Machine Approaches to Epistemic Learning with Application to Decision Strategy Recognition

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Qifeng Qiao

May

2012

APPROVAL SHEET

The dissertation

is submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Qifeng Qiao

AUTHOR

The dissertation has been read and approved by the examining committee:

[illegible]

Advisor

[illegible]

[illegible]

[illegible]

Gang Tao

Accepted for the School of Engineering and Applied Science:

[illegible]

Dean, School of Engineering and Applied Science

May

2012

abstract

In sequential decision problems, the forward planning problem is one in which an agent seeks to find a decision strategy that yields a sequence of actions that achieve a given goal. The inverse process, which is sometimes called plan or goal recognition, is to infer the goals of an agent on the basis of observations of its actions. One of the more general goal recognition problems may be framed in the context of reinforcement learning. The inverse reinforcement learning problem (IRL) is to determine the reward structure perceived by an agent on the basis of observations of its actions. A number of approaches have been proposed for IRL, but many rely on the assumption that the reward function can be linearly approximated, which may not be reasonable for many problems of practical interest. The ill-posed nature of the inverse learning problem also presents difficulties. Multiple reward functions may yield the same optimal policy, and there may be multiple observations at a state given the true reward function.

The focus of this dissertation is the design and analysis of algorithms for IRL to address several of the principal limitations of existing methods and to extend the domains of application for IRL. Specific areas of contribution include: (1) development of a non-parametric probabilistic model in a Bayesian framework that quantifies inference uncertainty and provides a prediction model in closed form, (2) introduction of the idea of using reward functions recovered through IRL as a signature for decision

agents in supervised or unsupervised learning of agent identity and decision strategy, (3) development of algorithms that can learn from supervised information from observed states and unsupervised information from unobserved states. IRL algorithms can present many computational difficulties. We prove that the principal computational procedures in IRL under a Gaussian process model can be formulated as convex optimization problems, and thus solved using efficient algorithms. We also develop a minorization-majorization approach to the computation of likelihood functions. For a variety of standard test problems, we present empirical results for experiments comparing our proposed methods with algorithms from the recent literature.

The study of behavior recognition, based on inverse reinforcement leaning, has considerable significance in many potential application areas that involve human or machine behavior in an interactive environment.

Acknowledgments

I owe a lot of gratitude to many people for their support of my research. First and foremost I would like to express the deepest gratitude to my advisor, Prof. Peter Beling, for not only guiding me in scientific research but also supplying me with help and knowledge to succeed in life. Prof. Bling directs my research towards interesting problems and innovative solutions, encourages the challenging discussions, always keeps patient, and gives me enough freedom to feel empowered by the journey. Particularly, I would like to thank him for spending a lot of time on patiently revising my writing, and being always available for me to discuss the problems even on weekends. He never said anything negative about my research and always encouraged me to explore the new ideas, which lets me stay in a fresh and healthy atmosphere.

I also thank Prof. Barry Horowitz, Prof. Steven Patek, Prof. Randy Cogill and Prof. Gang Tao for serving on my dissertation committee and providing their valuable insights and suggestions on my research. I am grateful that every time when I come to meet them, they spend time to discuss with me about the research and provide many valuable suggestions.

Many thanks to the people at Wireless Internet Center for Advanced Technology, which is directed by Prof. Barry Horowitz, for providing good facility and research environment for varieties of projects. I not only spend a lot of time doing research there but also meet good people to explore the unknown together. I thank the memorable time working with Jianping Wang, who is always at your hand when you need help, Jorge Alviar, Mingyi Hong, Chenyang Li, Roy Hayes and Jonathan Bao who lets me know an interesting research topic.

It was great fun to discuss academic questions with my officemates Zhou Zhou, Steve Yang and Kanshukan Rajaratnam. I am also grateful to Katherine Holcomb who works for cluster computing at UVA and is always willing to help me use clusters.

I owe gratitude to the people working in our department. Thank Jennifer Mauller for always helping me promptly and explaining the policies and questions patiently and clearly.

I would like to thank my wife, Ruizi Liu, and our parents, who are always supporting me and let me be able to completely focus on the research. My son, James, who was born during my doctorate study, encourages me to face more challenges. My wife and our parents have taken all the responsibility for the housework and care of baby. Without their help, I will not have so much time to work on my research.

"If we knew what it was we were doing, it would not be called research, would it?"

- Albert Einstein

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence(AI) aims to construct artifacts (automated decision systems) that are capable of performing tasks with a degree of intelligence that approaches that of human beings. Here we may define intelligence to be the ability to make decisions that form a successful solution to a problem. In contrast to older AI programs that mimic human behaviors using hard-coded rules, machine learning tools attempt to learn from demonstrations. Demonstration-based learning offers the promise of decision agents that, in effect, program themselves, adapting decisions and strategies to new circumstances in much the same manner as humans do. Conventional rule-based approaches are coded in action space. Machine learning approaches, by contrast, may be termed *principal-based* as they are generally defined in goal space. There is ever mounting empirical evidence in support of a hypothesis that principles may be more important than programmed rules as a means for achieving artifacts that might reasonably be called intelligent. Theoretically, too, the abstract nature of a principle-based artifact should leave it less vulnerable to environmental disturbances than a rule-based artifact, of necessity conceived and precisely specified for particular circumstances. Artifacts may never beat or even match human intelligence, but in many situations they can provide satisfactory solutions that approximate optimal behavior.

One may define a cognitive agent to be one that typically purposeful and adaptive to new information in the learning process. The cognitive decision processes of humans include: observing the environment, reasoning other agents' behavior and goals, thinking in minds, the selection of actions or plans, and the execution of plans [3]. We posit that to realize the intelligence that is "human like", artifacts will need to possess similar goal-directed, adaptive reasoning capabilities. These capabilities are necessary for enabling the artifacts to infer and understand, by observation, the underlying reasons why the agent takes the particular sequence of actions to solve a problem. Therefore the intelligent agent becomes robust to environmental changes, supporting the transfer of learned knowledge across similar domains.

A rational being would make decisions by computing the expected value or utility and choosing the alternative with maximum value [4]. The actual decision processes of humans, however, are not so simple. Human decision makers tend to choose the first satisfactory alternative instead of optimizing for the best alternative [5]. Different people often reach different decisions on the same problem, which may be attributed to biases in the decision making process [6, 7]. Because biases are widespread, goal-oriented decision-makers may be recognized, or discriminated from one another, on the basis of their decision principles. In artifacts that mimic human intelligence, then, one might look for the similar, recognizable distinctions in principles.

The past several decades have seen enormous volumes of research on the development of computer systems capable of performing intelligent tasks. Related areas of artificial intelligence research include planning and problem solving, machine learning, knowledge representation methodologies, and multi-agent systems. Unfortunately, integration across these areas of artificial intelligence has been lacking. Moreover, many current machine learning methods have little connection to planning and decision-making frameworks, though such frameworks have been the setting for many problems and experiments involving sequential decision making and interaction with human decision makers. In existing machine learning research in supervised learning, for example, it is common to consider behavior in the form of labels as a set of random variables without explicit models of the context in which the decision is made and how it relates to other alternative actions and the goals of the agent. Thus, many existing approaches lack an orientation toward the goal-directed and adaptive reasoning that is characteristic of high-level human intelligence.

There may be enormous potential benefit in research that integrates machine learning and planning frameworks; many real-world problems, such as robotics, semantic web searches, recommendation systems, and financial trading analysis, have proven difficult under either paradigm individually but might admit more progress under an integrated approach. We hypothesize that some of the most fruitful connections between machine learning and decision making will involve consideration of interactions between humans and machines and between machines and machines. Important research areas include identification of the limits of manual control, knowledge capture from human experts, and the development of new methods by which machines can learn from machines.

In this dissertation, we develop probabilistic models that enable inference of agent goals from observation of behavior. We model the problem faced by the decision agents as a Markov decision process (MDP) and model the observed behavior of the agents in terms of degrees of rationality with respect to optimal forward planning for the MDP. Within the MDP framework, we infer goals from observations using Bayesian models. The Bayesian framework for probabilistic inference provides a general approach to understanding how problems of induction can be solved in principle. This framework has been widely used to study the human mind in cognitive science, including Bayesian models to address animal learning [8], human inductive learning and generalization [9], and action understanding [10, 11]. The essential idea of our method is that a Bayesian probabilistic model in the MDP setting creates an inference model of decision making that is goal-oriented and adaptive and with uncertainty, providing effective approach to predict the behavior and recognize the decision rules hidden in incomplete observations.

1.1 Motivation

The growth in computing power and in data collection seen in the last decade has important implications for artificial intelligence. Embedded computers and ubiquitous sensing provide the basis for detailed description of the environments in which decisions are made. Personal portable devices connected to the Internet make it possible to record and retrieve vast amounts of behavior data. In our view, there is a clear need for new analytic methods that would contribute to understanding behavior through inference of the goals behind the observations. Possible objectives and benefits of technologies for learning the goals in a decision-making process include:

- Apprenticeship learning, in which the objective is to learn the decision policy of the agent. The decision policy could be used to predict future behavior under environments not yet observed. Such capability would support automatic creation of artifacts, e.g., robots that learn how to perform tasks through observation of demonstrations by experts.

- Goal-based filtering, in which knowledge of the user's interests is used to filter databases. Such capability would support recommendation engines. More generally, knowledge of user goals could be the basis for automated systems that interact with human beings to provide service according to individual preference.
- Decision strategy recognition, in which the learned goals are used as the basis for classifying or clustering users. Such capability would provide a robust way of identifying decision makers on the basis of observed actions.

1.2 Intelligent Agents and Theory of Mind

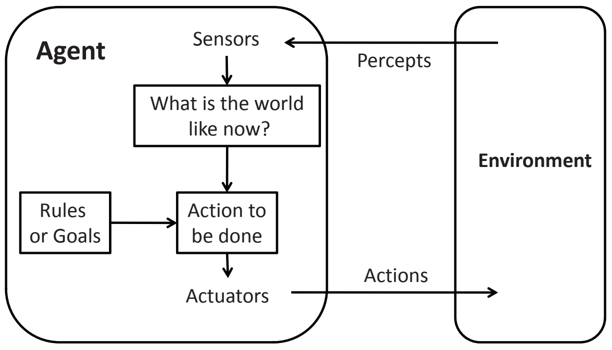


Figure 1.1: Goal-oriented agents interact with environment

An *agent* can be a person, a computer or a robot, anything that is able to perceive the environment through sensors and determine the actions that will act upon the environment. The long term objective of artificial intelligence is to develop machine agents that have "human like" perceptual, planning, reasoning and understanding capabilities. An agent is *intelligent*, if it has some of these characteristics. A simple idea for an intelligent agent is illustrated in Figure1.1, taken from [12].

Let us first review *behavior* in decision making. In [13], behavior is analyzed in hierarchy types. The first level of behavior is divided into the active and the passive, which are determined by whether the behavior outputs or inputs energy. Active behavior is further classified into purposeful and random. Purposeful behavior means that the agent is goal-oriented. For goal inference, the behavior we are interested in is active and purposeful. For the purposes of learning, observed behavior is useful only if it contains knowledge of goals and preferences.

Goals are simply states of affairs that the agent desires to achieve. The desirability is captured by a performance measure that evaluates the given sequence of environmental states. Note that we say environmental states, not agent states. Obviously, there is not one fixed measurement method for the different tasks. But as a general rule, the performance measure should be designed according to the environmental states, making evaluations on goals, instead of how one thinks the agent should act. In the cognitive architecture of a rational agent, the actions are determined for achieving the goal in the long term, rather than receiving the current reward. The *preferences* generally mean that humans can tell introspectively that they prefer one type of situation to another.

Russell and Norvig [12] have considered that the goal-directed behavior as the essence of intelligence and borrowed the conception of *rationality* from economics to describe the good behavior. We use their definition of a rational agent.

Definition (Russell and Norvig [12])

For each possible percept sequence, a rational agent should select an action that *is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*

This definition captures four things: the performance measure that defines the criterion of success, the agent's prior knowledge of the environment, actions the agent can perform and the percept sequence to date. However the perfect rationality is often not supported by the agents in practice. Herbert Simon [5] writes

> The capacity of the human mind for formulating and solving complex problems is very small compared with the size of the problems whose solution is required for objectively rational behavior in the real world - or even for a reasonable approximation to such objective rationality.

Simon suggests a notion of *bounded rationality.* The computer agents may display perfect rationality due to design features and limitations on computational resources. So a bounded rational agent behaves as well as possible, given its computational and knowledge resources. Note that the performance measurement is to evaluate the criterion for goals, not for the optimal solution. Thus we consider the agent in practice as a satisficer, one seeking a satisfactory solution rather than the optimal one. The satisfactory solution may often be near optimal, if the cost of decision-making process has been considered. In our work, we propose bounded rationality as the theoretic foundation to overcome the limitations of the rational-agent model in MDP that we rely on for explaining the observed behavior.

The above discussions related to the intelligent agents are mainly focused in the perspective of forward planning, which is how to make good decisions. However, another property is also a key to highlight human intelligence. Reasoning is the ability to infer intentions behind observations. How do humans understand and infer other agents' desires? The emerging Theory of Mind (ToM) is of interest for the understanding of normal human psychology. ToM refers to a broad areas studied in psychology and cognitive science. An informal definition given in [14] is:

> Theory of mind refers to the ability to explain and predict one's own actions and those of other intelligent agents (that is, agents that, unlike physical objects, are the causes of their own actions)

Understanding behaviors implies being able to infer that behaviors are induced by mental states, which in turn implies having ToM. Humans with ToM make inferences about and interpret others' behavior in terms of their mental states, such as goals, and beliefs. Inference of goals is a core area of ToM. Several authors have recently reported models for how people infer the goals or preferences as a kind of inverse planning [10, 15]. These models depend on the principle of rational behavior and formalize this principle at the heart of classical ToM. The key idea is that the purposeful agents are expected to choose the actions that reach their goals as effectively as possible, i.e., maximizing the expected utility. Goals or preferences are inferred by the criterion that the expected utility of observed action is maximized. Two key properties of ToM are self-propulsion and goal-oriented behavior. This means that the autonomous agents, learning how to infer like humans, are active and purposeful.

Now we have introduced some theory related to the intelligence in human decision making and goal inference. These theories drive our work on developing mathematical models to make the autonomous agents gain such 'human-like' intelligence. In the machine learning domains, we call these models *epistemic learning*, which refers to understanding of the nature and the principle of knowledge in observation of the interactive decision making process.

1.3 Inverse Reinforcement Learning

In the control field, the idea underlying inverse reinforcement learning (IRL) was first proposed by Kalman in the early 1960s as the problem of recovering the objective function for a deterministic linear system. Other researchers extended this idea as the inverse optimal control problem. Boyd has solved this control problem as a semidefinite program [16]. Another case is to consider a control-Lyapunov function as an optimal value function and find the cost. The theory related to the objective and attributes in decision analysis is well developed and given numerous applications [17]. From the viewpoint of economics, there have been empirical findings relating the behavior of people to structured models from decision theory and planning. Rust, for example, concluded from experiments that a MDP can be a good model of the way humans actually solve sequential decision problems under uncertainty [18]. Sargent developed an MDP model of a firm's decisions on hiring how many workers in a world with randomly changing prices and productivities [19]. This work can be viewed as the first example of structural estimation of MDP, term the "inverse stochastic control problem", that aims to not only uncover the form of the stochastic process generating the state-action pairs, but also the parameters that determine it.

The problem of IRL was first informally proposed in machine learning community by Russell [20] with the purpose of modeling natural learning using reinforcement learning (RL) ideas. Before applying RL algorithms, the reward function should be determined a priori and fixed. However, in some conditions the reward functions are unknown. Therefore, IRL is proposed in [20] to compute the reward function that is consistent with these given constraints: (1) measurements of an agent's behavior over time, (2) measurements of the sensory inputs to the agent, (3) a model of the physical environment. Here the measurements of (1) and (2) provide the observation of action and state respectively, which is sampled in the form of a state-action pair.

Markov decision processes provide a framework for modeling sequential decision-making, in which the objective is guided by the reward over state-action pairs. MDPs are well solved via dynamic programming and reinforcement learning. But uncertainty of the model parameters is the main bottleneck in these problems. Recent work called inverse reinforcement learning using machine learning tools has contributed positively to the process of eliminating this bottleneck. There have been some approaches being proposed for IRL in various applications. Ng [2] poses the IRL problem in the settings familiar to the machine learning community with MDP. The key idea there is to maximize the difference between the optimal and suboptimal policies by optimizing the reward function in a linear approximation. Another significant idea for problems of apprenticeship learning is to search mixed solutions in a space of learned policies with the goal that the accumulative feature expectation is near that of the expert [21], [22]. Similarly, a game-theoretic approach is given to solve IRL problem in the setting of playing a two-player zero-sum game, in which the apprentice chooses a policy and the environment chooses a reward function [23]. These methods of apprenticeship learning represent the reward as linear combinations of manually defined (prior) features. The method by which the features are selected has remarkable influence on the final performance. Another algorithm for IRL is policy matching that the loss function penalizing deviations from expert's policy is minimized by tuning the parameters of reward functions [24]. Maximum entropy IRL is proposed in the context of modeling real-world navigation and driving behaviors in [25]. From Bayesian perspective, the state-action samples from the expert are considered as the evidence that will be used to update a prior on the reward functions. Then the reward learning and apprenticeship learning can be solved using this posterior on the reward functions, and a modified Markov Chain Monte Carlo algorithm used to do the inference tasks [26].

1.4 Our Contribution

Learning goals from observed behavior in sequential decision making is challenging, as much uncertainty inherently exists in the inference process. There are several sources of uncertainty and difficulty in this learning problem, including:

- Goal inference is an ill-posed problem, because there may be multiple functions that can generate the same behavior observed. In turn, there may be multiple actions observed at one state given a reward function.
- In practice, the observed behavior is often incomplete, representing only a sample of the true policy. In MDPs, behavior is the choice of an action at a given state. In some real-world problems, the number of state-action examples observed may be too small for effective inference.
- The design of state features has a significant effect on performance guarantee of learning the goal from the behavior observed. How to design state feature is disputable.
- In IRL, the reward function is widely assumed as a linear function. Though it is helpful for making the computation efficient, real-world problems often violate this assumption. Therefore performance accuracy may be degraded.

We study new algorithms to deal with the above challenges. The main contributions of this thesis are as follows:

- **Probabilistic model with Bayesian rule** provides an approximation solution to infer the reward function more efficiently and effectively, particularly when the size of the observation data is small. We consider the reward function to be a stochastic process, such as Gaussian process, which allows for a tractable problem formulation and enables inference of a distribution with estimation of the mean and the variance. This is a general approach for modeling observed behavior that is bounded rational and purposeful, and may be applied in finite or continuous domains.

- **Preference graph in decision making** considers the ambiguity of observed policy in IRL with sampled decision trajectories. It deals with the condition that observed agents make decisions with uncertainties.

- **Semi-IRL** resolves the practical challenge when observations are incomplete and small in number. This work borrows the idea from semi-supervised learning that knowledge of the existence of states can help reduce the uncertainty in estimation, even though we do not have observed behavior at these states. In addition, previous IRL methods based on a Boltzmann distribution model of behavior require simulation of the value function during the inference procedure. Such simulations may present an overwhelming computational burden in the case of real world problems. To overcome this difficulty, we propose using the optimization techniques of majorization and minorization to make the computation more tractable and efficient. We prove theoretical convergence and report on a number of empirical studies.

- **On-line IRL** expands the probabilistic model to settings with addition uncertainty in observation, such as goals changing during observation, the actions observed being suboptimal or in adaptive learning process. This model validates the method of using MDP to model the observed behavior and using its reward function to represent the agent's goal.

- **Decision strategy recognition** studies the promising feasibility of using reward function to characterize intelligent agents. The idea is that an agent is described in a MDP environment and its decision strategy is represented by the reward function. IRL is considered as a tool to compute the features that describe the agent's decision strategy. The recognition tasks of clustering and classification have been validated in experiments on intelligent agents, including machines and simulated humans.

The central objective of proposed research is to study machine learning methodology for goal inference and pattern recognition of decision strategies in sequential decision-making. The approach to behavior recognition proposed here can be applied in various fields, such as abnormality detection, recommendation systems, robotics, personalized semantic webs, and financial trading.

1.5 Thesis Organization

The remainder of this dissertation is organized into six chapters. In Chapter 2, we review the decision-making frameworks that pose behavior as a utility maximization interaction with a stochastic process, as well as give problem formulations and a review of relevant previous work. In Chapter 3, we propose IRL methods designed for simple environments in which complete observations are available and a finite state space is appropriate. In Chapter 4, we propose Bayesian IRL with Gaussian process, which is a general approach to IRL designed to deal with infinite state space and incomplete observations. In Chapter 5, we propose Semi-IRL, a methodology that features a majorization-minimization approach to likelihood computation. In Chapter 6, we define the decision strategy recognition problem and propose a new algorithm for the problem. We also present a comparative empirical study. Finally, in Chapter 7, we offer conclusions and suggestions for future research.

Chapter 2

Background and Related Work

Epistemic learning relies upon existing concepts and ideas from machine learning and statistics. Specifically, the major theoretical work of this thesis is the extension of machine learning methods to sequential decision making processes that involve interaction between decision makers and the environment. We review existing decision-theoretic models of planning and decision making process, and define the model for the problems that we will solve. Lastly, we introduce some of the common notation and summarize the terminology that we employ throughout this thesis.

2.1 Quantifying Uncertainty and Inference

Agents have uncertainty in a decision-making process because the observation may be noisy or partial, and the result of next state after taking a sequence of actions is not deterministic.

2.1.1 Action under uncertainty

Partial, incomplete observability and the nondeterminism may cause uncertainty. Partial observability means that the observed state may be different from the true state of the environment, perhaps as a result of sensor or measurement errors. Incomplete observability means that the observed states are not good enough to describe the environment accurately, perhaps because the number of observations is small compared with the size of state space or the state model is a poor representation of the environment. Some techniques have been designed to handle uncertainties of these types by keeping track of a belief state, which is a presentation of the set of all possible states given the observation. The drawback of this method is that there is no plan to guarantee the achievement of the goal, but the agent must act or make inference. Additionally, computation grows arbitrarily large as the problem size increases.

Consider an example of uncertain reasoning: diagnosing the credit risk of a borrower failing to make a payment as promise. This example comes from a similar discussion in [12]. The logical approach may follow a simple rule: *Default* $\Rightarrow$ *Unemployed*. But this rule may be wrong. Not all the borrowers who default are unemployed. Some of them lose the ability to pay because of spending in other areas, some of them have business troubles, and others may change the original plan. One could follow a casual rule: *Unemployed* $\Rightarrow$ *Default*. But this may not be right either. Not all the unemployed borrowers will default. The connection between unemployed and default is not just a logical consequence in either direction. Instead, the agent's knowledge can at best provide only a degree of belief in the inference. One main tool we can use to deal with degree of belief is probability theory. The epistemological ability of the agent is described in probability, which is a numerical degree of belief between 0 and 1.

To make choices, the agent has preferences among different possible outcomes of the various actions. An outcome is a completely specified state. We use utility theory to represent and reason with preferences. Preferences, as expressed by utilities, are combined with probability theory in a general theory of rational decisions called decision theory. The fundamental idea of decision theory is that an agent is rational if and only if the action is selected by maximizing the expected utility, averaged over all the possible outcomes.

2.1.2 Inference and evaluation

A basic method for probabilistic inference is to compute posterior probabilities for query propositions given observed evidence. Given any sets of variables Y and X, the **marginalization** rule is

$$p(Y) = \sum_{x \in X} p(Y, x),$$

which sums over all the possible combinations of values of the set of variables X. A variant of this rule use conditional probability instead of joint probabilities, which is

$$p(Y) = \sum_{x \in X} p(Y|x)p(x),$$

which is called **conditioning**.

Two approaches are widely used for probabilistic inference: Maximum likelihood estimation (MLE) and Bayes' rule. In [27] the likelihood function is defined as

Definition Given independent and identical distributed sample $X_1, X_2, \cdots, X_n$ from a population with probability density function or probability mass function that is written as $f(x|\theta_1, \theta_2, \cdots, \theta_k)$, the likelihood function is

$$L(\theta_1, \theta_2, \cdots, \theta_k|x_1, \cdots, x_n) = \prod_{i=1}^{n} f(x_i|\theta_1, \theta_2, \cdots, \theta_k).$$

Similarly, in sequential settings of the sampling $X_1, X_2, \cdots, X_n$, given the transition probability function $p(x_{i+1}|x_i, \cdots, x_1)$, the likelihood function is

$$L(\theta_1, \cdots, \theta_k|x_1, \cdots, x_n) = f(x_1|\theta_1, \cdots, \theta_k) \prod_{i=2}^{n} f(x_i|x_{i-1}, \cdots, x_1, \theta_1, \cdots, \theta_k).$$

Then, MLE is defined as

Definition For each sample point x, let $\hat{\theta}(x)$ be a parameter value at which $L(\theta|x)$ is maximized as a function θ, with x held fixed. A MLE of the parameter θ based on a sample X is $\hat{\theta}(X)$.

Intuitively, MLE is a reasonable choice for an estimator, since the observed is most likely. A useful property of MLE is the invariance property [28, 29], which is

Theorem 1 *If $\hat{\theta}$ is the MLE of θ, for any function $\tau(\theta)$, the MLE of $\tau(\theta)$ is $\tau(\hat{\theta})$.*

The other approach is Bayes' rule, which underlies most AI systems for probabilistic inference. In the classical approach the parameter θ is thought to be an unknown, but fixed quantity. Consider data drawn from a distribution indexed by θ. Knowledge about the value of θ is obtained from the observations. In the Bayesian view, θ is a quantity that is described by a probability distribution called the prior. This is a subjective distribution that is based on the observer's beliefs and reflects past experience. The inference form of Bayes equation is

$$p(\text{cause}|\text{effect}) = \frac{\text{p(effect}|\text{cause)p(cause)}}{\text{p(effect)}}.$$

The conditional probability $p(\text{effect}|\text{cause})$ quantifies the relationship in the causal direction and encodes the evidence for inference of the reasons, whereas $p(\text{cause}|\text{effect})$

describes the inference direction. Learning from observed behavior can be well formulated in Bayes' framework. The behavior is the effect observed, and the cause is the goal, the desire of the agent that determines which actions are taken upon the corresponding states, and then the observed states and behavior follow. If we denote the prior distribution by $p(\theta)$ and the conditional probability $f(x|\theta)$, then the posterior probability, the conditional probability of θ given the observation x, is

$$p(\theta|x) = f(x|\theta)p(\theta)/p(x),$$

where $p(x)$ is the marginal probability of X, that is $p(x) = \int f(x|\theta)p(\theta)d\theta$. The posterior distribution is used to make statements about θ. For instance, the mean of the posterior distribution can be used as a point estimate of θ.

Given the increasing size of observation, we would like to see the performance of these estimators. **Consistency** in statistics concerns the question whether the estimator converges to the "correct" value as the observation size becomes infinite. The inference model should be consistent, because intuitively if we observe the policy at every possible state, we will completely understand how to solve the problem even though we do not know the internal goals of the agent. Besides comparing the distance between the true goal and the recovered goal, it is more interesting to evaluate whether the recovered goal is consistent. In [27], consistency is defined as follows:

Definition A sequence of inference $\hat{\theta}_n = \hat{\theta}_n(X_1, \cdots, X_n)$ is a consistent sequence of estimators of the parameter θ if, for every $\epsilon > 0$ and every $\theta \in \Theta$, we have $\lim_{n->\infty} P_\theta(|\hat{\theta}_n - \theta| < \epsilon) = 1$.

Informally, consistency means that the estimator will approach the true value with high probability, as the size of the observation increases. In experiments, we evaluate the accuracy as well as the consistency for each method. The performance measurement is defined to evaluate the distance between the true value and the estimate value. If the distance approaches zero, we say the estimate approaches the true value. A good model is expected to be consistent as well as always performs better than others when the size of observation is small or any choice.

2.2 Decision-Theoretic Model

Our research considers the goal-directness as a core property of decision-making process. As is well known, there are several axiomatizations of "rational" decision making that make the maximum expected utility criterion dominating in decision making under uncertainty [30].

2.2.1 Utility theory

The agents' preferences are captured by utility function $U(s)$, which rates a state s and formalizes the desirability of the state for agents. In a stochastic environment, taking action a given the evidence of observation e, is written $P(\text{Result}(a)|a, e)$. The expected utility is just the average utility value of the outcomes, weighted by the probability that the outcome occurs:

$$E[U(a|e)] = \sum_{s'} P(\text{Result}(a) = s'|a, e)U(s'). \tag{2.1}$$

The principle of maximum expected utility (MEU) says that a rational agent should choose the action that maximize the expected utility $E[U(a|e)]$.

The MEU principle can be derived from the constraints on rational preferences. Given two objects A and B, the following notations are used to represent the agent's preferences:

- $A \succ B$, the agent prefers A to B.
- $A \sim B$, the agent is indifferent between A and B.
- $A \succeq B$, the agent prefers A to B or is indifferent between them.

We can think the outcome for each action as a lottery, A lottery L with outcomes A and B that occur with probabilities p and q, is written as $L = [p, A, qB]$. In [31], the axioms of utility theory include

- Orderability: the agent must either choose one preference of $A \succ B$, $B \succ A$ or $A \sim B$.
- Transitivity: $(A \succ B) \ \wedge \ (B \succ C) \Rightarrow \ (A \succ C)$.
- Continuity: $A \succ B \succ C \Rightarrow \ \exists\, p\ [p, A; 1-p, C] \sim \ B$. If some state B is between A and C in preference, then there is some probability p for which the agent is indifferent between getting B for sure and the lottery that provides A with probability p and C with probability $1 - p$.
- Substitutability: $A \sim B \Rightarrow [p, A; 1-p, C] \sim [q, B; 1-q, C]$. If an agent is indifferent between A and B, the agent is indifferent between two lotteries involving them.
- Monotonicity: $A \succ B \ \Rightarrow (p > q \ \Leftrightarrow \ [p, A; 1-p, B] \ \succ \ [q, A; 1-q, B])$. If an agent prefers A to B, the agent must prefer the lottery that has a higher probability for A, and vice versa.

- Decomposability: $[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$. Compound lotteries can be reduced to simple ones, which is called "no fun in gambling" rule.

These constraints require the agents to be rational, and from them there are some conclusions in [32]:

- Existence of utility function: If an agent's preferences obey the axioms of utility, there exists function U that

$$U(A) > U(B) \Leftrightarrow \; A \succ B$$
$$U(A) = U(B) \Leftrightarrow \; A \sim B$$

- Expected utility of a lottery: The utility of a lottery is the averaged utility of all possible outcomes:

$$U([p_1, s_1; p_2, s_2; \ldots; p_n, s_n]) = \sum_{i=1}^{n} p_i U(s_i).$$

It is obvious that the states can be ranked in preference based on the utility.

2.2.2 Markov decision process

The prevailing model for discrete planning and sequential decision making is the Markov decision process (MDP). An MDP can be described by a graph structure of states and actions with the reward and stochastic transitions (see Figure2.1). Given a state, for each action, there is a stochastic matrix to determine the possible next state. A finite MDP is a tuple, $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$, where

- $\mathcal{S} = \{s_n\}_{n=1}^{N}$ is a set of states. Let $\mathcal{N} = \{1, 2, \cdots, N\}$, then $s_n, n \in \mathcal{N}$ is the n-th state.

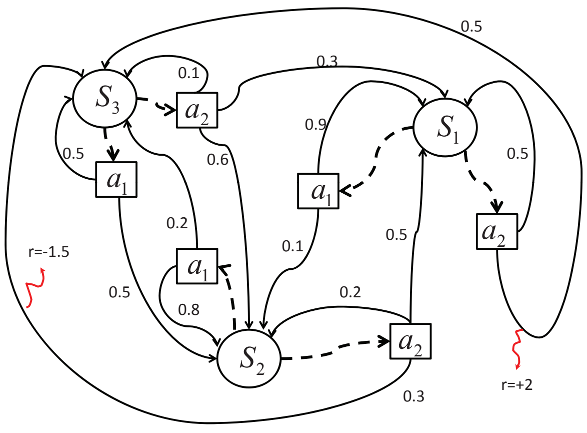


Figure 2.1: Goal-oriented agents interact with environment

- $\mathcal{A} = \{a_m\}_{m=1}^{M}$ is a set of actions. Let $\mathcal{M} = \{1, 2, \cdots, M\}$, then $a_m, m \in \mathcal{M}$ is the m-th action.

- $\mathcal{P} = \{\mathbf{P}_m\}_{m=1}^{M}$ is a set of state transition probabilities, where $m \in \mathcal{M}$. (Here $\mathbf{P}_m$ is a $N \times N$ matrix of transition probabilities after taking action a_m. Its n-th row, denoted as $\mathbf{P}_m(s_n, :)$, contains the transition probabilities upon taking action a_m in state s_n. The entry $\mathbf{P}_m(s_n, s_{n'})$, where $n' \in \mathcal{N}$, is the probability of moving to state $s_{n'}$ in the next stage. Note that the rows of $\mathbf{P}_m$ are probability vectors, whose components are non-negative and their sum is one.)

- $\gamma \in [0, 1]$ is a discount factor, which makes the reward contribution of future state and action less significant than the current ones. Discounting is a good model of preferences over time. A discount factor γ is equivalent to an interest rate of $\frac{1}{\gamma} - 1$.

- r denotes the reward function, mapping from $\mathcal{S} \times \mathcal{A}$ to $\Re$, which is

$$r(s_n, a_m) \triangleq \sum_{n'=1}^{N} \mathbf{P}_m(s_n, s_{n'}) r(s_n, a_m, s_{n'})$$

where $r(s_n, a_m, s_{n'})$ is the reward of moving to state $s_{n'}$ after taking action a_m in state s_n. The reward function $r(s_n, a_m)$ may be further simplified to $r(s_n)$, if we neglect the action's influence. The reward only depending on state is written as a $N \times 1$ vector $\mathbf{r}$.

In decision theory for Markov decision processes, the reward is considered to be a utility function representing the decision-maker's preferences and the transition probability represents the decision-maker's subjective beliefs about uncertain future states. If agents are assumed to be rational, they behave according to the rule of maximizing the expected utility. An MDP is completely determined by the transition matrix $\mathcal{P}$, the initial distribution and the policy π. Figure2.2, which is mainly based on the figure in [12], illustrates a goal-oriented MDP agent.

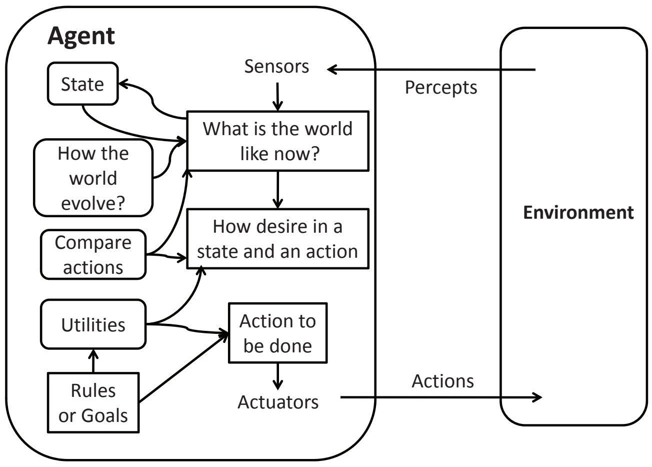


Figure 2.2: Goal-oriented MDP agent interacts with environment

In an MDP, the agent selects an action at each sequential stage. A rule describing the way the actions are selected, that is a mapping between states and actions, is called a **policy** (or **behavior**). A behavior of an agent defines a random state-action sequence $(s^0, a^0, s^1, a^1, \cdots s^t, a^t, \cdots)$, [1] where s^{t+1} is connected to (s^t, a^t) by $\mathbf{P}_{a^t}(s^t, s^{t+1})$. A policy that causes the agent to reach a specified goal is called a *proper policy*.

The rational agents in MDP model behave according to the optimal decision rule that each action selected at any stage should maximize the value function. This expectation is over the distribution of the state sequence $\{s^0, s^1, ...\}$ given policy $\pi = \{\mu^0, \mu^1, \cdots\}$, where $a^t = \mu^t(s^t)$, $\mu^t(s^t) \in W(s^t)$ and $W(s^t) \subset \mathcal{A}$. The objective at state s is to choose a policy maximizing the value of $V^\pi(s)$, where given initial state s^0, $V^\pi(s^0) = E[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t)|\pi]$. Similarly, there is another function called a *Q-function* (or *Q-factor*) that judges how good an action is performed in a given state. Notation $Q^\pi(s, a)$ represents the expected return from state s, taking action a and thereafter following policy π.

These MDP construct can be interpreted in terms of utility theory. The reward can be considered a short-term reward for being in a state, and the value function is the long-term expected reward from s onwards. The utility of a state depends on the utility of the state sequences that follow it. The value function V is considered to be the utility function for the state, and the Q-function, $Q(s, a)$, is considered to be the expected utility of a one-period decision that selects action a at state s. Actions are chosen based on the value function V and Q. Since an agent aims to reach the goal by choosing actions, we say that the agent prefers action a to another action because the agent prefers the expected utility of the possible outcomes induced by action a. To find the optimal actions based on MEU rule, solutions are given based on Bellman equations [33], which are described as

[1]Superscripts index time. E.g. s^t and a^t, with the upper-index $t \in \{1, 2, \cdots\}$, denote state and action at t-th horizon stage, while s_n (or a_m) represents the n-th state (or m-th action) in $\mathcal{S}$ (or $\mathcal{A}$).

Theorem 2 (Bellman Equations) *Given a stationary policy* π*,* $\forall n \in \mathcal{N}, m \in \mathcal{M}$*,* $V^{\pi}(s_n)$ *and* $Q^{\pi}(s_n, a_m)$ *satisfy*

$$\begin{aligned} V^{\pi}(s_n) &= r(s_n, \pi(s_n)) + \gamma \sum_{n' \in \mathcal{N}} \mathbf{P}_{\pi(s_n)}(s_n, s_{n'}) V^{\pi}(s_{n'}), \\ Q^{\pi}(s_n, a_m) &= r(s_n, a_m) + \gamma \sum_{n' \in \mathcal{N}} \mathbf{P}_{a_m}(s_n, s_{n'}) V^{\pi}(s_{n'}). \end{aligned}$$

The actions output by the algorithms using Bellman equations maximize the expected cumulative reward V at every state. Two popular algorithms in dynamic programming have been developed to solve the Bellman equations iteratively. The value iteration algorithm updates the value function by computing $V^{*}(s_n) = \max_a r(s_n, a) + \gamma \sum_{s_{n'}} \mathbf{P}_a(s_n, s_{n'}) V^{*}(s_{n'})$. The policy iteration algorithm selects a policy according to Theory 3 (policy improvement), and then repeatedly updates the value functions until convergence (policy evaluation). The algorithm alternates these two steps until the policy does not change [34].

Theorem 3 (Bellman Optimality) π *is optimal if and only if,* $\forall n \in \mathcal{N}$*,* $\pi(s_n) \in \arg\max_{a \in \mathcal{A}} Q^{\pi}(s, a)$*.*

The policy can be found deterministically, while the policy executed by the agent can also be stochastic. In a stochastic policy, each action is chosen according to a probability distribution, rather than deterministically choosing the action with maximum expected utility. Using the definition in [35], a randomized decision rule in MDP is defined as

Definition A randomized decision rule $\delta(s, \cdot)$ is, for each sate s, a probability distribution on $\mathcal{A}$, with the interpretation that if s is observed, $\delta(s, a)$ is the probability that an action in $\mathcal{A}$ will be chosen.

The deterministic decision rules can be considered a special case of randomized rules, in that they correspond to the randomized rules in the way that a specific action is chosen in probability one.

A natural approach to build a probability distribution is called Boltzmann probability distribution. In MDP settings, it is written as

$$p(a|s_n) = \frac{e^{\beta Q(s_n,a)}}{\sum_{m \in \mathcal{M}} e^{\beta Q(s_n,a_m)}}. \tag{2.2}$$

The parameter β encodes the confidence of choosing the action and determines the stochasticity of the policy. When $\beta \to \infty$, the policy becomes deterministic. A Bayesian approach to IRL using this model is presented in [26], which applies Morkov Chain Monto Carlo simulation to obtain the a posterior distribution.

2.2.3 Inverse Markov decision model

The agents in MDPs formalize their goals in terms of the reward passing through the interaction between the environment and themselves. In many real-world problems, however, the reward function is not known. IRL deals with the problem modeled by a variational MDP in which the reward is unknown. Standard algorithms formulate the problem as an optimization problem with respect to the reward with the constraints of observed state-action pairs. The variational MDP model in IRL is called inverse Markov decision (IMD) model $M_I = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \gamma)$, where $\mathcal{S}, \mathcal{A}, \mathcal{P}$ are prior known as in MDP model or recovered from observation. The notation $\mathcal{O}$ represents the observation data, which can be composed of explicitly given state-action pairs observed, written as $\mathcal{O} = \{o_l = (s_l, a_l)\}_{l=1}^{L}$, or a set of sampled decision trajectories, written as $\mathcal{O} = \{o_h = (s^1, a^1, s^2, \ldots, s^t, a^t)_h\}_{h=1}^{H}$. Note that s_l denotes the l-th state in $\mathcal{S}$, where a_l indicates the action taken at this state, rather than the l-th action in

$\mathcal{A}$. We use this notation only for simple representation. Figure2.3 explains an IMD model, including the observation of a decision-making agent, the environment, and modeling the observed behavior in MDP.

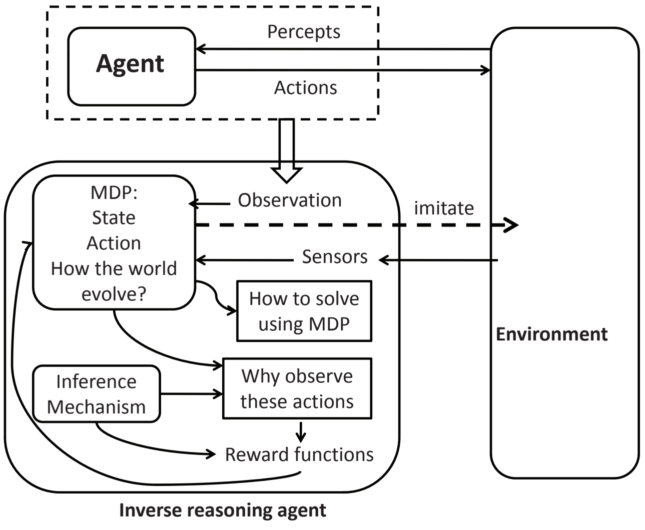


Figure 2.3: IMD model infers the goal of an agent who interacts with the environment and makes sequential decisions.

In some difficult real-world problems, the transition probability is unknown. We shall propose model-free reinforcement learning techniques to solve these problems. The task of learning the reward with unknown transition probability can be modeled as generalized inverse Markov decision process. This generalized model is defined as a tuple $\check{M} = (\mathcal{S}, \mathcal{A}, \gamma, C)$, where C is a simulated environment in which we will learn the reward r and the transition probability $\mathcal{P}$ at the same time.

Our algorithms for the inverse Markov decision model, introduced in later chapters, have the following advantages:

- The reward function is not restricted to be linear formalization of the state features. Though the reward depends on the state features because the reward is given according to state-action pairs, we argue there is little justification for assuming a linear relationship between the reward vector and feature vector. Considering the reward function values as realizations of random variables, we select the Gaussian process as the prior distribution for these random variables and then infer the reward based on Bayesian framework.

- Real-world applications allow for multiple actions for some certain states, which causes the existence of multiple actions observed at one state. We use preference graphs to represent the relations between states and actions. Each graph denotes the observations at a state.
- If the transition probability is not known as a prior, it will be approximated from the observed decision trajectories.
- If the observation has short-length trajectories and there are many states being absent from the observation, the Gaussian process model performs better than existing algorithms.

2.3 Related Work

2.3.1 Inverse optimal control

In control engineering, a linear quadratic setting is characterized by state dynamics that are distributed according to a linear function of past state and action variables with Gaussian noise. Cost functions are quadratic in the state variables, parameterized by Q. Inverse optimal control(IOC) consists in finding, for a given system and sampled policies, a cost function with respect to which such observed policies are optimal. In control engineering, the inverse problem is stated on the linear dynamical system, such as

$$\dot{s}(t) = As(t) + Ba(t)$$

and the control law

$$a(t) = Ks(t),$$

where $s(t)$ is the state vector and the control law $a(t)$ is equivalent to the policy in MDP model. The problem is to find a matrix $\hat{Q} = \hat{Q}' \geq 0$ such that the control law is optimal relative to the system and the cost function

$$J = \int_0^\infty [s^T(t)\hat{Q}s(t) + a^2(t)]dt.$$

The IOC system can be written as $M_{LQ} = \{A, B, K\}$. The objective idea behind IRL and IOC are similar from the perspective of learning an immediate reward or cost function given part of or complete policies.

Recently, IRL has received active attention in machine learning field. Below we review some fundamental algorithms.

2.3.2 Linear approximation

Most current IRL algorithms have assumed that the reward function depends only on state and can be represented by a linear approximation. The representation

$$r(s) = \omega^T \phi(s)$$

was proposed in [2] and has been widely adopted.

Here, we overview the algorithms proposed by Ng and Russell [2] and Abbeel and Ng [21] as the background to our work.

Ng's algorithm assumes that the policy of observed agent is optimal so that the reward function should guarantee the optimality of the observed policy π_E:

$$Q^{\pi_E}(s, \pi_E(s)) \geq \max_{a \in \mathcal{A} \setminus \pi_E(s)} Q^{\pi_E}(s, a) \forall s \in \mathcal{S},$$

which gives the regularized linear programming IRL (LPIRL) shown in Algorithm 1. The constraints in this algorithm guarantee the optimality of the observed behavior. But the trivial solution $\mathbf{r} = 0$ also satisfies these constraints, which highlights the underspecified nature of the problem and the need for reward selection mechanisms. Ng and Russel choose the reward function to maximize the difference between the optimal and suboptimal policies while favoring the sparseness in the reward function.

The IRL algorithms reviewed in this section will be used in comparison with our method in experiments.

Algorithm 1 Ng and Russel's LPIRL with complete observation

```
1: It can be formulated as a linear programming problem as follows,
```

$$\begin{aligned} \max_{\mathbf{r}} \sum_{s \in \mathcal{S}} \min_{a \in \mathcal{A} \setminus \pi_E(s)} & [Q^{\pi_E}(s, \pi_E(s)) - Q^{\pi_E}(s, a)] - \lambda \left\| \mathbf{r} \right\|_1 \\ \text{s.t. } & (\mathbf{P}_{\pi_E} - \mathbf{P}_m)(\mathbf{I} - \mathbf{P}_{\pi_E})^{-1} \mathbf{r} \geq 0, \ m \in \mathcal{M} \\ & R_{min} \leq r(s) \leq R_{max}, \ \forall s \in \mathcal{S}, \end{aligned}$$

where $\mathbf{P}_{\pi_E}$ is a $N \times N$ matrix whose (i,j) entry is $\mathbf{P}_{\pi_E(s_i)}(s_i, s_j)$ and $i, j \in \mathcal{N}$.

2.3.3 IRL from sampled trajectories

In many practical environments, it is impossible to obtain an observation of a complete decision policy, as this would require observing the agent in every possible state, no matter how rare. Partial policy observations, by contrast, would be available in many scenarios. In sequential decision-making, it is natural to collect these observations in the form of decision trajectories. Though the evolution of observed agent's behavior in MDP setting provides the information on both action selection and the environment dynamics, most IRL algorithms assume the dynamics, which is encoded by $\mathcal{P}$, is given beforehand. So the decision trajectory can be simply treated as a set of state-action pairs. If we always observe a fixed action at a state, we say the observed policy is deterministic. Otherwise, the policy is probabilistic.

In order to address the sampled trajectories in large state space, Ng and Russell [2] model the reward function by using a linear approximation that is written as

$$r(s) = \sum_{i=1}^{d} \omega_i \phi_i(s) = \omega^T \phi(s), \tag{2.3}$$

where $\phi : \mathcal{S} \rightarrow [0,1]^d$ and $\omega^T = [\omega_1, \omega_2, \cdots, \omega_d]$.

Then from the observed H decision trajectories, the value function can be estimated as

$$\hat{V}^{\pi_E} = \frac{1}{H} \sum_{h=1}^{H} \sum_{t=0}^{h_t} \gamma^t \omega^T \phi(s_{(h)}^t), \tag{2.4}$$

where h_t denotes the number of states in h-th decision trajectory. Ng's algorithm for sampled trajectories is given in Algorithm 2, which assumes that there is available computation resources for simulating decision trajectories for optimal policy and a number of random policies.

Based on Eq.2.3 and Eq. 2.4, Abbeel has given an apprenticeship learning algorithm which aims to find a policy that performs as well as the demonstrating agent (DA), in the measurement of *feature expectation*, defined as

$$\hat{\mu}^{\pi_E} = \frac{1}{H} \sum_{h=1}^{H} \sum_{t=0}^{\infty} \gamma^t \phi(s_{(h)}^t).$$

Algorithm 2 Ng and Russel's approximate LPIRL with sampled trajectories

```
1: Input $M_I = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$, where $\mathcal{O}$ is composed of H decision trajectories. A set of basis functions $\phi$.
2: A policy set $\mathbf{\Pi}$ which has the first randomly chosen policy $\pi_1$. Let k=1.
3: while $k <=$ IterMax or $\|V^{\pi_E}(s_0) - V^{\pi_k}(s_0)\| \leq \epsilon$
4:     Sample decision trajectories for the policies in $\mathbf{\Pi}$, then we have the value functions by calculating Eq. 2.4.
5:     Solve the linear programming problem
           $$\max_{\omega} \sum_{i=1}^{k} p(V^{\pi_E}(s_0) - V^{\pi_i}(s_0))$$
           $$s.t. |\omega_i| \leq 1,\ i = 1, 2, \cdots, d$$
6: Find the policy $\pi_{k+1}$ that is optimal with respect to $\hat{\mathbf{r}} = \hat{\omega}^T\phi$, where $\hat{\omega}$ is the solution to the above linear programming problem.
7: $\mathbf{\Pi} = \mathbf{\Pi} \cup \pi_{k+1}$
8: $k = k + 1$
9: end while
10: Output the reward function $\hat{\mathbf{r}} = \hat{\omega}^T\phi$.
```

It is obvious that the value function is a linear combination of the feature expectation and the coefficient vector is determined by the reward function. To find the policy whose performance is close to that of DA, Abbeel and Ng's algorithm is based on the fact that the difference between DA's value function and the estimated policy's value function is bounded from above by the difference between their feature expectations. So, the problem is reduced to finding a policy that induces feature expectations close to μ^{π_E}. Two apprenticeship learning algorithms are proposed in [21], using max-margin and projection method respectively to obtain the optimized $\hat{\omega}$ that minimize $\|\mu^{\pi_E} - \mu^{\pi_k}\|$ at k-th iterative step. After obtaining the optimized $\hat{\omega}$ in each step, another optimal policy π_{k+1} is computed and then used for the next iteration. It has been proved that both these algorithms will converge in c steps of iterations [2]. Finally, the IRL process outputs an estimation of the reward function $\hat{r} = \hat{\omega}\phi$, a set of policies $\mathbf{\Pi} = \{\pi_1, \pi_2, \cdots, \pi_c\}$ and their feature expectations $\Psi = \{\mu^{\pi_1}, \mu^{\pi_2}, \cdots, \mu^{\pi_c}\}$. The goal of AL is to find the best policy by manual examination or by the algorithm that finds a convex combination of the policies in $\mathbf{\Pi}$. The coefficients for this convex combination are obtained by solving the following QP: point closest to μ^{π_E} in the convex closure of the vectors in Ψ:

$$\min \left\|\mu^{\tilde{\pi}} - \mu^{\pi_E}\right\| \text{ s.t. } \mu^{\tilde{\pi}} = \sum_{i=1} c\lambda_i \mu^{\pi_i}, \lambda_i \geq 0, \sum_{i=1}^{c} \lambda_i = 1.$$

Though the selection of best policy guarantees its performance close to μ^{π_E}, the recovered reward function is not necessarily close to the true underlying reward function.

Based on the theory of feature expectation, Abeel presents another IRL algorithm in [21], which is called PROJ. The details have been shown in Algorithm 3. Though

[2]To find a policy $\tilde{\pi}$ such that $\left\|\mu^{\tilde{\pi}} - \mu^{\pi_E}\right\| \leq \epsilon$, then both max-margin and projection algorithms will terminate after at most $C = O(\frac{d}{(1-\gamma)^2\epsilon^2} \log \frac{d}{(1-\gamma)\epsilon})$, which means that $c \leq C$

the goal of apprenticeship learning is to learn a policy whose performance is close to the observed, IRL is the essential part of computation. Later , we will use PROJ to address the IRL for comparison.

Algorithm 3 PROJ in [21]

```
1: Input $B = (M \setminus r, \mathcal{G}, \mathcal{O})$, where $\mathcal{G}$ is a set of basis functions $\phi$ for linear approximation of the reward. $\mathcal{O}$ is composed of H decision trajectories.
2: Choose some policy $\pi^0$ and compute its feature expectation $\mu^0$ by repeating simulations.
3: Set index $i = 1$, $\bar{\mu}^0 = \mu^0$ and $v = \text{Inf}$.
4: while $v \leq \epsilon$ do
5:     Set $\omega = \mu^E - \bar{\mu}^{i-1}$.
6:     Compute the estimated reward $r = \omega^T \phi$ and use it to conduct forward planning using MDP. We have $\mu^i$.
7:     Compute orthogonal projection of $\mu^E$ onto the line through $\bar{\mu}^{i-1}$ and $\mu^i$ using
           $$\bar{\mu}^i = \bar{\mu}^{i-1} + \frac{(\mu^i - \bar{\mu}^{i-1})^T(\mu^E - \bar{\mu}^{i-1})}{(\mu^i - \bar{\mu}^{i-1})^T(\mu^i - \bar{\mu}^{i-1})}(\mu^i - \bar{\mu}^{i-1})$$
8:     Set $v = \left\|\mu^E - \bar{\mu}^i\right\|_2$ and i=i+1.
9: end while
10: Output the reward function $\hat{\mathbf{r}} = \omega^T \phi$.
```

2.3.4 Game-theoretic model

Using a game-theoretic approach, Syed and Schapire [23] have developed an apprenticeship learning algorithm that maximizes performance relative to the observed behavior for the worst-case choice of reward function. Their method is based on a multiplicative weights algorithm for solving two-player zero-sum games due to Freund and Schapire [36].

Based on the conception of feature expectation, they formulate an optimization problem

$$v^* = \max_{\psi \in \Psi} \min_{\omega \in \mathcal{W}} [\omega^T \mu(\psi) - \omega^T \mu_E], \tag{2.5}$$

where Ψ is a set of all mixed policies, and $\mathcal{W} = \left\{ \omega \in \Re^k : \|\omega\|_1 = 1, \text{ and } \omega \geq 0 \right\}$. The goal is to find the mixed policy ψ^* that achieves v^*. Since $V(\psi) = \omega^* \mu(\psi)$ for all ψ, we have that ψ^* is the policy in Ψ that maximizes $V(\psi) - V(\pi_E)$ with respect to the worst-case possibility for ω^*. As ω and ψ are both distributions, Eq. 2.5 is the form of a two-person zero-sum game. The "min player" specifies a reward function by choosing ω, and the "max player" chooses a mixed policy ψ. The algorithm to solve IRL problem in game-theoretic is summarized in Alg. 4.

Algorithm 4 The MWAL algorithm

```
1: Given a $M_I$ and the feature expectations $\hat{\mu}_E$.
2: Let $\beta = (1 + \sqrt{\frac{2 \ln k}{T}})^{-1}$.
3: Define $G(i, \mu) = ((1 - \gamma)(\mu(i) - \hat{\mu}_E(i)) + 2)/4$, and initialize $W^1(i) = 1$, for $i = 1, \ldots, k$.
4: for t=1,..., T do
5:     Set $\omega^t(i) = \frac{W^t(i)}{\sum_i W^t(i)}$, for $i = 1, \ldots, k$.
6:     Compute policy $\hat{\pi}^t$ for M with respect to $r(s) = \omega^t \phi(s)$.
7:     Compute $\mu^t = \mu(\hat{\pi}^t)$.
8:     $W^{t+1}(i) = W^t(i) \cdot e^{\ln \beta \cdot G(i, \hat{\mu}^t)}$, for $i = 1, \ldots, k$.
9: end for
10: Return the mixed policy by assigning weighting $\frac{1}{T}$ to $\hat{\pi}^t$.
```

Chapter 3

Bayesian IRL within Small Finite Space

The problems in this chapter are formulated in small finite space, which means that all the states, value functions, and transition probabilities are known as a prior and can be stored in the memory of a computer. We define the IRL problem here is to learn the reward vector r with the assumption that the observed actions come from the optimal policy for $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$. The IRL problem is, in general, highly underspecified, which has led researchers to consider various models for restricting the set of reward vectors under consideration. In a seminal consideration of IRL problems, Ng and Russel [2] observe that, by the optimality equations, the only reward vectors consistent with an optimal policy π are those that satisfy the set of inequalities

$$(\mathbf{P}_\pi - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{r} \quad \geq \quad \mathbf{0}, \quad \forall a \in \mathcal{A}, \tag{3.1}$$

where we use the notation $x \succ= 0$ to indicate a vector x that has all non-negative components. The $\mathbf{P}_\pi$ is the transition probability matrix relating to observed policy π, $\mathbf{P}_a$ denotes the transition probability matrix for other actions. Note that the trivial solution $\mathbf{r} = 0$ satisfies these constraints, which highlights the underspecified nature of the problem and the need for reward selection mechanisms. Ng and Russel [2] choose the reward function to maximize the difference between the optimal and suboptimal policies, which can be done using a linear programming formulation. In the sections that follow, we propose the idea of selecting reward on the basis of Maximum a posterior (MAP) estimation in a Bayesian framework.

3.1 Bayesian Framework with Gaussian Prior

Suppose that we have a prior distribution $p(r)$ for the reward in the M and we also have a model for $p(\mathcal{O}|r)$. The Bayesian inference is to maximize the posterior distribution $p(r|\mathcal{O})$. The MAP estimation problem can be formulated as a convex optimization problem by assuming $p(r)$ as a Gaussian distribution.

Specifically, let $\mathbf{r}$ be a random vector that depends on state. The entry $\mathbf{r}(s_n)$ denotes the reward at n-th state. We assign a Gaussian prior on the $\mathbf{r}$: $\mathbf{r} \sim \mathcal{N}(\mu_r, \Sigma_r)$. This is a subjective distribution; before anything is known about optimal policies for the MDP, the learner has characterized a prior belief distribution with mean μ_r and standard deviation Σ_r. An observation of data is taken from the population indexed by $\mathbf{r}$ and the prior distribution is updated with this new information using Bayes' Rule to form the *posterior distribution*

$$p(\mathbf{r}|\mathcal{O}) = \frac{p(\mathcal{O}|\mathbf{r})p(\mathbf{r})}{\int p(\mathcal{O}|\mathbf{r})p(\mathbf{r})d\mathbf{r}}.$$

One can envision two principal types of experiments for collecting a set of observations $\mathcal{O}$:

1. *Decision Mapping*: the observations are obtained by finding a mapping between state and action; e.g., we ask the expert which action he, she, or it would choose at state s, and then repeat the process. Ultimately, we will have a set

of independent state-action pairs, $\mathcal{O}_1 = \left\{(s^h, a^h)\right\}_{h=1}^t$.

2. *Decision Trajectory*: Given an initial state, we simulate the decision problem and record the history of the expert's behavior, which is written as $\mathcal{O}_2 = \{s^1, a^1, s^2, a^2, \cdots, s^t, a^t\}$.

Formally, we define an experiment E to be a triple $(O, \mathbf{r}, \{p(\mathcal{O}|\mathbf{r})\})$, where O is a random vector with probability mass function $p(\mathcal{O}|\mathbf{r})$ for some $\mathbf{r}$ in the function space. Given what experiment E was performed and a particular observation of $\mathcal{O}$, the experimenter is able to make inference and draw some evidence about $\mathbf{r}$ arising from E and $\mathcal{O}$. This evidence we denote by $Ev(E, \mathcal{O})$. Consider observations made using decision mapping $\mathcal{O}_1$ and decision trajectory $\mathcal{O}_2$, with corresponding experiments $E_1 = (O_1, \mathbf{r}, \{p(\mathcal{O}_1|\mathbf{r})\})$ and $E_2 = (O_2, \mathbf{r}, \{p(\mathcal{O}_2|\mathbf{r})\})$. We would like to show that $Ev(E_1, \mathcal{O}_1) = Ev(E_2, \mathcal{O}_2)$, if the states in $\mathcal{O}_1$ and $\mathcal{O}_2$ are the same. This fact implies that inference conclusions drawn from $\mathcal{O}_1$ and $\mathcal{O}_2$ should be identical.

Making use of independence of state-action pairs in decision mapping, we calculate the joint probability density as

$$p(\mathcal{O}_1|\mathbf{r}) = \prod_{h=1}^{t} p(s^h, a^h|\mathbf{r}) = \prod_{h=1}^{t} p(s^h)p(a^h|s^h, \mathbf{r}).$$

Considering Markov transition in decision trajectory, we write the joint probability density as

$$p(\mathcal{O}_2|\mathbf{r}) = p(s^1)p(a^1|s^1, \mathbf{r}) \prod_{h=2}^{t} p(s^h|s^{h-1}, a^{h-1})p(a^h|s^h, \mathbf{r}).$$

Finally, we get $p(\mathcal{O}_1|\mathbf{r}) = c(\mathcal{O}_1, \mathcal{O}_2)p(\mathcal{O}_2|\mathbf{r})$, where $c(\mathcal{O}_1, \mathcal{O}_2)$ is a constant. The above equation implies an equivalence of evidence for inference of $\mathbf{r}$ between the use of a decision map or a decision trajectory.

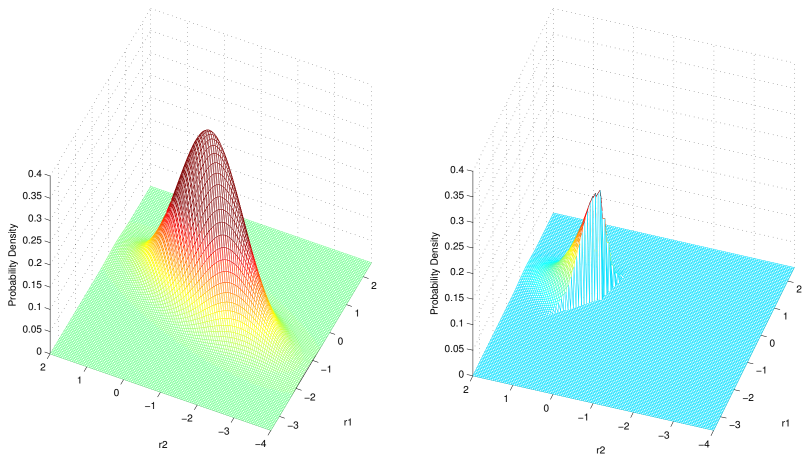


Figure 3.1: An example showing the Bayesian IRL given full observation of the decision maker's policy. The left is the prior and the right is the posterior.

3.2 Learning with Deterministic Policy Model

To simplify computation, we eliminate the elements in likelihood function $p(\mathcal{O}|\mathbf{r})$ that do not contain $\mathbf{r}$, which yields $p(\mathcal{O}|\mathbf{r}) = \prod_{h=1}^{t} p(a^h|s^h,\mathbf{r})$. Further, we model $p(a^h|s^h,\mathbf{r})$ by

$$p(a^h|s^h,\mathbf{r}) = \begin{cases} 1, \text{if } Q(s^h,a^h) \geq Q(s^h,a),\ \forall a \in \mathcal{A} \\ 0, \text{otherwise.} \end{cases} \tag{3.2}$$

This form for the likelihood function is based on the assumption that each observed action is an optimal choice on the part of the expert. Note that the set of reward values that make $p(a^h|s^h,\mathbf{r})$ equal to one is given by Eq. 3.1.

Proposition 4 *Assume a countable state and control space and a stationary policy. Then IRL using Bayesian MAP inference is a quadratic convex programming problem.*

Proof By Bayes rule, the posterior distribution of reward

$$p(\mathbf{r}|\mathcal{O}) = \frac{1}{(2\pi)^{N/2}|\Sigma_r|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{r}-\mu_r)^T\Sigma_r^{-1}(\mathbf{r}-\mu_r)\right).$$

This posterior probability $p(\mathbf{r}|\mathcal{O})$ quantifies the evidence that $\mathbf{r}$ is the reward for the observations in $\mathcal{O}$. Using Eq. 3.1, we formulate the IRL problem as

$$\begin{aligned} &\min_r \ \frac{1}{2}(\mathbf{r}-\mu_r)^T\Sigma_r^{-1}(\mathbf{r}-\mu_r) \\ &\text{s.t.} \ \ (\mathbf{P}_{a^*}-\mathbf{P}_a)(\mathbf{I}-\gamma\mathbf{P}_{a^*})^{-1}\mathbf{r} > 0, \ \forall a \in \mathcal{A} \\ &\quad\ \ \mathbf{r}_{\min} < \mathbf{r} < \mathbf{r}_{\max}. \end{aligned} \tag{3.3}$$

Since the objective is convex quadratic and constraints are affine, Problem 3 is a convex quadratic program.

Fig. 3.1 shows a Gaussian prior on reward and its posterior after truncation by accounting for the linear constraints on reward implied by observation $\mathcal{O}$. Note the shift in mode.

The development above assumes the availability of a complete set of observations, giving the optimal action at every state. If necessary, it may be possible to expand observations of partial policies to fit the framework. A naive approach would be to state transition probabilities averaged over all possible actions at unobserved states.

3.3 Learning with Stochastic Policy Model

Suppose that we are given observed optimal policies from decision trajectories. The observations can be transformed into the equivalent representation that is $\mathcal{O} = \{(s_1,a_1),(s_2,a_2),...,(s_K,a_K))\}$. The notation $a_k, k \in \{1,\ldots,K\}$ indicates the action taken at state $s_k \in \mathcal{S}$. Assume that there exists a set of reward functions, denoted as S_π, with the property that for every member of the set the optimal policy is precisely equal to the observed policies. Considering the optimal policy as stationary, we have the following likelihood model:

$$\begin{aligned} p(\mathcal{O}|r) &= \prod_{k=1}^{K} p(a_k, s_k|r) = \prod_{k=1}^{K} p(a_k|s_k, r) \prod_{k=1}^{K} p(s_k) \\ &= \prod_{k=1}^{K} \frac{e^{\beta Q^*(s_k,a_k)}}{Z_k} \prod_{k=1}^{K} p(s_k), \end{aligned} \tag{3.4}$$

where Z_k is a normalizing constant. We can use dynamic programming to calculate $Q(s_k, a_k)$. The constant β is a confidence parameter that encodes the agent confidence on the demonstration. Using Bayes' rule, we can write the posterior probability of the reward as,

$$\begin{aligned} p(r|\mathcal{O}) &= \frac{p(\mathcal{O}|r)p(r)}{p(\mathcal{O})} \\ &= \frac{1}{Z'} e^{\beta \sum_{k=1}^{K} Q^*(s_k,a_k)} p(r) \prod_{k=1}^{K} p(s_k), \end{aligned} \tag{3.5}$$

where Z' is a normalizing constant and $p(r)$ is the prior probability modeling the initial uncertainty in r. If we naively ignore r in the constant Z', we may drop the unrelated items in Eq.3.5, leading to

$$p(r|\mathcal{O}) \propto p(r) e^{\beta \sum_{k=1}^{K} Q(s_k,a_k)}. \tag{3.6}$$

This posterior probability $p(r|\mathcal{O})$ quantifies the evidence that r is the reward for the observations in $\mathcal{O}$. Using vector notation V^π and $\mathbf{r}$, we write Bellman equations as

$$V^\pi = \mathbf{r} + \gamma P_\pi V^\pi; Q^\pi(s, a) = r(s) + \gamma P_{as} V^\pi.$$

Now we can state a way to find the most likely reward $\mathbf{r}^*$, the maximum a posterior over the reward $\mathbf{r}$ constrained by the observed state-actions pairs. We write the optimization problem as

$$\begin{aligned} &\max \;\; p(r)e^{\beta \sum_{k=1}^{K} Q(s_k,a_k)} \\ &\text{s.t.} \;\; \mathbf{r} \in S_\pi. \end{aligned} \tag{3.7}$$

As mentioned for the ill-posed problem, there are multiple reward function values resulting in the same optimal policy. In the Problem 3.7, we intend to reduce the uncertainty in the reward by maximum a posterior. In practice we obtain the $\mathbf{r}^*$ by minimizing the negative log posterior,

$$\begin{aligned} &\min_r \;\; \frac{1}{2}(\mathbf{r}-\mu_r)^T \Sigma_r^{-1}(\mathbf{r}-\mu_r) - \beta \sum_{k=1}^{K} Q(s_k,a_k) \\ &\text{s.t.} \;\; \mathbf{r} \in S_\pi. \end{aligned} \tag{3.8}$$

To satisfy the constraints that $\mathbf{r} \in S_\pi$, we have some inequalities described as

Lemma 5 *For state s, let $\mathcal{A}_o$ be the set of observe actions and $\mathcal{A}_u$ be the set of unobserved actions. Then given a sufficiently large number of observations, we have*

$$(\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{r} \succ= \epsilon, \;\forall a^* \in \mathcal{A}_o \text{ and } a \in \mathcal{A}_u, \;\; 0 < \epsilon < C. \tag{3.9}$$

where C is a positive constant.

Proof The notation P_{a^*} is a $N \times N$ matrix whose row is the state transition probabilities upon taking the action observed at that state. According to utility theory, the observed actions are preferred by the agent to the unobserved, though the actions are determined by stochastic policy in 2.2. Given sufficient observations, the actions not observed indicates that the agent is not likely to select them according to the stochastic policy model. It follows that the Q-functions for the actions unobserved are much smaller than those for actions observed. For a positive value $\epsilon \in [0, C]$, we have the inequalities that are

$$Q(s, a^*) - Q(s, a) \geq \epsilon. \tag{3.10}$$

Since the Q-function is calculated as

$$Q(s, a) = r(s) + \gamma \mathbf{P}_a(s, :)V^\pi,$$

and the value function $V^\pi = \mathbf{r} + \gamma \mathbf{P}_\pi V^\pi$, we have $V^\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{r}$. After replacing the function V^π into Eq. 3.10, we have the inequalities shown in this lemma.

For finite-state MDPs, above inequality characterizes the set of all reward functions that make the observed actions preferred to the unobserved actions. Replacing inequalities (3.9) in Problem (3.8), we have

$$\begin{aligned}
&\min_{\mathbf{r}} \ \frac{1}{2}(\mathbf{r} - \mu_r)^T \Sigma_r^{-1}(\mathbf{r} - \mu_r) - \beta \sum_{k=1}^{K} Q(s_k, a_k) \\
&\text{s.t. } \ (\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{r} \succ= \epsilon, && (3.11) \\
&\forall a^* \in \mathcal{A}_o \text{ and } a \in \mathcal{A}_u, \ 0 < \epsilon < C. && (3.12) \\
&\quad \mathbf{r}_{\min} < \mathbf{r} < \mathbf{r}_{\max}.
\end{aligned}$$

The sum of $Q(s_k, a_k)$ is calculated as

$$\sum_{k=1}^{K} Q(s_k, a_k) = \mathbf{e}(\mathbf{I} + \gamma \mathbf{P}_{a^*}(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1})\mathbf{r}, \tag{3.13}$$

where $\mathbf{e}$ is a n dimensional row vector whose entries are ones for observed states in $\mathcal{O}$ or zeros for unobserved states. Replacing inequalities (3.9) and Eq. (3.13) in Problem

(3.8), we have

$$\begin{aligned} &\min_{\mathbf{r}} \ \frac{1}{2}(\mathbf{r}-\mu_r)^T\Sigma_r^{-1}(\mathbf{r}-\mu_r) - \beta\mathbf{e}(I+\gamma\mathbf{P}_{a^*}(I-\gamma\mathbf{P}_{a^*})^{-1})\mathbf{r} \\ &\text{s.t.} \ \ (\mathbf{P}_{a^*}-\mathbf{P}_a)(\mathbf{I}-\gamma\mathbf{P}_{a^*})^{-1}\mathbf{r} \succ = \epsilon, \\ &\forall a^* \in \mathcal{A}_o \text{ and } a \in \mathcal{A}_u, \ 0<\epsilon<C. \\ &\quad r_{\min} < r(\cdot) < r_{\max}. \end{aligned} \tag{3.14}$$

This is a quadratic program called *QPIRL*. Since the matrix Σ_r is positive-definite, Σ_r^{-1} is also positive-definite matrix. So the objective function is convex quadratic, and the constraint functions are affine. Finally, we formulate a convex quadratic program for IRL in the condition that we have the complete observations to know the optimal action at every state. If there are some states that we do not observe, we use the average state transition probabilities to replace that of taking the optimal action at that state. We summarize the whole process in Algorithm 5.

Algorithm 5 IRL via Quadratic Programming (QPIRL)

```
1: Initialization, Given $\mu_r$, $\Sigma_r$ and the observed data $\mathcal{O} = \{(s_k, a_k)_{k=1}^K\}$.
2: for $n = 1$ to $N$ do
3:     if $s_n \in \mathcal{O}$ then
4:         $\mathbf{P}_{a^* s_n} = \mathbf{P}_{a_n s_n}$
5:     else
6:         $\mathbf{P}_{a^* s_n} = \frac{\sum_{m=1}^M \mathbf{P}_{a_m s_n}}{M}$
7:     end if
8: end for
9: Solve the convex programming in Problem 3.14.
10: return the reward $r$.
```

Although the above method naively ignores the normalizing constant in the Bayes' inference, it still provides a solution better than [2] that maximizes the difference of Q-functions between the actions observed and unobserved. We give an intuitive analysis to explain the reasons for the superiority of our method based on the stochastic policy. First, in practice, many agents do not follow rational behavior by always choosing the action with maximum expected value functions. The strict constraints for rational behavior then may not yield a reward function that approaches the true, and the imitation performance may be not good. Second, we can show that the method of maximizing the difference is a particular approximation to our model.

Theorem 6 *Given complete observations, IRL problems can be defined as an optimization problem that has the constraints shown in 3.9. The cost function based on the stochastic policy is a general formulation to model the observed behavior, and the method based on deterministic policy is a particular approximation to the stochastic method.*

Proof First, the deterministic policy is a particular case of the stochastic policy, since when the confidence parameter in Eq. 2.2 goes infinity, the policy becomes deterministic. Second, to compare the cost functions, we write the complete cost function for stochastic policy

$$\min -\sum_{k=1}^{K} \beta N \cdot Q(s_k, a_k) + \sum_{k=1}^{K} \log \sum_{m=1}^{M} e^{\beta Q(s_k, a_m)} - \log p(r). \tag{3.15}$$

Without loss of generality, we multiply the constant number $N = |\mathcal{S}|$ by the Boltzmann distribution. In the equation $\sum_{m=1}^{M} e^{\beta Q(s_k, a_m)}$, without loss of generality, we assume $Q(s_k, a_1) = max_{m \in \mathcal{M}} Q(s_k, a_m)$, yielding

$$\sum_{m=1}^{M} \beta e^{Q(s_k, a_m)} \leq M \cdot e^{Q(s_k, a_1)}.$$

As we know the Q-function with discounted factor is bounded, written as $Q(s,a) \geq \frac{r_{min}}{1-\gamma}$, we have $e^{Q(s,a)} \geq e^{\frac{r_{min}}{1-\gamma}}$. When $e^{\frac{(M-1)\cdot r_{min}}{1-\gamma}} \geq M$, the equation $\prod_{m=2}^{M} e^{Q(s_k,a_m)} \geq e^{\frac{(M-1)\cdot r_{min}}{1-\gamma}} \geq M$ is true. Then, it follows that

$$\sum_{m=1}^{M} e^{\beta Q(s_k,a_m)} \leq \prod_{m=1}^{M} e^{Q(s_k,a_m)}. \tag{3.16}$$

The right equation is an upper bound of the left. To minimize the complete cost function, we solve the following relaxed optimization problem.

$$\begin{aligned}
&\min -\sum_{k=1}^{K} \beta N \cdot Q(s_k,a_k) + \sum_{k=1}^{K}\sum_{m=1}^{M} \beta Q(s_k,a_m) - \log p(r) \\
&\text{s.t. } \ (\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1} r \succ= \epsilon, \\
&\forall a^* \in \mathcal{A}_o \text{ and } a \in \mathcal{A}_u, \ 0 < \epsilon < C, \ r_{min} \leq r \leq r_{max}
\end{aligned} \tag{3.17}$$

Above problem is essentially the same as the Algorithm 1 in [2], which assumes the deterministic policy and maximizes the difference of Q-functions between the actions observed and unobserved.

Chapter 4

Inverse Reinforcement Learning with Gaussian Process

4.1 Introduction

The assumption that the reward function can be linearly approximated, which underlies a number of IRL approaches, may not be reasonable for many problems of practical interest. The ill-posed nature of the inverse learning problem also presents difficulties. Multiple reward functions may yield the same optimal policy, and there may be multiple observations at a state given the true reward function. To deal with these problems, we design algorithms that do not assume linear structure for reward function, but yet remain computationally efficient. In particular, we propose new IRL models and algorithms that assign a Gaussian prior on the reward function or treat the reward function as a Gaussian process. This approach is similar in perspective to that Ramachandran and Eyal [26], who view the state-action samples from the expert as the evidence that will be used to update a prior on the reward function, under a Bayesian framework. The solution in [26] depends on non-convex optimization using

Markov Chain Monte Carlo simulation. Moreover, the ill-posed nature of the inverse learning problem also presents difficulties. Multiple reward functions may yield the same optimal policy, and there may be multiple observations at a state given the true reward function. Our model aims to deal with the ill-posed nature by applying Bayesian inference and preference graphs. One of the main novelties of our approach is that it not only bears a probabilistically coherent view but also is computationally tractable.

The main contributions of our work are as follows. First, we model the reward function in a finite state space using a Bayesian framework with known Gaussian priors. We show that this problem is a convex quadratic program, and hence that it can be efficiently solved. Second, for the general case that allows noisy observation of incomplete policies, representation of the reward function is challenging and requires more computation. We show that a Gaussian process model is appropriate in that case. Our approach makes use of a preference graph in action space to represent the multiple observations at a state. Even in cases where the state space is much larger than the number of observations, IRL via Gaussian processes has the promise of offering robust predictions and results that are relatively insensitive to number of observations. Finally, we use the Gaussian process model to solve IRL problems in continuous domains where both dynamics and reward are unknown.

4.2 Gaussian Process Learning

4.2.1 Regression and classification

In recent years, Gaussian processes have found increasing employment in modeling random phenomena in a number of application domains. In this section, we provide a brief introduction to Gaussian processes. See [37, 38, 39, 40] for a more complete discussion.

Given a data set $\{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X}$ is a matrix that is composed of N input example vectors $\mathbf{x}_c, c \in \mathcal{N}$ and $\mathbf{y}$ is a vector of corresponding targets value y_c (real value for regression or categorical value for classification). The Gaussian process model treats the latent functions as random processes with Gaussian prior, which is different from the parametric form in classical statistical models. Denote the latent function by $u(\mathbf{x}_c)$, which is assumed to be a random process. Then the first and second order statistics of $u(\mathbf{x}_c)$ are its mean function $m(\mathbf{x}_c)$ and covariance function $k(\mathbf{x}_c, \mathbf{x}_d), \forall c, d \in \mathcal{N}$.

Both the estimation of mean function and variance function depend on the finite dimensional distribution. Since Gaussian process is a process whose finite dimensional distributions are Gaussian, a Gaussian process is determined by its mean and variance functions. For every set of realizations of random variables of a Gaussian process, the symmetric matrix $\mathbf{K}$, whose entries are calculated by the covariance function $k(\mathbf{x}_c, \mathbf{x}_d)$, is positive semi-definite[1]. It is sufficient that for $\mathbf{K}$ is positive semi-definite, there exists a Gaussian random field with this covariance matrix and zero-mean function $m(\mathbf{x}_c) = 0$ (Kolmogorov's theorem [41]).We denote such random field as $u(\mathbf{x}_c) \sim N(0, \mathbf{K})$.

The simplest Gaussian process model is $y_c = u(\mathbf{x}_c) + \epsilon$, where ϵ is an independent

[1]Positive semi-definite implies that $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$ for all $\mathbf{x}$.

Gaussian noise, written as $\epsilon \sim N(0, \sigma_\epsilon^2)$. Within a Bayesian framework, the inference of $u(\mathbf{x})$ at the test location $\mathbf{x}$ is described by maximization of the posterior probability

$$p(u(\mathbf{x})|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, u(\mathbf{x}))p(u(\mathbf{x}))}{p(\mathbf{y}|\mathbf{X})}.$$

The joint distribution of the observed target values and the function values follows a Gaussian distribution. Therefore the posterior conditional distribution is written as

$$u(\mathbf{x})|\mathbf{X}, \mathbf{y} \sim N(\mathbf{K}(\sigma_\epsilon^2\mathbf{I} + \mathbf{K})^{-1}\mathbf{y}, \sigma_\epsilon^2(\sigma_\epsilon^2\mathbf{I} + \mathbf{K})^{-1}\mathbf{K}).$$

The generative model based on Gaussian processes provides an efficient path for inference with finite observations in a large space, offering tractable computation and a guarantee of performance.

4.2.2 Preference learning

The area of machine learning research known as label preference learning is focused on methods for discovering a latent function that predicts preference relations among a finite set of labels. The label preference learning problem is a generalization of standard problems, such as classification and label ranking [42]. Considering the latent function values as Gaussian process, Chu and Ghahramani observed that a Bayesian framework is an efficient and competitive method for learning label preferences [38]. They proposed a novel likelihood function to capture the preference relations using *preference graph*, a directed graph encoding the label preferences of each sample [43, 44].

Let $u(\mathbf{x}, y)$ denote the latent function depending on both label y and instance $\mathbf{x}$, and $\mathcal{G}$ denote the observed preference graphs. The Bayesian inference is written as

$$\hat{u}(\mathbf{x}, y) \triangleq \arg\max_{u(\mathbf{x},y)} p(u(\mathbf{x}, y)|\mathcal{G}) \propto \arg\max_{u(\mathbf{x},y)} p(\mathcal{G}|u(\mathbf{x}, y))p(u(\mathbf{x}, y)) \tag{4.1}$$

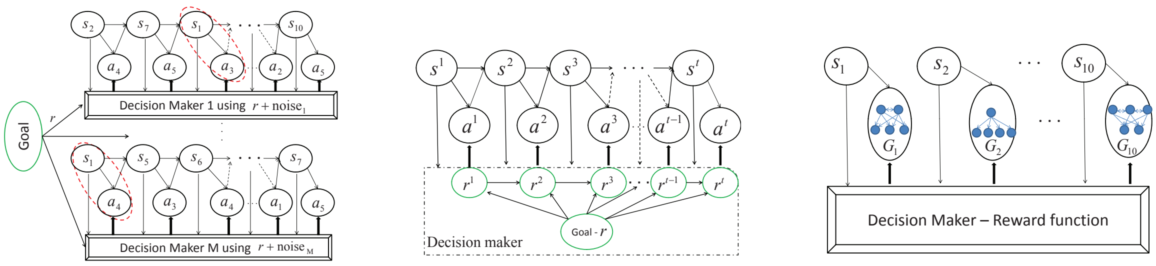
where $p(\mathcal{G}|u(\mathbf{x}, y))$ is the likelihood function derived from preference relations. Given a new instance $\mathbf{x}^*$, the labels y^* can be predicted by ranking the values of latent function, $y^* = \arg\max_y \hat{u}(\mathbf{x}^*, y), y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of labels.

We also use Bayesian inference and build off several of the ideas in [38] and related work, but our method differs from label preference learning for classification and label ranking. Our input data depends on states and actions in the context of an MDP. Moreover, we are learning the reward that indirectly determines how actions are chosen during the sequential evolution of an MDP, whereas preference learning studies the latent functions preserving preferences. On the grounds of Bellman optimality for MDPs, the decision maker chooses optimal actions with the maximum value of Q-function at a given state. The preference relation will be determined by Q-functions, the expected long-term reward, while the random variable we concern is the immediate reward function, the intrinsic function determining the decision maker's behavior.

4.3 IRL with Gaussian Process

We turn now to the general IRL problem defined on an IMDP. The ill-posed nature of the problem gives rise to two principal issues: (1) uncertainty about reward functions given the observation of decision behavior and (2) ambiguity associated with observation of multiple actions at a single state.

To address uncertainty about reward functions, we propose to use Bayesian inference. To disambiguate observations at each state, we adopt a model in which reward function is contaminated by a Gaussian noise. In particular, we assume that the reward function can be modeled as $r+\vartheta$, where $\vartheta \sim N(0, \sigma^2)$ and $N(0, \sigma^2)$ is a Gaussian noise of zero mean and unknown variance. Figure4.1 illustrates two scenarios that correspond to this notion. In Figure4.1 (a) we observe a group of decision makers who share a common reward structure but vary individually in their perception of that structure. In Figure4.1 (b) we observe a single decision maker whose perception of reward varies noisily with the stage of the process. To lessen the ambiguity of observing multiple actions at a state, we argue that the basis for Bayesian inference should be an understanding of the agent's preferences over the action space, This argument is reasonable because the goal of IRL is to learn the reward subjectively perceived by the decision maker from whom we have collected the observation data. Based on decision theory, the intuition is that the decision makers will select some actions at a given state because they prefer these actions to others. These preferences among countable actions can be used to represent the multiple observations at one state.


Goal
Decision Maker 1 using $r + \text{noise}_1$
Decision Maker M using $r + \text{noise}_M$
Decision maker
Goal - r
Decision Maker – Reward function

(a) (b) (c)

Figure 4.1: Examples of observation structures for noisy reward scenarios: (a) observations come from a group of decision makers; (b) observations come from a single decision maker; and (c) our proposed observation structure for MDP.

4.3.1 Action preference graph

Given a state and its Q-functions, we can select the optimal action according to Bellman optimality. So we consider Q-function as the utility function for the definition of preference relations in a finite countable action space. Next, we define action preference relations and action preference graphs.

Definition At state s_n, $\forall \hat{a}, \check{a} \in \mathcal{A}$, we define the *action preference relation* as:

1. Action $\hat{a}$ is weakly preferred to $\check{a}$, denoted as $\hat{a} \succeq_{s_n} \check{a}$, if $Q(s_n, \hat{a}) \geq Q(s_n, \check{a})$;

2. Action $\hat{a}$ is strictly preferred to $\check{a}$, denoted as $\hat{a} \succ_{s_n} \check{a}$, if $Q(s_n, \hat{a}) > Q(s_n, \check{a})$;

3. Action $\hat{a}$ is equivalent to $\check{a}$, denoted as $\hat{a} \sim_{s_n} \check{a}$, if and only if $\hat{a} \succeq_{s_n} \check{a}$ and $\check{a} \succeq_{s_n} \hat{a}$.

Definition An *action preference graph* is a simple directed graph showing preference relations among the countable actions at a given state. At state s_n, its action preference graph $G_n = (\mathcal{V}_n, \mathcal{E}_n)$ comprising a set $\mathcal{V}_n$ of nodes together with a set $\mathcal{E}_n$ of edges. About node and edge in graph G_n let us define

1. Each node represents an action in $\mathcal{A}$. Define a one-to-one mapping $\varphi : \mathcal{V}_n \to \mathcal{A}$.

2. Each edge indicates a preference relation.

The following lemma provides the basis for the construction of the preference graph.

Lemma 7 *At state s_n, if action $\hat{a}$ is observed, we have these preference relations: $\hat{a} \succ_{s_n} \check{a}, \forall \check{a} \in \mathcal{A} \setminus \{\hat{a}\}$.*

Proof According to Bellman optimality, $\hat{a}$ is observed if $\hat{a} \in \arg\max_{a \in \mathcal{A}} Q(s_n, a)$. Therefore, we have

$$Q(s_n, \hat{a}) > Q(s_n, \check{a}),\ \forall \check{a} \in \mathcal{A} \setminus \{\hat{a}\}$$

According to the definition on preference relations, it follows that if $Q(s_n, \hat{a}) > Q(s_n, \check{a})$, we have $\hat{a} \succ_{s_n} \check{a}$.

Remark Here we give an example to show the details on drawing preference graphs. Let $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$. From the observation data set $\mathcal{O}$, we get

1. An observation of action a_1 at s_n, then edges are shown in Figure4.2(a).
2. An observation of action a_3 at s_n, then new edges are added in Figure4.2(b).

Note that preference relation has the properties such as:

1. If $\hat{a}, \check{a} \in \mathcal{A}$, then at state s_n either $\hat{a} \succeq_{s_n} \check{a}$ or $\check{a} \succeq_{s_n} \hat{a}$.
2. If $\hat{a} \succeq_{s_n} \check{a}$ or $\check{a} \succeq_{s_n} \tilde{a}$, then $\hat{a} \succeq_{s_n} \tilde{a}$.

Therefore we have a simple representation of the action preference graph that is constructed by a two-layer directed graph. E.g. Figure4.2(c) showing multiple actions at s_n and Figure4.2(d) displaying the unique action at s_n. In this two-layer directed graph, the top layer $\mathcal{V}_n^+$ is a set of nodes representing the observed actions and the bottom layer $\mathcal{V}_n^-$ has the nodes denoting other actions. The edge in the edge set $\mathcal{E}_n$ can be represented by a formulation of its beginning node u and ending node v. We write the k-th edge as $(u \rightarrow v)_k$ if $u \in \mathcal{V}_n^+, v \in \mathcal{V}_n^-$, or the l-th edge $(u \leftrightarrow v)_l$ if $u \in \mathcal{V}_n^-, v \in \mathcal{V}_n^-$. Recall the mapping between $\mathcal{V}_n$ and $\mathcal{A}$, the representation $u \rightarrow v$ indicates that action $\varphi(u)$ is preferred to $\varphi(v)$. Similarly, $u \leftrightarrow v$ means that action $\varphi(u)$ is equivalent to $\varphi(v)$.

Recall the more realistic examples in Figure4.1 (a) and (b), where the observation data $\mathcal{O}$ may be multiple decision trajectories containing non-deterministic policies. To address IRL problems for realistic cases, we propose to process $\mathcal{O}$ into the form

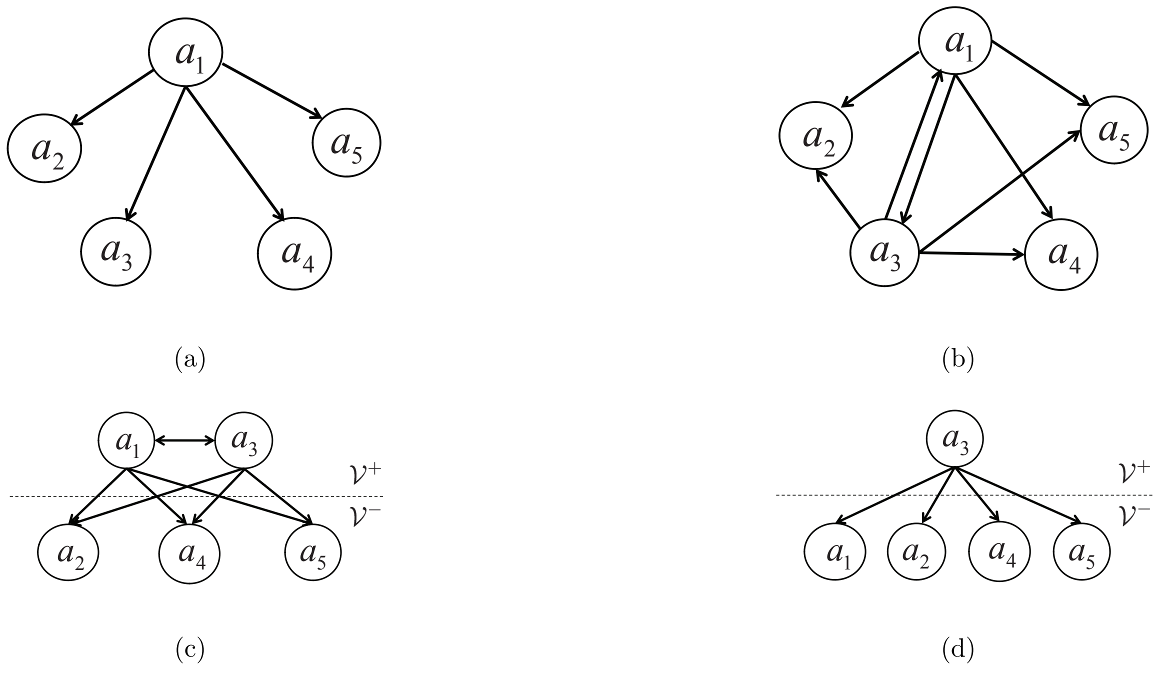


Figure 4.2: Examples of preference graph: (a) The first step to draw a preference graph. (b) The second step to draw a preference graph. (c) An example of observing two actions at a state. (d) An example of unique observation at a state.

of pairs of state and preference graph, e.g. the representation shown in Figure4.1(c), and then apply Bayesian inference using the new formulation. Toward this goal, we need to show the equivalence of evidence for inference of reward function between the use of decision trajectories and the independent pairs of state and preference graph (See Proposition 9). To show this proposition, we first state an obvious fact about the preference graph.

Lemma 8 *The set of preference relations, represented by a two-layered action preference graph G_n at state s_n, is an union of two preference set, written as*

$$\begin{aligned}\mathcal{E}_n &\equiv \left\{(\hat{a} \succ_{s_n} \check{a})_{k=1}^{n_n},\ \hat{a} \in \hat{\mathcal{A}}_n,\ \check{a} \in \mathcal{A} \setminus \hat{\mathcal{A}}_n\right\} \\ &\cup \left\{(\hat{a} \sim_{s_n} \hat{a}')_{l=1}^{m_n},\ \hat{a}, \hat{a}' \in \hat{\mathcal{A}}_n\right\}. \qquad (4.2)\end{aligned}$$

where $\varphi(\mathcal{V}_n^+) \rightarrow \hat{\mathcal{A}}_n$ and $\varphi(\mathcal{V}_n^-) \rightarrow \check{\mathcal{A}}_n$. The notation $(.)_k$ denotes the k-th strict preference relation and $(.)_l$ represents the l-th equivalent preference relation. The number of strict preference relations $n_n = |\hat{\mathcal{A}}_n| \times |\mathcal{A} \backslash \hat{\mathcal{A}}_n|$, and the number of equivalent preference relations $m_n = |\hat{\mathcal{A}}_n|(|\hat{\mathcal{A}}_n| - 1)/2$.

Then we have the following proposition.

Proposition 9 *The observation data set $\mathcal{O}_1$ is given as a set of decision trajectories. Assume independence among the observed decision trajectories. Observation of policy at a state can be specified by an action preference graph. Let $\mathcal{O}_2$ be the set of independent pairs of state and action preference graph, which is written as $\mathcal{O}_2 = \{(s_n, G_n)\}_{n=1}^N$. The inference of reward function drawn from $\mathcal{O}_1$ and $\mathcal{O}_2$ is identical. There is a constant factor C that makes likelihood function $p(\mathcal{O}_1|r) = Cp(\mathcal{O}_2|r)$.*

Proof Let $\mathcal{O}_1 \triangleq \{h_o\}_{o=1}^H$ denote a sampled decision trajectory, where h_o is a decision trajectory and

$$h_o =\triangleq \left\{s^{o1}, a^{o1} s^{o2}, a^{o2} \cdots, s^{ot_o}, a^{ot_o}\right\},$$

t_o is the length of the o-th trajectory and H is the number of sampled decision trajectories. The conditional probability of observing these decision trajectories in $\mathcal{O}_1$ is calculated by

$$\begin{aligned} p(\mathcal{O}|r) &= \prod_{o=1}^{H} p(s^{o1}) p(a^{o1}|s^{o1}, r) \\ &\quad \prod_{q=2}^{t_o} p(s^{oq}|s^{oq-1}, a^{oq-1}) p(a^{oq}|s^{oq}, r) \\ &\propto \prod_{o=1}^{H} \prod_{q=1}^{t_o} p(a^{oq}|s^{oq}, r) = \prod_{n=1}^{N} p(o_n|s_n, r) \end{aligned}$$

where $p(o_n|s_n, r) = \prod_{(o,q)\in\mathcal{B}} p(a^{oq}|s^{oq}, r)$,

and $\mathcal{B} \triangleq \{(o, q) : s^{oq} = s_n, o \in \{1, 2, \cdots, H\}, q \in \{1, 2, \cdots, t_o\}\}$.

Above equations show that the joint generating probability distribution of $\mathcal{O}$ is proportional to a multiplication of probabilities of taking observed actions at each state. According to definitions on preference relation, the preference among observed actions is equivalent relation and observed actions are strictly preferred to unobserved actions. Let $\hat{\mathcal{A}}_n = \{a^{oq}|s^{oq} = s_n\}$ denote the set of actions observed at state s_n. By Lemma 7, we have $p(\hat{a}|s_n, r) = p(\hat{a} \succ_{s_n} \check{a})$, $\forall \check{a} \in \mathcal{A} \setminus \hat{\mathcal{A}}_n$. Consider two actions $\hat{a}, \hat{a}' \in \hat{\mathcal{A}}_n$. Since $\hat{a} \succ_{s_n} \check{a}$ and $\check{a} \succ_{s_n} \hat{a}$, we have $\hat{a} \sim_{s_n} \check{a}$.

Thus, the likelihood function at state s_n is written as

$$p(o_n|s_n, r) = \prod_{k=1}^{n_n} p((\hat{a} \succ_{s_n} \check{a})_k) \prod_{l=1}^{m_n} p((\hat{a} \sim_{s_n} \hat{a}')_l) \tag{4.3}$$

where $\hat{a}, \hat{a}' \in \hat{\mathcal{A}}_n$ and $\check{a} \in \check{\mathcal{A}}_n$. Now, according to Lemma 4, we see the likelihood function for observations of sampled decision trajectories $p(\mathcal{O}_1|r)$ is proportional to the production of likelihood function $p(G_n|s_n, r)$, which is $p(\mathcal{O}_1|r) \propto p(\mathcal{O}_2|r) = \prod_{n=1}^{N} p(G_n|s_n, r)$. The proportional constant $C = \prod_{o=1}^{H} p(s^{o1}) \prod_{q=2}^{t_o} p(s^{oq}|s^{oq-1}, a^{oq-1})$ can be easily shown.

Remark In the proof of proposition 9, we find that at state s_n the observed set of actions $\hat{A}_n$ implies that

$$\begin{cases} Q(s_n, \hat{a}) > Q(s_n, \check{a}) \Rightarrow \hat{a} \succ_{s_n} \check{a}, \forall \hat{a} \in \hat{\mathcal{A}}_n, \check{a} \in \mathcal{A} \setminus \hat{\mathcal{A}}_n \\ |Q(s_n, \hat{a}) - Q(s_n, \hat{a}'))| \leq \eta \Rightarrow \hat{a} \sim_{s_n} \hat{a}', \forall \hat{a}, \hat{a}' \in \hat{\mathcal{A}}_n \end{cases}$$

where η is a small positive number. We will use this implication to formulate the likelihood function for Bayesian inference in next section.

Based on Proposition 9, we can represent $\mathcal{O}$ as shown in Figure4.1 (c). At state s_n, its action preference graph is constructed by a two-layer directed graph: a set of nodes $\mathcal{V}_n^+$ in the top layer and a set of nodes $\mathcal{V}_n^-$ in the bottom layer.

Consider the existence of alternative actions at a state. We adopt a reward structure depending on both state and action. Let $\mathbf{r}$ be the vector containing the reward for m possible actions at N observed states. We have

$$\begin{aligned}\mathbf{r} &= (\underbrace{\mathbf{r}_{a_1}(s_1), ..., \mathbf{r}_{a_1}(s_N)}, \ldots, \underbrace{\mathbf{r}_{a_m}(s_1), \ldots, \mathbf{r}_{a_m}(s_N)}) \\ &= (\quad \mathbf{r}_{a_1}, \quad \cdots, \quad \mathbf{r}_{a_m}) \end{aligned} \tag{4.4}$$

where $\mathbf{r}_{a_m}, \forall m \in \mathcal{M}$, denotes the reward for m-th action.

4.3.2 Bayesian inference

Bayesian analysis combines the prior knowledge about the hypotheses $\mathbf{r}$ and the observed data $\mathcal{O}$ into what is called posterior distribution of $\mathbf{r}$ given $\mathcal{O}$. We quantify the evidence that $\mathbf{r}$ is the underlying function for the true reward by this posterior distribution. To deal with Generalized IRL in large and infinite state space, we use Gaussian process to represent the prior knowledge.

Gaussian prior

Consider $\mathbf{r}_{a_m}$ a Gaussian process. For any $\{s_1, \cdots, s_N\} \in \mathcal{S}$, the random variables $\{\mathbf{r}_{a_m}(s_1), \cdots, \mathbf{r}_{a_m}(s_N)\}$ are normally distributed. We denote by $k_{a_m}(s_c, s_d)$ the function generating the value of entry (c, d) for covariance matrix $\mathbf{K}_{a_m}$, which leads to $\mathbf{r}_{a_m} \sim N(0, \mathbf{K}_{a_m})$. Then the joint prior probability of the reward is a product of multivariate Gaussian, namely $p(\mathbf{r}|\mathcal{S}) = \prod_{m=1}^{M} p(\mathbf{r}_{a_m}|\mathcal{S})$ and $\mathbf{r} \sim N(0, \mathbf{K})$. Note that $\mathbf{r}$ is completely specified by the positive definite covariance matrix $\mathbf{K}$.

A simple strategy is to assume that the M latent processes are uncorrelated. Then the covariance matrix $\mathbf{K}$ is block diagonal in the covariance matrices $\{\mathbf{K}_1, ..., \mathbf{K}_M\}$.

In practice, we use a squared exponential kernel function, written as:

$$k_{a_m}(s_c, s_d) = e^{\frac{1}{2}(s_c - s_d)\mathbf{M}_{a_m}(s_c - s_d)} + \sigma^2_{a_m}\delta(s_c, s_d)$$

where $\mathbf{M}_{a_m} = \kappa_{a_m}\mathbf{I}_N$ and $\mathbf{I}_N$ is an identity matrix of size N. The function $\delta(s_c, s_d) = 1$, when $s_c = s_d$; otherwise $\delta(s_c, s_d) = 0$. Under this definition the covariance is almost unity between variables whose inputs are very close in the Euclidean space, and decreases as their distance increases. An advantage of choosing this kernel function for Gaussian process is summarized in the following lemma in [45].

Lemma 10 *Assume a Gaussian process F with zero mean function and squared exponential covariance function. Using F to model the latent function is equivalent to learning using infinitely many Gaussian shaped basis functions placed everywhere*

After setting the prior distribution, we still need to know the marginal likelihood distribution $p(\mathcal{G}|\mathcal{S}, \mathbf{r})$ for Bayesian inference. In the following sections, we will show how to choose an appropriate likelihood distribution that favor the least complex model able to fit the data, rather than the best fitting model.

Likelihood

Consider the likelihood function for action preference graph. We generate two kinds of likelihood functions: one for strict preference relations and the other for equivalence preference relations.

Let us first establish a Gaussian form for the strict preference relation $(\hat{a} \succ_{s_n} \check{a})_k$. Its likelihood function is calculated by

$$p((\hat{a} \succ_{s_n} \check{a})_k) = \Phi\left(f(\mathbf{r}, s_n, k)\right) \tag{4.5}$$

where $f(\cdot)$ is a linear function of $\mathbf{r}$, and Φ is a cumulative normal distribution function. To derive the above Gaussian form, we adopt a variation of likelihood function proposed by Chu and Ghahramani in [38] to capture the strict preference relation. The likelihood function for the reward without noise is as follows,

$$p_{\text{ideal}}(\hat{a} \succ_{s_n} \check{a} | \mathbf{r}_{\hat{a}}(s_n), \mathbf{r}_{\check{a}}(s_n)) = \begin{cases} 1 & \text{if } Q(s_n, \hat{a}, \mathbf{r}) > Q(s_n, \check{a}, \mathbf{r}) \\ 0 & \text{otherwise} \end{cases}$$

This form of likelihood function requires that the preference relation should be consistent with the ranking hypothesis on values of Q-functions. Without loss of generality, we assume the transition probability matrices are known here. Then Q-function takes the form,

$$Q(s_n, a_m, \mathbf{r}) = \mathbf{r}_{a_m}(s_n) + \gamma \mathbf{P}_{a_m}(s_n, :) \left(\mathbf{I}_N - \gamma \mathbf{P}_{\pi(s_n)}(s_n, :)\right)^{-1} \hat{\mathbf{I}} \mathbf{r} \tag{4.6}$$

where $\hat{\mathbf{I}}$ is a sparse matrix with N rows and $N \times M$ columns, which is written as $\hat{\mathbf{I}} = (\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_2, \cdots, \hat{\mathbf{I}}_M)$. The subindex of $\hat{\mathbf{I}}_m$ denotes the relation with the j-th action. Let $\hat{\mathbf{I}}_m$ be an $N \times N$ matrix whose entires are 0 except $\hat{\mathbf{I}}_m(n, n) = 1,\ i \in \{1, 2, \ldots, N\}$ if, at state s_n, the action a_m is optimal. Hence, the n element of $\hat{\mathbf{I}}\mathbf{r}$ is $r_{a^*}(s_n), \forall n \in \mathcal{N},\ a^* = \text{argmax}_a Q(s_n, a)$. If the latent functions have been contaminated with Gaussian noise that has zero mean and unknown variance σ^2, the likelihood function for k-th strict preference relation is

$$\begin{aligned} & p((\hat{a} \succ_{s_n} \check{a})_k | \mathbf{r}_{\hat{a}}(s_n) + \delta_{\hat{a}}, \mathbf{r}_{\check{a}}(s_n) + \delta_{\check{a}}) \\ & = \int \int p_{\text{ideal}}(\hat{a} \succ_{s_n} \check{a} | \mathbf{r}_{\hat{a}}(s_n), \mathbf{r}_{\check{a}}(s_n)) \\ & N(\delta_{\hat{a}} | 0, \sigma^2) N(\delta_{\check{a}} | 0, \sigma^2) d\delta_{\hat{a}} d\delta_{\check{a}} = \Phi(f(\mathbf{r}, s_n, k)) \end{aligned} \tag{4.7}$$

where $f(\mathbf{r}, s_n, k) = \frac{Q(s_n, \hat{a}) - Q(s_n, \check{a})}{\sqrt{2}\sigma}$, and $\Phi(f(\mathbf{r}, s_n, k)) = \int_{-\infty}^{f(\mathbf{r}, s_n, k)} N(v|0, 1) dv$, which has the desired form.

Next, we give define likelihood function for the equivalenence preference relation. Consider the l-th preference relation $(\hat{a} \sim_{s_n} \hat{a}')_l$, where $\hat{a},\ \hat{a}' \in \hat{\mathcal{A}}_n$. Its likelihood function is

$$p((\hat{a} \sim_{s_n} \hat{a}')_l|\mathbf{r}) \propto e^{-\frac{1}{2}(Q(s_n,\hat{a})-Q(s_n,\hat{a}'))^2} = e^{-\sigma^2 f^2(\mathbf{r},s_n,l)} \tag{4.8}$$

This likelihood function is designed to that if the Q-values are close for two actions the corresponding nodes in the preference graph are more likely to have an equivalence preference relation, since the actions are close in attractiveness to the decision maker.

Combining Eq. 4.5 and Eq. 4.8, we arrive at the formulation of the likelihood function for observation data with preference graphs. We summarize the likelihood function in the following proposition.

Proposition 11 *The likelihood function, giving the evidence of the observation data $\mathcal{O}$ in the form of pairs of state and action preference graph, is calculated by*

$$p(\mathcal{G}|\mathcal{S},\boldsymbol{r}) = \prod_{n=1}^{N} p(G_n|s_n,\boldsymbol{r}) = \prod_{n=1}^{N}\prod_{k=1}^{n_n} \Phi(f(\boldsymbol{r},s_n,k))e^{\sum_{n=1}^{N}\sum_{l=1}^{m_n} -\sigma^2 f^2(\boldsymbol{r},s_n,l)} \tag{4.9}$$

To summarize, the probabilistic IRL model is controlled by kernel parameters κ_{a_m} and σ_{a_m} for computing the covariance matrix of reward realizations, and σ to tune the noise level in the likelihood function. We put these parameters into the hyper-parameter vector $\boldsymbol{\theta} = (\kappa_{a_m}, \sigma_{a_m}, \sigma)$. More often than not, we do not have *a priori* knowledge of the hyper-parameters. To address this problem, we apply maximum a posterior estimate to adjust the hyper-parameters in Section 4.3.3.

Posterior inference

Here we adopt a hierarchical model. At the lowest level are reward function values encoded as a parameter vector $\mathbf{r}$. At the top level are hyper-parameters in $\boldsymbol{\theta}$ controlling the distribution of the parameters at the bottom level. Inference takes place one level at a time. At the bottom level, the posterior over function values is given by Bayes' rule

$$p(\mathbf{r}|\mathcal{S},\mathcal{G},\boldsymbol{\theta}) = \frac{p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta},\mathbf{r})p(\mathbf{r}|\mathcal{S},\boldsymbol{\theta})}{p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})}. \tag{4.10}$$

The posterior combines the information from the prior and the data, which reflects the updated belief about $\mathbf{r}$ after observing the decision behavior. We can calculate the denominator in Eq.4.10 by integrating $p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta},\mathbf{r})$ over the function space with respect to $\mathbf{r}$, which requires a large amount of computation, as discussed in a later section. Fortunately, we are able to maximize the unnormalized posterior density of $\mathbf{r}$ without calculating the normalizing denominator, since the denominator $p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})$ is independent of the values of $\mathbf{r}$. In practice, we obtain the maximum posterior by minimizing the negative log posterior, which is written as

$$\begin{aligned} U(\mathbf{r}) &\triangleq \frac{1}{2}\sum_{m=1}^{M}\mathbf{r}_{a_m}^T\mathbf{K}_{a_m}^{-1}\mathbf{r}_{a_m} - \sum_{n=1}^{N}\sum_{k=1}^{n_n}\ln\Phi(\sum_{m=1}^{M}\rho_{a_m}^{nk}\mathbf{r}_{a_m}) \\ &+\sum_{n=1}^{N}\sum_{l=1}^{m_n}\frac{1}{2}(\sum_{m=1}^{M}\rho_{a_m}^{nl}\mathbf{r}_{a_m})^2 \end{aligned} \tag{4.11}$$

where given $(\hat{a}\sim_{s_n}\hat{a}')_l$, let $\boldsymbol{\Delta}_l \triangleq \gamma(\mathbf{P}_{\hat{a}}(s_n,:)-\mathbf{P}_{\hat{a}'}(s_n,:))(\mathbf{I}_N-\gamma\mathbf{P}_{\pi(s_n)}(s_n,:))^{-1}$, then we have

$$\rho_{a_m}^{nl} = \mathbf{e}_n[\mathbf{1}(a_m=\hat{a})-\mathbf{1}(a_m=\hat{a}')]+\boldsymbol{\Delta}_l\hat{\mathbf{I}}_{a_m}$$

where $\mathbf{e}_n$ is a $1\times N$ vector whose entry $\mathbf{e}_n(n)=1$, and $\mathbf{e}_n(j)=0,\forall j\in\mathcal{N}\setminus\{n\}$. The notation $\mathbf{1}(.)$ is an indicator function. Similarly, $\rho_{a_m}^{nk}$ denotes the coefficient vector for the k-th strict preference relation $\hat{a}\succ_{s_n}\check{a}$.

Proposition 12 *Minimizing Eq.4.11 is a convex optimization problem.*

We give the proof for Proposition 12 in Appendix 8.1.

At the minimum of $U(\mathbf{r})$ we have

$$\frac{\partial U}{\partial \mathbf{r}_{a_m}} = 0 \Rightarrow \hat{\mathbf{r}}_{a_m} = K_{a_m}(\nabla \log P(\mathcal{G}|\mathcal{S}, \hat{\mathbf{r}}, \boldsymbol{\theta})) \tag{4.12}$$

where $\hat{\mathbf{r}} = (\hat{\mathbf{r}}_1, \cdots, \hat{\mathbf{r}}_{a_m}, \cdots, \hat{\mathbf{r}}_m)$. In Eq.4.12, we can use Newton's method to find the maximum of U with the iteration,

$$\mathbf{r}_{a_m}^{\text{new}} = \mathbf{r}_{a_m} - (\frac{\partial^2 U}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}})^{-1} \frac{\partial U}{\partial \mathbf{r}_{a_m}}.$$

4.3.3 Model selection

The Gaussian probabilistic model discussed so far is fully specified by the hyper-parameter vector $\boldsymbol{\theta}$. In principle, we can do inference jointly over $\mathbf{r}$ and $\boldsymbol{\theta}$ using sampling techniques. Another approach in the view of a hierarchical model is to learn the hyper-parameters from data and prior knowledge at the top level, which can be considered training of a Gaussian process [37]. In other words, we are concerned with how to choose the parameters for the the Gaussian process to model the observed data well.

Now at the top level, we can optimize the hyper-parameters by maximizing the posterior over these hyper-parameters, $p(\boldsymbol{\theta}|\mathcal{G}, \mathcal{S})$. The marginal likelihood from the first level plays the role of the likelihood in Bayesian equation:

$$p(\boldsymbol{\theta}|\mathcal{G}, \mathcal{S}) = \frac{p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{G}|\mathcal{S})}. \tag{4.13}$$

Discard the items not containing $\boldsymbol{\theta}$ and consider the prior distribution of hyper-parameters with no population basis. Optimization over $\boldsymbol{\theta}$ becomes the problem of maximizing the marginal likelihood also known as the evidence framework [46]. Calculation of marginal likelihood calls for

$$p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta}) = \int p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta}, \mathbf{r})p(\mathbf{r}|\mathcal{S}, \boldsymbol{\theta})d\mathbf{r}.$$

We approximate the integral of the marginal likelihood $p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})$ using a Laplace approximation (LA) local expansion around the maximum, which yields

$$p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta}) \approx p(\mathcal{G}|\mathcal{S},\hat{\mathbf{r}},\boldsymbol{\theta}) \times p(\hat{\mathbf{r}}|\boldsymbol{\theta})\xi_{\mathbf{r}|\mathcal{S}} \tag{4.14}$$

where $\xi_{\mathbf{r}|\mathcal{S}} = |-\nabla\nabla \ln P(\mathbf{r}|\mathcal{G},\mathcal{S},\boldsymbol{\theta})|^{-\frac{1}{2}}$ is called an *Occam factor* that encodes the posterior uncertainty in $\mathbf{r}$. Thus the evidence is found by computing the best fit likelihood $P(\mathcal{G}|\mathcal{S},\hat{\mathbf{r}},\boldsymbol{\theta})$ that the model can achieve and multiply it by an Occam factor. By its nature, marginal likelihood incorporates a trade-off between model fit and model complexity. The approximation is expected to become increasingly accurate with increasing data. The marginal likelihood is approximated by Eq.4.14, yielding

$$\log p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta}) \approx -\frac{1}{2}\sum_{m=1}^{M}\hat{\mathbf{r}}_{a_m}^T\mathbf{K}_{a_m}^{-1}\hat{\mathbf{r}}_{a_m} \underbrace{-\frac{1}{2}\log|I+\mathbf{K}\Pi|}_{\text{complexity penalty}}$$
$$\underbrace{-\sum_{n=1}^{N}\sum_{l=1}^{m_n}\frac{1}{2}(\sum_{m=1}^{M}\rho_{a_m}^{nl}\hat{\mathbf{r}}_{a_m})^2+\sum_{n=1}^{N}\sum_{k=1}^{n_n}\ln\Phi(\sum_{m=1}^{M}\rho_{a_m}^{nk}\hat{\mathbf{r}}_{a_m}),}_{\text{data fit likelihood}}$$

where $\hat{\mathbf{r}}$ is the maximum of the posterior in Eq.4.10 and Π is the negative Hessian of $\ln p(\mathcal{G}|\hat{\mathbf{r}},\mathcal{S}\boldsymbol{\theta})$. Now we can find the optimal hyper-parameters by maximizing $\log p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})$. To this end, we seek partial derivatives on variables in $\boldsymbol{\theta}$. Gradient decent optimization methods may be adopted to determine the values of the hyper-parameters $\boldsymbol{\theta}$, as follows:

$$\frac{\partial \log P(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})}{\partial \kappa_{a_m}} = \frac{1}{2}\hat{\mathbf{r}}_{a_m}^T\mathbf{K}_{a_m}^{-1}\frac{\partial \mathbf{K}_{a_m}}{\partial \kappa_{a_m}}\mathbf{K}_{a_m}^{-1}\hat{\mathbf{r}}_{a_m}$$
$$-\frac{1}{2}tr\left[(I+\mathbf{K}\Pi)^{-1}(\frac{\partial \mathbf{K}}{\partial \kappa_{a_m}}\Pi+\mathbf{K}\frac{\partial \Pi}{\partial \kappa_{a_m}})\right]$$
$$\frac{\partial \log P(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})}{\partial \sigma} = -\frac{1}{2}tr\left[(I+\mathbf{K}\Pi)^{-1}\mathbf{K}\frac{\partial \Pi}{\sigma}\right]$$
$$\sum_{n=1}^{N}\sum_{k=1}^{n_n}\left[-\frac{\sum_{m=1}^{M}\rho_{a_m}^{nk}\mathbf{r}_{a_m}}{\sqrt{2}\Phi\left(f(\mathbf{r},s_n,k)\right)\sigma^2}N(\frac{\sum_{m=1}^{M}\rho_{a_m}^{nk}\mathbf{r}_{a_m}}{\sqrt{2}\sigma},0,1)\right].$$

The derivative of $\log P(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta})$ with respect to σ_{a_m} has the similar formulation as the derivative with respect to κ_{a_m}.

There are also other approximation techniques, e.g., expectation propagation (EP) which is an iterative method to find approximations based on approximate marginal moments. Experimental results generally show that EP can yield a more accurate approximation than Monte Carlo simulations, but this comes at the expense of additional computation. To balance the tradeoff between accuracy and computational efficiency, we adopt the LA approach in our algorithm. Extensive discussions about the approximation methods are given in [47].

4.3.4 Learning in large state space

Posterior predictive reward

If the state space is very large and even infinite, and we only have the computation ability to deal with a certain size of observation data, it is profitable to make the model learn from the limited number of observations and be able to predict the reward successfully at the new coming state. This also implies that the recovered reward function will be smoother. To make predictions at a new state s^*, we have to know the posterior distribution $p(\mathbf{r}^*|\mathcal{G}, \mathcal{S}, s^*, \boldsymbol{\theta})$, where $\mathbf{r}^*$ is a $M \times 1$ vector. We have,

$$p(\mathbf{r}^*|\mathcal{G}, \mathcal{S}, s^*, \boldsymbol{\theta}) = \int p(\mathbf{r}^*|\mathcal{G}, \mathcal{S}, s^*, \mathbf{r}, \boldsymbol{\theta}) p(\mathbf{r}|\mathcal{S}, \mathcal{G}, \boldsymbol{\theta}) d\mathbf{r}. \tag{4.15}$$

In Section 4.3.2, we have assumed that latent function values have Gaussian noise with variance σ^2. Denote the noisy function values as $\hat{r}(s) = r(s) + \delta$, and $\delta \sim N(0, \sigma^2)$. Laplace approximation provides a Gaussian approximation to $p(\mathbf{r}|\mathcal{S}, \mathcal{G}, \boldsymbol{\theta})$. Since both $p(\mathbf{r}^*|\mathcal{G}, \mathcal{S}, s^*, \mathbf{r}, \boldsymbol{\theta})$ and $p(\mathbf{r}|\mathcal{S}, \mathcal{G}, \boldsymbol{\theta})$ are Gaussian, $p(\mathbf{r}^*|\mathcal{G}, \mathcal{S}, s^*, \boldsymbol{\theta})$ is also

Gaussian. Therefore, we only need to compute the mean and the covariance. The predictive computation for action a_m is given by

$$\begin{aligned}
&\mu^*_{a_m} = E(\mathbf{r}^*_{a_m}|\mathcal{G}, \mathcal{S}, s^*, \boldsymbol{\theta}) = k_{a_m}(\mathbf{S}, s^*)^T(\mathbf{K}_{a_m} + \sigma^2 I)^{-1}\hat{\mathbf{r}}_{a_m} \\
&\mathrm{cov}(\mathbf{r}^*_{a_m}|\mathcal{G}, \mathcal{S}, s^*, \boldsymbol{\theta}) = \\
&k_{a_m}(s^*, s^*) - k_{a_m}(\mathbf{S}, s^*)^T(\mathbf{K}_{a_m} + \sigma^2 I)^{-1}k_{a_m}(\mathbf{S}, s^*)
\end{aligned} \tag{4.16}$$

where $\mathbf{S}$ is a matrix being composed of state vectors and $k_{a_m}(\mathbf{S}, s^*)$ is the vector of covariance between the test point and training points relating to action a_m.

Posterior predictive actions

Reward plays a role on the preference graph through Q functions. That is, if action a^* is optimal at state s, the function values satisfy these inequalities: $Q^\pi(s, a^*) \geq Q^\pi_{a\in\mathcal{A}}(s, a)$.

Definition An action $\hat{a}$ dominates at state s if and only if $\hat{a} \succeq_s \check{a}, \forall\check{a} \in \mathcal{A}$. An action set $\mathcal{A}^+$ dominates another set $\mathcal{A}^-$, denoted as $\mathcal{A}^+ \succ \mathcal{A}^-$, if and only if $\hat{a} \succ \check{a}$, $\forall\hat{a} \in \mathcal{A}^+$ and $\forall\check{a} \in \mathcal{A}^-$.

The preference prediction on two nodes is given by

$$p(u \succ v|\mathcal{S}, \mathcal{G}) \;=\; \int p(u \succ v|\mathcal{S}, \mathcal{G}, \mathbf{r})p(\mathbf{r}|\mathcal{S}, \mathcal{G})d\mathbf{r}$$

where we assign the uniform distribution over the state transition probability for the test point.

Proposition 13 *Given $\mathcal{A}_* \subset \mathcal{A}$, $\forall\check{\mathcal{A}} \subset \mathcal{A} \setminus \mathcal{A}_*$, $\mathcal{A}_* \succ \check{\mathcal{A}}$ and $\hat{a} \sim \check{a}, \forall\hat{a},\ \check{a} \in \mathcal{A}_*$, policy π is optimal if and only if, $\forall s \in \mathcal{S}$, $\pi(s) \in \mathcal{A}_*$.*

Proof If π is optimal, then $\forall s \in S$, $Q(s, \pi(s)) = \max_a Q(s, a)$. We have $\forall a \in \mathcal{A}$, $Q(s, \pi(s)) \geq Q(s, a)$, then $\pi(s) \succ a$. Assume $\pi(s) \notin \mathcal{A}_*$. By Definition, we get $\exists \hat{a} \in \mathcal{A}_*$, $\hat{a} \succ \pi(s)$, which conflicts the assumption that π is optimal. So we conclude $\pi(s) \in \mathcal{A}_*$.

If $\forall s \in S$, $\pi(s) \in \mathcal{A}_*$ and $A_* \succ \check{\mathcal{A}}$ $\hat{a} \sim \check{a}, \forall \hat{a}$, $\check{a} \in \mathcal{A}_*$ are true, we have $\pi(s) \succ a, \forall a \in \mathcal{A} \setminus \mathcal{A}_*$, and $\pi(s) \sim \hat{a}, \forall \hat{a} \in \mathcal{A}_*$. Hence $Q(s, \pi(s)) \geq Q(s, a), a \in \mathcal{A}$, so π is optimal.

4.3.5 Model checking and improvement

In Bayesian inference, the posterior distribution of the reward may underestimate the uncertainty because the model may be not able to fit the data, so we need to check the model against the observed data.

Model checking is for self-consistency purpose. We view model checking here as the comparison of observation data to replicated data under the model. If the replicated data generated by the model is similar to the observed data, we may say the model fits. A basic Bayesian data analysis technique for checking the fit of a model is to compare the observed data with the simulated samples from the posterior predictive distribution of replicated data [48]. Any obvious difference between them indicates the potential failure of the model. However the decision model is not concerned with the individual pair of instance-target but underscores whether the objective will be reached. Based on this basic idea, we design a technique to check whether the learned decision rules are able to replicate decision behaviors as similar as those of the expert. Later in our experiments, first we learn the reward from observed decision trajectories and then use this learned reward to solve new generated problems, the results of which will be considered as the simulated samples. At the same time we obtain the observation data from the expert's performance on the these new generated problems. Finally, we compare these observation data with the simulated samples to see whether our model is able to capture the expert's goal.

4.4 Approximate Gaussian Process IRL

The standing assumption in previous sections has been that the problems are in finite space. But in some real-world problems, the state space can be infinitely large or continuous, making it costly at best to observe a large number of decision trajectories and precluding the use of IRL techniques developed for finite space, which suffer from the curse of dimensionality.

In particular, if the state space is very large, one or more of the following may happen:

- The dynamics may be unknown as the system becomes more complicated.
- Prediction on the new coming state is required, as the observation data set is relatively small in comparison with the whole state space and the recovered values of the reward function are not enough to solve new problems.
- For small size problems, IRL can use pre-defined Gaussian prior distribution on the reward, while for a large complicated system, it is difficult to set the covariance function appropriately.

Most IRL methods deal with the problems in finite space with known transition probabilities. Ng proposed a method for IRL within infinite state space using linear function approximation. After sampling the optimal policy and generating randomly chosen policies, a linear programing problem is used to maximize the difference between the optimal and random policies' value functions. This method has a high computational and suffers from some ambiguity about how to differentiate two optimal policies.

IRL using Gaussian process can be used to address problems with infinitely large state space, provided the following two assumptions hold:

- Though the observation data is relatively small in the large or infinite space, the observations are sampled in a way the expert's behavior is well represented.
- The size of observation data is still in the range that a computer can deal with in memory.

Using observed decision trajectories, we are able to estimate an approximate state transition probability $\tilde{P}$ on a Cartesian space $\hat{\mathcal{S}} \times \hat{\mathcal{S}}$. Based on the notation of induced MDP in [49] and [50], we define an induced IRL model correspondingly.

Definition Given a number of decision trajectories as many as we can use to obtain $\tilde{\mathbf{P}}$ to approximate transition probabilities accurately in $\hat{\mathcal{S}} \times \hat{\mathcal{S}}$. Then the *induced inverse MDP* $\hat{M}_I = \left\{(\hat{\mathcal{S}} \cup s_0, \mathcal{A}, \hat{\mathcal{P}}, \gamma, \mathcal{O})\right\}$, where $\hat{\mathcal{P}} = \left\{\hat{\mathbf{P}}_a\right\}_{a=1}^{m}$ and

- $\forall a \in \mathcal{A}$, we have $\hat{\mathbf{P}}_{as_0}(s_0) = 1$. Thus s_0 is an absorbing state.
- $\forall i, j \in \hat{\mathcal{S}}$ and $a \in \mathcal{A}$, $\hat{\mathbf{P}}_{ai}(j) = \tilde{\mathbf{P}}_{ai}(j)$
- $\forall i \in \hat{\mathcal{S}}$ and $a \in \mathcal{A}$, $\hat{\mathbf{P}}_{ai}(s_0) = 1 - \sum_{j \in \hat{\mathcal{S}}} \tilde{\mathbf{P}}(j)$

There has been some work showing that if the number of trajectories exceeds a threshold value, then the estimate of transition probability has bounded error in comparison with the problem with known transition probability. Applying the work from Section 4.3 in induced IMDP, we summarize the approximate Gaussian IRL method in Algorithm 6.

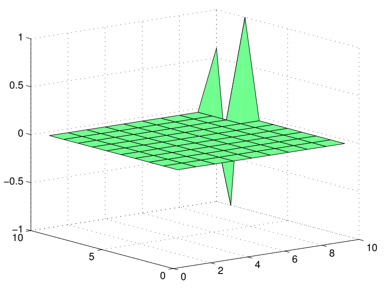
Algorithm 6 Gp IRL in continuous state space

```
 1: Initialization, Given trajectories
    $h_t = \{(s^1, a^1, s^2), (s^2, a^2, s^3), \cdots\}, t = \{1, 2, \cdots, H\}$.
 2: for $t = 1$ to $H$ do
 3:     for $(s^i, a^i, s^{i+1}) \in h_t$ do
 4:         $\tilde{\mathbf{P}}(s^i, a^i, s^{i+1}) += 1$
 5:     end for
 6: end for
 7: Normalize transition probability $\tilde{\mathbf{P}}$
 8: Construct induced inverse MDP model
 9: $\mathbf{r}$= $\arg\max p(\mathcal{G}|s, \mathbf{r}, \boldsymbol{\theta}) p(\mathbf{r}|\boldsymbol{\theta})$ solved in Section 4.3
10: Prediction on other unobserved states using Eq. 16
11: return $\mathbf{r}$
```

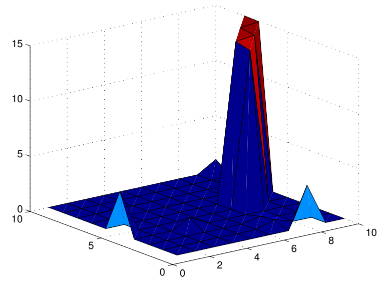
4.5 Experiments

4.5.1 *GridWorld* problem

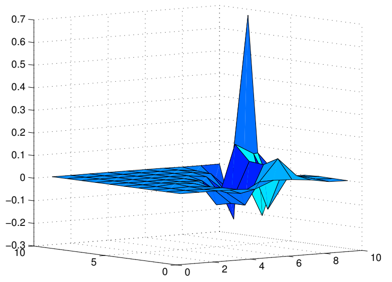
The grid world problem, in which the agent starts from an initial square on a grid and attempts to reach a destination square, has been extensively studied in many academic areas. In our experiments, we consider a grid that contains an obstacle that can block movement. The agent is able to take five actions, moving in four directions or staying



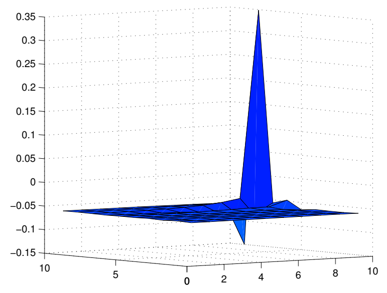
(a) True reward



(b) Reward by linear IRL



(c) Reward by QPIRL



(d) Reward by GPIRL

Figure 4.3: Comparisons of the recovered reward shape.

put. The outcome of the moving actions succeeds in probability 0.65, fails to move in probability 0.15, and results in unforeseen direction in probability 0.2. Considering this problem as benchmark experiment, we simulate an expert who navigates in the grid world with its own objective. We then use IRL techniques to learn the agent's reward function and compare the performances of our methods and other standard algorithms. Two kinds of experiments are discussed below.

Simple goal and complete observations

The first experiment is a simple grid world problem where complete observations at every state in a finite domain are available. The problem is simple in the sense of the goal; the expert starts from the left-lower corner square and moves until arriving at one destination corner. We do this experiment not to mimic the expert though complete observations, but to study the reward structure to compare different algorithms. We compare our method *QPIRL* in Algorithm 5 and [51], and *GPIRL* in Section 4.3 with the standard algorithm *LIRL* based on linear programming in [2]. Note that the reward is not known to the algorithms. But the complete observations of the expert's navigations are available. With complete observations, all the algorithms are able to recover the reward, by which reinforcement learning is able to output a best policy that is the same as the optimal policy of the expert. However, the shape of reward given by our method looks the most similar to the true reward, as can be seen in Figure4.3. In addition, we input the recovered reward functions to standard RL method, then we record the RL computation time in Table 4.1, in seconds. Each entry of time is calculated by averaging the RL computation time in 50 times of independent runs. These results suggest that our method tends to produce rewards that are shaped in the sense that they lead to fast convergence (cf. [52]). Here we do not go further in investigating the relationships between reward structure and convergence rate, but suggest this would be a fruitful area for future research.

Table 4.1: Time(sec) to find the apprentice policy

GridWorld Size	r-LIRL	r-QPIRL	r-GPIRL
10x10	2.61	2.06	1.20
20x20	20.05	15.75	9.32
30x30	75.12	64.30	35.11

Hybrid goal and incomplete observations

A hybrid goal in grid world problem implies that multiple squares may be the objective destination for the expert, with the desire for each destination square being governed by a probability distribution. From the view of apprenticeship learning, we access only partially sampled state-action pairs from the expert. According to the idea of model checking in Section 4.3.5, the IRL performance is proportional to the performance of the best policy output by the algorithm, which measures how the best policy is consistent with the observed policies. We report the performance of our algorithms, CPIRL and GPIRL, in comparison with the following methods:

- *Mimic the expert*, which for any state uses the expert's action if that action has been observed for the state, and a random action otherwise.
- *Linear IRL*, *Apprenticeship learning* (AL) in [21] [22].

Given the sampling number of decision trajectories, an average accuracy is calculated by steps:

1. Sample a certain number of decision trajectories.
2. Learn the reward using IRL algorithms.
3. Obtain the best policy for apprentice by dynamic programming using the recovered reward.
4. Generate 1000 grid-world tasks with random initial state as testing data, then let the expert and apprentice navigate in these grid-world tasks simultaneously and independently within fixed horizon limitation. Figure4.4 illustrates how the expert and the apprentice navigate in a grid world to reach the right-upper

corner with maximum reward. But due to noisy action, the real movement may differ from the planned direction. Both the expert and the apprentice reach the goal state, though their trajectories are quite different at the beginning. If the expert fails in the task, this task will not be counted in evaluation of the performance. The accuracy of IRL (*Acc*) is defined as the ratio of the performance of the apprentice and the expert, $Acc = \frac{n_a}{n_e}$, where n_a is the number of times the apprentice is able to achieve the goals and n_e is for the expert.

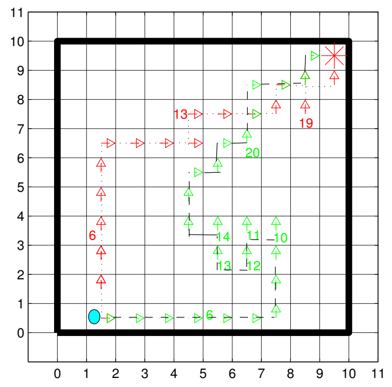
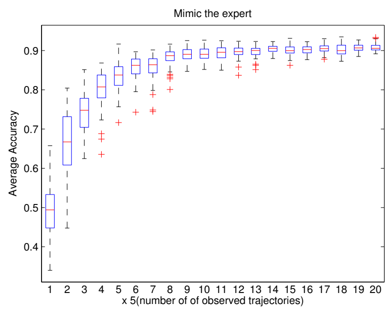
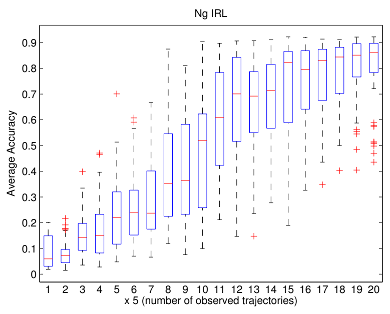


Figure 4.4: Noisy navigation: red decision trajectory for the expert and green decision trajectory for the apprentice. The arrow direction denotes the action taken by the agent.

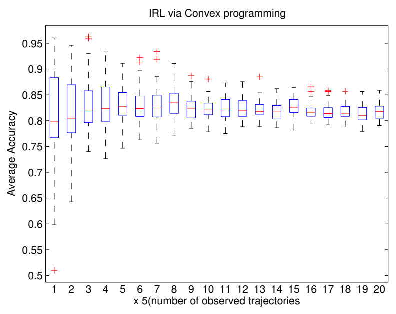
Results, shown in Figure4.5, suggest our algorithms can attain performance approaching that of the expert with fewer samples of decision trajectories. As the number of samples increases, apprenticeship learning using estimated reward becomes better at following the expert's behavior. The variance of performance accuracy decreases as the number of samples increases. The second advantage of our method is its robustness, which in the empirical tests is better than algorithms assuming linear approximation. Note that the estimated reward during apprenticeship learning is not



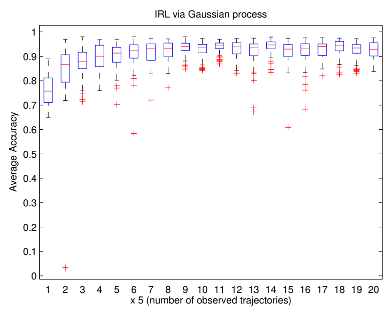
(a) Mimic the expert



(b) Linear approximation IRL



(c) IRL via convex programming



(d) IRL via Gaussian process

Figure 4.5: Plot of average accuracy vs. number of sampled trajectories from the expert. For each fixed number of sampled trajectory, we repeat 100 times of independent experiments and then take average over the accuracies. The value of x-axis times 5 is the number of sampled trajectories used in IRL.

able to let apprentice plan similarly as the expert, because AL is designed to find the best policy whose feature expectation is closest to the expert's, not to search the true underlying reward. So, in Figure4.5, we do not show the results of the experiments using the reward function estimated during apprenticeship learning. In Figure4.6, we plot the mean accuracy of 1000 testing problems in order to compare the differences among several IRL methods. These results show our algorithms' superiority to others, as the apprentice using the reward learned by our algorithms behaves more closely to the expert and exhibits more robust behavior.

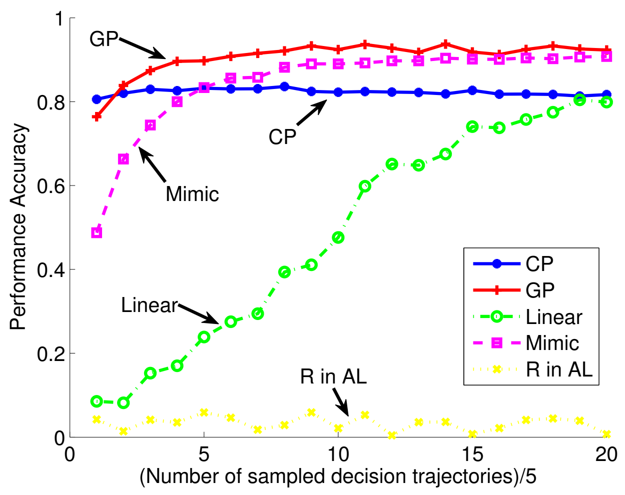


Figure 4.6: Comparison of average accuracy for testing problems.

4.5.2 Pendulum control problem

We analyze GPIRL in the continuous domains by applying it to a control problem known as the under-actuated pendulum swing up problem. This task was shown to be not trivial by Atkeson and Schaal in [53]. Doya pointed a challenge of this problem that the discretization can become prohibitively expensive [54]. Recently Deisenroth proposed a method called Gaussian process dynamic programming (GPDP) to counter the discretization challenge in [1]. Our algorithms are implemented using both the gpml[2] toolbox [37] and the GPDP code [1].

The dynamics in our experiment is determined by

$$\ddot{\phi}(t) = \frac{-\mu\dot{\phi}(t) + mgl\sin(\phi(t)) + u(t)}{ml^2},$$

where ϕ denotes the angle of pendulum and the vertical line, the coefficient of friction

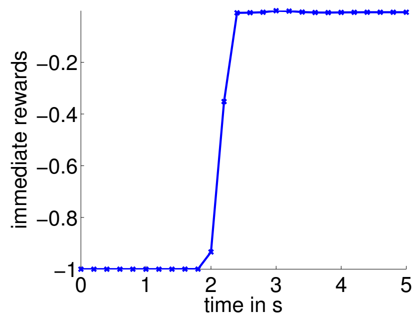
[2]http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html

$\mu = 0.05kgm^2/s$, the pendulum length $l = 1m$, the pendulum mass $m = 1kg$ and the gravitational constant $g = 9.81m/s^2$. The range of applied torque u is restricted to $[-5, 5]$, which is not enough to swing pendulum up directly. We define the state with two features, angle and angular velocity. The problem setup we used here is from the work in [1]. Starting from an initial state, the original control task is to learn the actions that swing the pendulum up and make it stay in the goal position at $[0, 0]^T$.

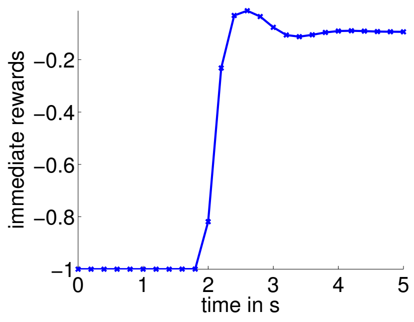
Our task is an inverse process that we learn the reward from samples of expert's decision trajectories. The expert is simulated by running RL with given reward functions. We learn the dynamics from the samples and randomly selected 200 states as the training data for IRL. To check the accuracy of estimation of reward, we compare the performance of expert and the apprentice using learned reward on a new test problem in 20 horizon length. If the estimation of reward can capture the structure of expert's reward, the apprentice should display similar behavior and reach close to the same objective, swinging the pendulum to the goal position and staying there. The experimental results are summarized in Figure4.7. The recovered Q function values and reward function values are shown in Figure4.8, providing the evidence that these random variables are well estimated by modeling with Gaussian process.

4.6 Conclusions

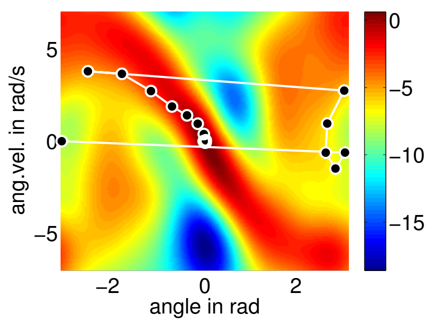
We have studied a Bayesian inference framework using Gaussian process for IRL problems that arise in decision making and convex optimization. The IRL problems can be solved efficiently by convex programming techniques, under the assumption that relationships between actions and reward can be modeled in a Bayesian framework.



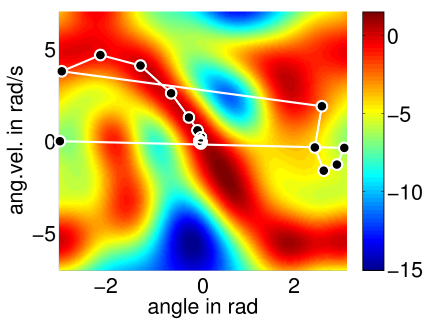
(a) Expert's reward



(b) Apprentice's reward



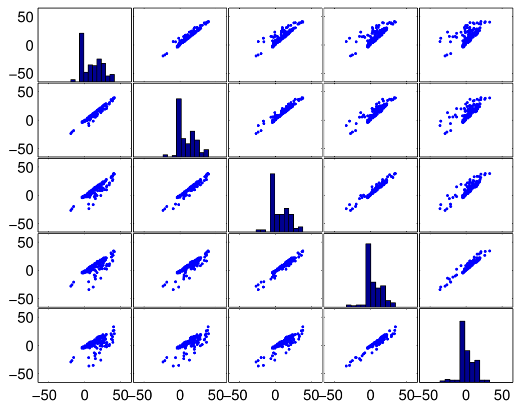
(c) Expert's trajectory



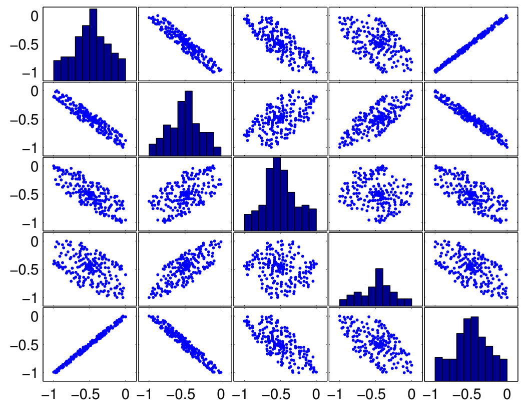
(d) Apprentice's trajectory

Figure 4.7: Plot of pendulum experimental results, where the first row contains the figures from [1] and the second row displays the results using the estimated reward by our *ALGPIRL* algorithm.

Furthermore, the use of preference graphs to present action allows for dealing with ill-posed IRL problems, which extends application of our algorithms to more practical areas. The experimental data shows that the apprentice using learned reward by our methods displayed more robust behavior and better performance with limited observations. Our Gaussian process method has other advantages over non-probabilistic kernel methods. Because our method has no assumption on function structure, we can use observations for model selection and hyper parameter optimization, and make probabilistic prediction on new coming data.



(a) Apprentice Q-factors



(b) Apprentice reward

Figure 4.8: Plot of Q-factors and reward values

The main contributions of this chapter are summarized as follows:

1. We develop a new probabilistic framework to solve IRL problems in convex programming domains.

2. We design preference graphs to represent the decision behavior, reducing the uncertainty and computation burden caused by the ill-posed nature. We build new likelihood functions for preference graphs and prove the effectiveness of

these formulations in experiments.

3. We model the reward function using Gaussian process, which offers the advantage that IRL using Gaussian process is relatively insensitive to the number of the observations and it performs better than other algorithms when the size of observation data set is small.

4. We use Gaussian process model to solve IRL problems in continuous domains where both dynamics and reward are unknown.

Future work in this area should focus on experiments on problems within continuous state domains, and should include consideration of methods for using Gaussian processes to model systems without knowledge of environmental dynamics.

Chapter 5

Semi-supervised Inverse Reinforcement Learning using MM

Most of existing IRL algorithms assume that there is a large number of decision trajectories being observed. If we consider the observation of action at state as the supervision given by the agent, we call IRL with a large number of observations as supervised IRL. However, we still have many practical problems in which it is not easy to obtain observations, and it is a non-trivial to apply IRL with limited observation. In this paper, we particularly study the problem of how to use IRL with limited observation, which is called semi-supervised IRL. Assume the environment information is known as a prior. Semi-supervised IRL(Semi-IRL) aims to utilize the known knowledge of state and action to reduce the uncertainty due to insufficient supervision. We show that an efficient semi-supervised IRL algorithm can be formulated by using the philosophy of majorization and minorization. A number of examples is considered to illustrate the quality and flexibility of the approach.

5.1 Introduction

The significance of IRL directly comes from the powerful application of DP/RL in varieties of research areas, such as optimal control [55, 56], animal learning [57], neurophysiology [58], behavioral neuroscience [59, 60] and economics [61, 62].

The difficulty of manually setting the reward function is a barrier to use RL in many real-world problems. Hence it is desirable to use IRL to learn the reward function from the demonstrating agent (teacher), and it follows that we can build an intelligent agent that imitates the teacher's behavior. Another potential advantage of IRL is that the reward function can be used to characterize the decision-making rule of an agent whose behavior has been sampled for demonstration. Though human may have bias in applying the decision-making rule, which results in noisy observations of behaviors, the abstract encoding of decision-making rule in the form of reward function is expected to be similar for the group of agents adopting the same decision-making rule. A study in how to recognize the decision-makers by using IRL is presented in [63].

Most of current IRL algorithms assume that the reward function can be well recovered in a MDP model that only considers the state and action in the decision trajectories observed. Although this assumption provides a good starting point for developing IRL algorithms with incomplete observations, the implication is that the agent is able to access a sufficient number of observations. But it is often difficult to get a number of observations under some practical conditions. For example, when we deal with the small-scale problems with incomplete observations, the number of observed states is too small to represent the true environment. Another widely used assumption in IRL algorithms is that the reward function is well approximated by a linear combination of a set of pre-defined basis functions, which is too strong in practice. In addition, a high computation capacity is required by some IRL algorithms in [2, 26] to simulate many random policies. Hence, to enhance the ability of learning with limited observations, reduce the computational burden and widen the formulation of reward function, we propose an new solution to IRL problem, which is based on probabilistic model and brings in the strategy of semi-supervised learning, utilizing not only the observation data but also the prior knowledge about state and action.

Semi-supervised learning, as the name suggests, extends either supervised learning or unsupervised learning to add additional information. It is attractive because it can potentially exploit both labeled and unlabeled data to achieve better performance than supervised learning [64], provided that there is a good match between the problem structure and the model assumption. The similar idea exists in IRL. Given a finite MDP model, the state and action space is always known as a prior. Observation consists of state and action taken upon the state by teacher. If we have an observation of action at a state, we say this state is observed; otherwise, the state is said unobserved. Semi-IRL can perform better with incomplete observation, which is based on the assumption that the unobserved states contain additional information that can be used to facilitate the learning process.

We propose to solve IRL using information from observation of teacher and the information on environment that is not included in observation. An important optimization method we used in formulating the Semi-IRL algorithm is called Minorization-Maximization (MM). The MM principle appears in lots of study and areas, such as robust regression [65], survival analysis [66], wavelet-based image restoration [67] and dictionary learning [68]. One of virtue of a successful MM is to substitute a simple optimization problem for a difficult one.

The first step of Semi-supervised IRL is to formulate the learning problem in the form of maximum likelihood estimation (MLE). The IRL problem with complete observation can be solved efficiently by using the quadratic convex programming [51]. Consider both observed and unobserved states, the likelihood depends on both observed state-action pairs and the unobserved - this is how the unobserved data might help according to the idea of semi-supervised learning. However, we are not able to solve the MLE analytically, if the likelihood function includes unobserved data. We propose a method under conceptually simple Expectation-Maximization (EM) framework to learn the reward function if the observation data is incomplete. It is proved that the optimization problem for IRL with incomplete observation is convex programming. In the E-step, we calculate the joint probability distribution of the observation data, in which the action at unobserved state is considered as the missing data. In the M-step, we maximize with respect to the reward function the likelihood function given by the previous E-step. But in the E-step the joint likelihood function is too complicated to be calculated efficiently, therefore we replace the original likelihood function with its minorize function which turns the calculation more smooth and tractable. In the optimization theory, EM is an special case of the class of MM algorithms [69]. That is the likelihood function in the E-step is just the process to construct a minorize function which has the promise of being easily solvable in the maximization step.

5.2 Preliminaries

5.2.1 Forward planning with MDP

To model the decision-making process for an agent, we consider an optimistic condition that the expert is seeking the optimal deterministic policy π and follows it. Following π means that at time t the expert selects an action a_t using $a_t = \pi(s_t)$. Therefore, the *value function* for a policy π evaluated at any initial state $s_{t=0}$, which is calculated by $V^{\pi}(s_0) = E[\sum_{t=0}^{\infty} \gamma^t r(s_t)|\pi]$, is always maximized at every stage, if π is optimal.

And we have the constraints for the optimality assumption as follows,

$$Q^{\pi}(s, \pi(s)) \geq Q(s, a) + \epsilon_s, \tag{5.1}$$

where ϵ_s is a parameter controlling the confidence of optimality. If $\epsilon_s \geq 0$, $\pi(s)$ is an *optimal action* at state s; If $\epsilon_s < 0$ and $|\epsilon_s|$ is smaller than a given positive number, $\pi(s)$ is a *suboptimal action* at state s.

An IRL problem is to learn a reward function for an MDP model, based on the observation model that is written as $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$. The variable $\mathcal{O}$ contains the measurements of teacher's behavior, which is action and the measurements of sensory inputs to the teacher, which is state. The observation is called incomplete[1], if the state-action pairs in $\mathcal{O}$ only cover part of the state space $\mathcal{S}$. Otherwise, the observation is called complete.

[1]The observation $\mathcal{O}$ can rewritten as a mapping between a subset $\hat{\mathcal{S}} \subset \mathcal{S}$ and $\hat{\mathcal{A}} \subset \mathcal{A}$, which is $\hat{\mathcal{S}} \to \hat{\mathcal{A}}$. Let $\check{\mathcal{S}} = \mathcal{S} - \hat{\mathcal{S}}$ denote the subset of states that we do not observe. Then we represent $\hat{\mathcal{S}}$ as $\hat{\mathcal{S}} = \{\hat{s}_l\}_{l=1}^{L} \subset \mathcal{S}$, and $\check{\mathcal{S}}$ as $\check{\mathcal{S}} = \{\check{s}_k\}_{k=1}^{K} \subset \mathcal{S}$, where $L = |\hat{\mathcal{S}}|$, $K = |\check{\mathcal{S}}|$ and $L + K = N$. We know $\hat{\mathcal{S}} \cup \check{\mathcal{S}} = \mathcal{S}$ and $\hat{\mathcal{S}} \cap \check{\mathcal{S}} = \emptyset$.

Let $\mathcal{Z} = \{\mathcal{O}, \mathcal{Y}\}$ denote the complete observation, where $\mathcal{O} = \{O_l : (\hat{s}_l, a_l)\}_{l=1}^{L}$ and $\mathcal{Y} = \{y_k : (\check{s}_k, a_{y_k})\}_{k=1}^{K}$. At K states in $\check{\mathcal{S}}$, an instantiation of the unknown actions at these states is defined as a vector $\mathbf{y} = (y_1, y_2, \ldots, y_K)$, where $y_k \in \mathcal{M},\ k \in \{1, 2, \ldots, K\}$ indicates that the action a_{y_k} is selected at k-th state in $\check{\mathcal{S}}$. A complete policy is written as $z = (\mathcal{O}, \mathbf{y})$.

The problem we aim to solve in this paper has these properties: 1) The observation only contains a subset of states; 2) The computing resources are too limited to run a large simulation.

5.2.2 Review of IRL as a supervised learning problem

Consider a learning machine has a set of functions $f(x, \alpha)$, where α is the parameter variable. The supervised learning problem is to learn a realization of α that makes $f(x, \alpha)$ best fit the supervisor's response. The selection of desired function is based on the training data set of independent and identically distributed observations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The learning problem can be formulated as an optimization problem to minimize the risk function $R(\alpha) = \int l((x, y), \alpha) dF((x, y))$, where $l(x, y)$ is a loss function. A commonly used method is to minimize the empirical risk function $R_{emp}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} l((x_i, y_i), \alpha)$.

Most of existing IRL algorithms can be viewed as a supervised learning problem, in which the observed agent is considered as a teacher providing the demonstrations as the supervisions. With the teacher's supervision, the IRL algorithms search for an optimal reward function. The forward planning using this reward function can find a policy that minimizes the cost function or is consistent with the teacher's supervision.

Ng's algorithm assumes that the policy of teacher is optimal so that the reward function should guarantee the optimality of the observed policy π_E:

$$Q^{\pi_E}(s, \pi_E(s)) \geq \max_{a \in \mathcal{A} \setminus \pi_E(s)} Q^{\pi_E}(s, a) \forall s \in \mathcal{S}$$

which gives the linear programming IRL (LPIRL) shown in Algorithm 1. Ng and Russel choose the reward function to maximize the difference between the optimal and suboptimal policies while favoring the sparseness in the reward function. This algorithm can viewed as a supervised learning problem.

To formulate IRL as a binary classification model, we let x denote the state-action pair (s, a) and y be a binary value in the set $(-1, 1)$. A positive y means that the teacher takes action a at state s. Otherwise y is assigned -1. The observed pairs in $\mathcal{O}$ are used as positive instances and the combinations of (s, a) not shown in $\mathcal{O}$ are negative instances. Let the reward function be parameters for the scoring function. The Q function becomes the score function $f(x, \alpha)$ for an instance (s, a). Based on the Bellman equation, the loss function can be defined as follows.

$$l(x, y) = \begin{cases} Q(s, a) - Q(s, a^*), \text{if y=-1;} \\ 0, \text{if y=1.} \end{cases} \tag{5.2}$$

In above equation $(s, a^*) \in \mathcal{O}$. Then, the objective function in Algorithm 1 is the empirical risk with L_1 regularization, and the constraints enforce the teacher's supervision.

5.2.3 Review of MM conception

MM is an iterative method that relies heavily on convexity arguments and particularly useful in high-dimensional problems. Here we review the philosophy of MM algorithm described in [69].

Use x as a variable, a function $g(x|x_t)$ is said to majorize a function $f(x)$ at x_t provided

$$\begin{cases} f(x_t) = g(x_t|x_t), \; x = x_t \\ f(x) \leq g(x|x_t), \; x \neq x_t, \end{cases}$$

where x_t represents the current iterate estimate of x in a search of the value of $f(x)$. The function $g(x|x_t)$ sets a upper boundary of the function $f(x)$ and its surface is tangent to that of $f(x)$ at the point $x = x_t$. Then, a minimization problem using MM algorithm is formulated by minimizing the surrogate majorizing function $g(x|x_t)$ rather than the original function $f(x)$. The point $x_{t+1} = arg\min \; g(x|x_t)$ also forces $f(x)$ downhill, which is $f(x_{t+1}) \leq f(x_t)$. This property makes MM algorithm remarkable numerical stability.

To maximize a function $f(x)$, we minorize it by a surrogate function $g(x|x_t)$ and maximize it to produce the next iterate x_{t+1}.

5.3 Semi-supervised IRL using MM

A semi-supervised IRL method learns from not only the set of observations $\mathcal{O}$ but also the set of states not seen in $\mathcal{O}$. The hypothesis underlying our approach is that the unobserved states can help us reduce the uncertainty in estimation and the formulation of problem with complete data might be more computationally tractable.

5.3.1 A Maximum likelihood estimation model

Consider a probabilistic model which we use to denote the joint distribution of observed variables $\mathcal{O}$ and hidden variables $\mathbf{y}$. teacher's behavior can be modeled by two approaches: a soft policy which is

$$\pi(s) \sim p(a|s),$$

and a deterministic policy that is

$$\pi(s) = arg \max_{a \in \mathcal{A}} p(a|s),$$

where the conditional probability

$$p(a = a_m|s) = \frac{e^{\beta Q(s,a_m)}}{\sum_{j=1}^{M} e^{\beta Q(s,a_j)}}. \tag{5.3}$$

Our goal is to maximize the likelihood function that is given by

$$p(\mathcal{O}|\mathbf{r}) = \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathcal{O}, \mathbf{y}|\mathbf{r}). \tag{5.4}$$

It is difficult to optimize $p(\mathcal{O}|\mathbf{r})$ directly, since we need complete information on policy-related probability matrix P_π to calculate the Q-function based on Bellman equation. We introduce a distribution $q(\mathbf{y})$ over the hidden variables. After taking logarithm on Eq.5.4, we have the following equation similar as that in [70].

$$\ln p(\mathcal{O}|\mathbf{r}) = \mathcal{L}(\mathbf{r}|\mathbf{r}^t) + K\mathcal{L}(q||p), \tag{5.5}$$

and

$$\mathcal{L}(\mathbf{r}|\mathbf{r}^t) = \sum_{\mathbf{y}} q(\mathbf{y}) \ln[\frac{p(\mathcal{O}, \mathbf{y}|\mathbf{r}^t)}{q(\mathbf{y})}],$$
$$\mathrm{KL}(q||p) = -\sum_{\mathbf{y}} q(\mathbf{y}) \ln[\frac{p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t)}{q(\mathbf{y})}].$$

Note that $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is a function of $\mathbf{y}$ and $\mathbf{r}$, and $\mathrm{KL}(q||p)$ is the Kullback-Leibler (KL) divergence between $q(\mathbf{y})$ and $p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t)$. Since $\ln p(\mathcal{O}|\mathbf{r})$ is difficult to calculate, generalized EM method provides a method to maximize the function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ iteratively,

Algorithm 7 EM IRL algorithm

```
1: Initialize $\mathbf{r}^0$.
2: Expectation: $\mathcal{L}(\mathbf{r}|\mathbf{r}^t) = \sum_{\mathbf{y}} p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t) \ln p(\mathcal{O}, \mathbf{y}|\mathbf{r})$.
3: Maximization: we have $\mathbf{r}^{t+1} = arg\max_{\mathbf{r}} \mathcal{L}(\mathbf{r}|\mathbf{r}^t)$.
4: Given $\delta$, if $||\mathbf{r}^{t+1} - \mathbf{r}^t||_2 \geq \delta$, go to step 2; Otherwise stop and output $\mathbf{r}^{t+1}$.
```

rather than directly dealing with $\ln p(\mathcal{O}|\mathbf{r})$, which is summarized in Algorithm 7. The complete proof of generalized EM algorithm in this formulation can be found in [70]. For the convenience of the readers, we will briefly review the proof of generalized EM as a particular case of MM method.

We have the following propositions.

Proposition 14 *Consider function* $\ln p(\mathcal{O}|\mathbf{r})$ *in MDP, we minorize it by function* $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$*, which is*

$$\begin{cases} \ln p(\mathcal{O}|\mathbf{r}^t) = \mathcal{L}(\mathbf{r}|\mathbf{r}^t) \\ \ln p(\mathcal{O}|\mathbf{r}^t) \geq \mathcal{L}(\mathbf{r}|\mathbf{r}^t), \;\; \mathbf{r} \neq \mathbf{r}_t \end{cases}$$

Proof By Eq. 5.5, we know $\ln p(\mathcal{O}|\mathbf{r}^t) \geq \mathcal{L}(\mathbf{r}|\mathbf{r}^t)$, because the KL divergence has the property that $\mathrm{KL}(q||p) \geq 0$, where if and only if $q(\mathbf{y}) = p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t)$, $\mathrm{KL}(q||p) = 0$.

In the E-step, the lower bound function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is maximized with respect to $q(\mathbf{y})$ by making KL divergence vanish. Then we have $\ln p(\mathcal{O}|\mathbf{r}^t) = \mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. The solution to this maximization is to set $q(\mathbf{y}) = p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t)$. Therefore, $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is a surrogate function that miniorizes $\ln p(\mathcal{O}|\mathbf{r}^t)$. Since $q(\mathbf{y})$ does not contain the parameter $\mathbf{r}$ we are optimizing, we drop it while writing the surrogate function in the E-step. Based on the philosophy of the MM algorithm introduced in Section 5.2.3, we say that EM IRL algorithm is a version of MM algorithm.

5.3.2 MM optimization

Using the Boltzman distribution to model the joint distribution of state and action shown in Eq. 5.3, we write the lower bound function as $\mathcal{L}(\mathbf{r}|\mathbf{r}^t) = \mathbf{w}^T\mathbf{r} - \kappa(\mathbf{r})$, where

$$\begin{aligned}\mathbf{w}^T &= \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}...\sum_{y_K=1}^{M}[\beta\mathbf{e}_1^T + (\sum_{l=1}^{L}\mathbf{P}_l(s_l,:) \\ &+ \sum_{k=1}^{K}\mathbf{P}_{y_k}(s_k,:))\mathbf{H}(\mathbf{y})]\prod_{k=1}^{K}p(y_k|\mathcal{O},\mathbf{r}^t),\end{aligned}$$

$$\kappa(\mathbf{r}) = \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}...\sum_{y_K=1}^{M}\beta\mathbf{e}_1^T\mathbf{r} + \log\prod_{i=1}^{N}\sum_{j=1}^{M}e^{\mathbf{P}_j(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}$$

and $\mathbf{H}(\mathbf{y}) = \beta\gamma(I - \gamma\mathbf{P}_{\pi(s)})^{-1}$. The details to compute function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ are shown in Appendix 8.2. We also have shown a proposition which is given below.

Proposition 15 *Function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is a concave function.*

Proof See Appendix 8.3.

To obtain an upperbound of $\mathbf{r}^{t+1}$ such that $\mathcal{L}(\mathbf{r}^{t+1}|\mathbf{r}^t) \geq \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)$, we may use Newton-Raphson algorithm as

$$\mathbf{r}^{t+1} = \mathbf{r}^t - (\nabla^2\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t))^{-1}\nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t) \tag{5.6}$$

But it is difficult to calculate the Hessian matrix $\nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. Here, we present an approximation technique that offers a minorize surrogate function not overshooting or lowerbounding the objective function.

In the MM algorithm, we attack the objective function using Taylor expansion of $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. This yields the minorization function $g(\mathbf{r}|\mathbf{r}^t)$. It is shown that $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is

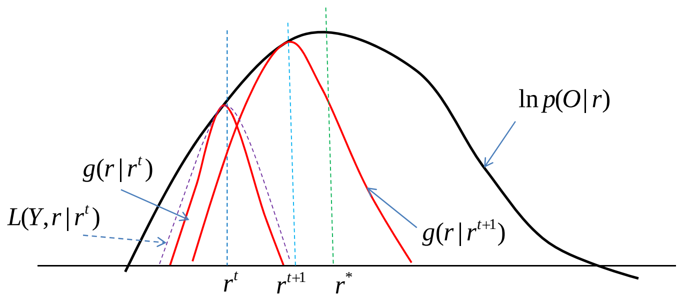


Figure 5.1: An illustration of minorization function

concave and twice differentiable. Therefore, given a negative definite matrix $\mathbf{B}$ such that $\mathbf{B} \leq \nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$,[2] we have

$$g(\mathbf{r}|\mathbf{r}^t) = \mathcal{L}(\mathbf{r}^t) + \nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}-\mathbf{r}^t) + \frac{1}{2}(\mathbf{r}-\mathbf{r}^t)^T\mathbf{B}(\mathbf{r}-\mathbf{r}^t), \tag{5.7}$$

which satisfies

$$\begin{cases} g(\mathbf{r}^t|\mathbf{r}^t) = \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t) = \ln p(\mathcal{O}|\mathbf{r}^t) \\ g(\mathbf{r}|\mathbf{r}^t) \leq \mathcal{L}(\mathbf{r}|\mathbf{r}^t) \leq \ln p(\mathcal{O}|\mathbf{r}^t). \end{cases} \tag{5.8}$$

The selection of matrix $\mathbf{B}$ can not be overshooting or lowerbounding the maximum of the likelihood along its current direction of search in the maximization step. Figure5.1 illustrates a good design of the minorization function for maximum of the likelihood.

We propose a choice of $\mathbf{B}$ in the following theory.

Theorem 16 *To satisfy* $\mathbf{B} \leq \nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$*, we define matrix* B *by calculating*

$$\mathbf{B} \;=\; -\sum_{i=1}^{N}\mathbf{H}(\mathbf{y}^*)^T\frac{\mathbf{\Pi}(\mathbf{y}^*)}{(\sum_{a=1}^{M}e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\boldsymbol{y}^*)\hat{\boldsymbol{r}}})^2}\mathbf{H}(\mathbf{y}^*).$$

Proof The proof and the process of computing hidden action $\mathbf{y}^*$ and the reward $\hat{\mathbf{r}}$ are shown in Appendix 8.4.

[2]It means that $\forall x \in R^N$, we have $x^TBx - x^T\nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t)x \leq 0$

This theory completes the miniorization step of the algorithm. The maximization step can be accomplished by introducing the regularization and finding the optimal point of the regularized likelihood function. The following theorem states that in the Bayesian inference setting, the prior distribution plays a role of regularization function in the maximum likelihood estimation.

Theorem 17 *Suppose we have a Laplace prior distribution* $f(\mathbf{r}|0,1)$ *on* $\mathbf{r}$. *The MAP estimator is the solution to*

$$\begin{aligned} &\max_{\mathbf{r}} g(\mathbf{r}|\mathbf{r}^t) + \lambda|\mathbf{r}| \\ &\qquad s.t.\ \ \mathbf{r} \in \Omega \end{aligned} \tag{5.9}$$

Proof The MAP estimation is formulated as follows

$$p(\mathbf{r}|\mathcal{O}) = \frac{p(\mathbf{r}|\mathcal{O})p(\mathbf{r})}{p(\mathcal{O})}$$

After taking logarithm and removing the element not containing the variable $\mathbf{r}$, we get

$$\ln p(\mathbf{r}|\mathcal{O}) \propto \ln p(\mathbf{r}|\mathcal{O}) + \ln p(\mathbf{r}).$$

This leads to the problem formulation that is given by

$$\max_{\mathbf{r}} \sum_{i=1}^{N} \log \frac{e^{\beta Q(s_i,\pi(s_i))}}{\sum_{j=1}^{M} e^{\beta Q(s_i,a_j)}} + \lambda|\mathbf{r}|. \tag{5.10}$$

Now compare this problem to the original maximum likelihood estimation, we can solve it using our proposed algorithm to formulate the surrogate function $g(\mathbf{r}|\mathbf{r}^t)$ for likelihood function. Finally, in the maximiation step, note that the problem is formulated as Eq. 5.9.

The regularization function we can choose from the following options:

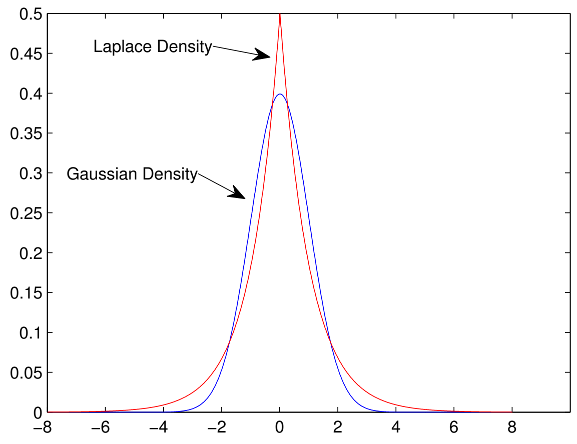


Figure 5.2: Graphs of prior distribution

1. Laplace distribution: $f(\mathbf{x}) = \frac{1}{2}e^{-|\mathbf{x}|}, \forall \mathbf{x} \in \Re^k$, which provides a l_1 regularization;
2. Gaussian distribution: $f(\mathbf{x}) = (2\pi)^{-\frac{k}{2}}e^{-\frac{1}{2}\mathbf{x}^T\mathbf{x}}$, $\forall \mathbf{x} \in \Re^k$, which provides a l_2 regularization.

See the Figure5.2 comparing the different kinds of regularization method. If we choose the Gaussian prior, the l_2 regularization term is selected.

5.3.3 Convergence analysis

Existing study in EM convergence theory [71] and MM convergence theory [69] can not be applied directly in our method, since the differentiability required of the update map does not hold.

The successive iteration in our method is denoted as $\mathbf{r}^t$. To prove the monotonicity of the maximum likelihood, we have the following lemma.

Lemma 18 *The likelihood function for incomplete observation is not decreased in MM IRL method, which is $\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) \geq \tilde{\mathcal{L}}(\mathbf{r}^t)$, with equality iff $\mathbf{r}^{t+1} = \mathbf{r}^t$.*

Algorithm 8 MM IRL - Main process

```
1: Input $M_I = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$.
2: Initialize $\mathbf{r}^0$ and t=0.
3: for $c = 1$ to $MaxIter$ do
4:     Minorization: Given $\mathbf{r}^t$, after calling Alg. 9, we have function $g(\mathbf{r}|\mathbf{r}^t)$.
5:     Maximization: Maximize the function $g(\mathbf{r}|\mathbf{r}^t)$ by calling Alg. 10.
6:     if $||\mathbf{r}^{t+1} - \mathbf{r}^t|| \leq \delta$ then
7:         Return $\mathbf{r}^{t+1}$.
8:     end if
9: end for
10: Return $\mathbf{r}^{t+1}$.
```

Algorithm 9 MM IRL - Part 1 Minorization

```
1: Given $\mathbf{r}^t$, we have $a_{\mathbf{y}^*} = \arg\max_a Q(s,a)$, and an estimate of $\hat{\mathbf{r}}$ obtained by solving
   the following problem
```

$$\max_{\mathbf{r}} \mathbf{e}^T \mathbf{H}(\mathbf{y}^*) \mathbf{r}$$
$$s.t.\ \mathbf{r} \in \Omega_r \tag{5.11}$$

```
2: To minorize $\ln p(\mathcal{O}|\mathbf{r})$, we have its surrogate function $g(\mathbf{r}|\mathbf{r}^t)$.
```

Algorithm 10 MM IRL - Part 2 Maximization

```
1: Maximization: Setting the feasible set $\Omega_r$ with these constraints:
     1. $r \geq 0, \text{and } \|r\| \leq R_{max}$;
     2. $Q(s_i, a_i) - \max_{a \in \mathcal{A} \backslash a_i} Q(s_i, a) \geq \epsilon_i, \epsilon_{min} \leq \epsilon \leq 0, \forall (s_i, a_i) \in \mathcal{O} \cup \{\mathcal{S}, \mathbf{y}^*\}$
   , we obtain $\mathbf{r}^{t+1} = \arg\max_{\mathbf{r}} \nabla \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)\mathbf{r} + \frac{1}{2}(\mathbf{r} - \mathbf{r}^t)^T \mathbf{B}(\mathbf{r} - \mathbf{r}^t) + \lambda|\mathbf{r}|$, s.t. $\mathbf{r} \in \Omega_r$.
   The parameters are updated by computing the following equations.
```

$$\mathbf{B} = -\sum_{i=1}^{N} \mathbf{H}(\mathbf{y}^*)^T \frac{\mathbf{\Pi}(\mathbf{y}^*)}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}})^2} \mathbf{H}(\mathbf{y}^*) \tag{5.12}$$

$$\hat{\mathbf{\Pi}}_{uu} = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} e^{(b^i + b^j)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}} (b_u^i - b_u^j)^2 \tag{5.13}$$

$$\hat{\mathbf{\Pi}}_{uv} = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} e^{(b^i + b^j)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}} |(b_u^i - b_u^j)(b_v^i - b_v^j)| \tag{5.14}$$

$$\nabla \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t) = \sum_{i=1}^{N} \mathbf{P}_{\pi(s_i)}(s_i,:)\mathbf{H}(\mathbf{y}^*) - \frac{\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\mathbf{r}^t}}{\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\mathbf{r}^t}}$$

In above equations, we define $\mathbf{b}^i := \mathbf{P}_i(s,:) = (b_1^i, b_2^i, \cdots, b_N^i),\ i \in \mathcal{M}$. The notation $\mathbf{\Pi}_{uu}$ is u-th diagonal entry for the Hessian matrix $\mathbf{\Pi}$ and $\mathbf{\Pi}_{uv}$ denotes the entry of (u, v).

Proof Standard EM theory [72] gives the decomposition

$$\tilde{\mathcal{L}}(\mathbf{r}') = \log p(\mathbf{y}|\mathbf{r}) = \mathcal{L}(\mathbf{r}'|\mathbf{r}) - \mathcal{H}(\mathbf{r}'|\mathbf{r})$$

Set $\mathbf{r} = \mathbf{r}^t$. It follows that $\mathcal{H}(\mathbf{r}^{t+1}|\mathbf{r}^t) \leq \mathcal{H}(\mathbf{r}^t|\mathbf{r}^t)$ and $\mathcal{L}(\mathbf{r}^{t+1}|\mathbf{r}^t) \geq \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)$ (this feature is not obvious due to the majorization step and is proved in Appendix 8.6). Thus

$$\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) + \mathcal{H}(\mathbf{r}^{t+1}|\mathbf{r}^t) \geq \tilde{\mathcal{L}}(\mathbf{r}^t) + \mathcal{H}(\mathbf{r}^t|\mathbf{r}^t).$$

Therefore

$$\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) \geq \tilde{\mathcal{L}}(\mathbf{r}^t) - \mathcal{H}(\mathbf{r}^{t+1}|\mathbf{r}^t) + \mathcal{H}(\mathbf{r}^t|\mathbf{r}^t) \geq \tilde{\mathcal{L}}(\mathbf{r}^t|\mathbf{r}^t)$$

with equality iff $\mathbf{r}^{t+1} = \mathbf{r}^t$.

Lemma 19 $\{\tilde{\mathcal{L}}(\mathbf{r}^t)\}_{t\geq 0}$ *is bounded and has a finite limit.*

Proof By $\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) \geq \tilde{\mathcal{L}}(\mathbf{r}^t)$, we know $\tilde{\mathcal{L}}(\mathbf{r}^t) \geq \tilde{\mathcal{L}}(\mathbf{r}^0)$. This establishes lower boundedness.By Bellman Equation, we have

$$(\mathbf{I} - \gamma \mathbf{P}_{\pi(s)})^{-1}\mathbf{V} = \mathbf{r}$$

Thus, the general solution to above equation is $\mathbf{V} = \mathbf{A}^\dagger \mathbf{r} + (\mathbf{I} - \mathbf{A}^\dagger \mathbf{A})\mathbf{w}$, where $\mathbf{A} = \mathbf{I} - \gamma \mathbf{P}_{\pi(s)}$ and $\mathbf{w}$ is an arbitrary vector. According to Bellman equation, we have

$$\mathbf{V}^*(s,r) = \max_a E[r(x,a) + \gamma \sum_{s'} \mathbf{P}_a(s,s')\mathbf{V}^*(s',r)],$$
$$\mathbf{V}_*(s,r) = \min_a E[r(x,a) + \gamma \sum_{s'} \mathbf{P}_a(s,s')\mathbf{V}^*(s',r)].$$

And we get $\mathbf{P}_a(s,:)\mathbf{H}(\mathbf{y})\mathbf{r} \leq \sum_{i=1}^N |\mathbf{V}^*(s_i)|$ and $\mathbf{P}_a(s,:)\mathbf{H}(\mathbf{y})\mathbf{r} \geq \sum_{i=1}^N \mathbf{V}_*(s_i)$. These properties lead to $\tilde{\mathcal{L}} \leq N\sum_{i=1}^N [|\mathbf{V}^*(s_i)| - \mathbf{V}_*(s_i)] - N\log M$. As $\tilde{\mathcal{L}}(\mathbf{r}^t)$ is nondecreasing and bounded, it must have a finite limit.

From Lemma 19, $\tilde{\mathcal{L}}(\mathbf{r}^t)$ converges monotonically to some $\tilde{\mathcal{L}}^*$. Depending on global convergence theorem (See [71, 73]), which is described in Appendix 8.7, and Wu's theorem on EM that if the complete likelihood function $\mathcal{L}(\mathbf{r}'|\mathbf{r})$ is continuous in both $\mathbf{r}'$ and $\mathbf{r}$, M is a closed point-to-set map over the complement of Γ, we have a theorem in the following

Theorem 20 *The limit points of sequence $\{\mathbf{r}^t\}$, optimized with l_1 regularization, are local maxima points of $\tilde{\mathcal{L}}(\mathbf{r})$ and $\tilde{\mathcal{L}}(\mathbf{r}^t)$ converges monotonically to $\tilde{\mathcal{L}}^* = \tilde{\mathcal{L}}(\mathbf{r}^t)$ for some points $\mathbf{r}^*$.*

Proof The condition 1) and 3) in Global convergence theorem follow from Lemma 18 and 19. Then Following from Wu'theorem, the convergence theorem becomes as a special case that has a simplified condition: $\mathcal{L}(\mathbf{r}^{t+1}) > \mathcal{L}(\mathbf{r}^t), \forall \mathbf{r}_t \notin \Gamma$. Then, all the limit points of $\{\mathbf{r}^t\}$ are local maxima of $\tilde{\mathcal{L}}$, and $\tilde{\mathcal{L}}(\mathbf{r}^t)$ converges to $\tilde{\mathcal{L}}^*$. Since it is not straightforward for this property, we show the proof in the following.

Let Γ be set of local maxima in the interior of Ω. Consider a $\mathbf{r}^t \notin \Gamma$, which is in the interior of Ω. From the proof for Lemma 18, we have $\mathbf{H}(\mathbf{r}^t|\mathbf{r}^t) \geq \mathbf{H}(\mathbf{r}'|\mathbf{r}^t)$, $\forall \mathbf{r}' \in \Omega$. As $\mathbf{r}^t \notin \Gamma$, it is not a local maximum of $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. Then, $\nabla \mathcal{L}(\mathbf{r}|\mathbf{r}^t) \neq 0$, and $\nabla g(\mathbf{r}|\mathbf{r}^t) \neq 0$ since $g(\mathbf{r}^t|\mathbf{r}^t) = \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)$. It follows that $g(\mathbf{r}^{t+1}|\mathbf{r}^t) > g(\mathbf{r}^t|\mathbf{r}^t)$. Together with $\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) \geq g(\mathbf{r}^{t+1}|\mathbf{r}^t) - \mathbf{H}(\mathbf{r}^{t+1}|\mathbf{r}^t)$ and $\tilde{\mathcal{L}}(\mathbf{r}^t) = g(\mathbf{r}^t|\mathbf{r}^t) - \mathbf{H}(\mathbf{r}^t|\mathbf{r}^t)$, this proves $\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) > \tilde{\mathcal{L}}(\mathbf{r}^t), \forall \mathbf{r}_t \notin \Gamma$. In addition, though l_1 regularization is not differentiable at $\mathbf{r} = \mathbf{0}$, the regularization penalizes the movement to this point (see discussion in [2]). For convex functions, the stationary points or local maxima points are global maxima.

Theorem 21 *The sequence $\{\mathbf{r}^t\}$ converges to some stationary point $\mathbf{r}^*$, the limit of $\tilde{\mathcal{L}}(\mathbf{r})$.*

Proof Since the surrogate function $g(\mathbf{r}|\mathbf{r}^t)$ is convex, the limit point of $\mathbf{r}$ is the maximum point of $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. If $\mathbf{w}$ is a maximum point of $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$, it is known that the matrices $\mathbf{C} = \nabla\mathcal{L}(\mathbf{w}|\mathbf{r}^t)$ and $\mathbf{D} = \nabla g(\mathbf{w}|\mathbf{w})$ are negative definite. So $\mathbf{C} - \mathbf{D}$ is positive semidefinite. In [69], it is shown that the iteration map $h(\mathbf{r})$ has differential $\mathbf{I} - \mathbf{D}^{-1}\mathbf{C}$ at $\mathbf{w}$, and the norm $\|\mathbf{I} - \mathbf{D}^{-1}\mathbf{C}\|_T \leq 1$. Then, Ostrowski's result applies. Hence, iterates are locally attracted to $\mathbf{w}$.

In maximization step, we have

$$\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) - \tilde{\mathcal{L}}(\mathbf{r}^t) \geq g(\mathbf{r}^{t+1}|\mathbf{r}^t) - g(\mathbf{r}^t|\mathbf{r}^t)$$
$$= \nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) + \frac{1}{2}(\mathbf{r}^{t+1} - \mathbf{r}^t)^T\mathbf{B}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t)$$

By projection theorem to solve maximization problem, $\mathbf{r}^{t+1} = \mathbf{r}^t + \beta_t\mathbf{A}\nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)$, where $0 < \beta_t < 1$. Then we have

$$(\mathbf{r}^t + \beta_t\mathbf{A}\nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t) - \mathbf{r}^{t+1})(\mathbf{r} - \mathbf{r}^{t+1}) \leq 0 \tag{5.15}$$

Applying with $\mathbf{r} = \mathbf{r}^t$, we get

$$\nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) \geq \frac{1}{\beta_t}\mathbf{A}^{-1}\left\|\mathbf{r}^{t+1} - \mathbf{r}^t\right\| \tag{5.16}$$

Let $\lambda_a = \sup_{\mathbf{r}}\{\text{max eigenvalue of } \mathbf{A}(\mathbf{r})\}$ and λ_b be the minimum eigenvalue of matrix $\mathbf{B}$, then

$$\tilde{\mathcal{L}}(\mathbf{r}^{t+1}) - \tilde{\mathcal{L}}(\mathbf{r}^t) \geq (\frac{1}{\beta_t\lambda_a} + \frac{\lambda_b}{2})\left\|\mathbf{r}^{t+1} - \mathbf{r}^t\right\|$$

which means $\lim_{t\to\infty}\|\mathbf{r}^{t+1} - \mathbf{r}^t\| = 0$. Therefore, $\mathbf{r}^t$ converges to the stationary point r^* with $\tilde{\mathcal{L}}(\mathbf{r}^*) = \tilde{\mathcal{L}}^*$.

If the regularization function is l_2, the objective function is continuous and differentiable. It is easily shown $\{\tilde{\mathcal{L}}(\mathbf{r}^t)\}_{t\geq 0}$ is bounded and the theorems in [71, 69] follow successfully. In general, if the sum of surrogate function and regularization function underlying nonlinear shrinkage function employed in the M-step is continuous and differentiable in $\mathbf{r}$, then our method converges to a stationary point of the regularized log-likelihood. The limit points may be local maxima or saddle-points.

The following points summarize the convergence properties of MMIRL algorithm.

1. It is difficult to write the formulation for the incomplete likelihood function due to uncertainty in optimal policy. One widely used method is to run a lot of simulations to obtain the estimation of value functions. Without running simulations, we can maximize the incomplete likelihood function by using EM method to iteratively maximize the complete likelihood function. EM is a special case of MM algorithm. To further simplify the problem, we turn to find another surrogate function which is easier to calculate, while we prove it is still effective under MM framework. Each iteration of MM algorithm produces a reward function with a penalized likelihood value greater than or equal to the previous estimation of the reward function. As $\{\tilde{\mathcal{L}}(\mathbf{r}^t)\}$ is bounded, the limit points of any instance of $\{\mathbf{r}^t\}$ converges monotonically to $\tilde{\mathcal{L}}(\mathbf{r}^*)$, for some stationary points $\mathbf{r}^*$.

2. Consider the regularization function in Bayesian inference framework. If the surrogate function and convex function are convex, then the sequence of complete log likelihood values converges to the global maximum. However, since there may be many points, the reward function is not fixed. If the incomplete likelihood function is unimodel and has only stationary point, then $\mathbf{r}^t$ converges to the unique maximizer of $\mathbf{r}^*$.

5.4 Experiments

In this section, we do experiments in some sequential decision making benchmark domains: *GridWorld*, Mountain car and Secretary problems. A purpose of these experiments is to show that our method is superior to other algorithms in dealing with a small number of observations. This is useful for potential application in many real world problems.

In our proposed method, there are some tuning parameters waiting to be selected: the temperature parameter β and the regularization coefficient λ. Here, we give some examples to find the optimal selection by using the Bayesian information criterion (BIC):

$$BIC(\beta, \lambda) = -2 \ln p(\mathcal{O}|\mathbf{r}^*) + (\ln n)\mathrm{Dim}(\mathbf{r}^*)$$

where n is the sample size of teacher and $\mathrm{Dim}(\mathbf{r}^*)$ is the number of parameters. In the following experiments, we compare our method with the popular IRL algorithms in two ways: qualitatively analyzing the reward function and its output policies; quantitatively calculating the performance accuracy that will be defined in the next section for *GridWorld* problem. The available IRL algorithms we used in comparison with our method contain

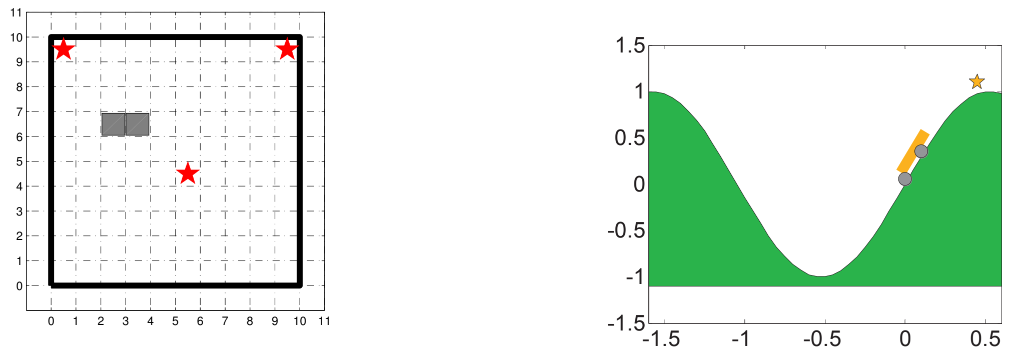
1. Naive mimic: choose the action if observed; otherwise randomly select an action.

2. Mimic IRL: we uniformly choose the random action at the states unobserved. Then, the value function for the unobserved state is an average over the Q functions of taking an action at that state. Finally, we can solve the IRL problem as a problem with complete observation, using the algorithm described in [2].

3. PROJ IRL: an algorithm in [21] and mentioned in Section 5.2.3.

4. WMAL IRL: a game theoretic approach algorithm in [23].

5.4.1 *GridWorld* problem



(a) Grid World (b) Mountain car

Figure 5.3: Picture of (a) Grid World (b) Mountain Car problem

The first experiment we study is *GridWorld* problem in which an agent starts from an initial square and is going to a destination square (See Figure5.3 (a) in which the stars indicate the destinations). In our environment, there may be some obstacle in the grid world. The agent is able to take five actions, moving in four directions or staying put. The outcome of the moving actions succeeds in probability 0.65, fails to move in probability 0.15, and results in unforeseen direction in probability 0.2.

In practical problems, we can observe the decision behavior from other agents using some techniques such as sensors. In this simulation, to generate the observation data, we first manually design a reward function that has encoded the objective functions, and then simulate a demonstrating agent who navigates in the grid world by forward planning with this designed reward function. Only a small number of observations are collected.

To evaluate the performance of IRL algorithms, a direct method is to compare the underlying true reward functions with the reward functions recovered by IRL. E.g., we can compute the Euclidean distance between the true reward function and the recovered reward function. However, it is not accurate and even meaningless in practical purpose due to the illness of IRL problems. Because an identical optimal policy may be induced by two different reward functions, and two similar reward functions may provide different optimal policies. Due to the noise added to observations by uncertainty in the environment dynamics, it is also not appropriate to compare the policy directly.

Since the reward function is considered as a function encoding the goal of an agent, the most concerned should be whether the recovered reward is also able to encode the original goal. To evaluate IRL algorithms quantitatively, we propose to count the percentage of successfully solved problems, which is called *performance accuracy*. A problem is formulated by randomly selecting an initial state in the same environment. We call a reward function has successfully solved a problem, if the best policy, which is output by forward planning with this reward function, can make the agent reach the true goal. The following steps describe how to compute the performance accuracy p_a for learned reward $\hat{\mathbf{r}}$ in 1000 runs of replicate simulations.

1. Initialize $C_r = 0$, which counts the number of problems solved by $\hat{\mathbf{r}}$, and $C_n = 0$ which counts the quantity of problems not solved by the true reward.

2. Generate a GridWorld problem by randomly choosing an initial state.

3. Apply RL to solve this problem using learned reward. If the objective is reached in finite time, $C_r = C_r + 1$. Consider the suboptimal solution, we have additional calculation for our MMIRL (which is represented by MMIRLsub): if the

destination is not the true objective, but adjacent to the objective (E.g. grid $(10, 2)$ is considered close to grid $(10, 1)$), $C_r = C_r + 0.5$.

4. Ask the teacher using the true reward to solve the problem the same as that in last step. If the teacher fails to arrive at the true goal in the finite time, $C_n = C_n + 1$ and $C_r = C_r - 1$.

5. Repeat the step 2 to step 4 for 1000 times. Calculate performance accuracy $p_a = \frac{C_r}{1000 - C_n}$.

We did four sets of experiments on *GridWorld* problem. The first two sets of experiments are designed to qualitatively show the learned reward function and its influence on policy, value function and goals. Then we compute the performance accuracy p_α to quantitatively compare the performance of our method with several prevailing IRL algorithms.

Qualitative evaluation - part of goals being observed

A set of experiments are conducted given the observation that includes partial goal states. We directly show the policy in Figure5.4 (a), (b) and (c). In the Figure5.4 (a), we see that the true reward function directs the agent to reach the goal states marked at the positions $(1, 10)$, $(10, 1)$ and $(10, 10)$. The observation data only contains 50% of states and has missed the goal state $(10, 1)$. For the learned reward function, we show its best policy in Figure5.4 (b) for LIRL and (c) for MMIRL, display the value function for the best policies respectively in Figure5.4 (d), (e) and (f). The policy maps show that the best policy output by our learned reward function in Figure5.4 (c) is more similar as the true policy than the result displayed in Figure5.4 (b). The proofs can be found in the arrow manifold and the number of and position of estimated goal states[3] . A comparison of the plots of value functions reveals more evidence that our learned reward approaches more closely to the true reward. From the color map in Figure5.4 (e), it is apparent that the value function estimated by LIRL has high values at many states, implying much more uncertainty on estimation of the goal.

Qualitative evaluation - goals not observed

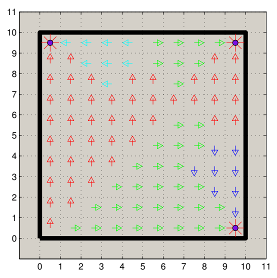
In the second set of experiments, we simulate a condition in which the the true goal states (Grid $(1, 10)$, $(6, 5)$ and $(10, 10)$) are completely unobserved. The percentage of states observed is still 50%. Using the similar steps in the first experiment, we compare the best policies output by learned reward function and their value functions.

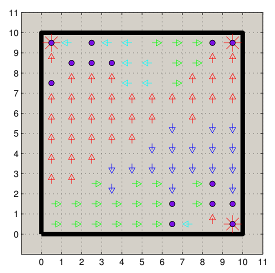
Figure5.5 displays the convergence rate for two sets of experiments respectively. As an illustration of goal and the actions taken to reach the goal, Figure5.6 (a), (b), and (c) show the best policy output by RL using the ture or recovered reward function. It is clear that MMIRL has less noise in goal estimation, and its policy map, drawed by directed arrows, is more similar as the true policy map than that of LIRL. Figure5.6 (d), (e) and (f), displaying the value functions, confirm that MMIRL is better at capturing the goal with incomplete observations. In comparision with the energy map of true value function, MMIRL provides a map with more accurate shape and energy distribution.

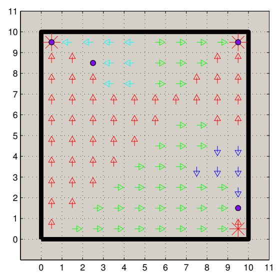
Quantitative evaluation

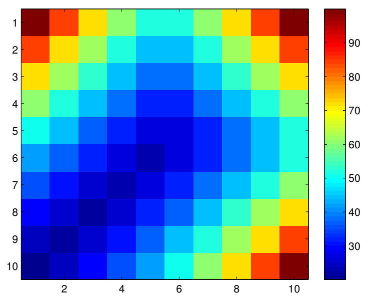
The third set of experiments are conducted to compare the IRL algorithms using performance accuracy p_α. Besides the settings mentioned in the previous two sets of experiments, we also consider the performance with respect to varying number

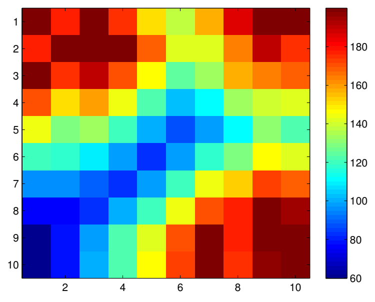
[3]The goal states are denoted as asterisks in the figure.


(a) True policy map


(b) Policy map by LIRL


(c) Policy map by MM IRL


(d) True value function


(e) Value function by LIRL

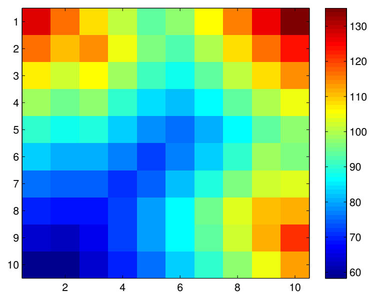
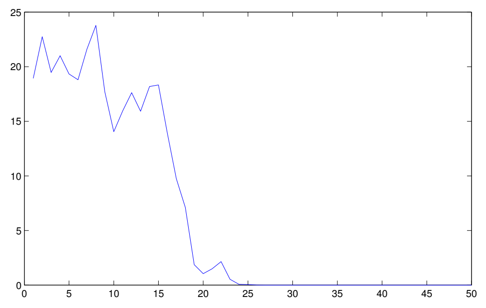
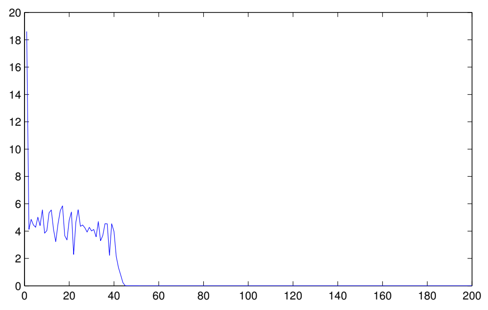

(f) Value function by MM IRL

Figure 5.4: The arrows represent the optimal moving actions selected at each state and the circle denotes the destination place.

of observations. The number of observations obtained through decision trajectories depend on two parameters: the number of trajectories H and the length of horizon stage T.



(a) First experiment

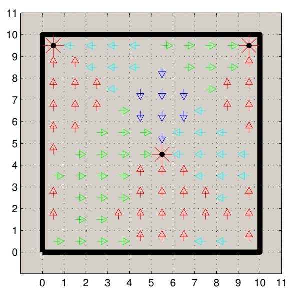


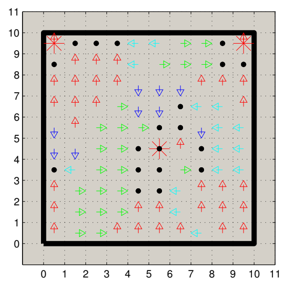
(b) Second experiment

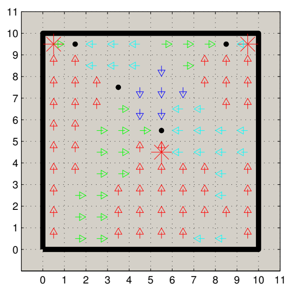
Figure 5.5: The convergence plot: $\|\mathbf{r}^{t+1} - \mathbf{r}^{t}\|$ as a function of iteration step t.

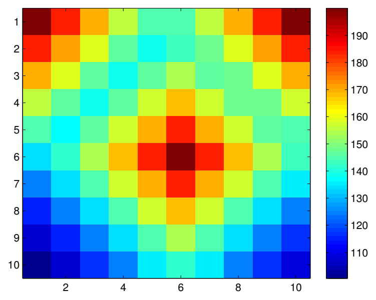
Since the GridWorld example we studied is a 10×10 square field, we set a fixed length of horizon stages $T = 10$ while comparing IRL algorithms in the third set of experiments. So the number of observations is completely specified by H[4]. In this

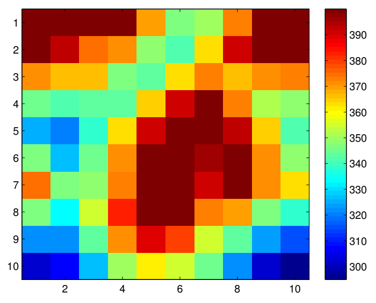
[4]The success of applying IRL algorithms based on the conception of feature expectation is more tricky than we first thought, which depends on both the number of trajectories H and the horizon length T. Further study shows that the horizon length T has more influence on the algorithm performance. It is reasonable because when the horizon is relatively larger than the problem size, we will arrive at the goal state almost sure. Consequently, the larger the horizon is, the more probable we will observe the true goal in every trajectory, and the weights in calculation of feature expectation will increase more heavily. Since we may observe true state every time and assign more weights to the goal state, the advantage of using IRL for goal inference diminishes. Therefore, it is more useful to set the horizon length short in the small problem, to make the observation incomplete.


(a) True policy map


(b) Policy map by naive mimic IRL


(c) Policy map by MMIRL


(d) True value function


(e) Value function by naive mimic IRL

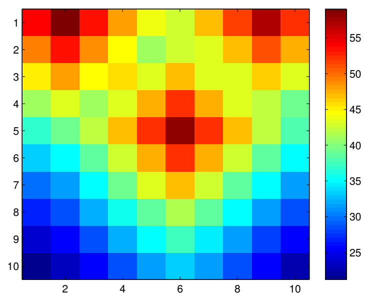

(f) Value function by MMIRL

Figure 5.6: The arrows represent the optimal moving actions selected at each state and the circle denotes the destination place.

experimentation, when H is in the range $(12, 15)$, the number of observed states is around 50% of the size of the state space. With the consideration of randomness in sampling, we run 50 replications of learning to get the averaged performance accuracy

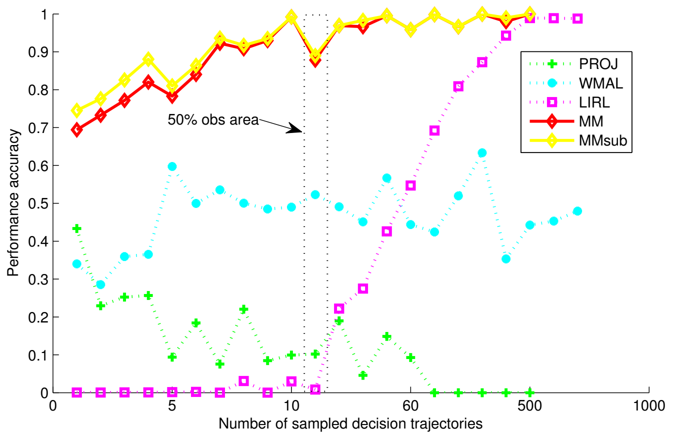


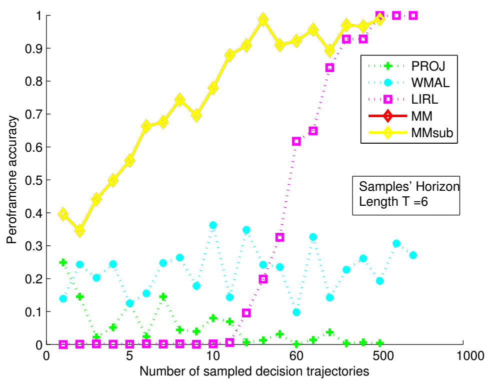
Figure 5.7: Plot of performance accuracy with respect to the number of sampled decision trajectories.

for a given H. The results with respect to performance accuracy p_α are shown in Figure5.7.

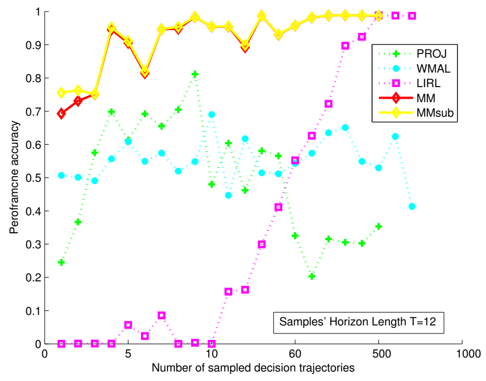
The fourth set of experiments aim to show that our method performs much better than the prevailing IRL algorithms when the horizon length of the sampled decision trajectories is small. In Figure5.8, we show the performance accuracy with different settings of horizon length, such as $T = 6, 12, 15$, and 50. It is obvious that our method, MMIRL, provides higher performance accuracy than other IRL algorithms while T is small. The PROJ and WMAL, which is based on feature expectation, fails to identify the reward functions until the horizon length T becomes larger than 15.

Table 5.1 lists the observation rate[5] as a function of horizon length, which is calculated by 1000 times of simulations. We have observed that the observation rate is mainly determined by the number of samples, rather than the horizon length. But the goodness of feature expectation heavily depends on the horizon length, since the long sampling length will make the goal state observed for many times, and then

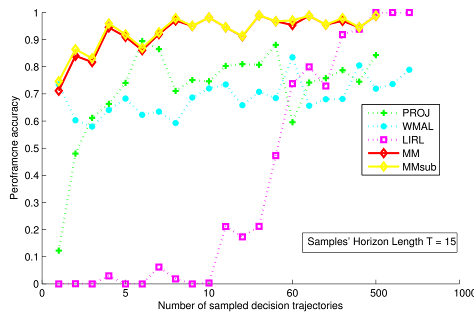
[5]The observation rate denotes ratio of observed states to all the states in state space.



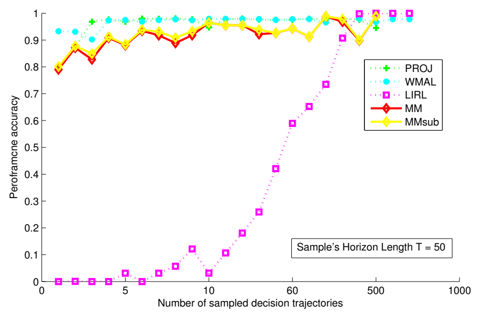
(a) T=6



(b) T=12



(c) T=15



(d) T=50

Figure 5.8: Plot of performance accuracy with respect to the number of sampled decision trajectories.

add more weights to the goal states. This property will restrict the applicability of algorithms based on feature expectation in real-world problems where the access to observation is restricted. For the 10×10 GridWorld problem, when $H \geq 15$, the feature expectation approaches to the true goal function. Consequently, our method is more useful when the access to observation is restricted, e.g. a few number of short-length decision trajectories.

Table 5.1: Observation rate

T	Number of sampled decision trajectories - H											
	1	3	5	6	9	10	15	25	40	60	80	100
10	0.1	0.22	0.27	0.28	0.4	0.46	0.52	0.63	0.73	0.86	0.86	0.96
12	0.09	0.17	0.39	0.29	0.41	0.46	0.5	0.64	0.86	0.88	0.9	0.95
13	0.06	0.19	0.28	0.37	0.35	0.48	0.46	0.58	0.8	0.87	0.88	0.92
15	0.1	0.19	0.23	0.36	0.41	0.49	0.54	0.74	0.77	0.94	0.99	0.98
50	0.01	0.17	0.21	0.18	0.38	0.58	0.49	0.51	0.8	0.86	0.88	0.97

5.4.2 Mountain car problem

We operate the second experiment on the simulation of an under-powered car that aims to reach a hill top (See Figure5.3(a)). The difficulty is that the gravity of the car is stronger than its engine, so the car must goes back and forth to gain enough momentum to arrive at the destination - hill top. Mountain car problem can be well solved by RL algorithms, such as SARSA [74, 75]. The state of this problem can be described by two continuous variables, car position x and its speed $\dot{x}$, which are constrained to $-1.5 \leq x \leq 0.55$ and $-0.07 \leq \dot{x} \leq 0.07$ respectively. The action consists of three countable values (forward, keep and backward). The true reward function is a Gaussian distribution function whose mean position is at the position of hill top.

In our experiments, we convert the continuous state into finite discrete state by applying grid discretization of state space. E.g. a 20×6 grid will provide a state space with 120 states. Given the predefined reward function, using SARSA, we are able to find the optimal policy to drive the car up the hill top. We call the driver an expert, if he/she knows the optimal policy. The IRL problem here is that if we have some incomplete observation of the expert, are we able to infer the goal of the expert and learn his/her skills to drive up the car?

We sample a set of short decision trajectories and use IRL algorithms to learn the reward function from these observations. Once the reward function is determined, we can find the optimal policy by SARSA or other DP algorithms based on a MDP with estimated transition probabilities. The quality of learned reward function is determined by the best policy output by the RL/DP algorithms using this reward function. A learned reward function is considered as correct if its best policy is able to drive the car up the hill top. Otherwise we say the IRL method fails to identify the true reward function.

In the experiments, the state transition probabilities are estimated from the decision trajectories and then used as a prior. When the horizon length of a decision trajectory $T = 50$ and the number of trajectory $H = 500$, experimental results show that PROJ, WMAL, LIRL all fail to learn a correct reward function[6]. But our method is able to infer a good reward function that drives the car up the hill top. The results are summarized in Figure5.9[7]. The second set of experiments studies the performance of reward function on Q-learning algorithms with unknown dynamics, e.g. SARSA. Our method is also the only one that is able to infer the correct reward function. Figure5.10 draws the number of steps the car has taken to reach the destination.

[6]The observation rate is near 90%

[7]Some videos, recording the car's behavior determined by these recovered reward functions, are available at *http://people.virginia.edu/ qq2r/mcarvideo.*

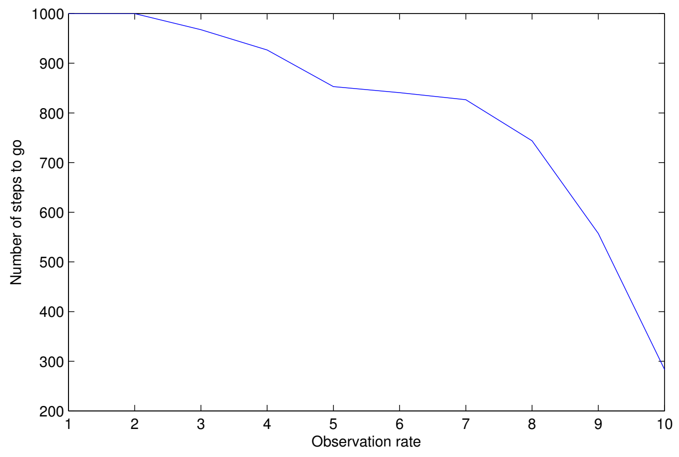


Figure 5.9: Plot of number of steps the car used to reach the goal vs. the observation rate.

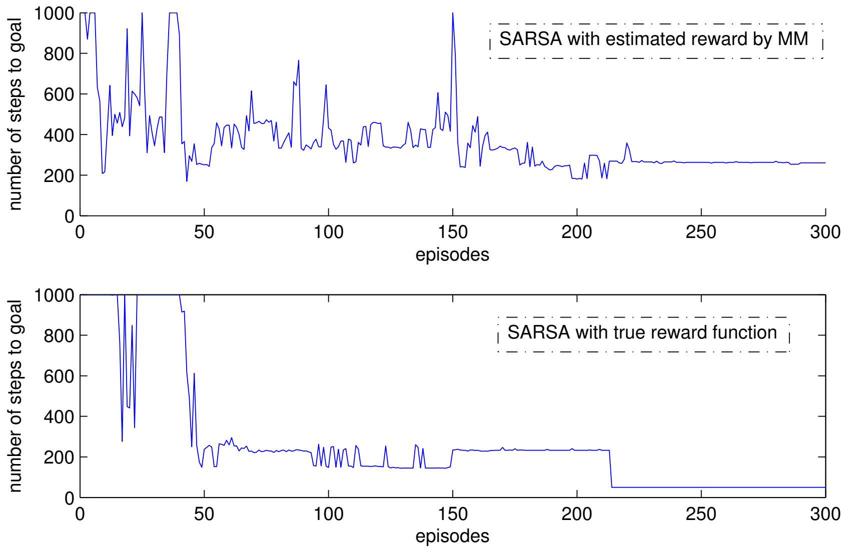


Figure 5.10: Plot of number of steps the car used to reach the goal.

5.4.3 Secretary problem

Secretary problem is a class of optimal stopping sequential decision tasks in which the binary decision to either stop or continue a search is made on the basis of objects already seen [76]. The study of secretary problem is to prove the hypothesis that in the framework of modeling decision making process by MDP, the learned reward function is able to capture the heuristic decision rule behind the observations of decision trajectories (See our work in [63]). Note that the decision makers in secretary problem are not required to follow MDP model.

In the archetypal problem, there is a job vacancy with a pool of applicants. Applicants are interviewed sequentially, one by one, in a random order. At the conclusion of each interview, the employer has to make a decision to accept or reject the applicant, with no opportunity to recall a rejected applicant. The interview process is terminated once an applicant is accepted.

Based on the study by Seale [76], many decisions in secretary problem can be explained by the following heuristic decision rules

- A cutoff rule: With a cutoff value h, the DM will reject the first $h - 1$ applicants and then accept the next candidate. The cutoff decision rule includes the optimal policy as a special case.
- A successive non-candidate counting rule: With a parameter value k, the DM will accept the first candidate who follows k successive non-candidate applicants since the last candidate.
- A candidate counting rule: The DM selects the next candidate once there have been l different candidates interviewed.

These decision rules all assume a single counter as the parameter but differ from each other in what will be counted. The dependency of the parameters are on the value of how many applicants have been interviewed.

Each of decision rules was evaluated for the secretary problem with 80 applicants. The simulation of decision strategy was carried according to the process described in [76]. We briefly summarize this process in Alg. 11. Each possible variant of decision rule with a specific parameter value (cutoff value h or non-candidate applicants k) was replicated $10,000$ times to obtain an estimate of its accuracy. This accuracy means that how many times the decision maker can find a correct secretary, given a specific decision rule. The simulation results indicate that different decision strategies have different shapes of accuracy curves that are plotted in a function of the rule parameter (See Figure5.11).

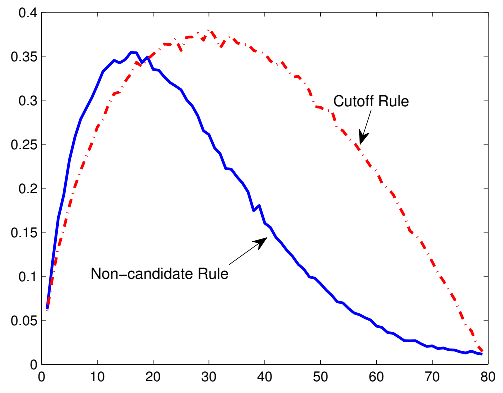


Figure 5.11: Plot of decision rules: cutoff rule and non-candidate rule

We study Secretary problem here because it is a sequential decision making problem widely existing in many practical areas, and it is more important that the simulated human agent, who adopts any of the three heuristic decision rules, behaves out of MDP rational property's constraints. Therefore, we aims to test whether our IRL method can recognize the decision strategy even when the decision making process is not in MDP world. Technically, to apply IRL algorithms, we model the observed behavior using MDP and learn the reward function of this MDP which is considered

Algorithm 11 Heuristic decision rule simulation

```
1: Set the number of applicants $N$.
2: for Set parameter value 1 to $N-1$ do
3:     for Repeat 1 to $10,000$ do
4:         Generate a set of $N$ applicants randomly.
5:         while Interview continues do
6:             Present applicants to the decision rule one at a time in a random order.
7:             if Applicant is a candite and meets the criteria of the decision rule then
8:                 Applicant is selected and interview is over.
9:             else
10:                Interview continues.
11:            end if
12:            if All the applicants have been interviewed then
13:                Accept the last applicant and interview is over.
14:            end if
15:        end while
16:        Reveal the rank of applicants.
17:        if The selected applicant is top ranked then
18:            We call a correct selection and accuracy counting $C = C + 1$.
19:        end if
20:    end for
21:    The accuracy of correct selection is estimated by $\frac{C}{10000}$.
22: end for
```

as a coding of the decision strategy. The experimental process includes:

1. Simulate the given number of interviewing examples using Alg. 11. Note that we only sample the behaviors in these examples, rather than computing the selection accuracy.

2. Model the interviewing process using MDP and apply IRL algorithms to estimate the reward function.

3. Generate a new set of 10000 interviewing problems with random applicants. Run forward planning with the reward function recovered in the previous step. Solve these interviewing problems according to the best policy output by the forward planning. Calculate the selection accuracy.

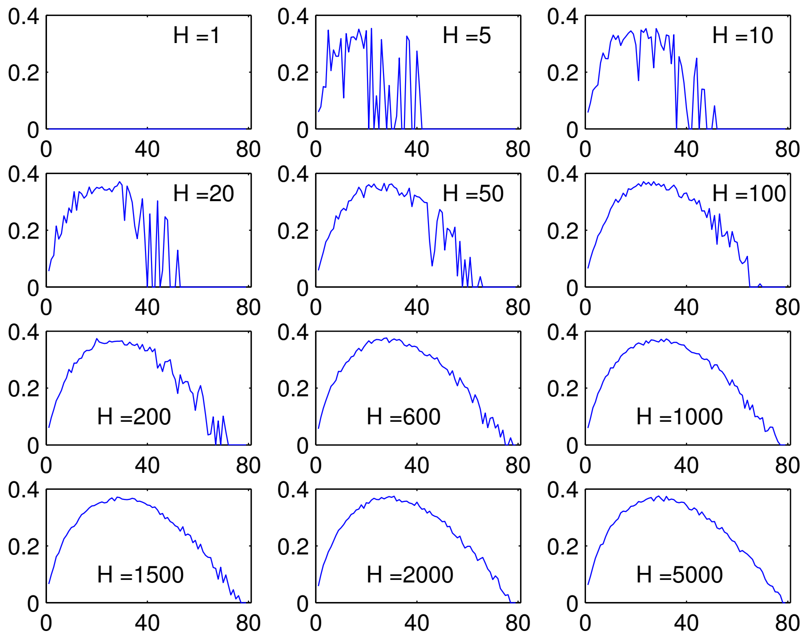


Figure 5.12: Plot of LIRL's selection accuracy vs the cutoff parameter

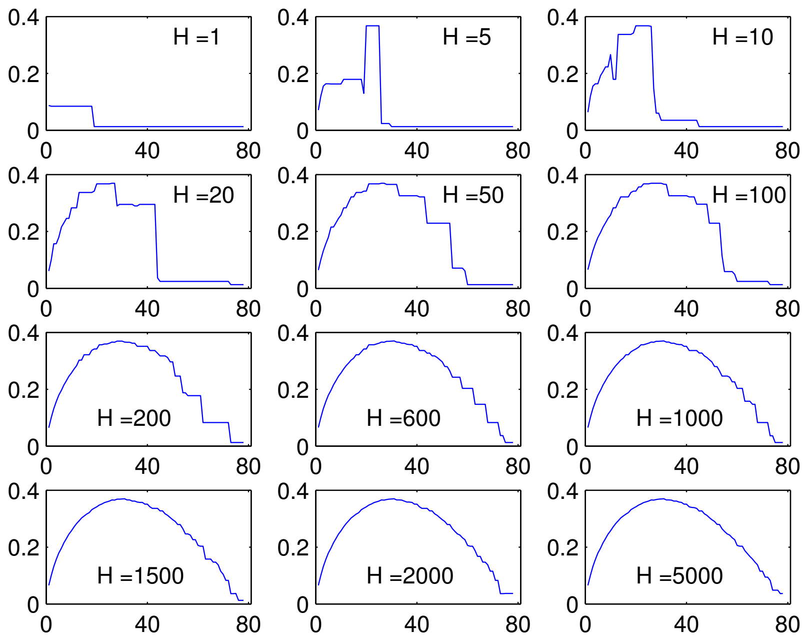


Figure 5.13: Plot of MMIRL's selection accuracy vs the cutoff parameter

A comparison of accuracy curves between the heuristic decision strategy and learned MDP reveals that IRL method can be applied to identify the decision strategy. It can been seen from the comparison between Figure5.12 and Figure5.13 for cutoff heuristic decision rule, and the comparison between Figure5.14 and Figure5.15 for noncandidate heuristic decision rule[8], that our method, given incomplete observations, approximates the true decision strategy better than IRL algorithms based on linear approximation without aid of the state knowledge.

[8]Here we only show the results of experiments in which the number of applicants $N = 80$. Because the replicate experiments with varied N show that the problem size does not affect performance report.

5.5 Conclusion

We propose an IRL framework that utilizes the philosophy of MM optimization techniques to deal with the question of learning with the help of unobserved information. The traditional IRL algorithms gain the information from observation of action on state, while if the number of observations is small, the performance of recovering a good reward function degrades severely. Like the idea of semi-supervised learning in machine learning field, we expect to reduce the uncertainty by mining more information from the unobserved states. At the same time, we solve the question of how to use IRL with incomplete and suboptimal observation. First, we developed the maximum likelihood formulation for incomplete observation with the constraints to guarantee the true of observation. Then, we built optimization problems using MM techniques to solve IRL with incomplete and suboptimal observation. Finally, we prove the convergence of our proposed method and conduct experiments to show the effectiveness in comparison with prevailing IRL algorithms.

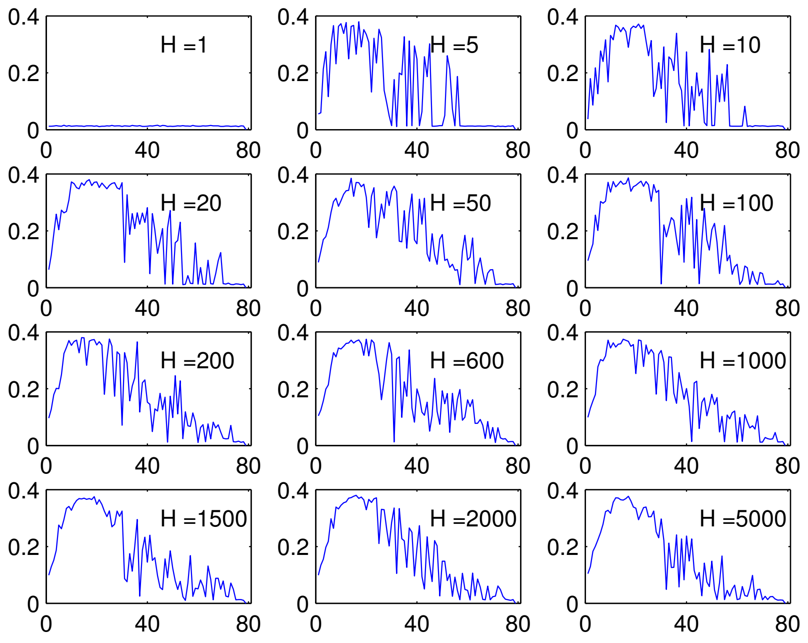


Figure 5.14: Plot of LIRL's selection accuracy for the non-candidate decision rule.

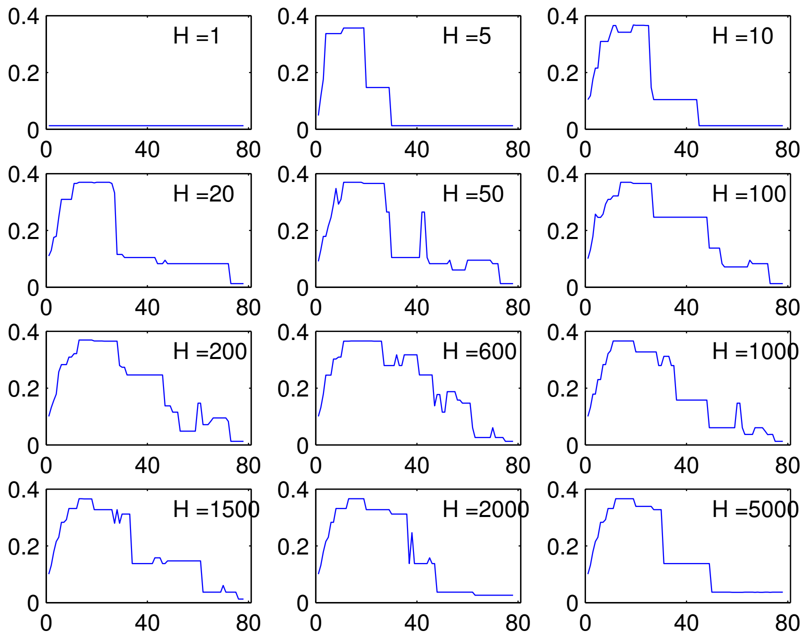


Figure 5.15: Plot of MMIRL's selection accuracy for the non-candidate decision rule.

Chapter 6

Decision Strategy Recognition in Reward Space

In this chapter, we study the use of inverse reinforcement learning (IRL) as a tool for the recognition of decision agents on the basis of observation of their actions in solving sequential decision problems. We model the problem faced by the decision agents as a Markov decision process (MDP) and model the observed behavior of the agents in terms in terms of degrees of rationality with respect to optimal forward planning for the MDP. To recognize the agents, we first use IRL to learn reward functions consistent with observed actions and then use these reward functions as the basis for clustering or classification models. Experimental studies with *GridWorld*, a navigation problem, and the *secretary problem*, an optimal stopping problem, suggest reward vectors found from IRL can be a good basis for classifying automated decision rules (e.g., cutoff rule, successive first candidates), even in the presence of action noise and variations in the parameters of the rule. We propose a new Bayesian IRL approach in which the likelihood function can be interpreted in terms of rationality models. Empirical comparisons of our method with several existing IRL algorithms and with direct methods that use feature statistics observed in state-action space suggest it may be superior for recognition problems.

6.1 Introduction

6.1.1 The decision strategy recognition problem

In sequential decision problems, the forward planning problem is one in which an agent seeks to find a decision strategy that yields a sequence of actions that achieve a given goal. The inverse process, which is sometimes called *plan or goal recognition*, is to infer the goal of an agent on the basis of observations of its actions [77, 78, 79]. Much of the existing work on plan recognition uses plan libraries that represent sets of alternative plans for solving domain-specific problems, and thus has a primary focus that independent of the method used for planning. More recent work has proposed the idea of solving the plan recognition problem using decision-theoretic planning models [10, 79].

Models for inferring the goals of decision makers as part of plan recognition or inverse decision theory have been the subject of considerable research in cognitive and psychology science (see, e.g., [80, 10, 81]). To take an example, it has been shown that before 18 months of age, children have difficulty in understanding the subjective nature of intentions [82], but two-year-olds readily recognize when another person holds a preference similar to their own and achieves desired outcomes using the shared preference [83]. Varieties of experiments on children and adults prove the effectiveness of some computational models for how people infer others' goals or preferences as a kind of inverse planning or inverse decision theory. Most such models have been developed on the basis of optimal control theory or theory of planning under uncertainty, using the principle of rational behavior. Rationality is also at the heart of the classical *theory of mind* [14] in which agents choose actions that achieve their desires through maximization of expected utility. Under such a framework, the goals or preferences of a decision maker can be inferred from observation of actions; the decision maker's utility function must have the property that the set of actions that maximize it in expectation include the observed behavior as a subset.

The IMDP problem defined in Chapter 2 can be considered a generalization of the plan recognition problem. The forward planning problem of achieving a goal, such as reaching a particular cell in GridWorld, can be modeled as an MDP with a restricted reward function (e.g., reward everywhere zero except positive at an absorbing state corresponding to the goal). In such a case, the goal recognition problem could be solved using an IRL formulation with side constraints to ensure the learned reward function has the desired structure. IMDP can be addressed using IRL without side constraints, and so much be a generalization of the restricted, goal recognition problem.

We now define a problem that is related to but distinct from goal recognition and IMDP.

Definition The *decision strategy recognition* problem is to identify the patterns of the decision rules that determine the sequential behavior observed in a particular environment.

The assumption is that each agent has adopted a decision rule that is hidden from the observer. Thus, the recognition problem can be framed as a supervised or unsupervised learning problem:

1. Given observations of decision trajectories consisting of sequential actions and given a label for each trajectory indicating which decision strategy was used by the agent, the problem is to determine the decision strategy for an unlabeled trajectory.

2. Given only observations of the decision trajectories, the problem is to assign trajectories to clusters on the basis of similarity of decision strategy.

We propose an IRL-based approach to the decision strategy recognition problem. Figure6.1.2 interprets the tasks of clustering and classification. In sub-figure (a), the input data consists of behavior observed from the agents, and for each agent the IRL learns his/her reward function, which is denoted by x. Finally, the agents have been grouped according to their reward functions. Similarly, in sub-figure (b), every agent's reward function has been recovered and input into the classifier for training or testing. The training set of agents have labels indicating the decision rule they have adopted.

The agents in our study are meant to model human decision makers as well as machine agents that automatically plan and choose actions. We model the problem faced by the decision agents as a Markov decision process (MDP) and model the observed behavior of the agents in terms of degrees of rationality with respect to optimal forward planning for the MDP. To recognize the agents, we first use IRL to learn reward functions consistent with observed actions and then use these reward functions as the basis for clustering or classification models. Adopting a Markov Decision Process (MDP) framework, we use variables of state and action to represent the environment and characterize the individual agent using its belief of environmental dynamics and the reward function that is assumed to encode the goals.

IRL depends on the principle of rationality and optimizes the reward function by minimizing a cost function of observation. It is more difficult to learn the reward function from the given behavior than forward planning, because the inverse problem is ill-posed. We propose the new IRL algorithm to reduce the ill-posed uncertainty and infer the reward function adaptively when the observed behavior is not strictly rational.

6.1.2 The secretary problem

In this section we introduce the secretary problem which, with GridWorld, is the subject of numerical experiments described later in the chapter. Secretary problem has been introduced in Chapter 5. Here we review this problem and show its assumptions. In the classical problem N rankable objects (job candidates, alternatives) appear sequentially and randomly, before a decision maker who has to accept or reject the object on the basis of rank relative to the objects already seen. Time is assumed to be discrete and the number of objects N is known. The goal of the decision maker is to maximize the probability that the item accepted is, in fact, the best. The secretary problem with uncertainty (SPU) is formulated in terms of the following assumptions:

1. The number of applicants is finite and known before solving this problem.
2. The applicants are interviewed sequentially in a random order.
3. There is only one position to be filled
4. At every interview, the interviewer has to make a decision whether to accept or reject the applicant based on the relative ranking of current applicant. Once the applicant is accepted, the problem is over. Otherwise the problem continues.

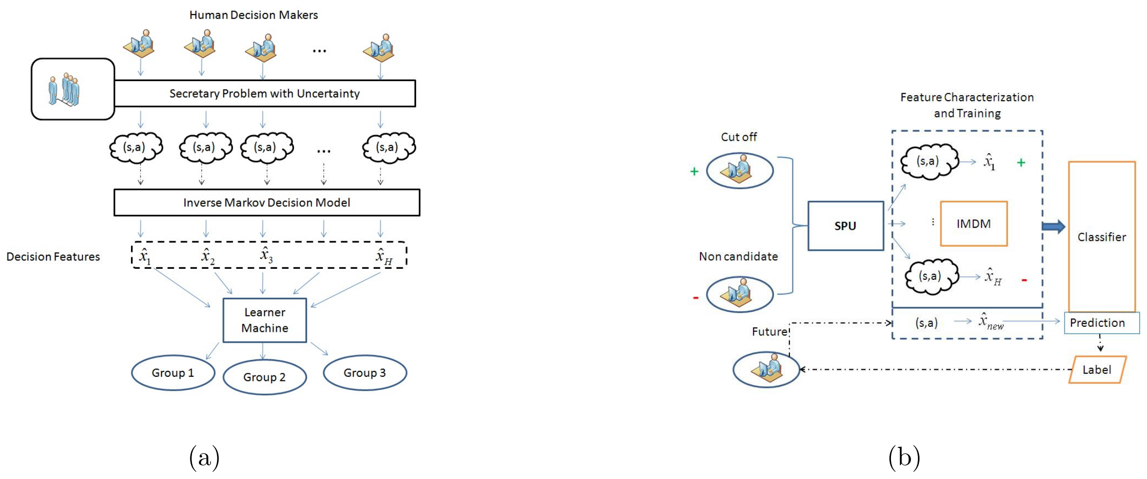


Figure 6.1: (a) clustering (b) classification

When the last applicant comes, the interviewer has to accept him/her. The rejected applicant can not be recalled.

5. The interviewer receives the reward if the best applicant is selected, otherwise no reward.

Though these assumptions place more constraints on the problem than apply in practice, they make a good example to explain the decision process numerically.

Using SPU as a study case, it is important that we do not add constraints to the agents that their actions are determined in MDP. In fact, the behavior we have observed in experiments are the results of using the heuristic decision rules, which are completely not related to MDP and the rationality assumption. However, we model the observed behavior in MDP and have successfully recognized the decision rule within the inverse MDP framework. By doing varieties of experiments, we have shown IRL method's superiority to other methods in the tasks of prediction, clustering and classification.

6.1.3 Bayes inverse learning

The Bayesian expected reward provides a way for us to evaluate the learned reward function, which is introduced in the following definition.

Definition *The Bayes inverse learning principle*

Given an observation of action a at state s_n, a reward function with distribution $\theta(r)$ is feasible if it satisfies

$$\rho(\theta(r), s_n, a) > \rho(\theta(r), s_n, \hat{a}) \ \forall \hat{a} \in \mathcal{A} \setminus a.$$

Remark We design an IRL rule that the optimal reward is to minimize a Bayes risk $L((\rho(\theta(r), s_n, a) - \rho(\theta(r), s_n, \hat{a}))$. The risk $L(\rho(\theta(r), s_n, a) - \rho(\theta(r), s_n, \hat{a}))) = L(Q(s_n, a|r = E_{\theta(r)}(r)) - Q(s_n, \hat{a}|r = E_{\theta(r)}(r)))$. This risk function should penalize the positive difference between $Q(s_n, a|r = E_{\theta(r)}(r))$ and $Q(s_n, \hat{a}|r = E_{\theta(r)}(r))$.

Definition *The principal of satisfying behavior*

Given observations $\mathcal{O} = \{O_h : (s_0, a_0, \cdots, s_t, a_t)_h\}$, where $h \in \{1, 2, \cdots, H\}$ and t is the length of the trajectory. Given the same problem, a decision satisfies the observation $\mathcal{O}$, if its output action a_m at state s_n is contained in the set of actions observed at s_n.

A solution to the IRL problem B outputs satisfying behavior in forward planning. The decision strategy behind the observed behavior has been represented by this recovered reward function.

6.2 Action Space and Goal Space

To solve the behavior recognition problem, the common approach is to process the observation data and formulate a feature vector on the basis of the observation, which we call learning in action space. E.g. there are five alternative actions, and we can build a feature vector by counting the frequency of utilization for each action. Here, we analytically discuss the properties of action space in short and why we propose to learn in goal space.

First, the action space is probably too large to describe in a finite set. The size of feature space directly characterizing the decision behavior may increase exponentially with the number of actions.

Second, there is no obvious approach to defining the action space, which varies as the problems change. Moreover, recognition results heavily depends on how well the features are constructed in the action space to represent and identify the decision rules.

Third, the manual designed features in action space often ignore some part of the decision process that can not be explicitly seen from the observation. The hidden goal functions, which determines the actions implicitly, are not guaranteed to be considered in the features of action space. It is like a relationship between rule-based decision making and principle-based decision making. Modeling of decision making using action is like to learn the decision making in the form of rule-based process, while the goal function behind the observations plays a role as the principle that influences the individual decision implicitly while giving a guarantee of reaching the objective state in the long run. As a result, the action space may be sensitive to noise for representation of the agents, while the goal space is more robust in this perspective.

6.3 Computational Framework

Our proposed computational framework formalizes the behavior recognition problem as Bayesian inverse learning of MDP. We consider two practical scenarios: the first condition is that the observation is given one at a time, providing an explicit action at current state. Therefore, the observer has the option of making a decision now or waiting more observations; the second condition is that a number of sampled behavior trajectories are ready for the observer at the beginning.

6.3.1 On-line inference

Given a sequence of observations, denoted as $O_h = (s_h^0, a_h^0, s_h^1, a_h^1, \cdots, s_h^t, a_h^t)$, the Bayesian inference of goal function is given by

$$p(\mathbf{r}|O_h) = \frac{p(s_h^0, a_h^0, s_h^1, a_h^1, \cdots, s_h^t, a_h^t|\mathbf{r})p(\mathbf{r})}{\int_{r\in D} p(s_h^0, a_h^0, s_h^1, a_h^1, \cdots, s_h^t, a_h^t|\mathbf{r})p(\mathbf{r})d\mathbf{r}} \tag{6.1}$$

where $p(s_h^0, a_h^0, s_h^1, a_h^1, \cdots, s_h^t, a_h^t|\mathbf{r}) = p(s_h^0|\mathbf{r})p(a_h^0|s^0, \mathbf{r})p(s_h^1|s^0, a^0, \mathbf{r}) \cdots p(a_h^t|s_h^t, \mathbf{r})$. By Markov property, this joint probability calculation is simplified by dropping the components not containing the reward function, then the function becomes

$$p(\mathbf{r}|O_h) \propto \prod_{i=0}^{t} p(a^i|s^i, \mathbf{r})p(\mathbf{r}) \tag{6.2}$$

In this equation, $p(a^i|s^i, \mathbf{r})$ is the likelihood function of choosing the action observed in the forward planning. The prior $p(\mathbf{r})$ sets up a hypothesis space of goals that are realizable in the environment, and it can be uninformative prior as a uniform distribution. The Bayes' rule integrates the exterior information from observed behavior and the inner constraints from the prior to infer the reward. When we have no prior information on the possible goals, the problem is solved by Maximum-likelihood methods. Our previous study shows that Bayesian inference with Gaussian prior is

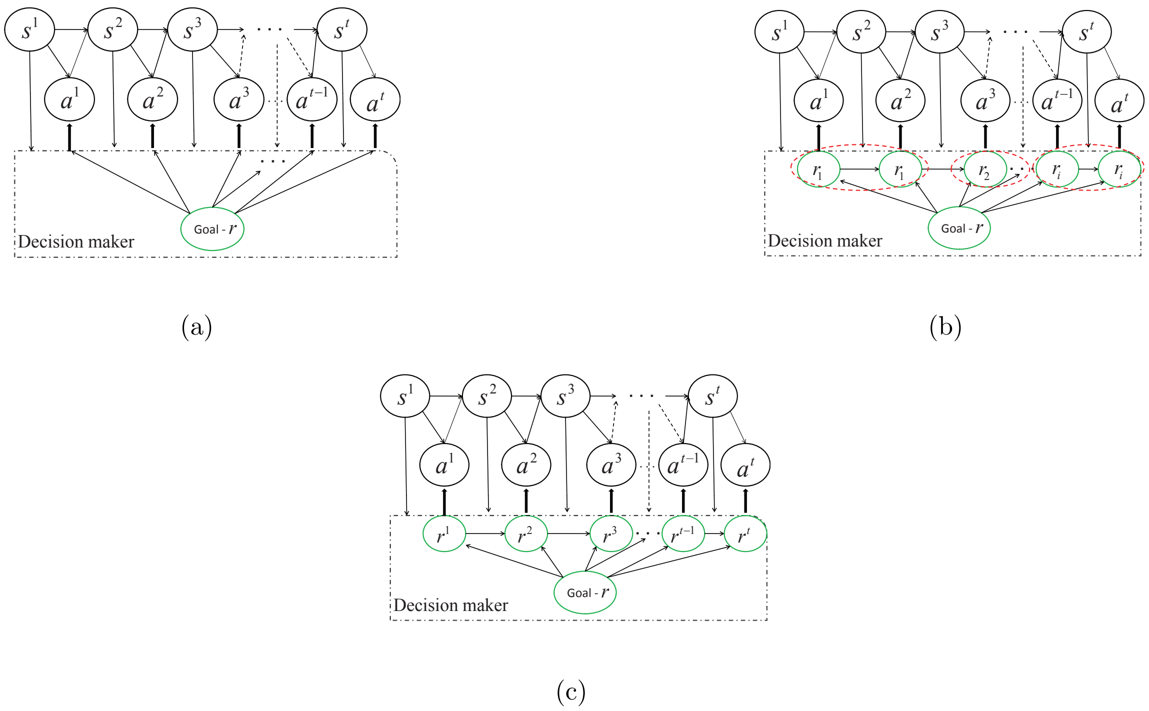


Figure 6.2: Stationary and Changing reward structure

an effective and tractable method for IRL in both finite and continuous domains [51], particularly being superior to other algorithms when the number of observations is small.

To formulate the likelihood function $p(\mathbf{r}|O_h)$, we should make *the principle of satisfying behavior* true in the formulation. This requirement adds a set of constraints in the settings of MDP, which means that at every state observed in $\mathcal{O}$, the reward function should be feasible. According to *the Bayes inverse learning principle*, we have, $\forall(s^i, a^i) \in O_t,\ a \in \mathcal{A} \setminus a^i, \forall r \in D,\ Q(s^i, a^i) \geq Q(s^i, a)$. Another concern we have is whether the observed agent's goals change over the course of making decisions. Figure6.2 interprets three kinds of reward functions: (a) is the deterministic reward function that is completely known and employed by the decision maker until the end of the process (denoted as R1), (b) is the reward representing the agent who changes the goal in the observation process (denoted as R2), and (c) is that the underlying reward does not change but the agent's perception of it varies noisily in the observation process (denoted as R3). The Eq. 6.2 has already modeled R1 and we modify it for R2 by introducing a parameter $\lambda \in (0, 1]$. With $\lambda = 1$, goal changing is prohibited, and R2 is equal to R1. The setting of λ close to 1 implies that the past and the recent observations are weighted equally in the inference. With λ close to 0, the mode R2 consider that the reward changes frequently, and only the most recent observation factors into the inference. The modified equation is written as

$$p(\mathbf{r}|O_t) \propto \prod_{i=1}^{t} \lambda^{t-i} p(a^i|s^i, \mathbf{r}). \tag{6.3}$$

The element of the above equation, $p(a^i|s^i, \mathbf{r})$, is calculated in two ways. We discuss them in the following sub-sections.

6.3.2 Existing IRL for rational behavior

With the assumption that the observed behavior modelled by using MDP is deterministic and optimal, we have the existing IRL algorithms like the linear IRL(LIRL) in [2], WMAL in [23] and PROJ in [84]. These algorithms consider the reward function as a linear formulation which is written as

$$r(s) = \sum_{i=1}^{d} \omega_i \phi_i(s) = \omega^T \phi(s),$$

where $\phi : \mathcal{S} \to [0, 1]^d$ and $\omega^T = [\omega_1, \omega_2, \cdots, \omega_d]$. Then using the observed decision trajectories, apprenticeship learning algorithms aim to find a policy that performs as well as the observation by comparing a metric called *feature expectation* defined as

$$\hat{\mu}^{\pi_E} = \frac{1}{H} \sum_{h=1}^{H} \sum_{t=0, s_t \in O_h}^{\infty} \gamma^t \phi(s_t).$$

Feature expectation can be considered to be a measurement in state-action space. We can use feature vectors to represent the observed agent and then recognize them in action space (This method is called as Action or Action Feature).

In [51], a Bayesian inference model adopts the likelihood function that is

$$p(a|s, \mathbf{r}) = \begin{cases} 1, \text{if } Q(s, a) > Q(s, \hat{a}), \quad \hat{a} \in \mathcal{A} \setminus a \\ 0, \text{otherwise.} \end{cases}$$

Consider Gaussian prior over the reward function which has mean vector $\mu_{\mathbf{r}}$ and covariance matrix $\Sigma_{\mathbf{r}}$. It has been proved that the IRL problem is a quadratic convex programming, yielding

$$\begin{aligned} &\min_{\mathbf{r}} \ \frac{1}{2}(\mathbf{r} - \mu_{\mathbf{r}})^T \Sigma_{\mathbf{r}}^{-1} (\mathbf{r} - \mu_{\mathbf{r}}) \\ &\text{s.t.} \ (\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1} \mathbf{r} \geq 0, \ \forall a \in \mathcal{A} \setminus a^* \\ &\quad \mathbf{r}_{\min} < \mathbf{r} < \mathbf{r}_{\max}, \end{aligned}$$

where the optimal transition matrix $\mathbf{P}_{a^*}$ is constructed by *approximate observation rule*. At state s_n, the transition probabilities are defined as

$$\mathbf{P}_{a^*}(s_n, :) = \begin{cases} \mathbf{P}_m(s_n, :) & \text{if} \quad (s_n, a_m) \in \mathcal{O} \\ \frac{1}{M} \sum_{m=1}^{M} \mathbf{P}_m(s_n, :) & \text{if} \quad (s_n, a_m) \notin \mathcal{O}. \end{cases}$$

Boltzmann distribution

Consider the case showed in Figure6.2 (c) that the reward is understood and used in the noisy perception. Boltzmann distribution can be adopted to model the decisions that are made in a learning process. The likelihood function based on this is written as follows.

$$\max \sum_{n=1}^{N} [\sum_{a \in \hat{U}(s_n)} \beta Q(s_n, a) - c_n \log \sum_{m=1}^{M} e^{\beta Q(s_n, a_m)}] + \log p(\mathbf{r}) \qquad (6.4)$$

where $\hat{U}(s_n)$ is a collection of observed actions at state s_n, and $c_n = \left|\hat{U}(s_n)\right|$ is the frequency weighting of observation at s_n. Note that in practice, many problems have infinite state space. It motivates us to adopt IRL algorithms being capable to learning with finite observations in large space. We propose to use the IRL method in [85]. Here, we give a summary of that method in the following subsection.

IRL with Gaussian process

The key idea of the method in [85] (GPIRL) to deal with the ill-posed difficulty employ the techniques of preference learning in the intersected field of machine learning and decision making. The basis argument is that Bayesian inferences should be an understanding of the observed agent's preferences over the action space. The argument complies with the principles of ToM.

First, GPIRL uses preference graph to represent the observed behavior at a state. A graph consists of nodes and edges. Each node indicates an action and each edge represents a preference relation at a state. The observed action is considered to be preferred to other unobserved actions. Given a set of decision trajectories, we transform the observation at a state into a preference graph.

Second, the reward is modeled by using Gaussian process, which is completely specified by a positive definite covariance matrix. A simple strategy is to assume that the latent processes are uncorrelated and the convariance matrix is calculated by using a squared exponential kernel function. With the likelihood function defined in MDP settings, a Bayesian inference is conducted to obtain the optimized reward functions by maximizing the posterior probability function. Since the hyper parameters, such as the parameters of kernel function, are unknown, appropriate model selection methods are employed to maximize the marginal likelihood function in order to obtain a trade off between the model's complexity penalty and the data fit likelihood.

Finally, the probabilistic model based on Gaussian process provides us a closed form posterior prediction on the new state coming in the future.

6.3.3 Bounded rationality

In practice, the individual bias may creep into the decision making process and distort the planning and decisions. The agent may also need to make decisions while being lack of sufficient computation and memory resources. Therefore, some of observed actions may not be consistent with the agent's underlying goals. When we model the behavior in MDP, these actions will violate the assumption that they are deterministic and completely rational in forward planning. As a result, the feasible reward space defined by the observed behavior may exclude the true reward representing the underlying goals.

In order to consider the reward in a larger feasible region, our model adopts the assumption of bounded rationality. The bounded rational agent behaves as well as possible based on its knowledge resources available. This means that the agent may take a suboptimal action which provides a Q function that is smaller than the possible maximum value but the difference is in a controlled range. Considering this requirement, we modify the IRL model by relaxing the constraints on observed behavior. Moreover, we propose a new likelihood function that includes the parameters to control the degree of rationality and the rate of learning process. This new likelihood function, denoting the observation of an action at a given state, is written as

$$p(a|s,\mathbf{r}) = \frac{1+\sum_{m=1}^{M} g(\Delta Q(a,a_m))(1-e^{-\beta(\Delta Q(a,a_m))_\alpha})}{M}, \tag{6.5}$$

where $g(\Delta Q(a,a_m))$ is a function to enable:

$$\begin{cases} 0 \leq p(a|s,\mathbf{r}) \leq 1 \\ \sum_{a\in\mathcal{A}} p(a|s,\mathbf{r}) = 1, \end{cases}$$

and the notation

$$\begin{aligned} \Delta Q(a,a_m) &= Q(s,a) - Q(s,a_m) \\ &= (\mathbf{P}_a - \mathbf{P}_{a_m})V^*. \end{aligned}$$

Remark The notation V^* denotes the optimal value function. The $\alpha \in \{1,2\}$, which denotes $|\Delta Q(a,a_m)|$ and $(\Delta Q(a,a_m))^2$ respectively. The function $g(\Delta Q(a,a_m))$ can be defined as

$$g(\Delta Q(a,a_m)) = \frac{\Delta Q(a,a_m)}{\sum_{a\in\mathcal{A}} Q(s,a)}.$$

Let us denote

$$\delta_1 = \min_{a_m \neq a}(Q(s,a) - Q(s,a_m)),$$
$$\delta_2 = \max_{a_m \neq a}|Q(s,a) - Q(s,a_m)|.$$

Then we have,

$$P\{\sup|p(a|s,\mathbf{r}) - 1| > \epsilon\} \xrightarrow[\delta_1\to\infty]{} 0$$
$$P\{\sup|p(a|s,\mathbf{r}) - \frac{1}{M}| > \epsilon\} \xrightarrow[\delta_2\to 0]{} 0.$$

When δ_1 goes infinity, the Q-function difference between the observation and the next best action is very large. This is the strong evidence for observation data. Consider the sub-optimal policy. The values of Q-functions are close to those of other policies. When δ_2 approaches zero, the differences between Q-functions are vary small. It follows that the likelihood of observing the actions becomes a uniform distribution. If an observed action is considered as optimal and contains more evidence, the value of δ_1 should be large and its likelihood is also large. So this observed action gains more weighting factors during the inference. If an observed action is treated as suboptimal, its Q-function will be close to others', and its likelihood is smaller with less influence on inference.

In summary, we state the problem formulation as:

Proposition 22 *The problem of learning from observations with sub-optimal decisions (we call SubIRL) is formulated as follows,*

$$\begin{aligned}
&\max\ C\sum_{i=1}^{t}\varepsilon_i + \log p(r) + \\
&\sum_{(s,a)\in\mathcal{O}} \log(1+\sum_{m=1}^{M} g(\Delta Q(a,a_m))(1-e^{-\beta[\Delta Q(a,a_m)]^2})) \\
&\text{s.t. } \Delta Q(a,a_m) \geq \varepsilon_i,\ \forall a_m \in \mathcal{A}\setminus a \qquad (6.6)\\
&\mathbf{r}_{\min} < \mathbf{r} < \mathbf{r}_{\max}, \varepsilon_i \leq 0,\ i \in 1,2,\dots,t,\ t=|\mathcal{O}|
\end{aligned}$$

Remark The parameter β and C are constants that can be tuned by employing cross-validation. This method introduces slack variables ε_i, which measure the degree of optimality of the observed behavior. The objective function in Problem 2 is increased by a function that penalizes non-zero ε_i, with the relaxed constraints, which allows the value function of the observed action is less than those of other actions. If we consider the constraints, which are difference of Q-functions with respect to actions at a given state, as a measurement of the degree of belief in rationality, slack variables make the bounded rationality possible. The optimization becomes a trade-off between a large degree of belief and a small error penalty. The objective function not only maximizes the posterior probability of the reward but also minimizes the Bayes risk as a function of $\Delta Q(a, a_m)$.

If the sampled trajectories have some states unobserved, we can apply the approximate observation rule to write Q functions. Or we can take an iterative algorithm to learn the Q function and the reward function at the same time, which has been shown well-behaved and effective of producing a well-defined solution in previous studies [2, 86]. The iterative method is summarized in Algorithm 12 .

This algorithm has the following property.

Proposition 23 *The SubIRL is a generalized optimization model, which can be approximated by Maximum likelihood IRL(MLIRL) using Boltzmann Distribution in [86] and LIRL provided with the strict rationality assumption.*

Proof Based on the rationality assumption, we have $\varepsilon_i > 0$. When the difference between the observed action and the second best action is maximized, we have $g(\Delta Q(a, a_m)) \to 1$. Let $\alpha = 1$. Then the objective function of SubIRL is

$$\max\ C\sum_{i=1}^{t}\varepsilon_i + \sum_{(s,a)\in\mathcal{O}} \log(M + 1 - \sum_{m=1}^{M} e^{-\beta\Delta Q(a,a_m)}).$$

The maximization of the lower bound of above function is equal to

$$\min \sum_{(s,a)\in\mathcal{O}} \log(\sum_{m=1}^{M} e^{-\beta\Delta Q(a,a_m)}).$$

Algorithm 12 SubIRL

```
1:  Input $B = (M \setminus r, \theta(r), \mathcal{O})$, where $\theta(r)$ consists of the Gaussian prior distribution
    for the reward $\mathbf{r}$. $\mathcal{O}$ is composed of the sequential observations.
2:  Choose the initial reward function $\mathbf{r}^0$. Set $c = 0$ and $v = Inf$.
3:  Set the initial optimal transition matrix $\mathbf{P}^0_{a^*}$.
4:  **if** $s \in \mathcal{O}$ **then**
5:      $\mathbf{P}^0_{a^*}(s,:) = \sum_{(s,a_i)\in\mathcal{O}} p(a_i|s)\mathbf{P}_{a_i}(s,:)$
6:  **else**
7:      $\mathbf{P}^0_{a^*}(s,:) = \sum_{a\in\mathcal{A}} \frac{\mathbf{P}_a(s,:)}{M}$
8:  **end if**
9:  **while** $c \leq C_{max}$ or $v \leq \epsilon$ **do**
10:     Solve the optimization Problem 2, where
            $$\Delta Q(a, a_m) = (\mathbf{P}_a - \mathbf{P}_{a_m})(I - \gamma \mathbf{P}^c_{a^*})^{-1}\mathbf{r}.$$
11:     Compute the Bayesian expected reward $Q(s, a|r = r^c)$ and choose the policy
        $\pi^c$ at the unobserved states.
12:     Update $\mathbf{P}^c_{a^*}$ and $\Delta Q(a, a_m)$.
13:     $c = c + 1$
14:     $v = \| \mathbf{r}^c - \mathbf{r}^{c-1} \|$.
15: **end while**
16: Output $\mathbf{r} = \mathbf{r}^c$.
```

Above equation can be further simplified into

$$\min -\sum_{(s,a)\in\mathcal{O}} \beta \cdot Q(s,a) + \sum_{(s,a)\in\mathcal{O}} \log \sum_{m=1}^{M} e^{\beta Q(s,a_m)},$$

which is the likelihood function in MLIRL that uses Boltzmann distribution. In the equation $\sum_{m=1}^{M} e^{\beta Q(s,a_m)}$, let $Q(s,a_1) = max_{m\in\mathcal{M}} Q(s,a_m)$, yielding

$$\sum_{m=1}^{M} e^{\beta Q(s,a_m)} \leq M \cdot e^{Q(s,a_1)}.$$

As we know the Q-function with discounted factor is bounded, written as $Q(s,a) \geq \frac{r_{min}}{1-\gamma}$. We have $e^{Q(s,a)} \geq e^{\frac{r_{min}}{1-\gamma}}$. When $e^{\frac{(M-1)\cdot r_{min}}{1-\gamma}} \geq M$, the equation $\prod_{m=2}^{M} e^{Q(s,a_m)} \geq e^{\frac{(M-1)\cdot r_{min}}{1-\gamma}} \geq M$ is true. Then, it follows that

$$\sum_{m=1}^{M} e^{\beta Q(s,a_m)} \leq \prod_{m=1}^{M} e^{Q(s,a_m)}.$$

The right equation is an upper bound of the left. Another relaxed formulation is proposed as follows.

$$\min -\sum_{(s,a)\in\mathcal{O}} \beta M \cdot Q(s,a) + \sum_{(s,a)\in\mathcal{O}} \sum_{m=1}^{M} \beta Q(s,a_m),$$

which is essentially the similar form of the LIRL in [2] that assumes the deterministic policy and maximizes the difference of Q-functions between the actions observed and unobserved.

6.4 Case Study and Experiments

Now, we present the results from experiments on *GridWorld* and secretary problem. The *GridWorld* problem gives insight into the tasks of goal inference and behavior predictions for the agents that are simulated by forward planning of MDP model. The secretary problem provides an environment in which the decision-makers are not based on MDP. Instead, the agents are imitating human behavior and they are simulated by employing the heuristic decision rules from the experimental study of human behavior in psychology and economics field.

About the behavior recognition including clustering and classification, we conduct the experiments using the following algorithms.

1. Clustering: Kmeans [87].
2. Classification: Support Vector Machine (SVM) [88], K-nearest neighbors (KNN), fisher discriminant analysis (FDA) and logistic regression (LR) [87].

In order to compare the clustering results, the following evaluation methods are employed:

1. *Clustering Accuracy*: The clustering accuracy is used to evaluate the final clustering performance in [89]. Specifically, we have the ground truth labels on the agents observed and know their true reward functions, but these are unknown to the observer. The observer learns the reward functions using IRL algorithms, and then relabel the agents observed using the clustering algorithms. Finally, the evaluator knows the ground truth label and the assigned label. A popular measure the evaluator can use is called clustering accuracy, which is to calculate the classification error rate using

$$Accuracy = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n} \tag{6.7}$$

where n is the number of examples, y_i and c_i denote the true category label and the obtained cluster label of the agent. The function $map(\cdot)$ is a permutation function that maps each cluster label to a category label and the optimal matching is found by the Hungarian algorithm [90]. It is clear that clustering accuracy measures the one-to-one matches between clusters. The greater the clustering accuracy, the better the performance of the clustering algorithm.

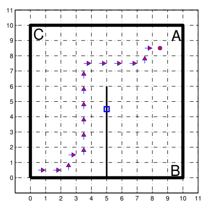
2. *Normalized Mutual Information* (NMI): It is a quantity bounded in $[0, 1]$, equaling 1 when the two clusterings are identical, and 0 when they are independent. If two clusters are independent, they share no information about each other. This criteria has been applied in cluster analysis in [91, 92]. More specifically, for two random variables, NMI is defined as $NMI(X, Y) = I(X, Y)\sqrt{H(X)H(Y)}$. The notation $I(X, Y)$ denotes mutual information between X and Y, telling how much knowing one clustering reduces the uncertainty about the other. The notation $H(X)$ and $H(Y)$ are entropies of X and Y respectively. Given two clustering results, we consider the clustering assignments as distributions of random variables. E.g. we compare the ground truth label and the label assigned by clustering algorithm. The NMI value is estimated as

$$NMI = \frac{\sum_{p=1}^{k}\sum_{q=1}^{k} n_{p,q}\log(\frac{n\cdot n_{p,q}}{n_p \cdot n_q})}{\sqrt{(\sum_{p=1}^{k} n_p \log\frac{n_p}{n})(\sum_{p=1}^{k} n_q \log\frac{n_q}{n})}} \tag{6.8}$$

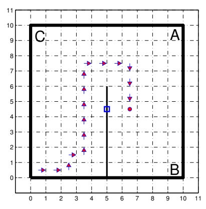
where k is the number of clusters, n_p and n_q refer to the number of agents assigned to the pth and qth cluster respectively, and $n_{p,q}$ represents the number of agents shared by the pth and qth clusters.

To compare the experimental results in the task of classification, we adopt the criteria of classification accuracy (ClassAcc). Specifically, given the ground truth label and the predicted label, we have the number of labels predicted as true positive (TP), false positive (FP), false negative (FN) and true negative (TN). Then, the accuracy is calculated using ClassAcc $= \frac{TP+TN}{TP+TN+FP+FN}$.

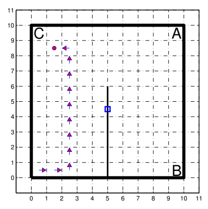
Now, we discuss the experiments one by one.



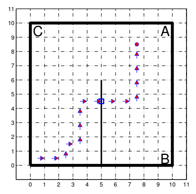
(a) Condition 1-1-A



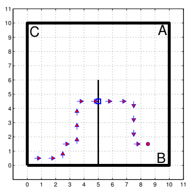
(b) Condition 1-1-B



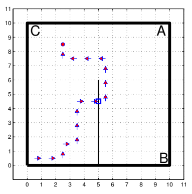
(c) Condition 1-1-C



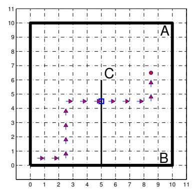
(d) Condition 1-2-A



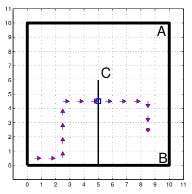
(e) Condition 1-2-B



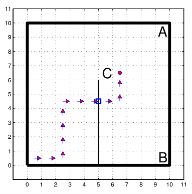
(f) Condition 1-2-C



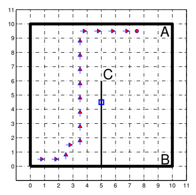
(g) Condition 2-1-A



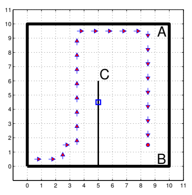
(h) Condition 2-1-B



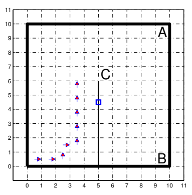
(i) Condition 2-1-C



(j) Condition 2-2-A



(k) Condition 2-2-B



(l) Condition 2-2-C

Figure 6.3: All stimuli from Experiment 1. We vary the goal configuration by changing the location of the goal candidate marked with C, obstacle influence and path trajectory.

6.4.1 Grid world problem

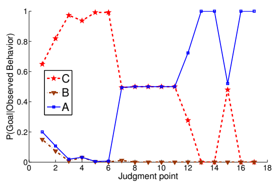
GridWorld problem is used as a benchmark experiment by Ng and Russell in [2], where the agent starts from a square and moves towards a destination square. The agent has five actions to take, moving in four directions or staying put. The outcome of an action succeeds in probability 0.65, fails to move in probability 0.15, or results in random direction in probability 0.2.

The IRL problem of *GridWorld* is that of recovering the reward structure given the observations of the moving behavior. It is obvious that the reward function in this problem well encodes the goal of the agent. The agent will assign fewer or penalty to the states where he/she is not going to. It is easy to build an autonomous agent using MDP to move in the *GridWorld*. Then the observation data is collected by sampling the autonomous agent's behavior. Note that the true reward function is used for simulation of autonomous agent, not known to IRL learner.

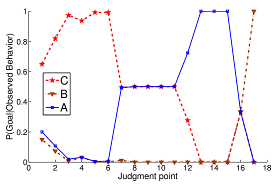
Using a 10×10 *GridWorld* problem, we did three sets of experiments to illustrate how to solve behavior recognition problems by employing IRL algorithms.

On-line goal inference

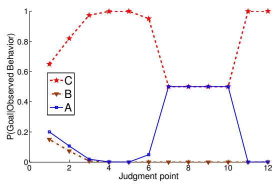
The authors in [93] conduct experiments on human beings' goal inference, and their results prove that the humans' goal inferences in their experimental domains can be explained as a process of inverse planning. This work provides quantitative evidence that Bayesian inference using probabilistic model can explain the human goal inference effectively. Our first experiment, inspired by the experimental design in [93], measures online goal inferences using IRL in response to animated stimuli of agents moving to reach destination in *GridWorld*. The stimuli alters the environmental settings, such as the configuration of the destination, the agent's moving paths and the point at which the IRL learner's judgments are collected. Given a problem, comparing the policy, which is output by forward planning with learned reward, tests whether the observed agent's goal has been captured by the IRL learner.



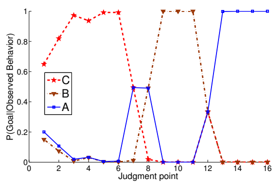
(a) Condition 1-1-A



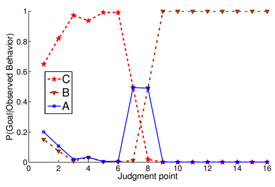
(b) Condition 1-1-B



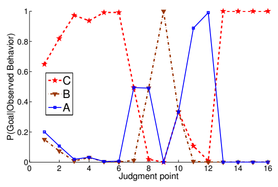
(c) Condition 1-1-C



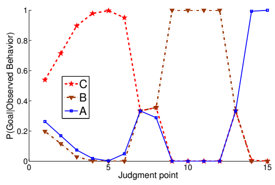
(d) Condition 1-2-A



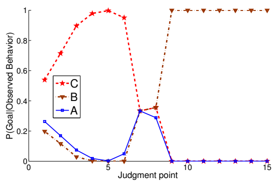
(e) Condition 1-2-B



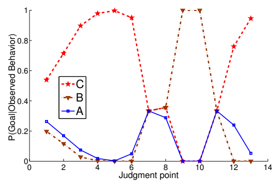
(f) Condition 1-2-C



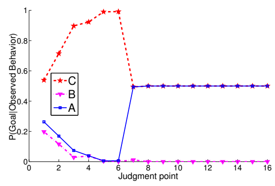
(g) Condition 2-1-A



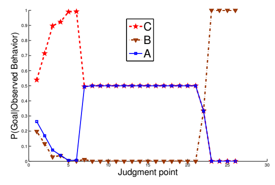
(h) Condition 2-1-B



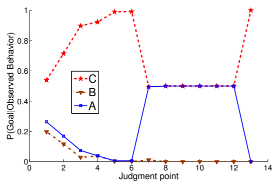
(i) Condition 2-1-C



(j) Condition 2-2-A



(k) Condition 2-2-B



(l) Condition 2-2-C

Figure 6.4: IRL prediction on the goal in Experiment 1.

Figure6.3 illustrates all the settings of Experiment 1[1]. They are different in three

[1]The sub-figure name, called condition x-y-Z, encodes the factors of environmental settings. The

factors: goal configuration, obstacle shape and the agent path. We compare two goal configurations. Only one of the candidate goals changes its location (In Figure6.3, the goal marked with C changes). Every environment shown has a wall obstacle built from the bottom. The line with a gap means a hole in the wall so that the agent can pass through the wall. The simulated agent is moving in the *GrodWorld* with the goal of arriving at one destination location.

The observed behavior is taken from an agent whose path is manually defined with two choices. One choice is which goal the agent is heading toward, and the other choice is whether the agent move around the obstacle or through it. The agent always starts from the left bottom corner grid and follows the path defined. The IRL learner is watching the agent online and infers its goal at the judgment points. In replicate trials, we set the judgment points at every observation point. E.g. we have an observed path with 10 states, then there are 10 judgment points where we infer the goal according to states and actions already seen.

The procedure of our experiment includes simulation of a robot moving in the *GridWorld* with manually defined path (See Figure6.3), simulation of an IRL learner observing the robot's behavior and modeling the solver to infer the robot's goal at the judgment points. Specifically, at each judgment point, a behavior recognition problem $B = (M \setminus r, \mathcal{G}, \mathcal{O})$ is proposed, and a reward function for M is estimated by the IRL learner. With the recovered reward function, the probability distribution of

notation $x \in \{1, 2\}$ denotes the two kinds of goal configurations, $y \in \{1, 2\}$ means that the agent move around the obstacle if $y = 1$ or through the obstacle if $y = 2$, $Z \in \{A, B, C\}$ represents the goal to which the agent is heading.

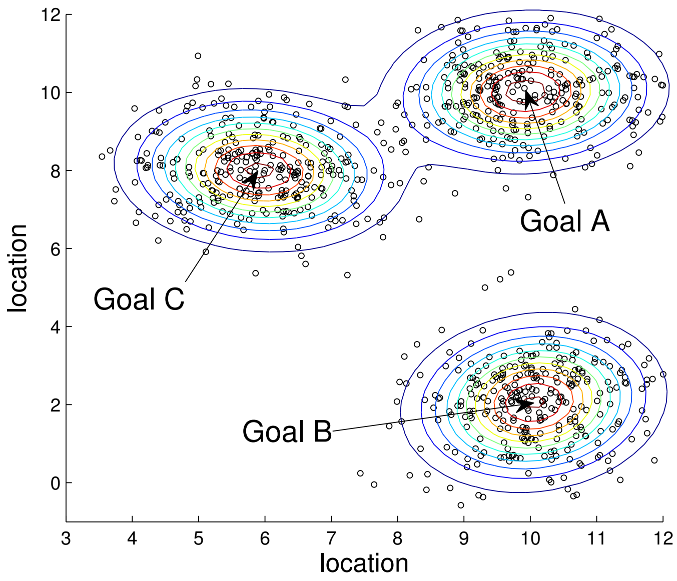


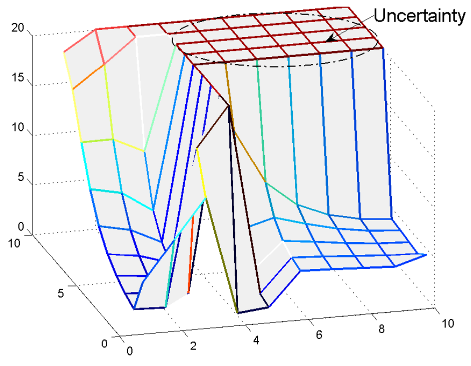
Figure 6.5: A Gaussian prior distribution with three possible goals

the goals is recalculated using

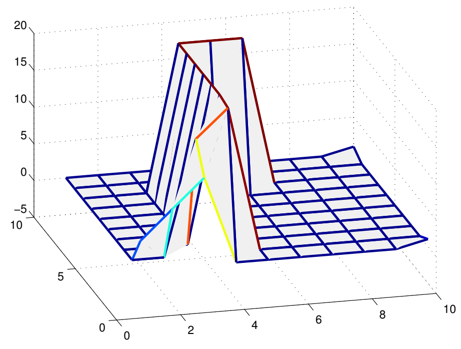
$$p(s_n \text{ is Goal}) = \frac{exp(V(s_n)|\mathbf{r})}{\sum_{s_i \in \mathcal{G}} exp(V(s_i)|\mathbf{r})},$$

where $\mathcal{G}$ describes the possible goal candidates known as a prior, given in the form of a set of states, or equivalently a Gaussian prior distribution. The Gaussian prior can be defined by its mean functions indicating the states as possible goal candidates and the variance function encoding the initial confidence. E.g. the mean vector of multivariate Gaussian prior defines its entries to be small quantities close to zero except the entries representing the goal candidates that have a positive value one (See Figure6.5, where the goal candidates are generated according to a Gaussian distribution).

Figure6.4 displays the results of using IRL to infer the goal from experiment 1. Here, we analyze these results qualitatively. In general, the results prove that IRL method estimates the goal very accurately across most conditions. Since the agent's behavior follows the pre-defined path, the observed behavior may be considered sub-



(a) incomplete path



(b) complete path

Figure 6.6: learned reward functions in condition 2-1-C, given observation of the goal marked with C (b) or not (a).

optimal in MDP settings. Before the agent passes the obstacle, the inverse learning algorithm assigns larger likelihood to the goal marked with C, because that potential goal's position is much more close to the trajectory and the action sequence is in the subset of optimal policy for goal marked with C. Beyond the obstacle, IRL model is able to identify the goal correctly. The distributions of the likelihood plotted in Figure6.4 comply with the study of human being's inference in [93].

Results on this side show that IRL method is also an effective model capturing humans' judgment and providing "human-like" intelligence. It brings new advantages for goal inference, in terms of employing a MDP model to explain and imitate the behavior, evaluating the goal quantitatively with an estimation of the probability distribution and dealing with the goals that change during observation. E.g. in Figure6.3 Condition 1-2-A, the agent moves up in the left area of obstacle, toward the goal marked with C, the learner has assigned the largest probability to C (See Figure6.4 Condition 1-2-A). As soon as the robot changes its direction to the obstacle, the learner turns uncertain about the possible goals. But after the robot passes through the obstacle and continues to east, the goal marked with B seems more probable. Finally, with more observations, the goal marked with A becomes more likely and clear, so the learner increases A's probability and decreases the probability of B and C.

Though most conditions are well solved by the IRL learner, condition 2-1-A and 2-1-C are difficult to solve if the observation is not long enough to include the goal state. We see that the observer has great uncertainty to make decisions, since two possible goal states have the same amount of reward. We are not surprised at this difficulty because given the prior distribution of the reward, the likelihood function of observed behavior does not have enough evidence to differentiate the two possible goal states. We draw the learned reward for these two conditions in Figure6.6: one does not include the goal marked with C and the other contains. Using the reward shown in Figure6.6(a), we are uncertain in the area circled and labeled with "Uncertainty", so the goal is not identified. But the more behavior the learner has observed, the better reward with less uncertainty can be recovered as shown in 6.6 (b).

In summary, Experiment 1 shows that the IRL learner based on goal priors can predict the online goal inference accurately in *GridWorld* problem. IRL models are stable and robust, providing the reward function within a MDP as a principle encoding of the goal. We do not need to design different models for different tasks when the environmental factors have changed. All kinds of observed behavior are studied in MDP settings, even the decision strategies are different and suboptimal. In turn, the MDP models with learned reward can be used to imitate the behavior or represent the decision strategy. We will study this topic in Experiment 2 on *GridWorld* problems.

Learning from decision trajectories

Experiment 2 presents a new task of learning with multiple decision trajectories. Without loss of generally, a 10×10 *GridWorld* problem has been studied. We investigate the behavior recognition problem in terms of clustering and classification. The experiments are conducted according to the steps in Algorithm 13.

Algorithm 13 Experimentation steps

```
1: Input the variables $\mathcal{S}, \mathcal{A}$ and $\mathcal{P}$. Design two base reward functions to represent
   two decision strategies, which are written as $r_1^*$ and $r_2^*$.
2: Simulate agents and sample their behavior.
3: **for** $i = 1 \to 2$ **do**
4:     **for** $j = 1 \to 200$ **do**
5:         Model agent $A_{ij}$ using $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where the reward $r$ is a variation
           of $r_i^*$ contaminated by Gaussian noise.
6:         Sample decision trajectories $\mathcal{O}_{ij}$ from $A_{ij}$.
7:     **end for**
8: **end for**
9: IRL learner has access to the problem $B = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}_{ij})$ for each agent $A_{ij}$, and
   then infers the reward $r_{ij}$ for the agent's decision strategy .
10: Recognize these agents based on the learned $r_{ij}$.
```

The simulated agents in our experiments have hybrid destinations. A small number of short decision trajectories brings more challenges for the existing IRL algorithms, which is of particular interest. Meanwhile, the length of the trajectory may impact the performance seriously. Thus, we evaluate and compare the performance in a scalability of the number of decision trajectories observed in combination with

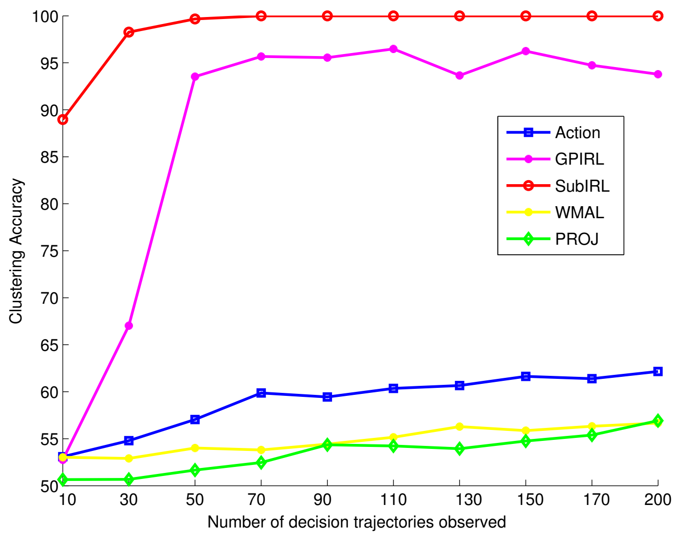


Figure 6.7: Clustering accuracy

the length of the trajectory.

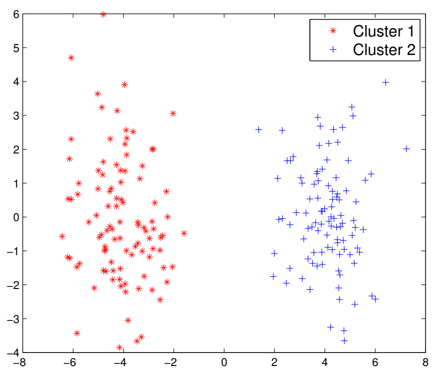
Table 6.1 displays the NMI scores and Figure6.7 shows the clustering accuracy. The length of the trajectory is ten, as we assume the observation is incomplete and the learner does not have sufficient information to infer the goal directly. Results are averaged over 100 replications. The clustering performance is improved by increasing the number of observations. When the number of observations is small, *SubIRL* method achieves high clustering accuracy and NMI scores due to the consideration of suboptimal behavior. The apprenticeship learning algorithms that depend on feature expectations, such as *PROJ* and *WMAL*, is not effective in this problem because the length of the observed decision trajectory is too small to provide a good feature expectation. The feature expectation is required to be good enough to approximate the agent's decision strategy. Otherwise the algorithms may find a divergent solution.

In Figure6.8, using reward functions to represent agents, we project the high-dimensional reward vectors into two dimensional space using principle component

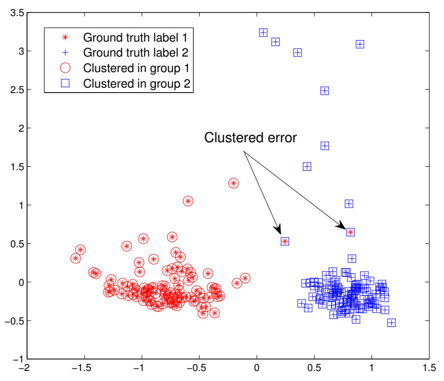
Table 6.1: NMI scores for *GridWorld* problem

H	Action	MWAL	PROJ	GPIRL	SubIRL
10	0.0077	0.0031	0.0068	0.0078	**0.6069**
30	0.0114	0.0060	0.0130	0.0932	**0.8299**
50	0.0177	0.0058	0.0165	0.7751	**0.9198**
70	0.0340	0.0039	0.0243	0.8113	**0.9328**
90	0.0321	0.0062	0.0365	0.8119	**0.9377**
110	0.0361	0.0114	0.0389	0.8123	**0.9306**
130	0.0387	0.0046	0.0388	0.8149	**0.9362**
150	0.0441	0.0035	0.0421	0.8095	**0.9362**
170	0.0434	0.0059	0.0478	0.8149	**0.9327**
200	0.0502	0.0050	0.0498	0.8149	**0.9372**
infinite	0.932	0.2343	0.871	0.931	**0.9391**

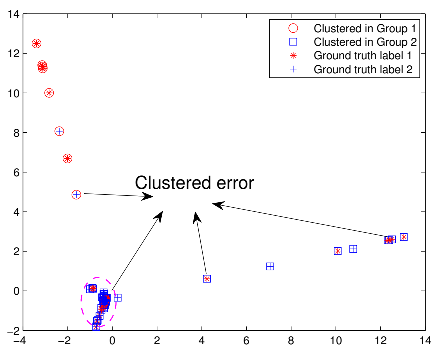
analysis. It is obvious that the reward functions recovered by *GPIRL* within one group have been clustered more closely than the true reward functions. Therefore, the clustering algorithm, k-means, works better in reward space than in action space, and the *GPIRL* algorithm achieves higher accuracy than other algorithms given the small number of observations due to the maximization of the ratio of between-class variance to within-class variance.



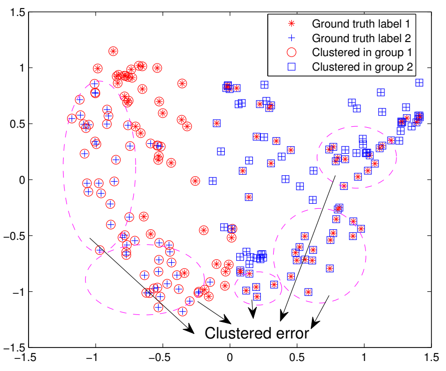
(a) True underlying reward



(b) Recovered reward by GPIRL



(c) Recovered reward by PROJ IRL



(d) Feature points in action space

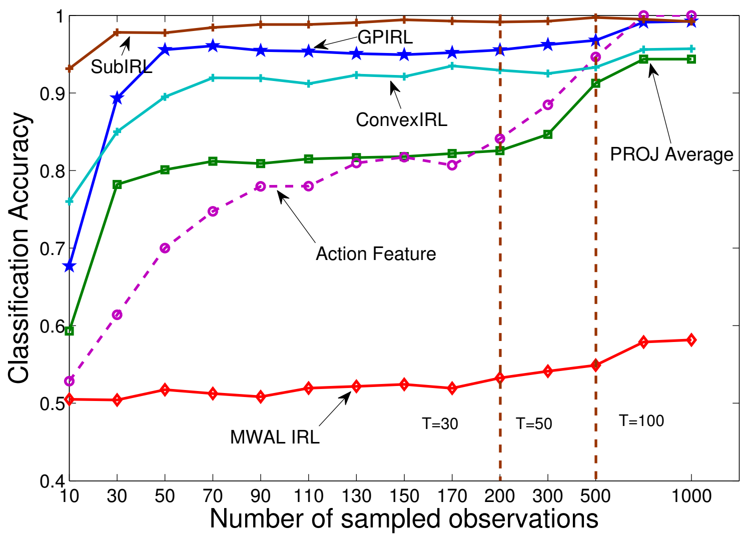
Figure 6.8: Clustering problem: the reward vectors are projected into the two dimensional space by principal component analysis.

The second recognition problem studied here is called classification of agents according to their decision strategy. Figure6.9 and Figure6.10 display the classification accuracy. A binary *GridWorld* classification problem consists of two hundred examples in our experiment. Every example is an agent that has been simulated by modeling a n MDP with a particular reward function. The reward functions of the agents in one class are generated from the same true reward function by adding Gaussian noise. Each true reward function corresponds to a class. Two true reward functions are used to simulate two groups of 100 agents. The classification accuracy for a classification problem is obtained by tenfold cross-validations. Given the size of observation data, we replicate 100 *GridWorld* classification problems to get an average accuracy.

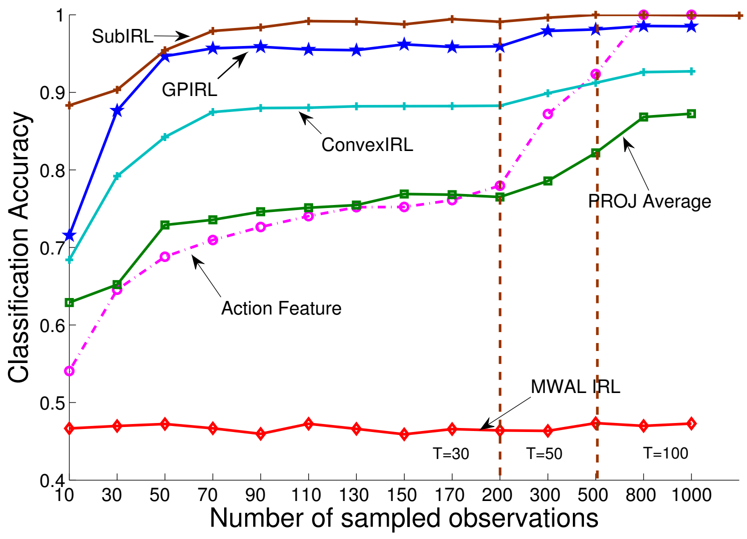
Four popular classifiers (SVM, KNN, FDA and LR) are employed to solve the classification problem. We compare the classification algorithms between using the reward function and using the feature expectation that is computed in state-action space. It indicates that the classifiers based on IRL perform better than the methods using the action features, particularly when the number of observation and the trajectory length are small. In addition, it is safe to say, in *GridWorld* problem, our probabilistic model for IRL provides the estimations of reward functions that are more robust and close to the true reward. Further, we are able to identify the agents based on the reward functions that have been learned for characterizing the decision strategies of these agents.

6.4.2 Secretary problem

GridWorld problem is used for conducting experiments on learning the goal of the autonomous agents. Now let us turn to simulated experiments on human decision



(a) SVM

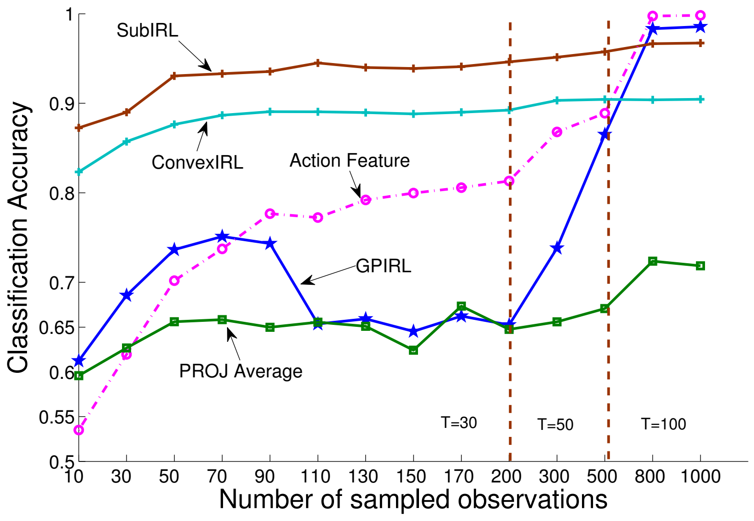


(b) KNN

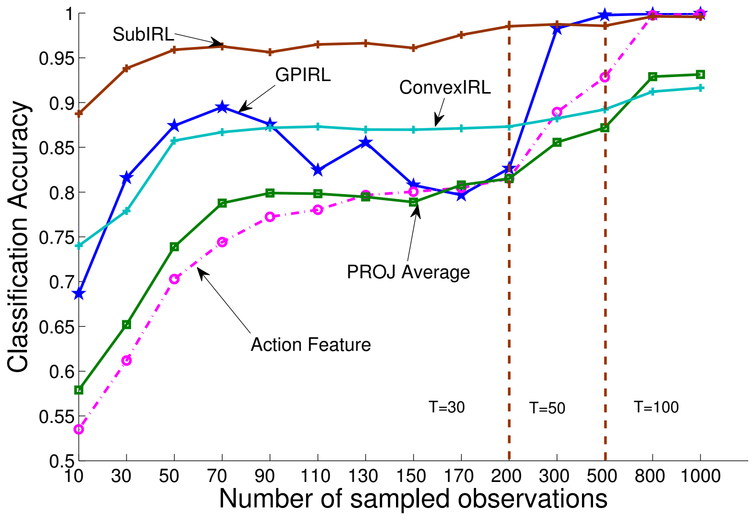
Figure 6.9: Classification accuracy using SVM and KNN. In these figures, the notation T denotes the length of the observed decision trajectory.

making process. We use Secretary problem to study the behavior recognition in human decision making process.

The details on Secretary problem have been introduced in Section 5.4.3 and 6.1.2.



(a) LR



(b) FDA

Figure 6.10: Classification accuracy using LR and FDA.

There heuristic decision rules (cutoff, successive non-candidate counting and candidate counting) will be studied in this section.

Experiment Design

In the following experiments, we simulate human agents by adding different amount of noise on decision rules. For example, if the actual decisions of a human decision maker are assumed to be consistent with a cutoff rule, we are able to simulate the DM using the cutoff rule with a predefined cutoff value. The cutoff value has been added white noise to simulate the bias in human thinking. Note that based on previous study in how humans are solving the secretary problem, we simulate the human behavior using the heuristic decision rules. Since humans using the same heuristic decision rule still have difference in thinking and making decisions, we add random noise to represent different people.

To apply IRL, we model the secretary problem using MDP framework. This idea supports the expectation that even two decision makers are using two different heuristic rules which are not easily differentiated by direct observation, the differences between two heuristic decision rules can still be found in the reward space.

The secretary problem is simulated by following steps: Given the number of applicants N, we generate a sequence of interviews by random permutation (each of $N!$ possibilities being equally likely) and then present the applicants to the observer one at a time. If the applicant is a candidate and the decision criteria is satisfied, the applicant is selected. Otherwise, the observer moves on to the next applicant. For each human observer, we will simulate R times of independent replications of the secretary problem. Each trial followed the same pattern. We will generate H human decision makers in the following experiments. Every human observer takes and follows a heuristic decision rule, while particular random noise is added to the rule's parameter because of the individual thinking bias. We add the Gaussian random noise to the parameters, such as h in cutoff rule and k in non-candidate rule. Same experiments are conducted with two different values of the number of applicants: $N = 40$ or $N = 80$. The simulation process is also summarized in Algorithm 14.

Algorithm 14 Simulation of an agent according to a heuristic rule

```
1: Given a heuristic rule with a parameter $h$, $k$ or $l$.
2: Add random Gaussian noise to the parameter, which is written as $\hat{p}$.
3: Generate new secretary problems and let the agent solve these problems using this
   heuristic rule with its own parameter $\hat{p}$. Save the observed decision trajectories
   into $\mathcal{O}$.
4: Model the secretary problem in MDP settings.
5: Infer the reward function by solving the problem $B = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$.
```

Behavior Prediction

From learning perspective, if a state-action pair has been observed, the apprentice knows the optimal decision at that state. However, it is often impossible to sample observations over all the environmental space, so the practical problems demand the model that is able to infer the optimal actions at unknown states, after being trained with the sampled observations in the decision state-action space.

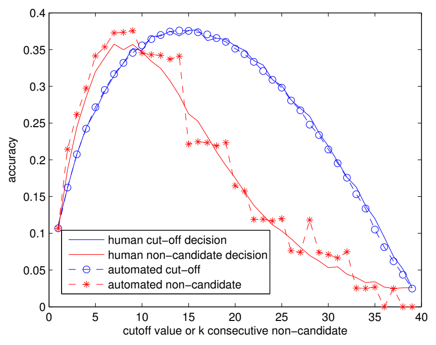
Given a Behavior recognition problem $B = (M \setminus r, \mathcal{G}, \mathcal{O})$, the problem of behavior prediction is a learning process based on MDP to output optimal actions for future problems. The reward function of model M is learned from $\mathcal{O}$. The predicted policy

μ_e satisfies the observation μ_h policy, if

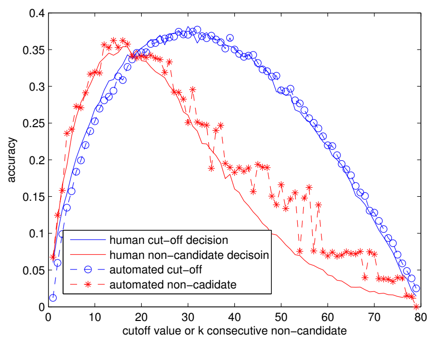
$$\begin{cases} |E(\mu_e) - E(\mu_h)| \leq \epsilon, \ \epsilon \geq 0, \\ \mathcal{G}_e \subset \mathcal{G} \end{cases} \tag{6.9}$$

where $E(\cdot)$ is a performance measurement of the policy.

The definition of behavior prediction expects that the predicted policy performs as well as the expert. Here, we define the measurement function $E(\cdot)$ as the accuracy of finding the best secretary, given the decision rule's parameter. E.g., in 1000 runs of interview problems, if the observer is able to choose the best secretary in 600 runs, the accuracy is 0.6. We conduct experiments on the interview problems with 40 and 80 applicants respectively. Given a heuristic decision rule and its specified parameters, we simulate a set of humans using this decision rule to solve Secretary problems, which is replicated 10000 times in our experiments to provide an average estimate of accuracy. In Figure6.11, we draw the accuracy curves of finding the best secretary and compare the curves of simulated agents and the behavior predicted by IRL. Different decision rules have own accuracy distribution. The recovered distribution reaches its peak at the parameter value the same as that of the simulated agents and the peak value is even higher than the value for humans. E.g. in the problem with 40 applications, the peak is at $k = 8$ applicants for the non-candidate rule, with a probability of correct selection of 0.365. These experimental discoveries in Figure6.11 back up the hypothesis that our estimated reward is close to the true reward with a guarantee of performance. It can be seen from the experiments with $N = 40$ and $N = 80$ that the number of applicants does not affect the learning result, since the shape of the distribution and comparison to the simulated subjects are analogous.

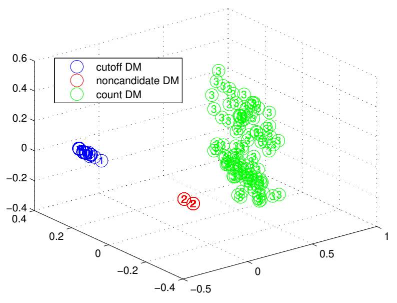


(a) N=40

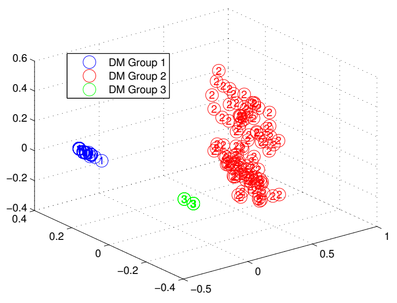


(b) N=80

Figure 6.11: A comparison between the predicted behavior and the true behavior



(a) True reward Distributed



(b) Clustered

Figure 6.12: Visualization of the reward feature vectors in 3 dimensions. In Figure (a) and (c) we assign the true group labels, while clustered group labels are shown in Figure (b) and (d) . The clustering accuracy is 100%. The high dimensional reward vectors are projected into the three dimensional space by principle component analysis.

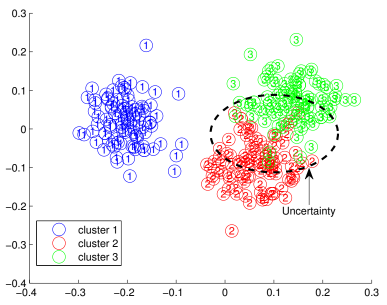
Clustering Analysis

One approach to drawing inferences about behavior recognition is to determine the behavior patterns for the agents even though no explicit feedback is supplied.

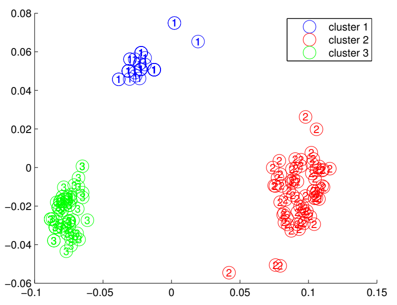
Here, we simulate the humans in the secretary problem on the basis of these three heuristic decision rules. A group of simulated humans (called subjects) are generated from a heuristic decision rule, and the individuals in the group are differentiated by adding Gaussian noise to the parameter of the heuristic decision rule. Every subject has to solve one hundred replicate secretary problems, which provides an averaged performance measurement. We assume that every subject follows exactly his/her decision rule until finishing all the secretary problems. In each problem, the relative rank of interviewed applicants is known to the subject. The subject can select the current applicant, or move on to the next applicant. For example, after interviewing 3 applicants, the relative rank is either $(1, 2, 3)$, $(1, 3, 2)$, $(2, 1, 3)$, $(2, 3, 1)$, $(3, 1, 2)$ or $(3, 2, 1)$. The subject selects the candidate according to the decision rule. IRL learner observes the decision making process of every subject, models the observed behavior on the basis of MDP and infers the reward function for every subject. Finally, k-means algorithm is employed to discover the patterns in the reward space.

An experiment is to group 300 subjects into 3 clusters. These subjects, simulated in this experiment, are equally distributed in three heuristic decision rules with specified parameters. So the number of clusters is known, but the IRL learner does not know the identity of every subject. The learner has the observed behavior for each subject, and aims to put the subject into the group where the subject comes from. Figure6.12 displays the distribution of true reward functions with the true group labels or the clustered labels[2]. In this experiment, each subject takes one of three decision rules with the parameter values that are fixed at $N = 80$ and $h = k = l = 16$

[2]Note the group label we mention here indicates the true group that the corresponding agent comes from, and it is only used for evaluation of the clustering performance. The clustered labels are predicted results from the clustering algorithm. It is different from the labels used in the classification task.



(a)



(b)

Figure 6.13: The distribution of constructed feature points that are assigned with ground truth group label. In Figure (a), the agent is represented by the feature vector constructed in the action space, while in Figure (b), the agent is denoted by its recovered reward function.

respectively. For each decision rule, we simulate 100 subjects. The results obviously prove the effectiveness of our method, as all the subjects have been grouped into the right cluster.

Another experiment is to examine the performance of clustering subjects that are simulated by the same heuristic decision rule but with different parameter values. E.x. all the subjects make decisions according to the cutoff rule but different groups have different cut-off values. The candidate cutoff values in our experiment are in a set $c = (6, 15, 45)$. We simulate three hundred subjects that randomly takes a cutoff value from the possible set c. Then the observed behavior is used for feature construction. To compare the clustering performance between the methods based on action space and reward space, we design an action space that consists of the frequencies of the heuristic decisions rules that have been used in the observations. Because the heuristic decision rules are known, we can program hard classification rules to determine which rule an observed agent has used. This method has been used in the research on Secretary problem and other behavior studies [76, 94]. Given a set of simulated agents and a fixed number of observations from them, we characterize the agents in action space or reward space. Then the performance is evaluated on the k-means algorithm. We show the data distribution in Figure6.13, where every point has been assigned the true group label. It is obvious that the data points in the reward space is easier to recognize, because we can qualitatively state that the data points has lower variance in the same group and higher variance between the different groups. Figure6.13 (a) displays an area marked with "uncertainty", containing the data points mixed from two underlying groups. The quantitative results are shown in Table 6.2, which displays the NMI for k-means clustering. We record the NMI scores as increasing the number of observed decision trajectories, with the expectation that the score will increase too. These quantitative evidence also prove that the cluster patterns can be discovered in reward space with a guarantee of performance that is better than in action space.

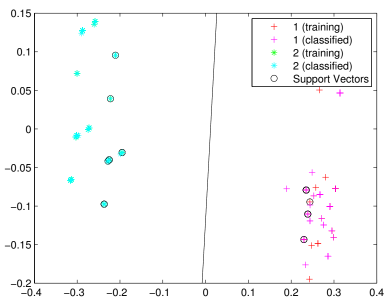
Classification Analysis

The classification analysis in behavior recognition has a similar procedure described in clustering analysis. The main difference between them is that the input subjects have labels that indicate the heuristic decision rule they have used. The subjects are simulated on the basis of three heuristic decision rule by the procedure described in clustering analysis.

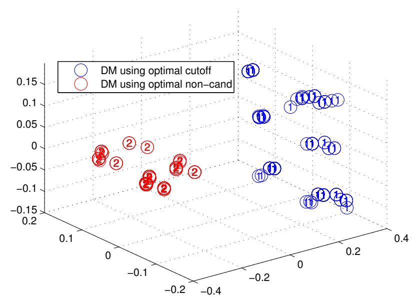
Each classification problem has a number of subjects demonstrating their behavior. These subjects are partitioned into training and testing data set. We use the SVM classifier and employ the cross-validation to estimate the classification accuracy. Assume there are M subjects in an experiment, we model every subject using

Table 6.2: NMI score for clustering the agents using the same heuristic decision rule but with different parameter values

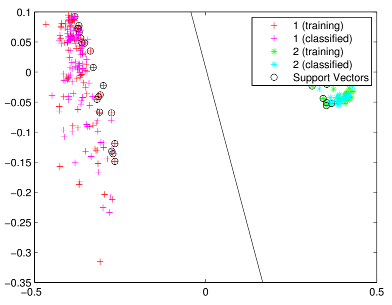
H	Cutoff rule			Noncandidate rule			Count-candidate rule		
	Action	SubIRL	LIRL	Action	SubIRL	LIRL	Action	SubIRL	LIRL
1	0.0557	**0.5497**	0.0000	0.0551	**0.1325**	0.0124	0.0229	**0.2081**	0.1356
11	0.3852	**0.6893**	0.4190	0.2916	**0.7190**	0.5772	0.1844	**0.4974**	0.3646
21	0.6017	**0.7898**	0.4478	0.4305	**0.8179**	0.6234	0.2806	**0.5181**	0.5312
31	0.7654	**0.8483**	0.4506	0.5504	**0.8641**	0.6624	0.4053	**0.6171**	0.5521
41	0.8356	**0.9676**	0.6349	0.5682	**0.9218**	0.6772	0.4524	**0.6533**	0.6123
51	0.8781	**0.9739**	0.6356	0.5894	**0.9423**	0.7121	0.5464	**0.6507**	0.6213
61	0.9102	**0.9913**	0.6580	0.5984	**0.9518**	0.7132	0.5492	**0.6513**	0.6255
71	0.9115	**0.9915**	0.6684	0.6460	**0.9639**	0.7204	0.6024	**0.6512**	0.6342
81	0.9532	**1.0000**	0.6702	0.6541	**0.9721**	0.7325	**0.6708**	0.6563	0.6378
91	0.9707	**1.0000**	0.7316	0.6494	**0.9864**	0.7423	**0.6884**	0.6544	0.6482



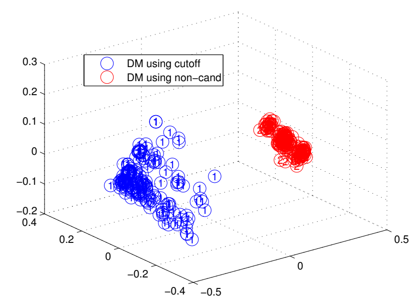
(a) complete observation



(b) complete observation



(c) incomplete observation

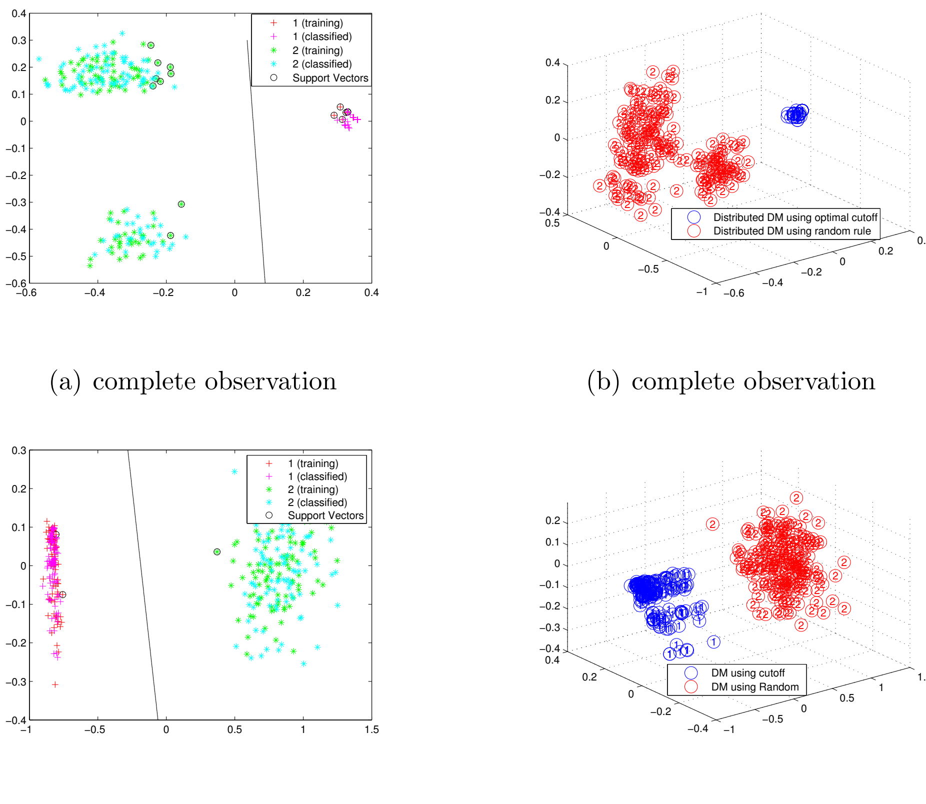


(d) incomplete observation

Figure 6.14: Visualization of a binary classification problem for subjects using cutoff rule and non-candidate rule. The left is the classified feature points projected in 2D space; The right is the classified feature points projected in 3d space. The classification accuracy is 100%. For the complete observation, we use linear IRL in [2] and use PROJ for the incomplete observation.

an MDP. The input data is written as $\{(M_1, y_1), (M_2, y_2), ..., (M_H, y_H)\}$, where M_i denotes a MDP model for i-th subject and y_i denotes a label for the subject's decision rule. Using x_i to denote the reward vector for model M_i, after using IRL, we have $X = \{(x_1, y_1), (x_2, y_2), ..., (x_H, y_H)\}$.

The first problem is to classify the subjects simulated from two decision rules. Assume we have $H = 400$ subjects, half of which are simulated by using cut-off rule



(a) complete observation (b) complete observation

(c) incomplete observation (d) incomplete observation

Figure 6.15: Visualization of a binary classification problem for subjects using cutoff rule and random rules. We use linear IRL in [2] or the problem given complete observation and use PROJ for the problem with incomplete observation.

and the remaining are simulated by using successive non-candidate rule. Figure6.14 displays the distribution of true reward functions that are labeled by the classifier. In the experiment, the cut-off value $h = 30$ and the number of non-candidate $k = 14$. The classification accuracy is 100%.

To give insight into the problem of abnormal detection, we design another experiment to classify the subjects using the heuristic decision rule from the subjects with random behavior. E.g. to hack into the network, some agent may repeat random behavior to detect the security holes. We simulate 200 subjects using the optimal cutoff decision rule $h = 30$ and another 200 agents choosing the secretary randomly. It is formulated as a binary classification problem: the agents using random rules are assigned with negative labels and the subjects using the heuristic rule are given positive labels. An experimental result is shown in Fig6.15.

Another set of experiments are conducted to show the classification performance on the input subjects who are simulated on the basis of one decision rule but with different parameter values. We show the experimental condition in which the observation is incomplete and the reward is recovered by applying PROJ algorithm. Figure6.16 gives the distribution of reward points with predicted labels. It is shown that when the distance between the parameter values increases, the classification accuracy becomes higher up to 100%, whereas when the parameter values are very close, the classification accuracy decreases. With the evidence that the margin between two classes increases as the distance between two parameter values increases, we may say that the reason for misclassification of subjects shown in Figure6.16 (c) and (d) is because the heuristic decision rules are very similar.

6.5 Conclusions

We have proposed the use of IRL to solve the problem of decision strategy recognition. The observed agent does not have to make decisions based on MDP. However, we model their behavior in a MDP environment and assume that the MDP's reward function has encoded the agent's underlying decision strategies.

Numerical experiments on *GridWorld* and the secretary problem suggest that the advantage that IRL enjoys over action space methods is more pronounced when observation are limited and incomplete. Though our experiments are conducted in

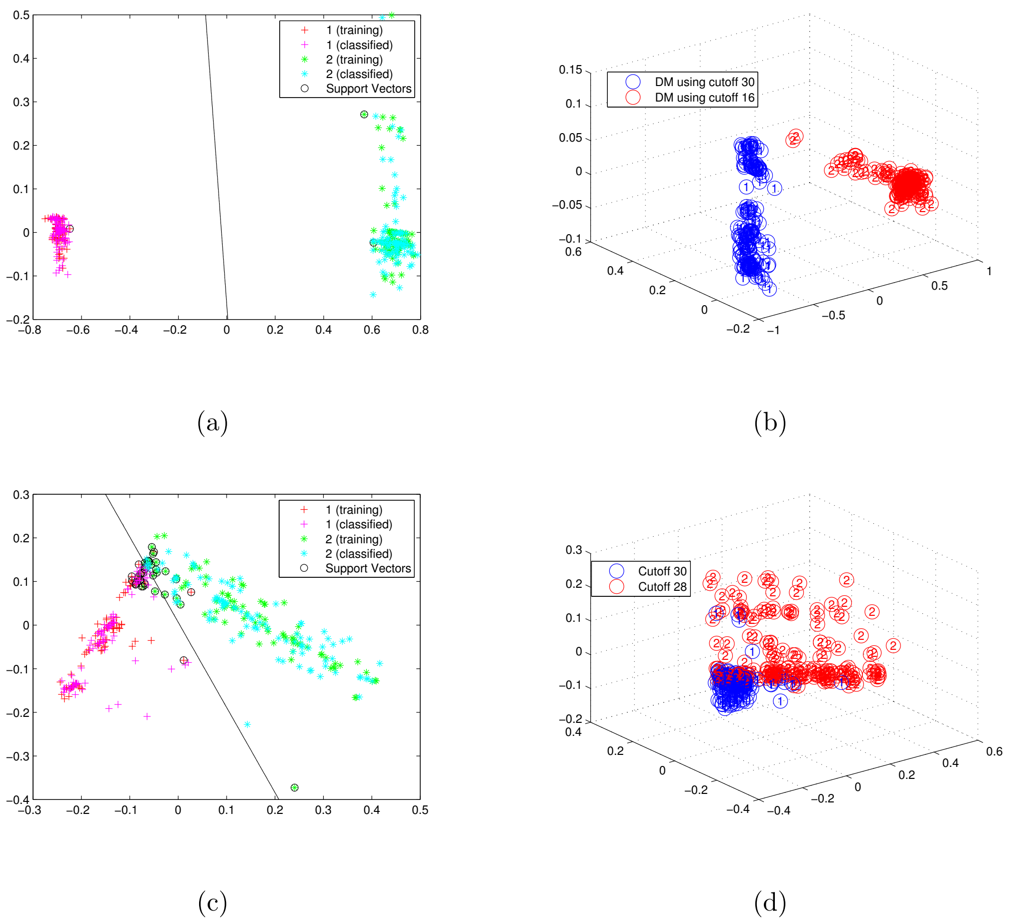


Figure 6.16: Visualization of a binary classification problem for subjects using the cutoff rule with different cutoff parameter values. The observation is incomplete and the reward is estimated by PROJ method.

simulations, we believe that IRL is also able to enhance the capacity of dealing with the real data in practice. More work here is certainly needed. Our expectation is that IRL will be successfully applied to many applications involving sequential decision-making, such as recommendation systems. To investigate the potential for application of our approaches beyond machine problems, we plan to consider more difficult sequential decision problems, such as periodic investment, for which data on human decisions has already been gathered.

Chapter 7

Conclusion

The central objective of this thesis is to develop new goal inference methods that are practical and efficient in terms of the number of observations required for learning, as well as to model the behavior observed in MDP settings for recognition of decision strategies. Based on well-developed Bayesian statistics, decision-making theory, and other machine learning techniques, we propose the GPIRL in Chapter 4. GPIRL is a probabilistic approach to general IRL problems that mimics two important features of biological learners: the ability to generalize and incorporation of uncertainty into the inverse reference process. It is also a non-parametric probabilistic model based on Gaussian process, which provides a closed form of prediction on the new states. The simple representations of unknown quantities using Gaussian process are sufficient to extract valuable information from small-sized datasets and support Bayesian inference with performance guarantees in only a few trials. In experiments with several benchmark problems, GPIRL has shown empirical performance superior to that of several of the prevalent IRL algorithms.

In Chapter 5, we propose an IRL framework that utilizes the philosophy of MM optimization techniques to deal with the question of learning with the help of unobserved states. We borrow the idea of semi-supervised learning, which potentially uses both labeled and unlabeled data to achieve better performance than supervised learning. In IRL settings, the observed behavior is like the labeled data and the learner aims to predict the latent functions on the states that are not observed. The reason that unobserved states provide useful information is because there exists a link between the marginal distribution of state and the distribution of actions conditional on state. At an unobserved state, we do not know which actions the agent may choose, while we consider these optimal actions as hidden variables. The reward function is optimized by maximizing the complete likelihood function. Experimental results support the hypothesis that the recovered reward function will be more accurate and close to the true reward than would be achieve if only observed state information is used.

In Chapter 6, we study the use of IRL as a tool for the recognition of decision agents on the basis of observation of their actions in solving sequential decision problems. We model the observed behavior of the agents in terms of degrees of rationality with respect to optimal forward planning for the MDP. To recognize the agents, we first use IRL to learn reward functions consistent with observed actions and then use these reward functions as the basis for clustering or classification models. Experimental studies with *GridWorld*, a navigation problem, and the *secretary problem*, an optimal stopping problem, suggest reward vectors found from IRL can be a good basis for classifying automated decision rules (e.g., cutoff rule, successive first candidates), even in the presence of action noise and variations in the parameters of the rule. We propose a new Bayesian IRL approach in which the likelihood function can be interpreted in terms of rationality models. Empirical comparisons of our method with several existing IRL algorithms and with direct methods that use feature statistics observed in state-action space suggest it may be superior for recognition problems.

In summary, we explore the IRL in the following main areas: (1) Bayesian probabilistic model based on Gaussian process, (2) Semi-supervised learning in IRL, where we learn from the observation as well as the states unobserved, (4) Goal inference and pattern recognition of the decision strategy behind the observations. Finally, we have a central claim that is that the principle of inverse reinforcement learning creates a MDP model of the decision making that is purposeful and approximately rational. This framework supports inference of the goals of the decision maker and also provides a pattern recognition mechanism for sequential decision making problems.

Recent years have seen an increase in the attention for IRL in a variety of fields, such as cognitive science, psychology, machine learning, financial engineering, computer graphics, and control engineering. However, the theory of IRL, in terms of interaction with the agents, utilization of environmental knowledge, and particularly multiple agents, has not yet seen full development. Additionally, the application of IRL to date has been focused in the area of control engineering. We would like to see more practical applications in goal inference and recognition of the agents based on IRL.

Chapter 8

Appendix

8.1 Proof of Proposition 12

Proof The second order derivative with respect to $\mathbf{r}_{a_m}$ is

$$\begin{aligned}\frac{\partial^2 U}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}^T} &= \mathbf{K}_{a_m}^{-1} + \frac{\partial^2 \sum_{i=1}^{n} \sum_{k=1}^{n_i} - \ln \Phi(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m})}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}^T} \\ &\quad + \sum_{i=1}^{n} \sum_{l=1}^{m_i} (\rho_{a_m}^{il})^T \rho_{a_m}^{il} \end{aligned} \tag{8.1}$$

It is obvious that $\mathbf{K}_{a_m}^{-1}$ and $(\rho_{a_m}^{il})^T \rho_{a_m}^{il}$ are positive definite matrix. If the second part in Eq.8.1 is also positive definite, the minimization of Eq.4.11 is a convex problem. Let $\mathbf{W}$ denote the $n \times n$ matrix for the second part and W_{cd} be the entry at c-th row and d-th column, which is calculated in the following,

$$W_{cd} = \frac{\partial^2 \sum_{i=1}^{n} \sum_{k=1}^{n_i} - \ln \Phi(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m})}{\partial \mathbf{r}_{a_m}(s_c) \partial \mathbf{r}_{a_m}(s_d)} \tag{8.2}$$

and let $z_i^k \triangleq \sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m}$. We have

$$\begin{aligned} &\frac{-\partial ln \Phi(z_i^k)}{\partial \mathbf{r}_{a_m}(s_c)} = -\frac{\rho_{a_m}^{ik}(x_c) N(z_i^k|0,1)}{\sqrt{2}\sigma \Phi(z_i^k)} \\ &\frac{-\partial^2 \ln \Phi(z_i^k)}{\partial \mathbf{r}_{a_m}(s_c) \partial \mathbf{r}_{a_m}(s_d)} = \\ &\frac{\rho_{a_m}^{ik}(s_c) \rho_{a_m}^{ij}(s_d) N(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m}|0,1)}{2\sigma^2 \Phi(z_i^k)} \left[z_i^k + \frac{N(z_i^k|0,1)}{\Phi(z_i^k)} \right] \end{aligned}$$

where let $\omega_{ik} = \frac{N(z_i^k|0,1)}{2\sigma^2\Phi(z_i^k)}\left[z_i^k + \frac{N(z_i^k|0,1)}{\Phi(z_i^k)}\right]$, we have

$$W_{cd} = \begin{cases} \sum_{i=1}^{n}\sum_{k=1}^{n_i}\left[\rho_{a_m}^{ik}(s_c)\right]^2\omega_{ik} \geq 0 \text{ if c=d} \\ \sum_{i=1}^{n}\sum_{k=1}^{n_i}\rho_{a_m}^{ik}(s_c)\rho_{a_m}^{ik}(s_d) = W_{dc} \text{ otherwise} \end{cases} \tag{8.3}$$

Let $y = [y_1, y_2, \cdots, y_n]$ denotes a $n \times 1$ vector. Then

$$\begin{aligned} y^TWy = &\sum_{i=1}^{n}\sum_{k=1}^{n_i} y_1 \sum_{b=1}^{n}\rho_{a_m}^{ik}(s_b)\rho_{a_m}^{ik}(s_1)y_b \\ &+\sum_{i=1}^{n}\sum_{k=1}^{n_i} y_2 \sum_{b=1}^{n}\rho_{a_m}^{ik}(s_b)\rho_{a_m}^{ik}(s_2)y_b \\ &+\cdots+\sum_{i=1}^{n}\sum_{k=1}^{n_i} y_n \sum_{b=1}^{n}\rho_{a_m}^{ik}(s_b)\rho_{a_m}^{ik}(s_n)y_b \\ &=\sum_{i=1}^{n}\sum_{k=1}^{n_i}[\sum_{b=1}^{n} y_b^2[\rho_{a_m}^{ik}(s_b)]^2 \\ &+2\sum_{b=1}^{n}\sum_{b'\neq b} y_b y_{b'}\rho_{a_m}^{ik}(s_b)\rho_{a_m}^{ik}(s_{b'})] \\ &=\sum_{i=1}^{n}\sum_{k=1}^{n_i}\left(\sum_{b=1}^{n} y_b\rho_{a_m}^{ik}(s_b)\right)^2 \geq 0 \end{aligned} \tag{8.4}$$

we prove the matrix W is semi-positive definite. So the Hessian matrix of Eq.4.11 is positive semi-definite on the interior of a convex set. Hence, minimizing Eq.4.11 is a convex programming problem.

8.2 Computation of lower bound function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$

The lower bound function $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ is computed as follows.

$$\begin{aligned}
\mathcal{L}(\mathbf{r}|\mathbf{r}^t) &= \sum_{\mathbf{y}}(\sum_{l=1}^{L}\log(p(\hat{s}_l, a_l|\mathbf{r})) + \sum_{k=1}^{K}\log(p(\check{s}_k, a_{y_k}|\mathbf{r})))p(\mathbf{y}|\mathcal{O}, \mathbf{r}^t) \qquad (8.5)\\
&= \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\left[\sum_{l=1}^{L}\log(p(\hat{s}_l, a_l|\mathbf{r})) + \sum_{k=1}^{K}\log(p(\check{s}_k, a_{y_k}|\mathbf{r}))\right]\prod_{k=1}^{K}p(y_k|\mathcal{O}, \mathbf{r}^t)\\
&= \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\left[\beta\sum_{l=1}^{L}Q(\hat{s}_l, a_l) + \beta\sum_{k=1}^{K}Q(\check{s}_k, a_{y_k})\right]\prod_{k=1}^{K}p(y_k|\mathcal{O}, \mathbf{r}^t)\\
&\quad -\sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\sum_{i=1}^{N}\log(\sum_{j=1}^{M}e^{\beta Q(s_i,a_j)})\prod_{k=1}^{K}p(y_k|\mathcal{O}, \mathbf{r}^t)
\end{aligned}$$

Recall that $Q(s,a) = r(s) + \gamma\mathbf{P}_a(s,:)(I - \gamma\mathbf{P}_{\pi(s)})^{-1}\mathbf{r}$. Given a complete observation z, we know the transition probability matrix $\mathbf{P}_{\pi(s)}$ related to policy $\pi(s)$. Let $\mathbf{H}(\mathbf{y}) = \beta\gamma(I - \gamma\mathbf{P}_{\pi(s)})^{-1}$. After simplifying the Eq.8.6, we have $\mathcal{L}(\mathbf{r}|\mathbf{r}^t) = \mathbf{w}^T\mathbf{r} - \kappa(\mathbf{r})$.

8.3 Proof of Proposition 15

Proof The first and second differentiations of $\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$ are

$$\begin{aligned}
\nabla\mathcal{L}(\mathbf{r}|\mathbf{r}^t) &= \mathbf{w} - \nabla\kappa\\
\nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t) &= -\nabla^2\kappa \qquad (8.6)
\end{aligned}$$

where

$$\nabla\kappa = \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\sum_{i=1}^{N}\frac{\sum_{a=1}^{M}\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}}{\sum_{a=1}^{M}e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}}\prod_{k=1}^{K}p(y_k|s_k, \mathbf{r}^t) \qquad (8.7)$$

Let $\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) = \boldsymbol{\omega}^T = (\omega_1, ..., \omega_N)$. We have some derivative facts as follows,

$$\begin{aligned}
\frac{\partial e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}}{\partial \mathbf{r}} &= [\omega_1, \cdots, \omega_n] e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
\frac{\partial^2 e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}}{\partial \mathbf{r}\partial \mathbf{r}^T} &= \begin{bmatrix} \omega_1^2 & \omega_1\omega_2 & \cdots & \omega_1\omega_N \\ \omega_2\omega_1 & \omega_2^2 & \cdots & \omega_2\omega_N \\ \vdots & \vdots & \cdots & \vdots \\ \omega_N\omega_1 & \omega_N\omega_2 & \cdots & \omega_N^2 \end{bmatrix} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
&= (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}))^T (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}. \qquad (8.8)
\end{aligned}$$

Then we can write the second derivative of κ with respect to $\mathbf{r}$ as

$$\begin{aligned}
&\nabla^2\kappa = \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\sum_{i=1}^{N} \frac{\prod_{k=1}^{K} p(y_k|s_k, \mathbf{r}^t)}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}})^2} \\
&\{\sum_{a=1}^{M} (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}))^T (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
&-[\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}]^2\} \qquad (8.9)
\end{aligned}$$

Suppose a function $\rho(s_i, \mathbf{y}, \mathbf{r})$ being written as

$$\begin{aligned}
&\rho(s_i, \mathbf{y}, \mathbf{r}) = \sum_{a=1}^{M} (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}))^T (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
&-[\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}]^T [\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}] \qquad (8.10) \\
&= \sum_{a=1}^{M} \left\{ (\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}))^T \sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} - [\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}]^T \right\} \\
&\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
&= \mathbf{H}(\mathbf{y})^T [\sum_{a=1}^{M} [\mathbf{P}_a(s_i,:)^T \sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} \\
&-(\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)^T e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}})] \mathbf{P}_a(s_i,:) e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}}] \mathbf{H}(\mathbf{y}) \\
&= \mathbf{H}(\mathbf{y})^T \mathbf{\Pi} \mathbf{H}(\mathbf{y})
\end{aligned}$$

where $\mathbf{\Pi}$ is a $N \times N$ matrix. Let $p_{a=i}(s) = \mathbf{b}^i = (b_1^i, b_2^i, \cdots, b_N^i)^T, i \in \mathcal{M}$, where $\mathbf{b}^i$ denotes the $N \times 1$ probability transition vector of taking action $a_i \in \mathcal{A}$ at state s and $b_j^i, j \in \mathcal{N}$ means the j-th feature value in the vector $\mathbf{b}^i$. Since we are not concerned with s_i during calculating ρ, s_i can be treated as a constant here. To calculate $\mathbf{\Pi}$, we have to know

$$\sum_{a=1}^{M} \mathbf{P}_a(s_i,:)^T e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} = \begin{bmatrix} \sum_{i=1}^{M} b_1^i e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ \sum_{i=1}^{M} b_2^i e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ \vdots \\ \sum_{i=1}^{M} b_N^i e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \end{bmatrix} \tag{8.11}$$

$$\mathbf{P}_a(s_i,:)^T \sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}} = \begin{bmatrix} \sum_{i=1}^{M} b_1^a e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ \sum_{i=1}^{M} b_2^a e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ \vdots \\ \sum_{i=1}^{M} b_N^a e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \end{bmatrix} \tag{8.12}$$

Using above equations, we finally write $\mathbf{\Pi}$ as

$$\mathbf{\Pi} = \sum_{a=1}^{M} e^{b^a \mathbf{H}(\mathbf{y})\mathbf{r}} \tag{8.13}$$

$$\begin{bmatrix} b_1^a \sum_{i=1}^{M} (b_1^a - b_1^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & b_2^a \sum_{i=1}^{M} (b_1^a - b_1^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & \cdots & b_N^a \sum_{i=1}^{M} (b_1^a - b_1^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ b_1^a \sum_{i=1}^{M} (b_2^a - b_2^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & b_2^a \sum_{i=1}^{M} (b_2^a - b_2^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & \cdots & b_N^a \sum_{i=1}^{M} (b_2^a - b_2^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \\ \vdots & \vdots & \cdots & \vdots \\ b_1^a \sum_{i=1}^{M} (b_N^a - b_N^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & b_2^a \sum_{i=1}^{M} (b_N^a - b_N^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} & \cdots & b_N^a \sum_{i=1}^{M} (b_N^a - b_N^i) e^{b^i \mathbf{H}(\mathbf{y})\mathbf{r}} \end{bmatrix}$$

It is obvious that $\mathbf{\Pi}^T = \mathbf{\Pi}$. Denote the element in $\mathbf{\Pi}$ as $[\mathbf{\Pi}]_{uv}$, particularly $[\mathbf{\Pi}]_{uu}$ representing the element along the diagonal. Then we have

$$
\begin{aligned}
\mathbf{\Pi}_{uu} &= b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}}[b_u^1 \sum_{i\neq 1}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^2 \sum_{i\neq 2}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ b_u^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^3 \sum_{i\neq 3}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ \cdots \\
&+ b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^{M-1} e^{b^{M-1}\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^M \sum_{i\neq M}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&= \sum_{j=2}^{M} e^{(b^1+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^1 - b_u^j)^2 + \sum_{j=3}^{M} e^{(b^2+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^2 - b_u^j)^2 \\
&+ \cdots + e^{(b^{M-1}+b^M)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^{M-1} - b_u^M)^2 \\
&= \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} e^{(b^i+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^i - b_u^j)^2 \qquad (8.14)
\end{aligned}
$$

and

$$
\begin{aligned}
[\mathbf{\Pi}]_{uv} &= b_v^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}}[b_u^1 \sum_{i\neq 1}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ b_v^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^2 \sum_{i\neq 2}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ b_v^3 e^{b^3\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^3 \sum_{i\neq 3}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&+ \cdots \\
&+ b_v^M e^{b^M\mathbf{H}(\mathbf{y})\mathbf{r}}[-b_u^1 e^{b^1\mathbf{H}(\mathbf{y})\mathbf{r}} - b_u^2 e^{b^2\mathbf{H}(\mathbf{y})\mathbf{r}} - \cdots - b_u^{M-1} e^{b^{M-1}\mathbf{H}(\mathbf{y})\mathbf{r}} + b_u^M \sum_{i\neq M}^{M} e^{b^i\mathbf{H}(\mathbf{y})\mathbf{r}}] \\
&= \sum_{j=2}^{M} e^{(b^1+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^1 - b_u^j)(b_v^1 - b_v^j) + \sum_{j=3}^{M} e^{(b^2+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^2 - b_u^j)(b_v^2 - b_v^j) \\
&+ \cdots + e^{(b^{M-1}+b^M)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^{M-1} - b_u^M)(b_v^{M-1} - b_v^j) \\
&= \sum_{i=1}^{M-1}\sum_{j=i+1}^{M} e^{(b^i+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}(b_u^i - b_u^j)(b_v^i - b_v^j) \qquad (8.15)
\end{aligned}
$$

Now it can be easily shown that $\mathbf{\Pi}$ is positive definite, by $\forall x = [x_1, x_2, \cdots, x_N]^T \in \Re^N$, $x^T\mathbf{\Pi}x > 0$.

$$
x^T\mathbf{\Pi}x = \sum_{i=1}^{M-1}\sum_{j=i+1}^{M} e^{(b^i+b^j)\mathbf{H}(\mathbf{y})\mathbf{r}}[\sum_{k=1}^{N} x_k(b_k^i - b_k^j)]^2 > 0 \qquad (8.16)
$$

Replace it in $\nabla^2\kappa$ to obtain

$$
\nabla^2\kappa = \sum_{y_1=1}^{M}\sum_{y_2=1}^{M} \dots \sum_{y_K=1}^{M}\sum_{i=1}^{N}\prod_{k=1}^{K} p(y_j|s_k, \mathbf{r}^t)\mathbf{H}(\mathbf{y})^T \frac{\mathbf{\Pi}}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}})^2}\mathbf{H}(\mathbf{y}) \qquad (8.17)
$$

So $\nabla^2\kappa$ is positive definite and $\nabla^2\Phi$ is negative definite.

8.4 Proof of Theory 16

Proof Consider the learning process as follows:

1. Upteacherte the the value functions in the rational strategy. The value functions are upteacherted according to Bellman optimality.

2. Select the policy using soft strategy with the Boltzmann distribution in Eq. 5.3.

At t-step, we have the estimation $\mathbf{r}^t = \arg\max_{\mathbf{r}} \mathcal{L}(\mathbf{r}|\mathbf{r}^t)$. Then, in the next step, we have $a_{\mathbf{y}^*(s)} = \arg\max_a Q(s,a)$, where $\mathbf{y}^*$ is a vector containing the optimal policy at the unobserved states. Consider a feasible region Ω_r

$$\begin{aligned}\Omega &= \left\{\mathbf{r} : Q(s_i, a_i) \geq \max_{a \in \mathcal{A} \setminus a_i} Q(s_i, a), \forall (s_i, a_i) \in \mathcal{O} \cup \left\{\check{\mathcal{S}}, \mathbf{y}^*\right\}\right\} \\ &\cup \quad \{\mathbf{r} : \|\mathbf{r}\| \leq R_{max}\} \end{aligned} \tag{8.18}$$

$\forall \mathbf{r} \in \Omega_r$, $\mathbf{H}(\mathbf{y}^*)$ is the inverse matrix that is computed by using $\mathbf{y}^*$. So, at the $t+1$ step, we have matrix

$$\begin{aligned}\mathbf{L}_1 &= -\sum_{i=1}^{N} \mathbf{H}(\mathbf{y})^T \frac{\mathbf{\Pi}(\mathbf{y})}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y})\mathbf{r}})^2} \mathbf{H}(\mathbf{y}) \\ \mathbf{L}_2 &= -\sum_{i=1}^{N} \mathbf{H}(\mathbf{y}^*)^T \frac{\mathbf{\Pi}(\mathbf{y}^*)}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\mathbf{r}})^2} \mathbf{H}(\mathbf{y}^*)\end{aligned} \tag{8.19}$$

To obtain $\mathbf{e}^T\mathbf{H}(\mathbf{y}^*)\mathbf{r} \leq \mathbf{e}^T\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}$, we get $\hat{\mathbf{r}}$ by solving the following problem.

$$\begin{aligned}&\max_{\mathbf{r}} \mathbf{e}^T\mathbf{H}(\mathbf{y}^*)\mathbf{r} \\ &\quad s.t.\ \mathbf{r} \in \Omega_r\end{aligned} \tag{8.20}$$

In $\mathbf{L}_2$, let

$$\hat{\mathbf{\Pi}}(\mathbf{r}) = \frac{\mathbf{\Pi}(\mathbf{y}^*)}{(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\mathbf{r}})^2} \tag{8.21}$$

Consider for arbitrary vector $\mathbf{x} \in \Re^N$ $\mathbf{x}^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})\mathbf{x} \geq \mathbf{x}^T\hat{\mathbf{\Pi}}(\mathbf{r})\mathbf{x}$, which has been proved in Appendix 8.5. Therefore, setting $\mathbf{B} = -\sum_{i=1}^{N} \mathbf{H}(\mathbf{y}^*)^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})\mathbf{H}(\mathbf{y}^*)$, we find that $\mathbf{B} \leq \mathbf{L}_2$. Using similar techniques in Appendix 8.5, we are able to show $\mathbf{L}_2 \leq \mathbf{L}_1$ when $\mathbf{r} = \mathbf{r}^t$. Thus $\forall \mathbf{r} \in \Omega_r,\ \mathbf{B} \leq \nabla^2\mathcal{L}(\mathbf{r}|\mathbf{r}^t)$.

8.5 Proof for the inequality in Theorem 16

Proof Here we prove the claim that $\hat{\mathbf{\Pi}}(\hat{\mathbf{r}}) \geq \hat{\mathbf{\Pi}}(\mathbf{r})$ in Theorem 16. Consider for arbitrary vector $\mathbf{x} \in \Re^N$ the form:

$$\mathbf{x}^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})\mathbf{x} = \frac{\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} e^{(b^i+b^j)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}}[\sum_{k=1}^{N} x_k(b_k^i - b_k^j)]^2}{(\sum_{j=1}^{M} e^{\mathbf{P}_{a_j}(s_i,:)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}})^2}$$

where

$$(\sum_{a=1}^{M} e^{\mathbf{P}_a(s_i,:)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}})^2 = \sum_{m=1}^{M} e^{2\mathbf{P}_{a_m}(s_i,:)\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}} + 2\sum_{m_1=1}^{M-1}\sum_{m_2=m_1+1}^{M} e^{(\mathbf{P}_{a_{m_1}}(s_i,:)+\mathbf{P}_{a_{m_2}}(s_i,:))\mathbf{H}(\mathbf{y}^*)\hat{\mathbf{r}}} \tag{8.22}$$

Denote the first part in Eq. 8.22 as $\tau(\hat{\mathbf{r}})$ and the second part as $2\upsilon(\hat{\mathbf{r}})$. Let $\Delta_\upsilon = \upsilon(\hat{\mathbf{r}}) - \upsilon(\mathbf{r})$. Then $\Delta_\upsilon \geq 0$ is guaranteed in the previous computation. We can write

$$\mathbf{x}^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})\mathbf{x} = \frac{C_1\upsilon(\mathbf{r}) + C_1\Delta_\upsilon}{\tau(\mathbf{r}) + 2\upsilon(\mathbf{r}) + (\frac{2C_2}{M-1}+2)\Delta_\upsilon} \tag{8.23}$$

where C_1 and C_2 are two constant factors, and it is easily shown that $C_1 \geq 0,\ 0 \leq C_2 \leq 1$. Now making the difference between $x^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})x$ and $x^T\hat{\mathbf{\Pi}}(\mathbf{r})x$ leads to

$$\mathbf{x}^T\hat{\mathbf{\Pi}}(\hat{\mathbf{r}})\mathbf{x} - \mathbf{x}^T\hat{\mathbf{\Pi}}(\mathbf{r})\mathbf{x} = \frac{C_1\Delta_\upsilon\left(\tau(\mathbf{r}) + 2\upsilon(\mathbf{r}) - (\frac{2C_2}{M-1}+2)\upsilon(\mathbf{r})\right)}{\left(\tau(\mathbf{r}) + 2\upsilon(\mathbf{r}) + (\frac{2C_2}{M-1}+2)\Delta_\upsilon\right)C_1\upsilon(\mathbf{r})} \tag{8.24}$$

Since the denominator in above equation is greater than zero, we are only concerned with

$$\begin{aligned} &\tau(\mathbf{r}) + 2\upsilon(\mathbf{r}) - (\frac{2C_2}{M-1}+2)\upsilon(\mathbf{r}) \\ &= \tau(\mathbf{r}) - C_2\frac{2}{M-1}\upsilon(\mathbf{r}) \geq \tau(\mathbf{r}) - \frac{2}{M-1}\upsilon(\mathbf{r}) \geq 0 \end{aligned} \tag{8.25}$$

To prove Eq. 8.25, denoting

$$\begin{aligned} \alpha_i &= e^{\mathbf{P}_{a_i}(s_i,:)\mathbf{H}(\mathbf{y}^*)}, i \in \mathcal{M} \\ \boldsymbol{\alpha} &= (\underbrace{\alpha_1, \alpha_1, \ldots, \alpha_1}_{M-1}, \underbrace{\alpha_2, \alpha_2, \ldots, \alpha_2}_{M-2}, \ldots, \underbrace{\alpha_{M-1}}_{1})^T \\ \boldsymbol{\beta} &= (\underbrace{\alpha_2, \alpha_3, \ldots, \alpha_M}_{M-1}, \underbrace{\alpha_3, \alpha_4, \ldots, \alpha_M}_{M-2}, \ldots, \underbrace{\alpha_M}_{1})^T \end{aligned}$$

, we have

$$\upsilon(\mathbf{r}) = \boldsymbol{\alpha}^T\boldsymbol{\beta} \leq \sqrt{\boldsymbol{\alpha}^T\boldsymbol{\alpha}}\sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}$$

This inequality can be easily proved by Cauchy's inequality

$$\begin{aligned}
&\boldsymbol{\alpha}^T\boldsymbol{\beta} \leq \sqrt{(M-1)\alpha_1^2 + (M-2)\alpha_2^2 + \cdots + \alpha_{M-1}^2} \\
&\sqrt{\alpha_2^2 + \alpha_3^2 + \cdots + (M-2)\alpha_{M-1}^2 + (M-1)\alpha_M^2} \\
&\leq \frac{M-1}{2}\sum_{i=1}^{M}\alpha_M^2 \leq \frac{M-1}{2}\tau(\mathbf{r}) \qquad (8.26)
\end{aligned}$$

, where the second inequality follows from $x_1x_2 \leq x_1^2 + x_2^2, \forall x_1, x_2 \in \Re$. Indeed, Eq. 8.26 implies for this inequality

$$\tau(\mathbf{r}) \geq \frac{2}{M-1}\upsilon(\mathbf{r})$$

from which the inequality $\mathbf{x}^T\hat{\mathbf{\Pi}}(\hat{r})\mathbf{x} \geq \mathbf{x}^T\hat{\mathbf{\Pi}}(r)\mathbf{x}$ follows.

8.6 Proof of Lemma 18

We have

$$\begin{aligned}
\mathcal{L}(\mathbf{r}|\mathbf{r}^t) &= E\left\{\log p(z|\mathbf{r})|\mathbf{y}, \mathbf{r}^t\right\}, \\
\mathcal{H}(\mathbf{r}|\mathbf{r}^t) &= E\left\{\log p(z|\mathbf{Y}, \mathbf{r})|\mathbf{y}, \mathbf{r}^t\right\}.
\end{aligned}$$

First, the Jensen's inqeuality ensures

$$\begin{aligned}
&\mathcal{H}(\mathbf{r}|\mathbf{r}^t) - \mathcal{H}(\mathbf{r}^t|\mathbf{r}^t) \\
&= \int p(z|\mathbf{y}, \mathbf{r}^t) \log \frac{p(z|\mathbf{y}, \mathbf{r})}{p(z|\mathbf{y}, \mathbf{r}^t)} dz \\
&\leq \log \int p(z|\mathbf{y}, \mathbf{r}) dz = 0.
\end{aligned}$$

It is obvious that when $\mathbf{r} = \mathbf{r}^t$, $\mathcal{H}(\mathbf{r}|\mathbf{r}^t) = \mathcal{H}(\mathbf{r}^t|\mathbf{r}^t)$. In the majorization step, we have

$$\mathcal{L}(\mathbf{r}^{t+1}|\mathbf{r}^t) - \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t) = \nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) + \frac{1}{2}(\mathbf{r}^{t+1} - \mathbf{r}^t)^T\nabla^2\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) + \cdots .$$

Due to maximization step, we have

$$\begin{aligned} g(\mathbf{r}^{t+1}|\mathbf{r}^t) - g(\mathbf{r}^t|\mathbf{r}^t) &= \nabla\mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) + \frac{1}{2}(\mathbf{r}^{t+1} - \mathbf{r}^t)^T\mathbf{B}(\mathbf{r}^t|\mathbf{r}^t)(\mathbf{r}^{t+1} - \mathbf{r}^t) \\ &\leq \mathcal{L}(\mathbf{r}^{t+1}|\mathbf{r}^t) - \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t). \end{aligned}$$

By $g(\mathbf{r}^{t+1}|\mathbf{r}^t) \geq g(\mathbf{r}^t|\mathbf{r}^t)$, we prove $\mathcal{L}(\mathbf{r}^{t+1}|\mathbf{r}^t) \geq \mathcal{L}(\mathbf{r}^t|\mathbf{r}^t)$.

8.7 Global Convergence Theory

Theorem 24 *Let the sequence $\{x_k\}_{k=0}^{\infty}$ be generated by $x^{k+1} \in F(x^k)$, where F is a point-to-set map on $\mathbf{X}$ Let a solution set $\Gamma \subset \mathbf{X}$ be given. The limits points of $\{x_k\}$ are in the solution set and a continuous function $l(x)$ converges monotonically to $l(x^*)$ for some points $x^* \in \Gamma$, if*

1. *all points x^k are in a compact set;*

2. *M is closed over the complement of Γ;*

3. *The function $l(y) > l(x)$, if $x \notin \Gamma$, $\forall y \in F(x)$, and if $x \in \Gamma, l(y) \geq l(x), \forall y \in F(x)$.*

8.8 Proof of Proposition 22

Proof Without loss of generality, assume that at state $s^1 = s^2 = s_n \in \mathcal{A}$, we have observed action a_1 and a_2. Then $\hat{U}(s_n) = \{a_1, a_2\}$. Based on Bellman optimality, we have $\forall a \in \mathcal{A} \setminus \hat{U}(s_n)$,

$$Q(s_n, a_1) > Q(s_n, a);\ Q(s_n, a_2) > Q(s_n, a) \tag{8.27}$$

And $\forall a \in \mathcal{A}$, the Q-function is calculated by

$$Q(s_n, a) = r(s_n) + \gamma \mathbf{P}_a(s_n, :)V^\pi \tag{8.28}$$

where the value function is written as

$$V^\pi(s_n) = r(s_n) + \gamma \sum_{a^* \in \hat{U}(s_n)} \frac{1}{|\hat{U}(s_n)|} \mathbf{P}_{a^*}(s_n, :)V^\pi \tag{8.29}$$

So given observation (s_n, a_1), we have

$$V^\pi = r + \gamma \mathbf{P}_{a^*} V^\pi \Rightarrow V^\pi = (\mathbf{I}_n - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{r}$$

where $\mathbf{P}_{a^*}$ is the transition matrix, whose n-th row written as $\mathbf{P}_{a^*}(s_n, :) = \sum_{a^* \in \hat{U}(s_n)} \frac{\mathbf{P}_{a^*}(s_n,:)}{|\hat{U}(s_n)|}$ if s_n is observed, otherwise $\mathbf{P}_{a^*}(s_n, :) = \sum_{m=1}^{M} \frac{\mathbf{P}_{a_m}(s_n,:)}{M}$.

Then $\forall a \in \mathcal{A} \setminus \hat{U}(s_n)$, we have

$$(\mathbf{P}_{a_1}(s_n, :) - \mathbf{P}_a(s_n, :))(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{r} \geq \varepsilon_1$$

$$(\mathbf{P}_{a_2}(s_n, :) - \mathbf{P}_a(s_n, :))(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{r} \geq \varepsilon_2$$

where ε_1 and ε_2 are slack variables. Generally, $\forall a^* \in \hat{U}(s_n)$, its constraint is written as that in Eq. 6.7.

Bibliography

[1] Marc Peter Deisenroth, Carl Edward Rasmussen, and Jan Peters, “Gaussian process dynamic programming,” *Neurocomputing*, vol. 72, no. 7-9, pp. 1508–1524, 2009.

[2] Andrew Y. Ng and Stuart Russell, “Algorithms for inverse reinforcement learning,” in *Proc. 17th International Conf. on Machine Learning*. 2000, pp. 663–670, Morgan Kaufmann.

[3] John L. Pollock, *Thinking about Acting*, Oxford University Press, 2006.

[4] Edwards, “The theory of decision making,” *Psychological Bulletin*, vol. 51, no. 4, pp. 380–417, 1954.

[5] Simon H.A., *Models of Man: Social and Rational*, Wiley, 1957.

[6] Elizabeth F. Chua, Erin Rand-giovannetti, Daniel L. Schacter, Marilyn S. Albert, and Reisa A. Sperling, “Dissociating confidence and accuracy: Functional magnetic resonance imaging shows origins of the subjective memory experience,” *J. Cognitive Neuroscience*, vol. 16, no. 7, pp. 1131–1142, 2004.

[7] Scott Plous, *The psychology of judgment and decision making*, McGraw-Hill, 1993.

[8] A. C. Courville, N. D. Daw, and D. S. Touretzky, “Bayesian theories of conditioning in a changing world,” *Trends in Cognitive Sciences*, vol. 10, pp. 294–300, 2006.

[9] J. B. Tenenbaum, T.L. Griffiths, and C. Kemp, “Theory-based bayesian models of inductive learning and reasoning,” *Trends in Cognitive Sciences*, vol. 10, pp. 309–318, 2006.

[10] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum, “Action understanding as inverse planning,” *Cognition*, 2009.

[11] Ullman Tomer David, Tenenbaum Joshua B., Baker Christopher Lawrence, Macindoe Owen, Evans Owain Rhys, and Goodman Noah Daniel, “Help or hinder: Bayesian models of social goal inference,” in *Advances in Neural Information Processing System*, 2009.

[12] Stuart Russell and Norvig Peter, *Artificial Intelligence: A Modern Approach (3rd Edition)*, Prentice Hall, 3 edition, Dec. 2009.

[13] A. Rosenblueth, N.Wiener, and J. Bigelow, “Behavior, purpose and teleology,” *Philosophy of Science*, vol. 10, pp. 18–24, 1943.

[14] D. G. Premack and G. Woodruff, “Does the chimpanzee have a theory of mind?,” *Behavioral and Brain Sciences*, vol. 1, pp. 515–526, 1978.

[15] L. Bergen, O.R. Evans, and J.B. Tenenbaum, “Learning structured preferences,” in *the Thirty-second Annual Conference of the Cognitive Science Society*, 2010.

[16] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, vol. 15 of *Studies in Applied Mathematics*, SIAM, Philadelphia, PA, June 1994.

[17] Ralph L. Keeney and Howard Raiffa, *Decisions with multiple objectives: preferences and value tradeoffs*, Cambridge Univsersity Press, 1993.

[18] John Rust, "Estimation of dynamic structure models: Problems and prospects part i: Discrete markov decision processes," in *Proc. 6th World Congress of the Econometric Society*, 1994.

[19] T.J. Sargent, "Estimation of dynamic labor demand schedules under rational expectations," *Journal of Political Economy*, vol. 86, pp. 1009–1044, 1978.

[20] Stuart Russell, "Learning agents for uncertain environments (extended abstract)," in *11th Annual Conference on Computational Learning Theory*, 1998, pp. 101–103.

[21] Peter Abbeel and Andrew Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st International Conf. on Machine learning.* 2004, p. 1, ACM.

[22] Umar Syed, Michael Bowling, and Robert E. Schapire, "Apprenticeship learning using linear programming," in *Proc. 25th international Conf. on Machine learning.* 2008, pp. 1032–1039, ACM.

[23] Umar Syed and Robert E. Schapire, "A game-theoretic approach to apprenticeship learning," in *Advances in Neural Information Processing Systems.* 2008, pp. 1449–1456, MIT Press.

[24] Gergely Neu and Csaba Szepesvari, "Apprenticeship learning using inverse reinforcement learning and gradient methods," in *Proc. Uncertainty in Artificial Intelligence*, 2007.

[25] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008.

[26] Ramachandran Deepak and Amir Eyal, "Bayesian inverse reinforcement learning," in *Proc. 20th International Joint Conf. on Artificial Intelligence*, 2007.

[27] George Casella and Roger Berger, *Statistical Inference*, Duxbury Resource Center, June 2001.

[28] Peter W. Zehna, "Invariance of maximum likelihood estimators," *Annals of Mathematical Statistics*, vol. 37, pp. 744, 1966.

[29] Nabendu Pal and J. Calvin Berry, "On invariance of maximum likelihood estimation," *The American Statistics*, vol. 46, pp. 209–212, 1992.

[30] S.French, *Decision Theory*, Ellis Horwood, Chichester, West Sussex, England, 1988.

[31] Barbera Salvador, Hammond Peter, and Seidl Christian, *Handbook of Utility Theory*, Springer, 1999.

[32] von Neumann John and Morgenstern Oskar, *Theory of Games and Economic Behavior (Commemorative Edition) (Princeton Classic Editions)*, Princeton University Press, May 2004.

[33] Bellman R., *Dynamic programming*, Princeton University Press, 1957.

[34] Puterman Martin L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley-Interscience, April 1994.

[35] James O. Berger, *Statistical decision theory and Bayesian analysis*, Springer series in statistics. Springer, New York, NY [u.a.], 2. ed edition, 1985.

[36] R. E. Schapire Y. Freund, "Adaptive game playing using multiplicative weights," *Games and Economic Behavior*, vol. 29, pp. 79–103, 1999.

[37] Carl Edward Rasmussen and Christopher K.I.Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[38] Chu Wei and Ghahramani Zoubin, "Preference learning with gaussian processes," in *Proc. 22th Iinternational Conf. on Machine learning.* 2005, pp. 137–144, ACM.

[39] Matthias Seeger, "Gaussian processes for machine learning," *International Journal of Neural Systems*, vol. 14, pp. 2004, 2004.

[40] Takeyuki Hida and Masuyuki Hitsuda, *Gaussian Processes*, American Mathematical Society, 1993.

[41] A.N.Kolmogorov, *Foundations of the Theory of Probability*, AMS Chelsea, 2nd edition, 1956.

[42] J. Fürnkranz and E. Hüllermeier, "Preference learning," in *Künstliche Intelligenz*, 2005.

[43] Fabio Aiolli and Alessandro Sperduti, "Learning preferences for multiclass problems," in *Advances in Neural Information Processing Systems 17.* 2004, pp. 17–24, MIT Press.

[44] Ofer Dekel, Christopher D. Manning, and Yoram Singer, "Log-linear models for label ranking," in *21st International Conference on Machine Learning*, 2004.

[45] Christopher Williams Neural, Christopher K. I. Williams, and Carl E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems*. 1996, pp. 514–520, MIT press.

[46] David C. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[47] Hannes Nickisch and Carl Edward Rasmussen, "Approximations for Binary Gaussian Process Classification," *Journal of Machine Learning Research*, vol. 9, pp. 2035–2078, October 2008.

[48] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis, Second Edition*, Chapman & Hall/CRC, 2003.

[49] Umar Syed, Robert E. Schapire, and Let ??, "A game-theoretic approach to apprenticeship learning," in *In Advances in Neural Information Processing Systems*. 2008, pp. 1449–1456, MIT Press.

[50] Michael Kearns, "Near-optimal reinforcement learning in polynomial time," pp. 209–232, 2002.

[51] Qifeng Qiao and Peter A. Beling, "Inverse reinforcement learning via convex programming," in *Americon Control Conference*, 2011.

[52] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*

1999: The Sixteenth International Conference on Machine Learning, 1999, pp. 278–287.

[53] Atkeson Christopher G. and Schaal Stefan, "Robot learning from demonstration," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, San Francisco, CA, USA, 1997, pp. 12–20, Morgan Kaufmann Publishers Inc.

[54] Kenji Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, pp. 219–245, 2000.

[55] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.

[56] Dimitri P. Bertsekas and John N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[57] Rutledge R.B., Lazzaro S.C., Lau B., Myers C.E., Gluck M.A., and Glimcher P.W., "Dopaminergic drugs modulate learning rates and perseveration in parkinson's patients in a dynamic foraging task," *The Journal of Neuroscience*, vol. 29, no. 48, pp. 15104–15114, 2009.

[58] Michael X Cohen and Charan Ranganath, "Reinforcement learning signals predict future decisions," *Journal of Neuroscience*, vol. 27, no. 2, pp. 371–378, 2007.

[59] Peter Dayan and Nathaniel D. Daw, "Decision theory, reinforcement learning, and the brain," *Cognitive, Affective, and Behavioral Neuroscience*, vol. 8, pp. 429–453, 2008.

[60] Botvinick Matthew M., Niv Yael, and Barto Andrew C., "Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective," *Cognition*, vol. 113, no. 3, pp. 262–280, Dec. 2009.

[61] Hopkins Ed, "Reinforcement learning signals predict future decisions," *Journal of Economic Behavior and Organization*, vol. 64, no. 3-4, pp. 348–368, 2007.

[62] Choi James Jinwoo, Laibson David, Madrain Brigitte, and Metrick Andrew, "Reinforcement learning in investment behavior," Tech. Rep., UCLA Department of Economics, 2007.

[63] Qifeng Qiao and Peter Beling, "Behavior recognition as bayesian inverse learning problem," Tech. Rep., University of Virginia Department of Systems Engineering, 2011.

[64] Xiaojin Zhu and Andrew B.Goldberg, *Introduction to Semi-Supervised Learning*, Morgan and Claypool, 2009.

[65] Huber P.J., *Robust Statistics*, Wiley, 1981.

[66] Hunter D.R. and Lange K., "Computing estimates in the proportional odds model," *Annuals of the Institute of Statistical Mathematics*, vol. 54, pp. 155–168, 2002.

[67] Mrio A. T. Figueiredo, Jos M. Bioucas-Dias, and Robert D. Nowak, "Majorization-minimization algorithms for wavelet-based image restoration.," *IEEE Transactions on Image Processing*, pp. 2980–2991, 2007.

[68] Mehrdad Yaghoobi, Thomas Blumensath, and Mike E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2178–2191, June 2009.

[69] Kenneth Lange, Springer, 2004.

[70] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2007.

[71] C.F.Jeff Wu, "On the convergence properties of the em algorithm," *Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1998.

[72] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, series B*, vol. 39, no. 1, pp. 1–38, 1977.

[73] Willard I. Zangwill, *Nonlinear Programming; a Unified Approach*, Prentice Hall, 1969.

[74] G. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," Tech. Rep., University of Cambridge, 1994.

[75] Richard S. Sutton and Andrew G. Barto, "Reinforcement learning i: Introduction," 1998.

[76] Darryl A. Seale, "Sequential decision making with relative ranks: An experimental investigation of the 'secretary problem'," *Organizational Behavior and Human Decision Process*, vol. 69, pp. 221–236, March 1997.

[77] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson, "The plan recognition problem: An intersection of psychology and artificial intelligence," *Artificial Intelligence*, vol. 11, pp. 45–83, 1978.

[78] Miquel Ramirez and Hector Geffner, "Probabilistic plan recognition using off-the-shelf classical planners," in *AAAI*, 2010.

[79] Miquel Ramirez and Hector Geffner, "Goal recognition over pomdps: Inferring the intention of a pomdp agent," in *IJCAI*, 2011, pp. 2009–2014.

[80] C.Lucas, T.L.Griffiths, F. Xu, and C.Fawcett, "A rational model of preference learning and choice prediction by children," in *Advances in Neural Information Processing Systems 21*, 2009.

[81] W. Yoshida, R.J.Dolan, and K.J.Friston, "Game theory of mind," *PloS Computational Biology*, vol. 4, no. 12, pp. 1–14, 2008.

[82] B.M.Repacholi and A. Gopnik, "Early reasoning about desires: Evidence from 14- and 18- mont-olds," *Developmental Psychology*, vol. 33, no. 1, 1997.

[83] Christine A Fawcett and Lori Markson, "Children reason about shared preferences," *Developmental Psychology*, vol. 46, pp. 299–309, 2010.

[84] Pieter Abbeel and Andrew Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[85] Qifeng Qiao and Peter A.Beling, "Gaussian process inverse reinforcement learning with bayesian inference and convex optimization," *manuscript*, 2011.

[86] Monica Babes-Vroman, Vukosi Marivate, Kaushik Subramanian, and Michael Litman, "Apprenticeship learning about multiple intentions," in *the 28th International Conference on Machine learning*, WA, USA, 2011.

[87] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley, 2001.

[88] C.-C. Chang and C.-J. Lin, "Libsvm : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[89] L. Xu amd J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *Av. Neural Inf. Process Syst.*, 2005, pp. 1537–1544.

[90] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, New York: Dover, 1998.

[91] A. Strehl and J.Ghosh, "Cluster ensembles? a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 9, pp. 386–396, 2002.

[92] A. Banerjee, I.S.Dhillon, J.Ghosh, and S.Sra, "Clustering on the unit hypersphere using von mises-fisher distributions," *Journal of Machine Learning Research*, vol. 6, pp. 1345–1382, 2005.

[93] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, pp. 329–349, 2009.

[94] Daniel Schunk and Joachim Winter, "The relationship between risk attitudes and heuristics in search tasks: A laboratory experiment," *Journal of Economic Behavior and Organization*, vol. 71, pp. 347–360, 2009.