Deep PQR: Solving Inverse Reinforcement Learning using Anchor Actions

Sinong Geng [1] **Houssam Nassif** [2] **Carlos A. Manzanares** [2] **A. Max Reppen** [3] **Ronnie Sircar** [3]

Abstract

We propose a reward function estimation framework for inverse reinforcement learning with deep energy-based policies. We name our method PQR, as it sequentially estimates the Policy, the Q-function, and the Reward function by deep learning. PQR does not assume that the reward solely depends on the state, instead it allows for a dependency on the choice of action. Moreover, PQR allows for stochastic state transitions. To accomplish this, we assume the existence of one anchor action whose reward is known, typically the action of doing nothing, yielding no reward. We present both estimators and algorithms for the PQR method. When the environment transition is known, we prove that the PQR reward estimator uniquely recovers the true reward. With unknown transitions, we bound the estimation error of PQR. Finally, the performance of PQR is demonstrated by synthetic and real-world datasets.

1. Introduction

Inverse reinforcement learning (IRL) (Russell, 1998; Ng et al., 2000) aims to estimate the reward function of an agent given its decision-making history (often referred to as ***demonstrations***). IRL assumes that the policy of the agent is optimal. The reward function characterizes the preferences of agents used for decision-making, and is thus crucial in many scenarios. In reinforcement learning, the reward function is important to replicate decision-making behavior in novel environments (Fu et al., 2018); in financial markets, the reward function depicts the risk tolerance of agents, suggesting how much compensation they require in exchange for asset volatility (Merton, 1973; 1971); in industrial organization, reward functions are synonymous with firm profit functions (Abbring, 2010; Aguirregabiria and Nevo, 2013).

However, different reward functions may lead to the same behaviour of agents, making it impossible to identify the true underlying reward function (Ng et al., 2000). This ***identification issue*** has been a bottleneck for policy optimization in a transferred environment (Fu et al., 2018), and for recovering utility and profit functions in economics (Abbring, 2010; Bajari et al., 2010; Manzanares et al., 2015). To partly address this issue, Maximal Entropy Inverse Reinforcement Learning (MaxEnt-IRL) (Ziebart et al., 2008; Wulfmeier et al., 2015; Finn et al., 2016; Fu et al., 2018) introduces stochastic policies to the problem formulation, and estimates the reward function using maximal entropy. However, the reward function is still not fully identifiable (Fu et al., 2018). Based on MaxEnt-IRL, to deal with this issue, Adversarial Inverse Reinforcement Learning (AIRL) assumes that the reward is solely a function of state variables, and is independent of actions. However, even with state-only reward functions, AIRL and related methods (Kostrikov et al., 2018) are guaranteed to work only for deterministic environment transitions with the so-called decomposability condition. In some scenarios (especially in many real-world settings in economics), these requirements are hard to satisfy.

This work proposes a Policy Q-function Reward (PQR) approach, combined with an anchor-action assumption, to identify and flexibly estimate reward functions in infinite-horizon Markov decision processes with stochastic transitions. We assume that agents follow energy-based policies derived from optimally solving an entropy-augmented reinforcement learning objective (Kappen, 2005; Todorov, 2007; Haarnoja et al., 2017). We estimate the policy by taking advantage of recent developments in Maximal Entropy Adversarial Imitation Learning methods (MaxEnt-AIL), such as Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) and the aforementioned AIRL (Fu et al., 2018). These methods either do not attempt to identify the reward function, or require strong assumptions to do so. However, they provide efficient and effective methods to replicate the policy from agent demonstrations. We aim to recover the reward function by leveraging the accurately estimated policies.

Our estimators for the Q-function and reward function de-

[1]Department of Computer Science, Princeton University, Princeton, New Jersey, USA [2]Amazon, Seattle, Washington, USA [3]Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey, USA. Correspondence to: Sinong Geng <sgeng@princeton.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

pend on the estimated policy. We handle the ***identification issue*** by assuming the existence of an ***anchor action*** with known rewards. One natural example is an action that provides no rewards across all possible realizations of states. Such a zero-reward action may correspond to doing nothing, and exists in many applications (Bajari et al., 2010). Using this assumption, under fairly general conditions, we show that, if (i) the environment transition is given and (ii) policy functions are identifiable from observed demonstrations, the reward estimator uniquely recovers the true underlying reward function. When the environment transition is unknown, the calculation of the proposed estimator is equivalent to first solving a Markov Decision Process (MDP) with only one available action, and then solving a supervised learning problem. Leveraging this characteristic, we provide effective learning algorithms. We also prove that the error of PQR is bounded and show how it decreases as the sample size increases.

Many imitation learning approaches (Reddy et al., 2019a; Ross et al., 2011; Mahler and Goldberg, 2017; Bansal et al., 2018; Gupta et al., 2020; de Haan et al., 2019) do not attempt to recover reward functions, ineased focusing on identifying policy functions. Another related approach involves agent alignment (Leike et al., 2018; Reddy et al., 2019b), which focuses on (i) learning reward functions from demonstrations, and (ii) scaling these reward functions to even more complex domains. We focus on the first part and provide a more accurate reward function estimator. In summary, our main contributions are two fold:

- We propose a reward estimator that is guaranteed to uniquely recover the true reward function under the anchor-action assumption when the environment transition is given.
- Using the same anchor-action assumption, we propose a learning algorithm to calculate the estimator when the environment transition is unknown, and provide its convergence analysis.

The utility of PQR is demonstrated on synthetic data and an illustrative airline market entry analysis.

2. Deep Energy-Based Modeling

In this section, we characterize our setting for how agents make decisions. Similar to MaxEnt-IRL, we assume that agents maximize the entropy-augmented long-term expected reward (Kappen, 2005; Todorov, 2007; Haarnoja et al., 2017).

We describe an MDP by the tuple $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \gamma, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition distribution, $\gamma \in (0,1)$ is the discount factor, and r is the reward function. Let $\mathbf{S}_t$ take values in $\mathcal{S}$ and $\mathbf{A}_t$ in $\mathcal{A}$ to denote state and action variables, respectively. Given a starting state $\mathbf{S}_0 = \mathbf{s}$, the value function is defined as

$$V(\mathbf{s}) := \max_{\pi} \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}[r(\mathbf{S}_t, \mathbf{A}_t) + \alpha \mathcal{H}(\pi(\mathbf{S}_t, \cdot)) \mid \mathbf{S}_0 = \mathbf{s}]. \tag{1}$$

$$\mathcal{H}(\pi(\mathbf{s}, \cdot)) := -\int_{\mathcal{A}} \log(\pi(\mathbf{s}, \mathbf{a}))\pi(\mathbf{s}, \mathbf{a}) \, d\mathbf{a}$$

is the information entropy. $\alpha\mathcal{H}(\pi(\cdot \mid \mathbf{S}_t))$ with $\alpha > 0$ acts as an encouragement for the randomness of agent behavior. $\pi(\mathbf{s}, \mathbf{a})$ denotes the stochastic policy of agents, representing the conditional probability $\mathrm{P}(\mathbf{A}_t = \mathbf{a}_t \mid \mathbf{S}_t = \mathbf{s})$. Thus, the expectation in (1) is over both the transition $\mathcal{P}$ and the stochastic policy π. We assume that the agent takes a stochastic energy-based policy:

$$\pi(\mathbf{s}, \mathbf{a}) = \frac{\exp(-\mathcal{E}(\mathbf{s}, \mathbf{a}))}{\int_{\mathbf{a}' \in \mathcal{A}} \exp(-\mathcal{E}(\mathbf{s}, \mathbf{a}')d\mathbf{a}')},$$

where $\mathcal{E}$ is an energy function represented by deep neural networks. Such an energy distribution is widely used in machine learning, see Boltzmann machines (Hinton, 2012), Boltzmann bandits (Biswas et al., 2019), and Markov random fields (Geng et al., 2018b). The likelihood of the decision-making via such stochastic energy-based policies is reported in Lemma 1.

Lemma 1 (Summary of Haarnoja et al. (2017))**.** *By assuming that agents solve* (1) *optimally with energy-based policies, the likelihood of the observed demonstrations* $\mathbb{X} = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T\}$ *can be derived as*

$$L(\mathbb{X}; r) = \prod_{t=0}^{T} \pi^*(\mathbf{s}_t, \mathbf{a}_t) \prod_{t=0}^{T-1} \mathrm{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t),$$

where the optimal policy function taken by agents follows

$$\pi^*(\mathbf{s}, \mathbf{a}) = \frac{\exp\left(\frac{1}{\alpha} Q(\mathbf{s}, \mathbf{a})\right)}{\int_{\mathbf{a}' \in \mathcal{A}} \exp\left(\frac{1}{\alpha} Q(\mathbf{s}, \mathbf{a}')\right) d\mathbf{a}'}, \tag{2}$$

with

$$Q(\mathbf{s}, \mathbf{a}) := r(\mathbf{s}, \mathbf{a}) + \max_{\pi} \mathbb{E}\left\{ \sum_{t=1}^{\infty} \gamma^t \left[r(\mathbf{S}_t, \mathbf{A}_t) + \alpha \mathcal{H}(\pi(\mathbf{S}_t, \cdot)) \right] \mid \mathbf{s}, \mathbf{a} \right\}.$$

Lemma 1 derives the likelihood for our inverse reinforcement learning problem. The behavior of agents is governed by Q-function: $(\mathbf{s}, \mathbf{a})$ pairs with higher Q values are more likely to appear in the dataset. In other words, given $\mathbf{s}$, an agent is more likely (but not guaranteed) to select better actions. This behavior is more realistic in practice than

assuming agents always deterministically make the best decision. The two hyper parameters γ and α correspond to the patience and the rationality of agents, respectively. The larger the γ is, the more agents value future returns. As α increases, the difference between Q values of different actions diminishes, and the influence of randomness in agent decision-making increases. At the extreme, this is equivalent to choosing actions randomly, which could be interpreted as agents not knowing or trusting existing information. When α approaches 0^+, (1) is equivalent to a classic control or RL problem with a deterministic policy.

3. Related Models

In this section, we review related settings for reward function estimation: Inverse Reinforcement Learning (IRL) in machine learning and Dynamic Discrete Choice (DDC) in economics. We show that our setting is more general and facilitates more efficient and effective estimation methods.

3.1. Inverse Reinforcement Learning

The most relevant model for our problem is MaxEnt-IRL (Ziebart et al., 2008), where the likelihood of the observation set $\mathbb{X} = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T\}$ is

$$L^{MaxEnt}(\mathbb{X}; r) \propto \prod_{t=1}^{T} \exp\left(r(\mathbf{s}_t, \mathbf{a}_t)\right). \tag{3}$$

Comparing (3) with the likelihood in Lemma 1, MaxEnt-IRL tends to treat the Q-function in Lemma 1 as the reward function and thus simplifies the problem. As a result, the reward function of MaxEnt-IRL incorporates some environment information, an issue known as environment entanglement (Fu et al., 2018). Instead, we properly represent the relationship between r and Q, and aim to estimate the reward function. When $\gamma = 0$, $r(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$, our setting degenerates to the MaxEnt-IRL setting. When $\gamma \neq 0$, MaxEnt-IRL methods can still be applied to our setting, albeit with the simplification of using our Q-function as their reward function.

Classic IRL models (Ng et al., 2000) with deterministic policies are also a special case. As α approaches 0, (2) gets close to hard maximization over $\mathbf{a}_t \in \mathcal{A}$. This recovers classic IRL with deterministic policies, where agents take the best decision deterministically every time. This behavior is not realistic in practice: in almost all real-world settings that model human behavior, the reward function includes information available to the agent but not available to the statistician. Classic IRL also suffers identification issues.

3.2. Dynamic Discrete Choice

DDCs (Rust, 1987) are dynamic structural models with discrete and finite action spaces, popular in economics. A very popular DDC setting assumes that there exists two sets of state variables: $\mathbf{S}_t$ and ϵ_t, where ϵ_t are random shocks following a Type-I extreme value distribution (Ermon et al., 2015). While $\mathbf{S}_t$ is observable and recorded in the dataset, ϵ_t is not observable in the dataset, and is only known by the agent when making decisions at time t. With some further assumptions listed in Section A of the Supplements, the likelihood of DDC becomes

$$L^{DDC}(\mathbb{X}; r) \propto \prod_{t=1}^{T} \exp\left[Q^{DDC}(\mathbf{s}_t, \mathbf{a}_t)\right],$$

where $Q^{DDC}(\mathbf{s}, \mathbf{a})$ is a counterpart to our Q-function for a discrete finite action space with $\alpha = 1$. Details about $Q^{DDC}(\mathbf{s}, \mathbf{a})$ is deferred to the Section A. The estimation method in Section 4 can also be applied to DDCs to efficiently recover reward functions.

4. Policy-Q-Reward Estimation

We now provide our Policy-Q-Reward estimation, which proceeds by estimating agent policies, Q-functions and rewards in that order. We only propose the estimators and algorithms in this section, and theoretically justify our estimation strategy in Section 5.

4.1. Anchor-Action Assumption

Assumption 1. *There exists a known anchor action* $\mathbf{a}^A \in \mathcal{A}$ *and a function* $g : \mathcal{S} \longmapsto \mathbb{R}$*, such that* $r(\mathbf{s}, \mathbf{a}^A) = g(\mathbf{s})$.

To correctly identify the reward function, we require Assumption 1, which stipulates that there exists an anchor action $\mathbf{a}^A$ whose reward function value is known a priori. A special case is $g(\mathbf{s}) = 0$, indicating that there exists an anchor action providing no rewards. This class of assumption is widely used in economics and has been proved crucial for even much simpler static models (Bajari et al., 2010). Assumption 1 is more realistic in many scenarios (Hotz and Miller, 1993; Bajari et al., 2010) than those in Fu et al. (2018) which require a state-only reward function and a deterministic environment transition. In settings where firms are involved, any non-action (like not selling a good, not entering a market) typically leads to zero rewards and provides a natural anchor (Sawant et al., 2018).

4.2. Policy estimation

Let $\hat{\pi}$ be an estimator to the true policy function $\pi^*(\mathbf{a}, \mathbf{s})$. Our PQR procedure works with any stochastic policy estimator for energy distributions (Geng et al., 2017; Kuang et al., 2017; Fu et al., 2018; Geng et al., 2018a) . We use the Adversarial Inverse Reinforcement Learning (AIRL) procedure of Fu et al. (2018) as an example. Recall from Section 2 that $\pi^*(\mathbf{s}, \mathbf{a}) = \mathrm{P}(\mathbf{A}_t = \mathbf{a} \mid \mathbf{S}_t = \mathbf{s})$. AIRL estimates

the conditional distribution of $\mathbf{A}_t$ given $\mathbf{S}_t$ with a GAN, where the discriminator is $D(s,a) := \frac{\exp\{f(\mathbf{s},\mathbf{a})\}}{\exp\{f(\mathbf{s},\mathbf{a})\}+P(\mathbf{a}\mid\mathbf{s})}$, and $P(\mathbf{a}\mid\mathbf{s})$ is updated by solving (1) substituting the Q-function with $\log(1-D(\mathbf{s},\mathbf{a}))-\log D(\mathbf{s},\mathbf{a})$. At optimality, this procedure leads to $f(\mathbf{s},\mathbf{a})=\log(\pi^*(\mathbf{s},\mathbf{a}))$. While, the aforementioned procedure does not estimate the Q-function or the reward function, Fu et al. (2018) provides a disentangling procedure to estimate the reward function. We refer to AIRL with the disentangling procedure as D-AIRL. D-AIRL requires a state-only reward function and deterministic transitions. The proposed PQR does not use the disentangling procedure, and we compare PQR with D-AIRL in Section 6.

4.3. Reward Estimation

For ease of explanation, we tackle the reward estimator before the Q-function estimator.

Definition 1. *Given Q-function estimator $\hat{Q}(\mathbf{s},\mathbf{a})$ and policy estimator $\hat{\pi}(\mathbf{s},\mathbf{a})$, the reward estimator is defined as:*

$$\hat{r}(\mathbf{s},\mathbf{a}) := \hat{Q}(\mathbf{s},\mathbf{a}) - \gamma\hat{\mathbb{E}}_{\mathbf{s}'}\Big[-\alpha\log(\hat{\pi}(\mathbf{s}',\mathbf{a}^A)) + \hat{Q}(\mathbf{s}',\mathbf{a}^A)\mid\mathbf{s},\mathbf{a}\Big], \quad \text{(R-ESTIMATOR)}$$

where $\mathbf{a}^A$ is the anchor action. $\hat{\mathbb{E}}_{\mathbf{s}'}$ denotes the estimated expectation over $\mathbf{s}'$, the one-step look-ahead state variable. When the environment transition is known, we obtain the exact expectation $\mathbb{E}_{\mathbf{s}'}$.

Note that R-ESTIMATOR shares a very similar form as the Bellman equation for Q:

$$Q(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}[V(\mathbf{s}')\mid\mathbf{s},\mathbf{a}],$$

except the value function is represented by $\hat{Q}(\mathbf{s},\mathbf{a}^A)$ and $\hat{\pi}(\mathbf{s},\mathbf{a}^A)$. This representation avoids directly calculating the value function, which is very challenging and unavailable by many methods including AIRL. Also, the representation requires a one-step-ahead expectation instead of the entire trajectory, making the calculation or estimation for the expectation much easier.

The intuition behind this representation is to replace the value function in the Bellman equation with a specification that makes agent reward functions explicitly a function of agent policies. Representing reward functions in this way can be traced back to Hotz and Miller (1993), and has been used by various subsequent economics publications (Hotz et al., 1994; Train, 2009; Arcidiacono and Miller, 2011). We extend this rational to our more general PQR setting, which eventually facilitates more efficient estimation methods. Given $\hat{\pi}(\mathbf{s},\mathbf{a})$ and $\hat{Q}(\mathbf{s},\mathbf{a})$, Algorithm 1 provides an implementation of R-ESTIMATOR to estimate the reward, and uses a deep neural network to estimate the expectation. The calculation of R-ESTIMATOR reduces to a supervised learning problem.

Algorithm 1 Reward Estimation (RE)

Input: Dataset: $\mathbb{X} = \{\mathbf{s}_0,\mathbf{a}_0,\cdots,\mathbf{s}_T,\mathbf{a}_T\}$.
Input: $\hat{Q}(\mathbf{s},\mathbf{a})$ and $\hat{\pi}(\mathbf{s},\mathbf{a})$.
Output: $\hat{r}(\mathbf{s},\mathbf{a})$.
1: **Initialize:**
  Initialize a deep function $h : \mathcal{S}\times\mathcal{A}\longmapsto\mathbb{R}$.
2: **for** $t\in[T]$ **do**
3: $\quad y_t \leftarrow -\alpha\log(\hat{\pi}(\mathbf{s}_{t+1},\mathbf{a}^A)) + \hat{Q}(\mathbf{s}_{t+1},\mathbf{a}^A)$
4: **end for**
5: Train h using $\{y_t\}_{t=0}^{T-1}$ with $\{\mathbf{s}_t\}_{t=0}^{T-1}$ and $\{\mathbf{a}_t\}_{t=0}^{T-1}$.
6: **return** $\hat{r}(\mathbf{s},\mathbf{a}) = \hat{Q}(\mathbf{s},\mathbf{a}) - \gamma h(\mathbf{s},\mathbf{a})$.

4.4. Q-function Estimation

Q-function estimation is susceptible to the identification issue. Let $Q(\mathbf{s},\mathbf{a})$ be the ground-truth Q-function. For any function $\phi:\mathcal{S}\longmapsto\mathbb{R}$,

$$Q'(\mathbf{s},\mathbf{a}) := Q(\mathbf{s},\mathbf{a}) + \phi(\mathbf{s})$$

leads to the same behaviour of agents as $Q(\mathbf{s},\mathbf{a})$, making it impossible to recover $Q(\mathbf{s},\mathbf{a})$ without further assumptions. The Q-function is ***shaped*** by $\phi(\mathbf{s})$, as per Ng et al. (1999) terminology. Even worse, if we wrongly take $Q'(\mathbf{s},\mathbf{a})$ as the true $Q(\mathbf{s},\mathbf{a})$, Algorithm 1 can lead to a highly shaped reward estimation:

Lemma 2. *Let Q estimator (denoted by Q') be shaped by a function $\phi(\mathbf{s})$. R-ESTIMATOR leads to a wrongly estimated reward function, even when the expectation and policy function are calculated uniquely and exactly:*

$$r'(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \Phi(\mathbf{s},\mathbf{a}),$$

with $\Phi(\mathbf{s},\mathbf{a}) := \phi(\mathbf{s}) - \gamma\hat{\mathbb{E}}[\phi(\mathbf{s})\mid\mathbf{s},\mathbf{a}]$.

Lemma 2 is proved in Section B.1 of the Supplements. According to Lemma 2, the wrongly estimated $Q(\mathbf{s},\mathbf{a})$ induces a reward function estimation shaped in terms of ***both*** the action variable and the state variable. This additive shaping is a bottleneck for policy optimization in a transferred environment (Fu et al., 2018). It is also a long-studied problem in economics, where identification of primitives of utility functions is a first step in modeling customer choices (Bajari et al., 2010; Abbring, 2010). An entire field called partial identification (Tamer, 2010) also focuses on this issue; namely, the conditions under which utility and reward functions (or parameters of these functions) are uniquely identified, or whether they can instead be characterized by identified sets. To deal with this identification issue, we use a two-step strategy. We first identify the Q-function for $\mathbf{a}^A$, and then use it to recover $Q(\mathbf{s},\mathbf{a})$ for other $\mathbf{a}\in\mathcal{A}$.

4.4.1. IDENTIFYING $Q(\mathbf{s},\mathbf{a}^A)$

Let $Q^A(\mathbf{s}) := Q(\mathbf{s},\mathbf{a}^A)$. We define our $Q^A(\mathbf{s})$ estimator as a fixed point to an operator:

Algorithm 2 FQI-I

Input: Dataset: $\mathbb{X}$.
Input: γ, α, N, and $\hat{\pi}(\mathbf{s}, \mathbf{a})$.
Output: $\hat{Q}(\mathbf{s}, \mathbf{a})$.
Initialize:
  Initialize a deep function $h : \mathcal{S} \longmapsto \mathbb{R}$.
$y_t = 0$ for $t \in \left\{t \mid \mathbf{a}_t = \mathbf{a}^A\right\}$
for $k \in [N]$ **do**
  for $t \in \left\{t \mid \mathbf{a}_t = \mathbf{a}^A\right\}$ **do**
    $y_t \leftarrow g(\mathbf{s}_t) - \gamma\alpha \log(\hat{\pi}(\mathbf{s}_{t+1}, \mathbf{a}^A)) + \gamma h(\mathbf{s}_{t+1})$
  end for
  Update h using $\{y_t\}_{\{t \mid \mathbf{a}_t = \mathbf{a}^A\}}$ and $\{s_t\}_{\{t \mid \mathbf{a}_t = \mathbf{a}^A\}}$.
end for
$\hat{Q}(\mathbf{s}, \mathbf{a}) \leftarrow \alpha \log(\hat{\pi}(\mathbf{s}, \mathbf{a})) - \alpha \log(\hat{\pi}(\mathbf{s}, \mathbf{a}^A)) + h(\mathbf{s})$
return $\hat{Q}(\mathbf{s}, \mathbf{a})$

Definition 2. *Let $f(s) := Q(\mathbf{s}, \mathbf{a}^A)$ and $C(\mathcal{S})$ the set of continuous bounded functions $f : \mathcal{S} \longmapsto \mathbb{R}$. Under Assumption 1, $f(\mathbf{s})$ is the unique solution to*

$$\hat{Q}^A(\mathbf{s}) = \hat{\mathcal{T}}\hat{Q}^A(\mathbf{s}), \qquad \text{(Q}^A\text{-ESTIMATOR)}$$

where $\hat{\mathcal{T}}$ is an operator on the set of continuous bounded functions $f : \mathcal{S} \longmapsto \mathbb{R}$. $\hat{\mathcal{T}}$ is defined as

$$\hat{\mathcal{T}} f(\mathbf{s}) := g(\mathbf{s}) + \gamma \hat{\mathbb{E}}_{\mathbf{s}'}\left[-\alpha \log(\hat{\pi}(\mathbf{s}', \mathbf{a}^A)) + f(\mathbf{s}') \mid \mathbf{s}, \mathbf{a}^A\right].$$

$\hat{\mathbb{E}}_{\mathbf{s}'}$ denotes the estimated expectation over one-step transition of state variables.

The definition of $\hat{\mathcal{T}}$ follows from inserting $\mathbf{a}^A$ into R-ESTIMATOR. When $\hat{\mathbb{E}}'_{\mathbf{s}}$ in $\hat{\mathcal{T}}$ is exactly calculated, $\hat{\mathcal{T}}$ becomes a contraction, which guarantees the uniqueness of Q^A-ESTIMATOR (Lemma 7 in the Supplements). We derive Q^A-ESTIMATOR by repeatedly applying the operator $\hat{\mathcal{T}}$. The $\hat{\mathcal{T}}$ operator is similar to the Q-function Bellman operator when solving an MDP with only one available action $\mathbf{a}^A$. We will show that identifying $\hat{Q}^A(\mathbf{s})$ reduces to solving a simple one-action MDP.

4.4.2. IDENTIFYING $Q(\mathbf{s}, \mathbf{a})$

We now derive the estimator to $Q(\mathbf{s}, \mathbf{a})$.

Definition 3. *Given anchor estimator $\hat{Q}^A(\mathbf{s})$ and policy estimator $\hat{\pi}(\mathbf{s}, \mathbf{a})$, we define the Q-function estimator as*

$$\hat{Q}(\mathbf{s}, \mathbf{a}) := \alpha \log(\hat{\pi}(\mathbf{s}, \mathbf{a})) - \alpha \log(\hat{\pi}(\mathbf{s}, \mathbf{a}^A)) + \hat{Q}^A(\mathbf{s}). \qquad \text{(Q-ESTIMATOR)}$$

Q-ESTIMATOR uses the fact that $\phi(s)$ depends only on the state variable. Accordingly, the difference $Q(\mathbf{s}, \mathbf{a}^A) - Q(\mathbf{s}, \mathbf{a})$ is equal to the shaped Q-function difference $Q'(\mathbf{s}, \mathbf{a}^A) - Q'(\mathbf{s}, \mathbf{a})$. Thus, knowing $Q(\mathbf{s}, \mathbf{a}^A)$ is sufficient to correctly identify $Q(\mathbf{s}, \mathbf{a})$ for any $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$. When the environment transition is unknown, we estimate $Q^A(\mathbf{s})$ using fitted-Q-iteration (Riedmiller, 2005). The algorithm uses a deep function for $\hat{Q}^A(\mathbf{s})$, and implements the operator $\hat{\mathcal{T}}$ by repeatedly leveraging the observed demonstrations. With Q^A-ESTIMATOR, we can easily derive Q-ESTIMATOR. The 2-step procedure for the Q-function is summarized as the Fitted-Q-Iteration Identification (FQI-I) method in Algorithm 2. Again, FQI-I reduces to solving a simple one-action MDP. Note that when $\hat{\pi}(\mathbf{s}, \mathbf{a})$ or $\hat{\pi}(\mathbf{s}, \mathbf{a}^A)$ are too small, the log in Q-ESTIMATOR will explode. In practice, one may resort to clipping by capping the policy probability (Ionides, 2008).

Algorithm 3 PQR Algorithm

Input: $\mathbb{X} = \{\mathbf{s}_0, \mathbf{a}_0, \cdots, \mathbf{s}_T, \mathbf{a}_T\}$.
Input: γ, α and N.
Output: $\hat{r}(\mathbf{s}, \mathbf{a})$.
$\hat{\pi}(\mathbf{s}, \mathbf{a}) \leftarrow$AIRL($\mathbb{X}$)
$\hat{Q}(\mathbf{s}, \mathbf{a}) \leftarrow$FQI-I($\mathbb{X}$, N, $\hat{\pi}(\mathbf{s}, \mathbf{a})$)
$\hat{r}(\mathbf{s}, \mathbf{a}) \leftarrow$RE($\mathbb{X}$, $\hat{\pi}(\mathbf{s}, \mathbf{a})$, $\hat{Q}(\mathbf{s}, \mathbf{a})$)
return $\hat{r}(\mathbf{s}, \mathbf{a})$.

4.5. Full Algorithm

We now tie the PQR components together and provide our overall framework in Algorithm 3. Our method has three steps: (i) estimate the policy function by AIRL or other policy estimation methods; (ii) estimate $Q(\mathbf{s}, \mathbf{a})$ using the FQI-I method in Algorithm 2; (iii) estimate the reward function using Algorithm 1. Note that the deep neural network method used in the proposed algorithms can be replaced by many machine learning methods (Nassif et al., 2012; Kuusisto et al., 2014; Athey and Wager, 2019).

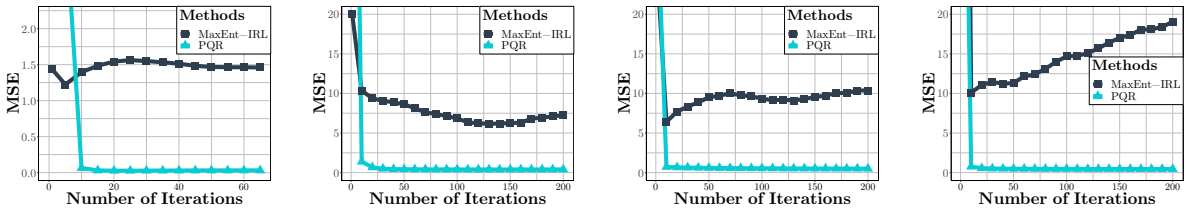
5. Theoretical Analysis

We theoretically study the estimation error of PQR. We first demonstrate the intuition behind PQR by focusing on a simplified setting with (i) known transitions, and (ii) an accurate estimation to the policy function. Then, we study the more general setting and provide a convergence analysis. All the proofs and required extra technical assumptions are deferred to Section B of the Supplements.

5.1. Estimators Error with Known Transitions

In this section we assume that the environment transition is given and that that the policy estimator $\hat{\pi}(\mathbf{s}, \mathbf{a})$ is accurate. We show that the R-ESTIMATOR accurately recovers the true reward function without identification issues. By using this idealized setting, we focus on the errors of the proposed estimators.

Theorem 1. *Let $Q(\mathbf{s}, \mathbf{a})$ and $r(\mathbf{s}, \mathbf{a})$ denote the true Q and*



(a) $p = 5$ (b) $p = 10$ (c) $p = 20$ (d) $p = 40$

Figure 1: MSE for Q-function recovery with different state variable dimensions p

reward function. Let $\hat{r}(\mathbf{s}, \mathbf{a})$, $\hat{Q}^A(\mathbf{s})$ and $\hat{Q}(\mathbf{s}, \mathbf{a})$ be the R-ESTIMATOR*, Q^A-*ESTIMATOR*, and* Q-ESTIMATOR*, whose expectations are exactly calculated. Assume that the policy estimator $\hat{\pi}(\mathbf{s}, \mathbf{a})$ is accurate. Then, under the formulation in Section 2 with Assumption 1,*

- $\hat{Q}^A(\mathbf{s}) = Q(\mathbf{s}, \mathbf{a}^A)$;
- $\hat{Q}(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$;
- $\hat{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a})$.

Once the transition information is known, R-ESTIMATOR is able to uniquely recover the true reward function. In other words, the error of estimators comes from the estimated expectation over the transitions. This justifies the intuition behind PQR.

5.2. PQR Error with Unknown Transitions

Next, we study the errors induced by PQR when the environment transition is unknown. As we suggested in Section 4, Algorithm 1 and Algorithm 2 requires deep supervised learning and deep-FQI. While extensively used in practice, these methods are not yet endowed with theoretical guarantees: theoretical analysis is very challenging due to the nature of deep learning. Existing theoretical work inevitably calls for a series of extra assumptions. We follow the assumptions in Munos and Szepesvári (2008); Du et al. (2019); Arora et al. (2019); Yang et al. (2020). For ease of presentation, we provide the minimal assumptions required to clarify the results. Other assumptions are deferred to Section B.3 of the Supplements. By Assumption 2 and Assumption 3, we assume that the error of the log policy estimation from existing methods like AIRL is bounded, and constrain the training data.

Assumption 2. *Let $\hat{\pi}(\mathbf{s}, \mathbf{a})$ be the estimated policy function in Algorithm 3. The estimation error can be bounded by $\max_{(\mathbf{s},\mathbf{a})\in\mathcal{S}\times\mathcal{A}} |\log(\hat{\pi}) - \log(\pi)| \leq \epsilon_\pi$.*

Assumption 3. *Let there be n IID training data extracted from $\mathbb{X} = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T\}$ for the neural networks in Algorithm 1 and Algorithm 2. Specifically, we use $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathcal{S} \times \mathcal{A}$ to denote the training data for Algorithm 1, and $\{(\mathbf{x}_i, y_i^k)\}_{i=1}^n$ for the k^{th} iteration of Algorithm 2.*

We now provide the convergence result for PQR.

Theorem 2. *Let $Q(\mathbf{s}, \mathbf{a})$ be the true Q-function, $r(\mathbf{s}, \mathbf{a})$ the true reward function, $\hat{Q}(\mathbf{s}, \mathbf{a})$ the result of Algorithm 2, and $\hat{r}(\mathbf{s}, \mathbf{a})$ the result of Algorithm 3. Let Assumptions 1 to 3 and the assumptions listed in Section B.3.1 of the Supplements be satisfied. For any $P \in (0, 1)$, there exists a constant $C > 0$, such that the error of both $\hat{Q}(\mathbf{s}, \mathbf{a})$ and $\hat{r}(\mathbf{s}, \mathbf{a})$ can be bounded with probability at least $1 - P$ by:*

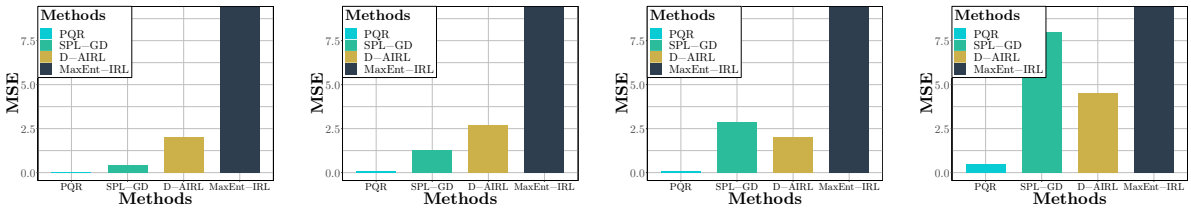
$$\begin{aligned}
&\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}_{\mathbf{s},\mathbf{a}}}\left[\left|\hat{Q}(\mathbf{s}, \mathbf{a}) - Q(\mathbf{s}, \mathbf{a})\right|\right] \\
&\quad\leq \frac{C}{1-\gamma} \max_{k\in[N]} \sqrt{\frac{2\mathbf{y}_k^T(\mathbf{H}^\infty)^{-1}\mathbf{y}_k}{n}} \\
&\quad\quad + \gamma^N \frac{2C}{1-\gamma} + O\left(\sqrt{\frac{\log(\frac{n(N+2)C}{P})}{n}}\right) \\
&\quad\quad + \frac{\gamma\alpha\epsilon_\pi}{1-\gamma},
\end{aligned} \tag{4}$$

$$\begin{aligned}
&\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}_{\mathbf{s},\mathbf{a}}}[|\hat{r}(\mathbf{s}, \mathbf{a}) - r(\mathbf{s}, \mathbf{a})|] \\
&\quad\leq \frac{(1+\gamma)C}{1-\gamma} \max_{k\in[N]} \sqrt{\frac{2\mathbf{y}_k^T(\mathbf{H}^\infty)^{-1}\mathbf{y}_k}{n}} \\
&\quad\quad + \sqrt{\frac{2\mathbf{y}^T(\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}} + \gamma^N C(1+\gamma) + \frac{2\alpha\epsilon_\pi}{1-\gamma} \\
&\quad\quad + O\left(\sqrt{\frac{\log(\frac{n(N+2)C}{P})}{n}}\right),
\end{aligned} \tag{5}$$

where $\mathbf{y} = \{y_1, \cdots, y_n\}^\top$, $\mathbf{y}^k = \{y_1^k, \cdots, y_n^k\}^\top$ and the components of $\mathbf{H}^\infty$ are defined as

$$H_{ij}^\infty = \frac{\mathbf{x}_i^T\mathbf{x}_j(\pi - \arccos(\mathbf{x}_i^\top\mathbf{x}_j))}{2\pi}.$$

Following the rationale in (Arora et al., 2019), $\sqrt{\frac{2\mathbf{y}^T(\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}}$ and $\max_{k\in[N]}\sqrt{\frac{2\mathbf{y}_k^T(\mathbf{H}^\infty)^{-1}\mathbf{y}_k}{n}}$ quantify the generalization errors of the neural networks used



(a) $p = 5$ (b) $p = 10$ (c) $p = 20$ (d) $p = 40$

Figure 2: MSE (truncated at 9) for reward recovery with different state variable dimensions p

in Algorithms 1 and 2, respectively. For the Q-function estimation, the error of the neural network in Algorithm 2 is amplified by $\frac{C}{1-\gamma}$ due to the accumulation in each iteration, according to (4). However, this coefficient is smaller than that of a general FQI method, which is $\frac{2\gamma C}{(1-\gamma)^2}$ (Theorem 4.5 of Yang et al. (2020)). This is consistent with the fact that Algorithm 2 solves a much simplified MDP with only one available action. For the reward estimation, the policy estimation error ϵ_π is amplified by the coefficient $\frac{2\alpha}{1-\gamma}$, according to (5). ϵ_π does not significantly accumulate or explode to harm the proposed reward estimation.

6. Experimental Results

We now demonstrate the utility of PQR. Throughout this section, we assume that $g(\mathbf{s}) = 0$ in Assumption 1, since this is the most common case in practice.

Competing Methods For IRL, we consider the classic MaxEnt-IRL proposed in Ziebart et al. (2008), which estimates the Q-function and treats it as the reward function. The estimated Q-function faces the identification issue described in Section 4.4. Further, we include the D-AIRL method (Fu et al., 2018), which attempts to distinguish the reward function from the Q-function by a disentangling procedure. D-AIRL assumes that the reward does not depend on actions, and that the transition is deterministic. As for DDCs, we consider the Simultaneous Planning and Learning - Gradient Descent (SPL-GD) method (Ermon et al., 2015). SPL-GD directly formulates the reward function as a linear function.

6.1. Synthetic Experiments

For simulation, we build an MDP environment with p-dimensional state variables taking values in $\mathbb{R}^p$, one-dimensional action variables, and a nonlinear reward function. We take $\delta = 0.9$ and $\alpha = 1$ and sovle the MDP with a deep energy-based policy by the soft Q-learning method in Haarnoja et al. (2017). By conducting the learned policy for 50000 steps, we obtain the demonstration dataset, on which we compare PQR, MaxEnt-IRL, AIRL, and SPL-GD. We assume that γ and α are known as required by existing methods. The detailed data-generation procedure is provided in Section C.1 of the Supplements. We let $\mathbf{a}^A = 0$ which means $r(\mathbf{s}, 0) = 0$.

Q Recovery We first consider Q-function estimation. We compare our method with MaxEnt-IRL which also generates an estimated Q-function. The results are summarized in Figure 1. With sufficient training iterations, Algorithm 2 provides a much more accurate Q-function estimation. This result is consistent with the analysis in Section 4 and also Fu et al. (2018): without a proper identification method, Q-function estimation may not recover the true Q-function. On the other hand, when not well trained, the proposed procedure provides poor estimates. This is not surprising since FQI-I outputs the sum of three deep neural networks (line 9 of Algorithm 2) and thus has a bigger error at an early stage of training.

Reward Recovery Since only the proposed method uses the information that $r(\mathbf{s}, 0) = 0$, for a fair comparison, we ground all the other reward estimators by the estimated $r(\mathbf{s}, 0)$. In other words, by this normalization procedure, all the methods provide reward estimates with zero reward at the anchor action, and the estimated reward function should have the same scale as the true reward function. Figure 2 reports the results of reward function estimation. PQR performs the best and provides the most accurate estimates. Reward recovery may not be a fair task for MaxEnt-IRL, since it estimates the Q function instead of the reward function. It also performs the worst of the methods considered. D-AIRL improves upon MaxEnt-IRL by attempting to disentangle the reward from the environment effects. However, since the assumptions required (deterministic transitions and state-only reward functions) are too strong, the estimates are still inaccurate. SPL-GD assumes a linear reward function, and performs worse than PQR in our nonlinear-reward setting. As the dimension of the problem increases, the error increases. Our results show that PQR provides accurate Q-function and reward function estimates.

Note that the performance of the proposed method may not be guaranteed if the anchor-action assumption is not satis-

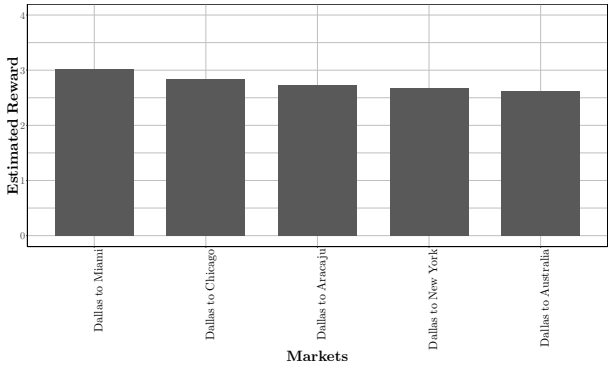


Figure 3: Estimated Reward for American Airlines

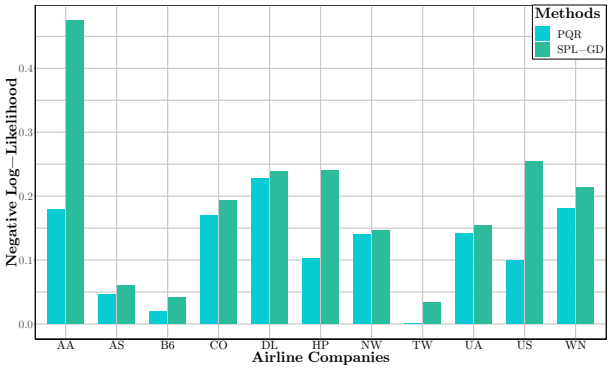


Figure 4: Negative log-likelihood for the airline market entry analysis.

fied (such as the setting of D-AIR with state-only reward functions). More detailed analysis with experiments is provided in Section C.2.1 of the Supplements. We conduct sensitivity (to α and δ), in Section C.2.2. An α selection procedure is provided in Section C.2.3 of the Supplements. PQR may benefit tasks like imitation learning and agent alignment by identifying the reward function accurately. We leave such extensions to future work.

6.2. Airline Market Entry Analysis

After demonstrating the performance of PQR on a synthetic, machine-learning setting, we use the method in a heavily-studied setting in economics. Specifically, we consider the dynamic market entry decisions of airline carriers, where markets are defined as unidirectional city pairs (Berry and Jia, 2010; Manzanares, 2016). For example, Denver to Chicago is one market. Airline entry competition has been studied extensively in economics and other literature, and it is well-known that these problems are computationally hard to solve. In fact, among the top 60 composite statistical areas (CSA's) alone, there are $60 \times 59/2 = 1770$ different markets. We combine public data from the Official Activation Guide[1] for scheduled flight information, with public data from the Bureau of Transportation Statistics[2] for airline company information.

First, we follow Benkard (2004) to aggregate airports by Composite Statistical Areas (CSAs) where possible, and Metropolitan Statistical Areas (MSAs) otherwise. We focus on the top 60 CSAs with the most itineraries in 2002, and thus consider 1770 markets. The action variable $\mathbf{A}_t$ is to select markets from the 1770 options: select entry or not for each market. We study 11 airline companies, see Section C.3 for a list of considered CSAs and airline companies. The considered state variables include origin/destination

[1] https://www.oag.com/

[2] https://www.transtats.bts.gov

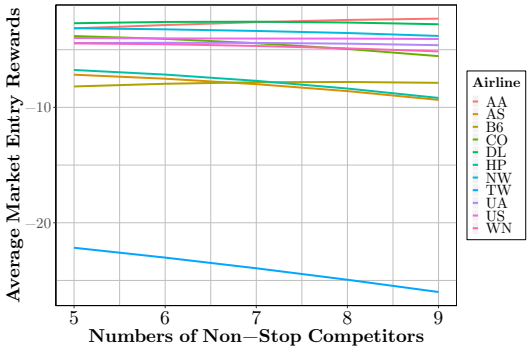


Figure 5: Average market entry rewards v.s. numbers of non-stop competitors.

city characteristics, company characteristics, competitor information for each market and so on. We assume airline companies make decisions given the observed state information, in order to maximize their reward function.

Reward Estimation We implement the proposed PQR method on the airline dataset (including the recorded state variables and airline market entry history) to estimate the reward function. Intuitively, if an airline company decides to not enter a specific market, there will be no reward from this decision, which acts as the anchor action. We then apply the estimated reward function to each company for each potential market, to estimate the profit an airline company could make by entering a specific market. Figure 3 plots the top five PQR-estimated reward markets of American Airlines. The top five most profitable markets are related to Dallas, where American Airlines has its biggest hub. The hub can reduce the cost of entering connecting markets, and thus generate higher rewards (Berry et al., 1996), which is consistent with our empirical results.

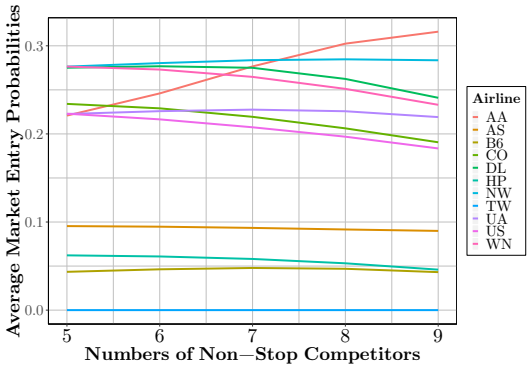


Figure 6: Average market entry probabilities v.s. numbers of non-stop competitors.

Behaviour Prediction We predict the decision-making of airline companies using the estimated reward function, and compare our method to SPL-GD in Figure 4. Our proposed method achieves a smaller negative log-likelihood than SPL-GD, reflecting a better prediction of the decision-making of airline companies.

Counterfactual Analysis We also conduct counterfactual analysis on the number of non-stop competitors using the estimated reward function. Specifically, we change the number of non-stop competitors of each market in year 2015 from 5 to 9. Then, we calculate the average entry reward and predict the average market entry likelihood over the 1770 markets for each airline company. The results are summarized in Figure 5 and Figure 6. Note that, as the number of competitors increases, most airline companies tend to choose not to enter the market. This is consistent with the analysis in Berry and Jia (2010). This trend is more significant for smaller airline companies, which experience harsher negative consequences from competition, given traditionally small profit margins.

American Airlines is the only exception, as its chances of entering markets appear to increase with additional competitors. This phenomenon suggests that there exists unobserved confounders distorting the effect of the number of non-stop competitors. This motivates combining PQR with an instrumental variable (IV) identification strategy for modeling non-stop competition, which in principle would eliminate the confounder impact. See Kuang et al. (2019; 2020) for an example of an IV identification strategy. We leave the combination of IV strategies with PQR for future work.

7. Conclusions

We propose a novel Policy-Q-Reward method to uniquely identify and estimate reward functions in IRL. We provide effective estimators that use the anchor-action assumption to recover the true underlying reward function, under both known and unknown transitions. We demonstrate the utility of PQR using both synthetic and real-world data. While we focus on reward function estimation, PQR can also be applied to imitation learning in the machine learning community and counterfactual analysis in the economics community.

Acknowledgements

The authors would like to thank Amazon Web Services for providing computational resources for the experiments in this paper. We acknowledge the Amazon Prime Economics team as well as Dr. Zongxi Li for very helpful discussions. Dr. Max Reppen is partly supported by the Swiss National Science Foundation grant SNF 181815.

References

Jaap H Abbring. Identification of dynamic discrete choice models. *Annual Review of Economics*, 2(1):367–394, 2010.

Victor Aguirregabiria and Aviv Nevo. Recent developments in empirical IO: Dynamic demand and dynamic games. *Advances in economics and econometrics*, 3: 53–122, 2013.

Peter Arcidiacono and Paul B Ellickson. Practical methods for estimation of dynamic discrete choice models. *Annual Review of Economics*, 3(1):363–394, 2011.

Peter Arcidiacono and Robert A Miller. Conditional choice probability estimation of dynamic discrete choice models with unobserved heterogeneity. *Econometrica*, 79(6): 1823–1867, 2011.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332, 2019.

Susan Athey and Stefan Wager. Estimating treatment effects with causal forests: An application. *arXiv preprint arXiv:1902.07409*, 2019.

Patrick Bajari, Han Hong, and Stephen P Ryan. Identification and estimation of a discrete game of complete information. *Econometrica*, 78(5):1529–1568, 2010.

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

C Lanier Benkard. A dynamic analysis of the market for wide-bodied commercial aircraft. *The Review of Economic Studies*, 71(3):581–611, 2004.

Steven Berry and Panle Jia. Tracing the woes: An empirical analysis of the airline industry. *American Economic Journal: Microeconomics*, 2(3):1–43, 2010.

Steven Berry, Michael Carnall, and Pablo T Spiller. Airline hubs: costs, markups and the implications of customer heterogeneity. Technical report, National Bureau of Economic Research, 1996.

Ari Biswas, Thai T Pham, Michael Vogelsong, Benjamin Snyder, and Houssam Nassif. Seeker: Real-time interactive search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2867–2875, 2019.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, pages 11693–11704, 2019.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685, 2019.

Stefano Ermon, Yexiang Xue, Russell Toth, Bistra Dilkina, Richard Bernstein, Theodoros Damoulas, Patrick Clark, Steve DeGloria, Andrew Mude, Christopher Barrett, et al. Learning large-scale dynamic discrete choice models of spatio-temporal preferences with application to migratory pastoralism in east africa. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 49–58, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Sinong Geng, Zhaobin Kuang, and David Page. An efficient pseudo-likelihood method for sparse binary pairwise Markov network estimation. *arXiv preprint arXiv:1702.08320*, 2017.

Sinong Geng, Zhaobin Kuang, Jie Liu, Stephen Wright, and David Page. Stochastic learning for sparse discrete Markov random fields with controlled gradient approximation error. In *Uncertainty in artificial intelligence: proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 2018, page 156. NIH Public Access, 2018a.

Sinong Geng, Zhaobin Kuang, Peggy Peissig, and David Page. Temporal poisson square root graphical models. *Proceedings of machine learning research*, 80:1714, 2018b.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, pages 1025–1037, 2020.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

V Joseph Hotz and Robert A Miller. Conditional choice probabilities and the estimation of dynamic models. *The Review of Economic Studies*, 60(3):497–529, 1993.

V Joseph Hotz, Robert A Miller, Seth Sanders, and Jeffrey Smith. A simulation estimator for dynamic models of discrete choice. *The Review of Economic Studies*, 61(2): 265–289, 1994.

Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311, 2008.

Hilbert J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011, 2005.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2018.

Zhaobin Kuang, Sinong Geng, and David Page. A screening rule for l1-regularized Ising model estimation. In *Advances in neural information processing systems*, pages 720–731, 2017.

Zhaobin Kuang, Aldo Cordova-Palomera, Fred Sala, Sen Wu, Jared Dunnmon, Chris Re, and James Priest. Mendelian randomization with instrumental variable synthesis (IVY). *bioRxiv*, page 657775, 2019.

Zhaobin Kuang, Frederic Sala, Nimit Sohoni, Sen Wu, Aldo Córdova-Palomera, Jared Dunnmon, James Priest, and Christopher Ré. Ivy: Instrumental variable synthesis for causal inference. *arXiv preprint arXiv:2004.05316*, 2020.

Finn Kuusisto, Vitor Santos Costa, Houssam Nassif, Elizabeth Burnside, David Page, and Jude Shavlik. Support vector machines for differential prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer, 2014.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524, 2017.

Carlos A Manzanares. *Essays on the Analysis of High-Dimensional Dynamic Games and Data Combination*. PhD thesis, Vanderbilt University, 2016.

Carlos A Manzanares, Ying Jiang, and Patrick Bajari. Improving policy functions in high-dimensional dynamic games. Technical report, National Bureau of Economic Research, 2015.

Robert C Merton. Optimum consumption and portfolio rules in a continuous-time model. In *Stochastic Optimization Models in Finance*, pages 621–661. Elsevier, 1971.

Robert C Merton. An intertemporal capital asset pricing model. *Econometrica*, 41(5):867–887, 1973.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Houssam Nassif, Vitor Santos Costa, Elizabeth Burnside, and David Page. Relational differential prediction. In *European Conference on Machine Learning (ECML)*, pages 617–632, 2012.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, volume 1, page 2, 2000.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2019a.

Siddharth Reddy, Anca D Dragan, Sergey Levine, Shane Legg, and Jan Leike. Learning human objectives by evaluating hypothetical behavior. *arXiv preprint arXiv:1912.05652*, 2019b.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial intelligence and Statistics*, pages 627–635, 2011.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

John Rust. Optimal replacement of GMC bus engines: An empirical model of harold zurcher. *Econometrica: Journal of the Econometric Society*, pages 999–1033, 1987.

Neela Sawant, Chitti Babu Namballa, Narayanan Sadagopan, and Houssam Nassif. Contextual multi-armed bandits for causal marketing. In *Proceedings of the International Conference on Machine Learning (ICML'18) Workshops*, 2018.

Elie Tamer. Partial identification in econometrics. *Annual Review of Economics*, 2(1):167–195, 2010.

Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376, 2007.

Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Zhuoran Yang, Yuchen Xie, and Zhaoran Wang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489, 2020.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.