Parameterized Modular Inverse Reinforcement Learning

APPROVED BY

SUPERVISING COMMITTEE:

Prof. Dana Ballard, Supervisor

Prof. 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 like to express my gratitude to Prof. Dana Ballard and Prof. Mary Hayhoe for the useful comments, remarks and engagement through the process of this master thesis. Furthermore I would like to thank Prof. Peter Stone for introducing me to the topic of reinforcement learning, and reading this thesis as a committee member.

I would like to thank Matthew Tong, who conducted the human experiments. The human experiment section would not be possible without his work. I would like to thank Embodied Cognition Lab, directed by Prof. Dana Ballard, for their feedback on this work. I would especially thank Ruohan Zhang, who I work with on this project and on a paper submission.

Parameterized Modular Inverse Reinforcement Learning

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

Supervisor: Prof. Dana Ballard

Reinforcement learning and inverse reinforcement learning can be used to model and understand human behaviors. However, due to the curse of dimensionality, their use as a model for human behavior has been limited. Inspired by observed natural behaviors, one approach is to decompose complex tasks into independent sub-tasks, or modules. Using this approach, we extended an earlier work on modular inverse reinforcement learning, and developed what we called parameterized modular inverse reinforcement learning algorithm. We first demonstrate the cor rectness and efficiency of our algorithm in a simulated navigation task. We then show our algorithm able to estimate a reward function and discount factor for real human navigation behaviors in a virtual environment, and train an agent that imitates the behavior of human subjects. [1]

[1] This is the extension of a nonarchived paper in RLDM 2015, and overlaps largely with a recent submission co-authered with Ruohan Zhang (equal contribution), Matthew Tong, Mary Hayhoe and Dana Ballard.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Reinforcement learning has shown advantages in learning with only the task specifications and feedback signals. A learning agent can interact with the task, observe a current state, and take an action to observe transition to the next state and receive a reward signal. The underlying model is either known or not by the agent. Concretely, the task is formulated as a Markov Decision Process (MDP), which will be defined in details later. The objective of the task is to take actions to maximize the accumulated payoff. In recent years, reinforcement learning has been successfully applied to helicopter control [11] and Atari games [10].

Reinforcement learning is analogous to the way that human learns. Human infants have limited prior knowledge of the world. They can learn to optimize their behavior by observing the environment and perceiving joy or pain. So it is of interest to discover the underlying decision model of human. In both fields of artificial intelligence and neuroscience, it is believed that reinforcement learning is the model that can explain humans' learning.

However, humans are able to learn and accomplish complex tasks more efficiently than reinforcement learning agents. The possibilities are that hu-

mans have some prior knowledge on how to accomplish smaller and easier tasks. When they tackle a complex task, they only need to decide which basic skills they need to use, and how to combine them. Taking driving for example, humans learning to drive don't take it as a completely new task. Many of their preliminary skills can contribute to this new task. It is a complex behavior but a combination of object avoidance, object following, etc.

To understand how human subjects make decisions and to test our hypothesis, we propose an novel inverse reinforcement learning approach in this thesis. We first tested the correctness and efficiency of this approach in a toy domain, and further collected humans' behavior in a navigation task. We used such method to analyze how the behavior is generated.

This thesis is organized as follows. Chapter 2 introduces the preliminary concepts of reinforcement learning, inverse reinforcement learning, modular reinforcement learning and describes the proposed algorithm. Chapter 3 reviews the literature on task decomposition and inverse 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) [18] as a tuple of five elements, (S, A, P, R, γ), where S is the set of states; A is the set of actions; $P : S \times A \times S \to \mathcal{R}$ denotes the probability of a state-action-state transition; $R : S \to \mathcal{R}$ 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 [18]. 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 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 [7].

For example, consider a domain that an agent navigates in a room with targets, obstacles and a path. The agent tries to collect the targets, avoid the obstacles, and follow the path in the meantime. So there is positive reward given when the agent reached an target, or close to a path waypoint. A penalty will be given when the agent reaches an obstacle (the obstalces don't "block" the agent from reaching it).

Apparantly, 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, obstacles 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 obstacles and the path. So we can define R^T, R^O, R^P respectively. Where $R = R^T + R^O + R^P$.

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. [16]

We make 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 $s^{(n)}$ is a component of s and $s'^{(n)}$ is a component of 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 obstacle 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 transition functions in all the modules are consistent

with the global one. However, if we involve, say, an block module, an action that taking the action into an block would make it get stuck at its place, while transitions in other modules get it through. This causes discrepancy between local and global MDPs. Therefore, such modularization is not valid.

We notice that different modules have different Q functions, which lead to different policies. We can only have one global policy. So one issue in modular reinforcement learning is how the modules compromise with each other. This remains to be an open problem [20]. While in this thesis, we assume that the global Q values can be obtained by summing up module Q values [15, 16]:

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

2.1.4 Inverse Reinforcement Learning

While reinforcement learning tries to find the optimal policy given the states, transitions 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 [1]. 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 Parameterized 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 [14].

The input of this learning algorithm is modular Q functions, and samples from an 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. So we let η denote the confidence level in optimality (to what extent the expert agent follows the optimal policy).

In modular reinforcement learning, we assume that the gloabl Q func-

tion 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. This is proposed in [14]. 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 attain the optimal solution. That is, the optimal solution may not be represented by weighted sum of some fixed priorly assumed modules. Second, it is unlikely that we know exactly what the modular MDPs are, and define the Q functions consistent with the expert's. This motivates 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 fundamental behaviors of the modules. Take the target module for instance. Let M^L be the module of target collection. Its configuration parameter can be the reward of the target r^L, the discount factor γ^L ($p^L = (r^L, \gamma^L)$). This captures the significance of the target module compared to other modules, and the agent's sensitivity to this module. Given these parameters, the value function can be determined.

In general, the parameter can be a general attribute of a module. Figure 2.1 shows an example of a value function configured by reward, discount factor, and the radius of the object.

We made the same assumption as modular RL that the global Q function is the sum of modular Q functions. We redefine the modular Q functions

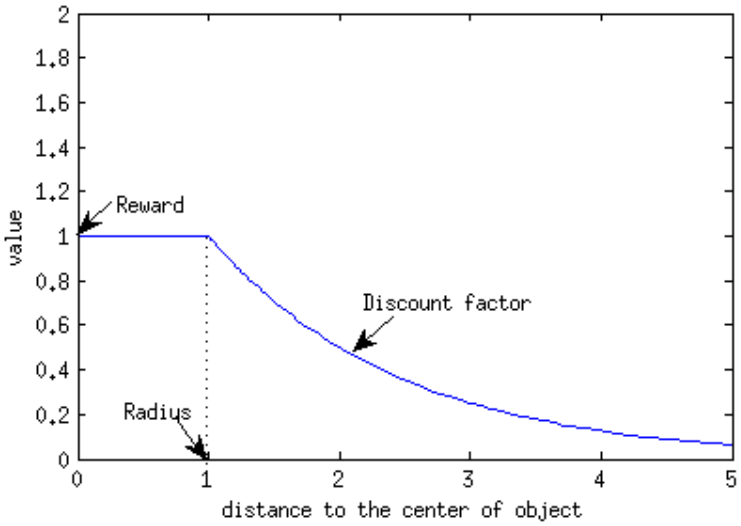


Figure 2.1: A modular value function configured by reward, discount factor and radius of the object.

here to accept the parameter p.

$$Q(s_t, a_t) = \sum_{n=1}^{N} Q^{(n)}(s_t^{(n)}, a_t, p^{(n)}) \tag{2.2}$$

We didn't make any assumption on the prior distribution of the parameters, so the objective is to maximize the sample likelihood. We assume that the expert uses a softmax function in action selection. The objective function is

$$\max_p \prod_t \frac{e^{\eta Q(s_t, a_t)}}{\sum_b e^{\eta Q(s_t, b)}} \tag{2.3}$$

, where $a^{(t)}$ is the action selected by the expert, and b iterates over other

possible actions. This is equivalent to maximizing the log of this function.

$$\max_p \log \prod_t \frac{e^{\eta Q(s_t,a_t)}}{\sum_b e^{\eta Q(s_t,b}} \tag{2.4}$$

$$= \max_p \sum_t (\eta Q(s_t, a_t)) - \log \sum_b e^{\eta Q(s_t,b)} \tag{2.5}$$

$$= \max_p \sum_t (\eta \sum_i Q_i(s_t^{(i)}, a_t, p^{(i)}) - \log \sum_b e^{\eta \sum_i Q_i(s_t^{(i)},b,p^{(i)})}) \tag{2.6}$$

In some cases, we may want to add a regularization factor. In general, we modify the objective function to be as follows.

$$\max_p \sum_t (\eta \sum_i Q_i(s_t^{(i)}, a_t, p^{(i)}) - \log \sum_b e^{\eta \sum_i Q_i(s_t^{(i)},b,p^{(i)})}) - \lambda ||p'||_1 \tag{2.7}$$

, where p' is a vector such that $p'_i = p_i \mid\mid 0$. λ is the regularization parameter. So we can regularize some supports of p. For example, in a driving task, human subjects can only pay attention to certain number of modules. To immitate such behavior, we may force the reward vector to be sparse. In this way, we can define many possible modules and feed them into the algorithm, the algorithm will determine which are the non-significant modules, which would have rewards of 0.

We notice that this function may not be convex. So we use a nonconvex optimization solver. Concretely, we use evolution strategy with covariance matrix adaptation (CMA-ES) [6] for the experiments in the evaluation chapter.

Chapter 3

Literature Review

3.1 Task Decomposition

Modularization is one approach to reduce the dimension of the state space. In this section, we will discuss some major approaches of MDP decomposition as solutions to the curse of dimensionality, and analyze their similarity and difference compared to the modular approach. We will discuss a hierarchical approach and a layered approach.

Some complex tasks have explicit phases. Take the taxi domain for example (Figure 3.1). The taxi driver picks up a passenger from a location, and drives and drops him at a different location. This task requires some shared low-level skills.

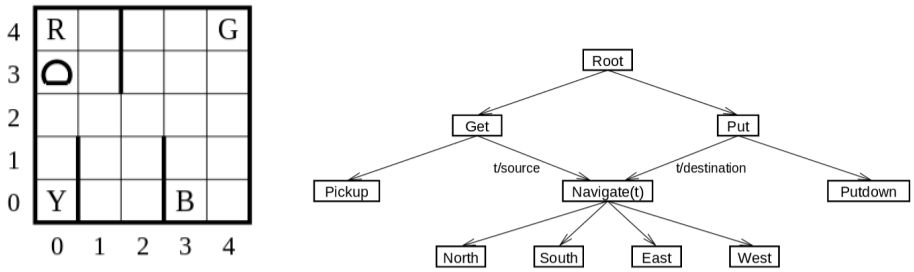


Figure 3.1: (Left) Taxi domain. (Right) A decomposition of the task.

This is generally known as hierarchical reinforcement learning (hierar-

chical RL) [4]. We assume that there is a known structure of the task, and the learning is decomposed hierarchically. At a higher level, an action can be which lower level task to execute. For example, in the Get task, the two actions available would be Pickup and Navigate. At the bottom level, primary actions are accessible.

A similar approach is skill chaining [8], which can be seen as a hierarchical RL approach. Figure 3.2 shows an example that a robot navigates in a domain with obstacles. The red circle is the goal. The robot starts from the left-bottom corner. It has several trained skills, represented by different colors. The robot learns which skill to use, and when to terminate and switch to another skill.

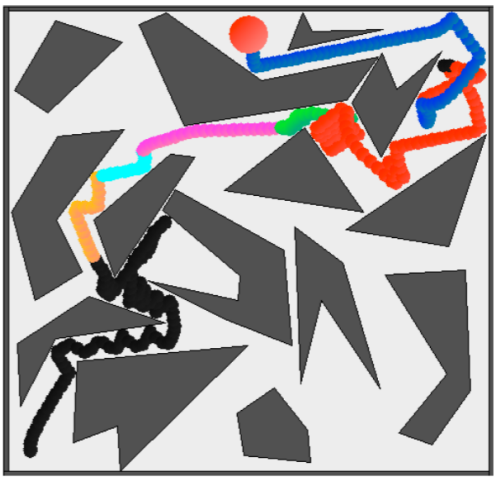


Figure 3.2: Skill chaining in robot motion.

Hierarchical reinforcement learning is different from modular reinforcement learning in a way that the former approach doesn't involve concurrent

modules (or subtasks, skills, depending on the context). For example, the Pickup module doesn't run parallel with Navigate module. Therefore, there is no compromise in policies proposed by different modules.

Apart from hierarchical RL, layered learning is another approach to decompose a complex task [17]. This is a general learning method that may not restricted to reinforcement learning. Similar to hierarchical learning, it also assumes a hierarchy of the task. A module has some parameters. The optimization startes from the bottom layer. When the parametres in modules in the same layer are optimized, the optimization for the upper layer starts. A module will use the modules in the lower layers.

One example is learning in robot soccer. In Figure 3.3, each block is an module. The number in the parenthesis is the number of parameters involved in each module. The arrows indicate the dependency relations between modules. Training starts from the bottom layer, then propagate upwards to higher level layers. A recent work in layered learning makes some parameters in lower level modules available to change in the process of training of a higher level module [9]. This is the same idea as parameterizing modules in modular inverse reinforcement learning.

Layered learning is similar to modular reinforcement learning, as the modules are run in parallel. However different modules correspond to different controllers or agents. They need to compromise to contribute to a higher level module, but not sharing a uniform output interface.

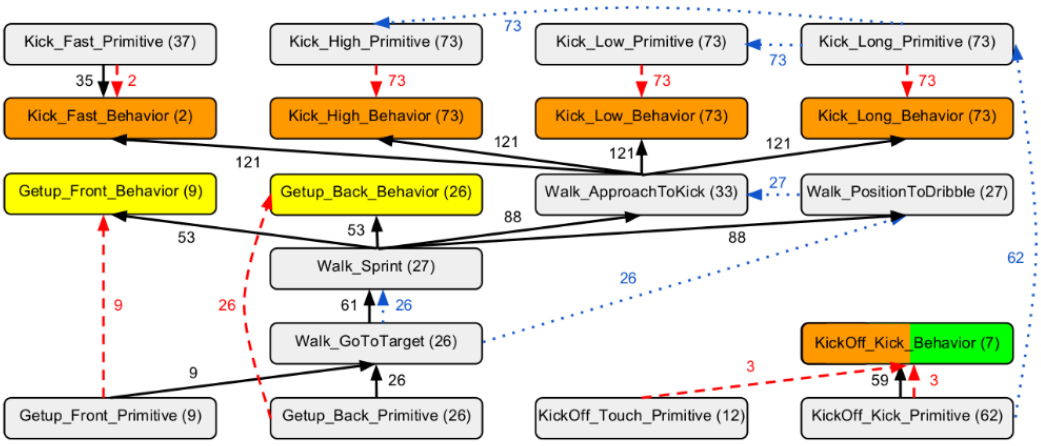


Figure 3.3: Layered learning in robot soccer. [9]

3.2 Inverse Reinforcement Learning

We introduced basic concepts of inverse reinforcement learning in the last chapter. In this section, we discuss some advances in IRL and compare them with our methods.

The original IRL algorithm is formulated by [1, 12]. A popular following work is Bayesian inverse reinforcement learning (Bayesian IRL) [13]. It is motivated by the observation that rewards are not completely arbitrary. We can reasonably assume some prior knowledge of the rewards. Bayesian IRL assumes that the rewards are drawn from a prior distribution. 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. Concretely, according to the

Bayes rule,

$$P(R|O) = \frac{P(O|R)P(R)}{P(O)}$$

, where O is the observed data, R is the reward vector. Bayesian IRL assumes $P(R)$ to be non-uniform.

Other recent developments in Bayesian IRL include [2, 3, 5]. [2] uses MAP inference to solve the Bayesian IRL problem. [3, 5] tackle the problem of recovering rewards in domains with multiple tasks or multiple rewards.

Chapter 4

Evaluations and Applications

4.1 Preliminary Evaluation

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

4.1.1 Sanity Check

We test the algorithm in a grid world domain with 2 modules. Each has several objects. Different modules have different parameters, including rewards and discount factors. The objects of same module share the same parameters. The agent only considers the closest object in each module as its state space. It can obtain the corresponding reward when reaching in a grid with an object.

According to the definition of the global function, $Q(s,a) = Q^{(1)}(s^{(1)},a) + Q^{(2)}(s^{(2)},a)$. Where $s^{(1)}, s^{(2)}$ are the distances to the closest objects in each module.

We show the heatmaps of the value functions of some sample domains

with one in Figure 4.1. This helps us to visualize the effect of different parameters with same configuration of objects. For simplicity, we only show the case with one object in each module.

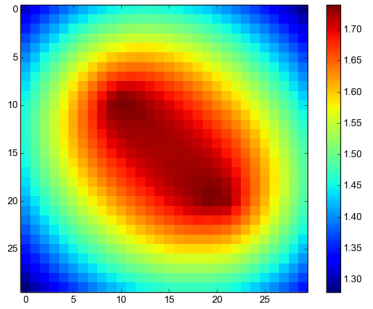
We use the modular Q function and observed policies on all the states as input to our modular IRL algorithm, and run our algorithm to recover the underlying parameters for both modules. The dimension of the gridworld is 20×20. There are 5 objects in each module. The results are shown in Table 4.1. We can observe that in some cases the discount factors are not recovered correctly. This happens when a range of parameters can explain the same observed policy.

	r_1	r_2	γ_1	γ_2
Task 1, Ground Truth	1	1	0.9	0.9
Task 1, Recovered	1.0	1.0	0.87	0.91
Task 2, Ground Truth	1	1	0.9	0.1
Task 2, Recovered	1.0	1.0	0.89	0.01
Task 3, Ground Truth	1	-1	0.9	0.9
Task 3, Recovered	1.0	-1.0	0.86	0.54

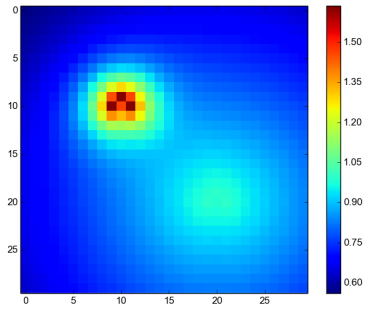
Table 4.1: Ground truth and recovered parameters in three gridworld domains.

4.1.2 Comparison with Non-Modular Inverse Reinforcement Learning

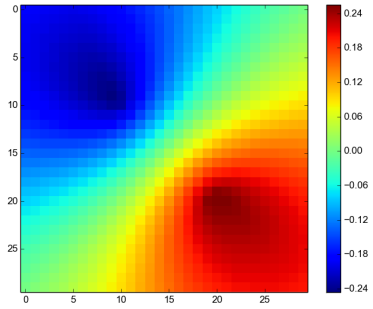
We compare our algorithm with non-modular Bayesian inverse reinforcement learning [13] to demonstrate the sample efficiency advantage of the modular approach. In Bayesian IRL, we assume a prior distribution of the reward function. We use a Laplacian prior in Bayesian IRL since the rewards



(a) $r_1 = 1, r_2 = 1, \gamma_1 = .9, \gamma_2 = .9$



(b) $r_1 = 1, r_2 = 1, \gamma_1 = .9, \gamma_2 = .1$



(c) $r_1 = 1, r_2 = -1, \gamma_1 = .9, \gamma_2 = .9$

Figure 4.1: Heatmap of value function in a navigation domain. The red color indicates higher value. There are 2 modular objects. Object #1 is in upper-left and Object #2 is in lower-right. Rewards and discount factors are specified for these two objects (as $r_1, r_2, \gamma_1, \gamma_2$).

are sparse. Because traditional IRL algorithms can only recover rewards, we make our modular IRL algorithm to recover the same.

In Figure 4.2, we report the sample efficiency of modular IRL versus Bayesian IRL. There are 4 modules and each has 4 objects. 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 policies as the ground truth. We use the metric of policy agreement in our comparison since the outputs of these two algorithms 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.

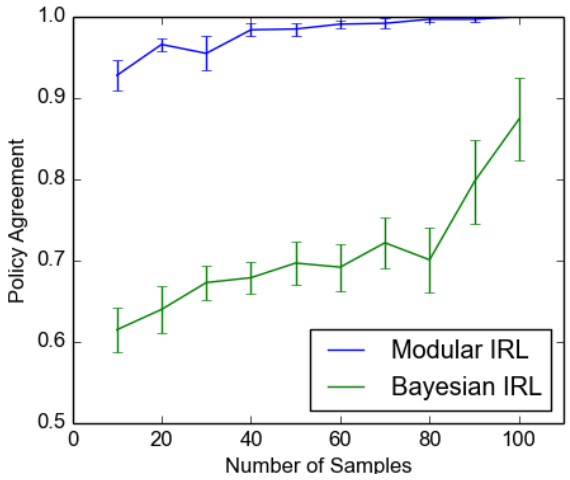


Figure 4.2: Modular IRL vs Bayesian IRL on sample efficiency, measured by policy agreement.

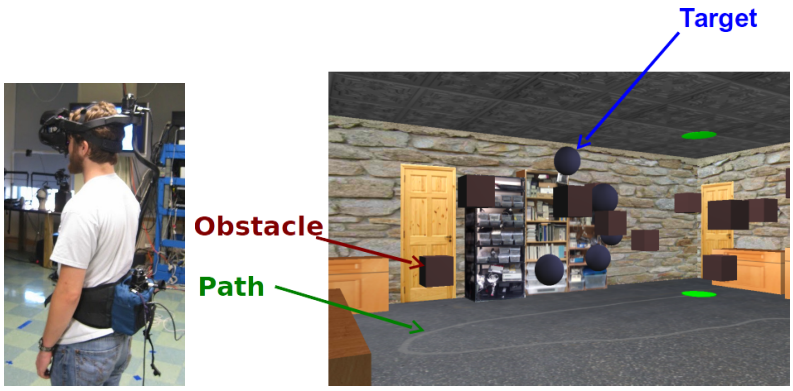


Figure 4.3: The second test domain. (Left) A human subject wears a head mounted display (HMD) and trackers for eyes, head, and body. (Right) The virtual environment as seen through the HMD. The red cubes are obstacles and the blue spheres are targets. There is also a gray path on the ground which the human subject were told to follow.

4.2 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 4.3 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 [14] and to study human navigation and gaze behavior [19].

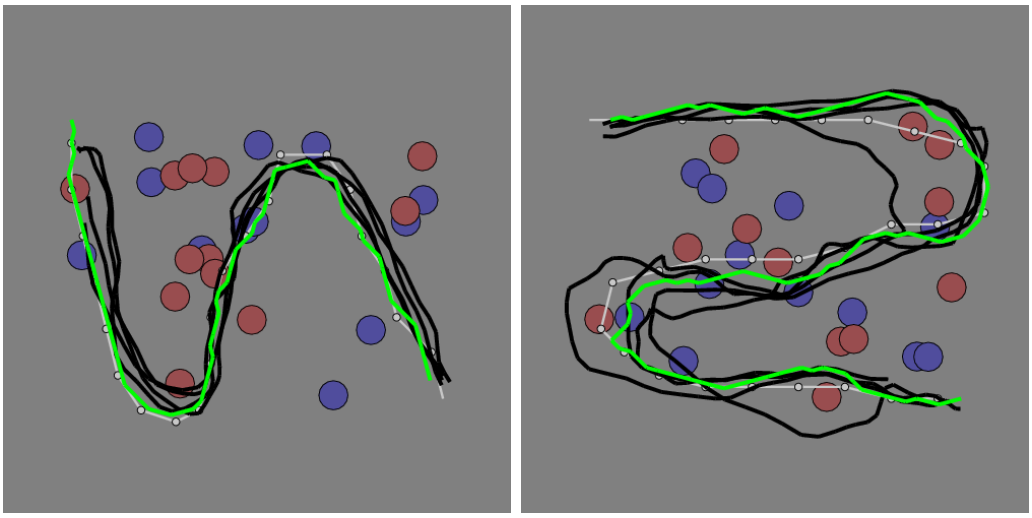
We gave subjects four types of task instructions, resulting in 4 experimental conditions:

- **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.

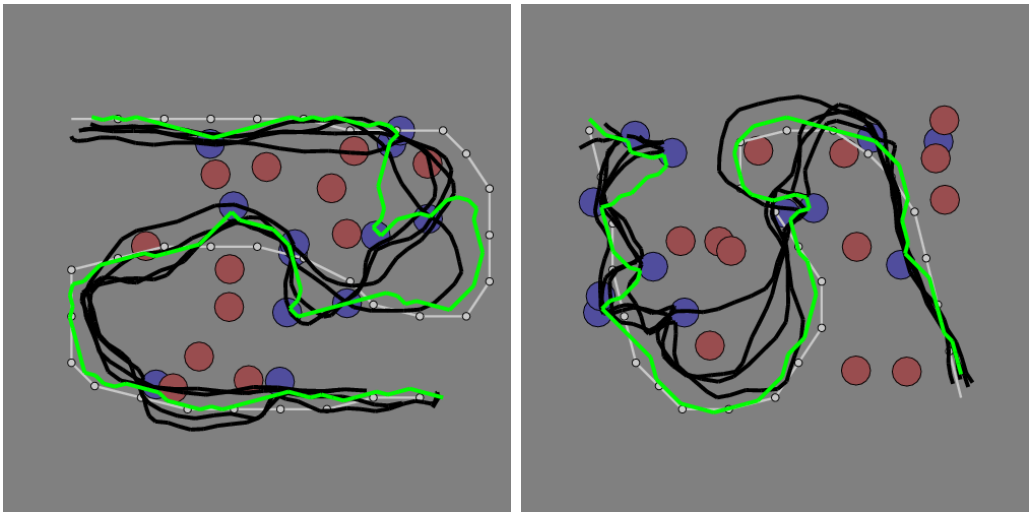
We use Equation 2.5 as our objective function to recover r and γ. Our agent uses the distance and angle to the module instances as state information. We constrain the action set of the agent to be human-like; it takes discrete forward actions, ranging from turning left 90 degrees to turning right 90 degrees.

The results are shown in Figure 4.4. It is clear that the estimated r agreed with our task instructions. We then trained an agent with the recovered r 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 4.4.



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

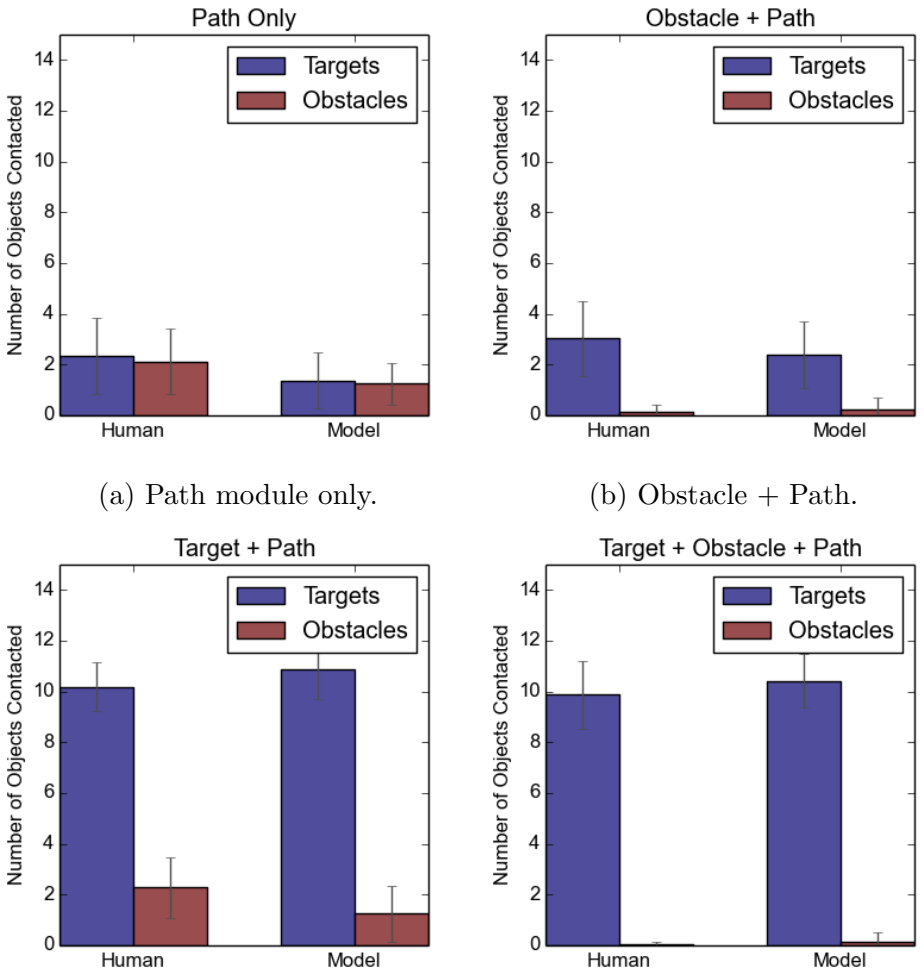
(b) Path + Obstacle
Rewards: [.000, **-.227**, **.773**]
Discount factors: [.900, .134, .618]



(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 4.4: 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.



(a) Path module only.

(b) Obstacle + Path.

(c) Target + Path.

(d) Target + Obstacle + Path.

Figure 4.5: Number of targets hit and number of obstacles hit of the human subjects and the agent.

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

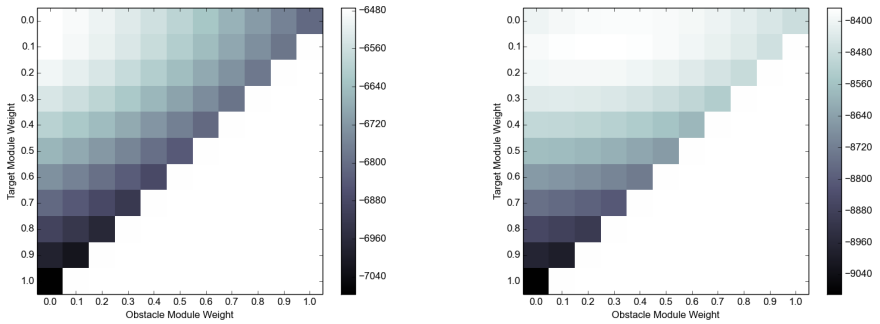
We compared the performance of our agent with two baseline agents, shown in Table 4.2. The *Random Agent* takes an action randomly without considering state information. The *Reflex Agent* greedily chases the nearest target or avoids the nearest obstacle, depending on which is closer. We considered two 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 4.2. 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.

Figure 4.5 evaluates the performance by showing the number of targets hit and number of obstacles hit. The number of targets hit should be large, and the number of obstacles hit should be small, when the corresponding module is active. We can observe that humans still do better than the agent in these tasks. Nevertheless, the agent performance is quite similar to that of the human subjects.

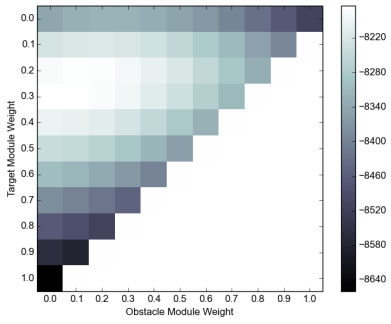
It is of interest to see the objective function value of different rewards. To present the rewards, we use the term *weights* to represent the normalized rewards which sum to 1. For example, the reward vector of (1, -1, 2) corresponds to the weight vector (0.25, 0.25, 0.5). It would be difficult to visualize the whole solution space because of the dimensionality. We report the results for different rewards, but with the discount factors fixed to be the solved optimal values. This is shown in Figure 4.6. The colors represent the log of values of Equation 2.5 for different weights. The white color represents higher probability. We can observe the centroids of white zones move for different tasks. It stays at the origin in Task 1, so neither the target module nor the obstacle module is active. It moves away from the origin when a module is active. From the heatmaps, we find that optima exist for these tasks and match our solutions. We also find out that with the discount factors fixed, the solution space is convex. This can be verified in Equation 2.5.

We then look at differences of r and γ within subjects and between subjects in Task 4. The results are shown in Figure 4.7. Within-subjects

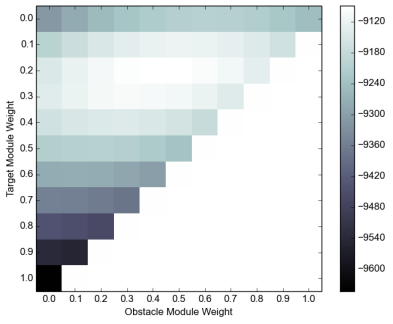


(a) Path following only.

(b) Obstacle + Path.



(c) Target + Path.



(d) Target + Obstacle + Path.

Figure 4.6: Heatmaps of the log of the values of Equation 2.5 for different rewards for the four tasks, respectively. The white zones indicate higher probabilities. The weights of all three modules sum to 1, so we only show the weights on the target and the obstacle modules.

consistency indicates if the same subject has similar r and γ in different trials of Task 4, measured by the confidence interval (the errorbar). Between-subjects consistency indicates if different subjects have similar r and γ on average in Task 4, measured by the mean value (the height of bar).

First, we observe that both rewards and discounters are similar for each module for all the human subjects. For example, the discount factors for the path module are generally large, which means that the value is still large even not strictly following the path. This intuitively matches human's behavior on path following. On the contrary, the discount factors for the target module are small, because people want to make sure that they reach the target and get the positive reward. The value is small even slightly off the target. For r, 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.

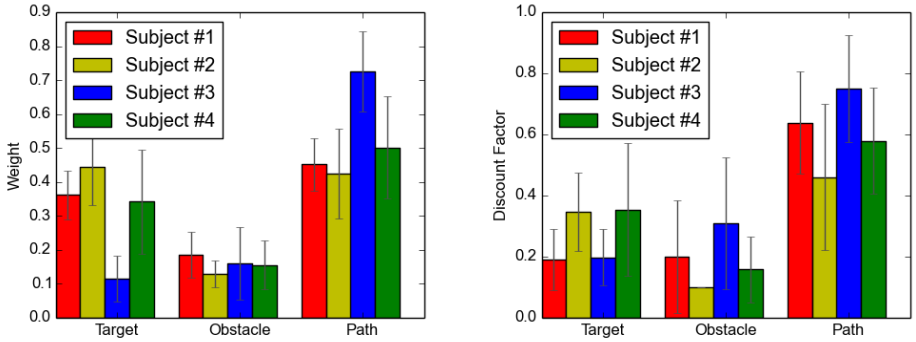


Figure 4.7: The weights (normalized rewards, on the left) and discount factors (on the right) of different human subjects in Task 4. The error bars are 95% confidence intervals.

Chapter 5

Conclusion

In this thesis, we followed an earlier work on modular inverse reinforcement learning. We proposed a new modular inverse reinforcement learning algorithm, and applied it to collected human subjects' data to analyze their behavior. The experimental results show that modular reinforcement learning can explain human's navigation behavior. By our evaluation metrics, this method is better than other baseline assumptions.

We have discussed in the literature review that modular approach is just one way to combine multiple sub-MDPs. There are other assumptions that can be evaluated in the future work. For example, scheduling between different modules, with only one active at one time. This is close to the skill switching method [8]. However, we adopt the weighted sum approach because this is more reasonable for human behavior. When a human tries to collect targets while avoiding obstacles, these two modules are expected to be both active. A scheduling approach may yield frequent oscillation between these two modules.

Static combination of modules throughout a task is another assumption that should be removed in the future work. Parameters may be dynamic and

different from state to state. However, with such an assumption we need to learn a mapping from state to parameters. In this case, the curse of dimensionality still exists, and inverse learning would be difficult.

So far, we only used the motion data of human subjects. There is also gaze data collected but not used. Gazes are the points that human subjects are looking at at a time step, which indicate humans' attention. This is actually a useful information on how human combines different modules. When a human is paying attention to a module, it is reasonable to assume that his action would be affected mainly by this module.

In sum, we show in this thesis that modular reinforcement learning is a promising explanation for human's navigation behavior. It is of interest to see its analysis on different humans' behavior, and using the recovered parameters for a learning agent to tackle similar tasks.

Appendices

Bibliography

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

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

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

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

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

[6] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.

[7] Anders Jonsson and Andrew Barto. A causal approach to hierarchical decomposition of factored mdps. In *Proceedings of the 22nd international conference on Machine learning*, pages 401–408. ACM, 2005.

[8] George Konidaris and Andre S Barreto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023, 2009.

[9] Patrick MacAlpine, Mike Depinet, Jason Liang, and Peter Stone. Ut austin villa: Robocup 2014 3d simulation league competition and technical challenge champions. In *RoboCup 2014: Robot World Cup XVIII*, pages 33–46. Springer, 2015.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[11] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

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

[13] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement

learning. In *Proceedings of the 20th IJCAI*, pages 2586–2591. Morgan Kaufmann Publishers Inc., 2007.

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

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

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

[17] Peter Stone and Manuela Veloso. Layered learning. In *Machine Learning: ECML 2000*, pages 369–381. Springer, 2000.

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

[19] 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.

[20] Ruohan Zhang. Action selection in modular reinforcement learning. 2014.