow the path and avoid the obstacles

- **Task 3**: Follow the path and collect targets
- **Task 4**: Follow the path, collect targets, and avoid obstacles.

Subjects received auditory feedback when running into obstacles or targets, but only when the objects were task relevant. Here we examined data collected from 4 human subjects. Each subject walked through the environment 8 times for each experimental condition, resulting in 32 experimental trials. In each trial the configuration of objects was different.

We use Equation **??** as our objective function to recover w and γ. Our agent uses the distance and angle to the module instances as state information. We constrain the action set of the agent to be human-like; it takes discrete forward actions, ranging from turning left 90 degrees to turning right 90 degrees.

The results are shown in Figure 4.3. Weights are normalized for comparison between modules. It is clear that the estimated w agreed with our task instructions. We then trained an agent with the recovered w and γ, and let the agent navigate in the same environment. The resulting agent trajectories (shown in green) are compared with human trajectories (shown in black) in Figure 4.3.

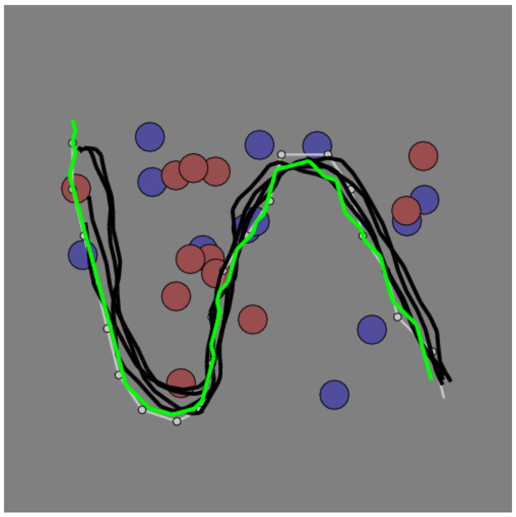
We compared the performance of our agent with two baseline agents, shown in Table 4.1. The *Random Agent* takes an action randomly without considering state information. The *Reflex Agent* greedily chases the nearest

Task	Agent	Angular Diff.	Log Likelihood
Path only	Modular	25.820	-3914.196
	Reflex	35.600	-3916.157
	Random	55.219	-3926.847
Obstacle + Path	Modular	33.988	-4950.079
	Reflex	63.884	-4985.290
	Random	55.717	-4989.314
Target + Path	Modular	33.918	-4855.531
	Reflex	37.176	-4838.832
	Random	55.012	-4909.531
All	Modular	39.034	-6175.692
	Reflex	45.307	-6164.702
	Random	55.961	-6221.075

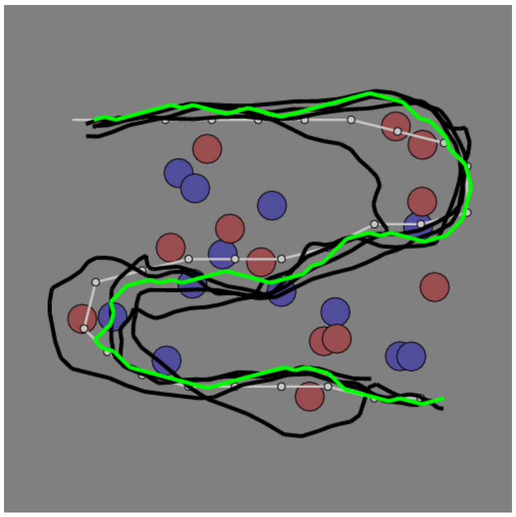
Table 4.1: Evaluation on the modular agent's performance compared with two baseline agents.

target or avoids the nearest obstacle, depending on which is closer. We considered two evaluation metrics. The first is the *angular difference* of the policies between the human subjects and the agent. For every state-policy pair in the human data, we compared the action the human took to the action our agent would select in the same state, taking the difference in angle between the chosen actions. The second metric is the *logarithm of the likelihood*, which is the probability that the human data is generated by the learned parameters. The results are shown in Table 4.1. The modular agent is more similar to the human subjects than the other two agents in terms of angular difference. However, in the last two tasks, it has similar performance with the reflex agent using the likelihood metric.

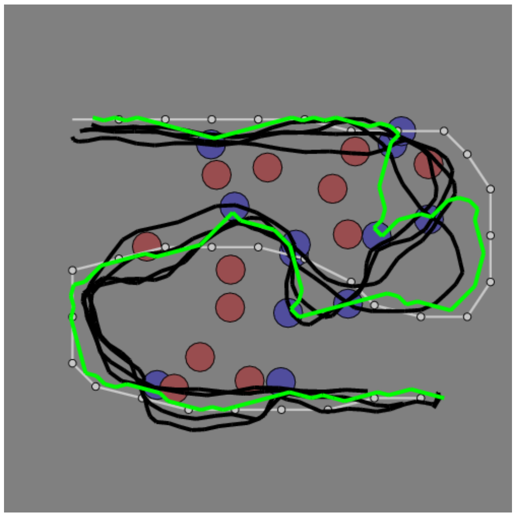
We then look at differences of w and γ within subjects and between subjects in Task 4. The results are shown in Figure 4.4. Within-subjects consistency indicates if the same subject has similar w and γ in different trials of Task 4, measured by the confidence interval (the errorbar). Between-subjects consistency indicates if different subjects have similar w and γ on average in Task 4, measured by the mean value (the height of bar). For w, an interesting observation is that subjects may have different rewards for modules, even though they are given the same task instruction. Subject #3 is different than the other subjects, as he/she clearly weighted collecting targets less and following path more. For γ, we do not find any significant within-subjects or between-subject consistency.



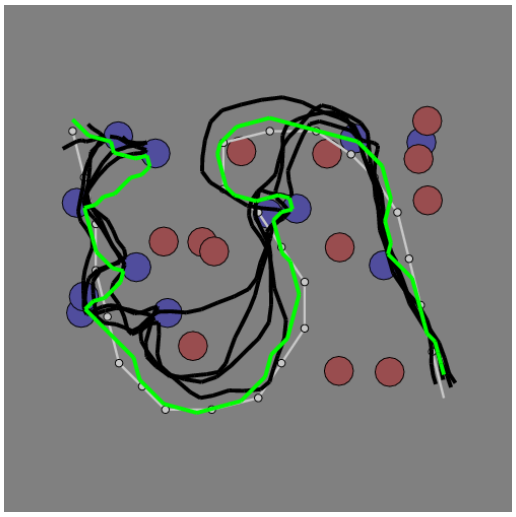
(a) Path only
Rewards: [.035, .017, **.948**]
Discount factors: [.892, 0.181, .900]



(b) Path + Obstacle
Rewards: [.000, **.227**, **.773**]
Discount factors: [.900, .134, .618]

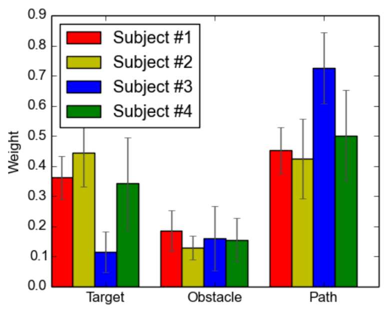


(c) Path + Target
Rewards: [**.395**, .098, **.506**]
Discount factors: [.189, .100, .407]



(d) Path + Target + Obstacle
Rewards: [**.312**, **.180**, **.508**]
Discount factors: [.148, .100, .570]

Figure 4.3: The trajectories of the human subjects and the agent in four conditions. Targets are blue and obstacles are red. The black lines are trajectories of human subjects, and the green lines are trajectories of the RL agent trained using the recovered rewards and discount factors. The rewards and discount factors are shown in [Target, Obstacle, Path] format. The rewards are normalized to sum to 1. The rewards that correspond to task instructions are bold.



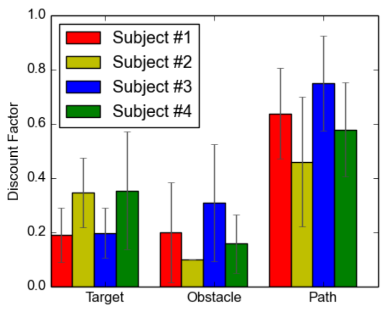


Figure 4.4: The rewards (left) and discount factors (right) of different human subjects in Task 4. The error bars are 95% confidence intervals.

Chapter 5

Conclusion

Appendices

Bibliography

[1] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first ICML*, page 1. ACM, 2004.

[3] Dana H Ballard, Dmitry Kit, Constantin A Rothkopf, and Brian Sullivan. A hierarchical modular architecture for embodied cognition. *Multisensory research*, 26:177–204, 2013.

[4] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[5] Sooraj Bhat, Charles L Isbell, and Michael Mateas. On the difficulty of modular reinforcement learning for real-world partial programming. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 318. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[6] Justin Boyan and Andrew W Moore. Generalization in reinforcement

learning: Safely approximating the value function. *In NIPS*, pages 369–376, 1995.

[7] Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. In *NIPS*, pages 1989–1997, 2011.

[8] Jaedeug Choi and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *NIPS*, pages 305–313, 2012.

[9] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[10] Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse reinforcement learning. In *Recent Advances in Reinforcement Learning*, pages 273–284. Springer, 2012.

[11] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

[12] Katie Genter, Shun Zhang, and Peter Stone. Determining placements of influencing agents in a flock. 2015.

[13] Samuel J Gershman, Bijan Pesaran, and Nathaniel D Daw. Human reinforcement learning subdivides structured action spaces by learning

effector-specific values. *The Journal of Neuroscience*, 29(43):13524–13531, 2009.

[14] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *NIPS*, volume 1, pages 1523–1530, 2001.

[15] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, pages 399–468, 2003.

[16] Masahiko Haruno, Tomoe Kuroda, Kenji Doya, Keisuke Toyama, Minoru Kimura, Kazuyuki Samejima, Hiroshi Imamizu, and Mitsuo Kawato. A neural correlate of reward-based behavioral learning in caudate nucleus: a functional magnetic resonance imaging study of a stochastic decision task. *The Journal of Neuroscience*, 24(7):1660–1665, 2004.

[17] Clay B Holroyd and Michael GH Coles. The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity. *Psychological review*, 109(4):679, 2002.

[18] Mark Humphrys. Action selection methods using reinforcement learning. *From Animals to Animats*, 4:135–144, 1996.

[19] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.

[20] Mitsuo Kawato and Kazuyuki Samejima. Efficient reinforcement learning: computational theories, neuroscience and robotics. *Current opinion in neurobiology*, 17(2):205–212, 2007.

[21] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *NIPS*, pages 19–27, 2011.

[22] Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf, and Jan Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108(5):603–619, 2014.

[23] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *UAI*, 2012.

[24] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

[25] Norihiko Ono and Kenji Fukumoto. Multi-agent reinforcement learning: A modular approach. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 252–258, 1996.

[26] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th IJCAI*, pages 2586–2591. Morgan Kaufmann Publishers Inc., 2007.

[27] Mark Ring and Tom Schaul. Q-error as a selection mechanism in modular reinforcement-learning systems. In *IJCAI*, volume 22, page 1452, 2011.

[28] Khashayar Rohanimanesh and Sridhar Mahadevan. Coarticulation: an approach for generating concurrent plans in markov decision processes. In *Proceedings of the 22nd ICML*, pages 720–727. ACM, 2005.

[29] Constantin A Rothkopf. Inferring human intrinsic rewards through inverse reinforcement learning. *Frontiers in Computational Neuroscience*, (50), 2013.

[30] Constantin A Rothkopf and Dana H Ballard. Modular inverse reinforcement learning for visuomotor behavior. *Biological cybernetics*, 107(4):477–490, 2013.

[31] Stuart Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *ICML*, pages 656–663, 2003.

[32] Kazuyuki Samejima, Kenji Doya, and Mitsuo Kawato. Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16(7):985–994, 2003.

[33] Nathan Sprague and Dana Ballard. Multiple-goal reinforcement learning with modular sarsa (0). In *IJCAI*, pages 1445–1447. Citeseer, 2003.

[34] Nathan Sprague, Dana Ballard, and Al Robinson. Modeling embodied visual behaviors. *ACM Transactions on Applied Perception (TAP)*, 4(2):11, 2007.

[35] Rainer Storn and Kenneth Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, volume 3. ICSI Berkeley, 1995.

[36] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[37] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[38] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[39] Philip S Thomas and Andrew G Barto. Motor primitive discovery. In *ICDL-EPIROB*, pages 1–8, 2012.

[40] M. H. Tong and M. M. Hayhoe. Modeling uncertainty and intrinsic reward in a virtual walking task. *Journal of Vision*, 14(10):5, August 2014.

[41] Eiji Uchibe, Minoru Asada, and Koh Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *IROS*, volume 3, pages 1329–1336. IEEE, 1996.

[42] Adam Vogel, Deepak Ramachandran, Rakesh Gupta, and Antoine Raux. Improving hybrid vehicle fuel efficiency using inverse reinforcement learning. In *AAAI*, 2012.

[43] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[45] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, James A Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *IROS*, pages 3931–3936. IEEE, 2009.

Vita

Permanent address: 9905 Chukar Circle
Austin, Texas 78758

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.