Proceedings of the Fifteenth AAAI Conference on Artificial
Intelligence and Interactive Digital Entertainment (AIIDE-19)

Beyond Winning and Losing: Modeling Human Motivations and Behaviors with Vector-Valued Inverse Reinforcement Learning

Baoxiang Wang,[1] Tongfang Sun,[2] Xianjun Sam Zheng[3]
[1]The Chinese University of Hong Kong
[2]University of Washington
[3]Deephow & Tsinghua University
bxwang@cse.cuhk.edu.hk, tongfs@uw.edu, sam.zheng@deephow.com

Abstract

In recent years, reinforcement learning (RL) methods have been applied to model gameplay with great success, achieving super-human performance in various environments, such as Atari, Go and Poker. However, these studies mostly focus on winning the game and have largely ignored the rich and complex human motivations, which are essential for understanding humans' diverse behavior. In this paper, we present a multi-motivation behavior model which investigates the multifaceted human motivations and learns the underlying value structure of the agents. Our approach extends inverse RL to vectored-valued rewards with Pareto optimality which significantly weakens the inverse RL assumption. Our model therefore incorporates a wider range of behavior that commonly appears in real-world environments. For practical assessment, our algorithm is tested on World of Warcraft datasets and demonstrates the improvement over existing methods.

1 Introduction

Reinforcement learning methods (Sutton and Barto 2018) have been intensively applied to game environments with great success. Some landmark research works have been conducted on games such as Atari, Go, and Poker where reinforcement learning algorithms achieve super-human performance ((Mnih et al. 2015; Silver et al. 2017; Heinrich and Silver 2016)). However, the majority of the studies have focused on winning the game or achieving high scores which largely ignore other human motivations in games such as relaxing, enjoyment and engagement. Hence, the understanding of these motivations and their corresponding reward mechanism has long been open. Extending the unitary, scalar reward to model multifaceted human motivations is non-trivial (Mossalam et al. 2016; Nguyen 2018). In fact, the vector-valued reward signal does not describe the task as clear and succinct as the scalar reward does, and consequently, the optimality of the policy can be ill-defined.

Previous studies have discussed the setting of multiple reward signals, in terms of *features*, which commonly show up in inverse reinforcement learning (IRL) since its inception ((Ng, Russell, and others 2000; Abbeel and Ng 2004)). The objective is then defined as a combination of these features, either linearly or through a parameterized function like neural networks (Wulfmeier, Ondruska, and Posner 2015; Finn, Levine, and Abbeel 2016; Li and Wang 2018; Wang 2019; Young, Wang, and Taylor 2018). When the objective is a scalar, the assumption of IRL is that the agent achieves a higher or equal objective for the action it has taken than an alternative action it would have taken. This assumption does hold in general, as the scalarization of features for the objective is not the same for all agents. In practice, the formulation induced by this assumption has to be relaxed, for example in (Ziebart et al. 2008; Parameswaran and Weinberger 2010), to make the problem even feasible. Especially for real-world environments where the motivations are rather multifaceted, algorithms with weaker assumptions are desired.

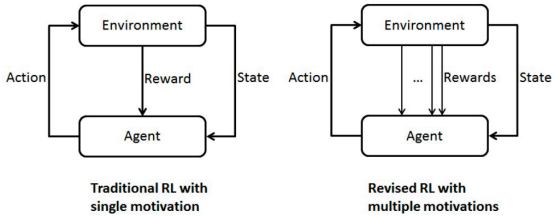

Figure 1: In a typical RL model (left), an agent has only one motivation and maximizes the scalar reward. In a revised RL model (right) for real-world settings, an agent has multiple motivations and the goal is to optimize a combination of different rewards based on the agent's own value.

We investigate the IRL problem under the setting of vector-valued rewards, where the utility measure is the cumulative reward vector. In this setting, the utility of all actions will not form a totally ordered universe, thus there will not exist an optimal action. Instead, an action is considered not dominated by an alternative action, if at least one element of the utility vector is strictly greater than the alternative utility. The IRL assumption we impose is that for any feasible action, the observed action is either not dominated by that action, or they yield exactly the same utility vector. This assumption on the optimality of

Table 1: Components of game motivation

Components	Sub-components
Achievement	advancement, mechanics, competition
Social	socializing, relationship, teamwork
Immersion	discovery, role playing, customization, escapism

the demonstrated trajectories is strictly weaker than the one used in previous studies (Ng, Russell, and others 2000; Abbeel and Ng 2004). We show that the setting is rigorously equivalent to where there exists a monotonic increasing scalarization function per agent such that under the mapping of this function the agent is optimal with respect to the scalar utility, which is intuitive if we treat this function as the value of the agent.

We propose a novel method called multi-motivation behavior modeling (MMBM) which takes the multifaceted, complex, and diverse motivations into consideration. Instead of finding the scalarization in the reward space, our method is based on the vectored-reward setting and the vector-valued IRL formulation, as shown in Figure 1. We build our analysis upon the theorem that the policy is not dominated if and only if it is optimal under a linear scalarization to connect the set of optimal policy with the simplex of the coefficient vector. In this way, we characterize the range of the value function under optimal policies as a convex hull. We further estimate the range via value learning on the extreme points and define the suboptimality of a policy as the distance between its value function and the range. Armed with our analyses, we design an efficient algorithm via a linear program to minimize the distance and find the value of the agents.

A significant advantage of MMBM is that it utilizes only off-policy learning. Both the range of the value function and the linear program depend on only the trajectories and does not require the dynamics of the state transition. In this way, the algorithm is applicable to complex environments such as online games. MMBM is tested on the World of Warcraft Avatar History dataset (Lee et al. 2011), which records the movement of players over a three-year period. We present our results based on the motivation theory of gameplay ((Yee 2006) and compare them with previous knowledge-based studies on WoW ((Ducheneaut et al. 2006; Nardi and Harris 2006)). We further show the significant improvement on the inverse learning error (Ng, Russell, and others 2000; Choi and Kim 2011) over existing studies.

2 Formulation of Vector-valued MDP

2.1 Vector-valued Markov Decision Process

We adopt the vector-valued Markov decision process (MDP) studied by (Wakuta 1995). In the vector space the utility does not form a totally ordered universe. We follow the convention that for two vectors $r, r' \in \mathbb{R}^d$, $r > r'$ if and only if $e_i^T r > e_i^T r'$ for all $i \in \{1,\dots,d\}$, where e_i^T has its i-th element equal to 1 and 0 otherwise. We define $r < r'$, $r \geq r'$, and $r \not< r'$ similarly. The vector-valued infinite-horizon MDP is characterized by the finite state space $\mathbb{S}$, the finite action space $\mathbb{A}$, the transit probability $\mathbb{P}(s'|s,a)$, the reward signal $r = r(s,a) \in \mathbb{R}^d$, and the discount factor γ. At each step of the process, the agent chooses a feasible action $a \in \mathbb{A}$ and the system's state s transits to s' with probability $\mathbb{P}(s'|s,a)$, while the agent receives an immediate reward of $r(s,a,s')$. To describe the problem, we first define the value function $v^\pi(s) \in \mathbb{R}^d$ with respect to a fixed policy π by

$$v^\pi(s) = \mathbb{E}_{s'\sim\mathbb{P},s\sim\rho_0(s)}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi], \tag{1}$$

where π chooses actions at each state, either deterministically or stochastically, also without ambiguity we write $r_t = \mathbb{E}_{s'_t}[r(s_t, a_t, s'_t)]$ and let s_0 be the initial state. We will omit the distribution terms $s' \sim \mathbb{P}, s_0 \sim \rho_0(s)$ in the rest of the discussion. The objective of the agent is to find a policy π such that the inequality

$$v^\pi(s_0) \not< v^{\pi'}(s_0) \tag{2}$$

holds for every feasible policy π'. Note that the objective of scalar-valued MDP is the special case where $r(\cdot) \in \mathbb{R}$ and $d = 1$. To characterize the optimal policies, we first denote Π the set of policies that satisfy the condition (2). Also define $V(s) = \{v^\pi(s), \pi \in \Pi\}$ to be the set of value vectors that are only smaller or equal than ($\leq$) themselves in the range of the value function. $V(s)$ is also denoted as the Pareto frontier of the range of the value function (Van Moffaert and Nowé 2014).

We describe the properties of Π's elements. Similar to the recent studies in multi-object MDPs ((Vamplew et al. 2008; Van Moffaert and Nowé 2014; Jaderberg et al. 2016; Mossalam et al. 2016)), we define the scalarization of the reward signal as

$$\tilde{r} = \phi^T r(s,a) \tag{3}$$

where $\phi \in \Phi \subset \mathbb{R}^d$ is the weight vector of the scalarization and $\Phi = \{\phi \in \mathbb{R}^d, \|\phi\|_1 = 1, \phi \geq 0\}$ is the simplex. We then define the set Π_ϕ of the optimal policies under the scalarized reward signal $\Pi_\phi = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi^T r_t | s_0, \pi]$. The following theorem shows the connection between the optimality of policies under the vectored reward and the optimality of policies under the scalarized reward.

Theorem 1. *For any policy π, $\pi \in \Pi$ if and only if there exists a vector $\phi \in \Phi$ such that $\pi \in \Pi_\phi$.*

Theorem 2. *There exists finite number of policies π_1, dots, π_m, such that the range of the value function $v^\pi(s)$ with respect to π is the convex hull with $v^{\pi_1}(s)$, …, $v^{\pi_m}(s)$ being all its extreme points.*

Theorem 1 maps the optimal policies to the reward weight space Φ which is a standard simplex. Theorem 2 then maps the corresponding values to a convex hull based on which the suboptimality measure can be established. Both the theorems are important to us and our analysis is build upon them. We refer the readers to (Wakuta 1995) for the proofs of both the theorems.

2.2 Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) reverses the input and output pairs of RL algorithms, computing the rewards function according to the policies or trajectories of the agents. The topic has been intensively studied and developed since its inception ((Ng, Russell, and others 2000)), including max-margin based methods ((Abbeel and Ng 2004; Ratliff, Bagnell, and Zinkevich 2006; Syed and Schapire 2008; Neu and Szepesvári 2012)), max-entropy IRL ((Ziebart et al. 2008; Boularias, Kober, and Peters 2011; Finn, Levine, and Abbeel 2016)), Bayesian IRL ((Ramachandran and Amir 2007; Levine, Popovic, and Koltun 2011; Michini and How 2012)), and etc. The algorithms vary widely but they can be roughly categorized into two classes. The first idea is to update the policy function and the reward function coordinately. In each policy update, the policy is optimized for one step to maximize the value of the current reward. Then the reward function is optimized for one step to ensure that the expert demonstration is optimal under the reward function. The second idea approaches learn the function approximation of the action-value function. Then it tests, for each state-action pair in the demonstration, if all the alternative feasible actions lead to values that are no greater than the expert action does ((Hester et al. 2018)).

We extend the IRL to the vector-valued case which allows much weaker assumptions than the existing methods. In fact, in both the classes of algorithms it assumes the optimality of the demonstration policy ((Ng, Russell, and others 2000; Abbeel and Ng 2004)). Such optimality can be furthered formulated as the $\not\leq$ relationship between the value function of the expert policy and the value function of the alternative feasible policy. Hence, when both sides of the value function are extended to the vectored case, $\not\leq$ is strictly weaker than the $\geq$ relation of the scalar value functions. To highlight the significance of the difference in both the assumptions, we note that it is common that existing IRL approaches to solve the system induced by the assumption by assigning a penalty on the violation of the assumption. Such relaxation will drive the actual algorithm from its theory and motivation, which can be largely avoided by using vector-valued formulation.

2.3 Value Function Approximation

We discuss the approximation of both the state-value function and the action-state value function under the scalar reward case. This will be used in our algorithm for vector reward settings. Without further specification, the notation we used in this section is still vectorized, i.e. $r \in \mathbb{R}^d$. The action-state value function is defined as

$$Q^\pi(s,a) = \mathbb{E}_{s'\sim\mathbb{P},s\sim\rho_0(s)}[\sum_{t=0}^{\infty}\gamma^t r_t | s_0 = s, a_0 = a, \pi], \quad (4)$$

where by definition we have $v^\pi(s) = \mathbb{E}_{\pi(a|s)}[Q^\pi(s,a)]$.

The approximation is based on the Bellman equation, which describes the recursive relation

$$Q^\pi(s_t,a_t) = \mathbb{E}_{\mathbb{P}(s_{t+1}|s_t,a_t)}[r_t + \gamma\mathbb{E}_{\pi(a'|s_t)}Q^\pi(s_{t+1},a')]. \quad (5)$$

We have then $\pi \in \Pi$ if and only if that whenever a' is feasible under state s_{t+1}

$$Q^\pi(s_t,a_t) \not< \mathbb{E}_{\mathbb{P}}[r_t + \gamma Q^\pi(s_{t+1},a')]. \quad (6)$$

This inequality is not tractable in general, but we describe the previous studies that address the case of $d = 1$ below and present our method in the next section.

The approximate action-state value function can be learned by parameterizing the $Q(\cdot)$ function and minimize the discrepancy between both sides of Equation (6), known as Q-learning ((Watkins and Dayan 1992; Sutton and Barto 2018; Mnih et al. 2015; Dabney et al. 2017)). Specifically when $d = 1$ and $\not<$ degenerates to $\geq$, the discrepancy can be quantified as the Bellman error

$$\frac{1}{2}(Q^\pi(s_t,a_t) - \max_{a'}\mathbb{E}_{\mathbb{P}}[r_t + \gamma Q^\pi(s_{t+1},a')])^2. \quad (7)$$

Combined with the fact that the action-state value function is parameterized (e.g. approximated by a neural network), the Bellman error can be minimized by running optimization program such as SGD. The Q-learning can be conducted off-policy as described.

The optimization in Q-learning is unstable in general and we borrow the techniques and tricks from (Mnih et al. 2015) in the implementation. We re-state the techniques for completeness of our presentation. First, the order of expectation and maximization are swapped, resulting in the estimator $\frac{1}{2}(Q^\pi(s,a) - (r_t + \gamma\max_{a'}Q^\pi(s',a')))^2$ which is biased but easier to compute. It then replace the $\max_{a'}Q^\pi(s',a')$ term by $\max_{a'}Q^\pi(s',a'|\theta^-)$, where θ^- is the parameter of the function approximation of a previous iteration. The tweak aims to reduce the instability due to the correlation of both the approximation terms in (7). Note that our algorithm is compatible with consecutive improvements over Q-learning ((Hasselt 2010; Schaul et al. 2015; Dabney et al. 2017)).

3 Methods

3.1 Estimating Range of the Value Function

We first construct a superset of the range of the value function, which can be used to derive the distance lower bound.

Theorem 3. *When the reward function is linear with respect to the action a, $\{v^\pi(s)|\pi\in\Pi\} \subseteq conv\{\{v^\pi(s)|\pi\in\Pi_{e_1}\},\ldots,\{v^\pi(s)|\pi\in\Pi_{e_d}\}\}$, where conv denotes the convex hull and e_i is the one-hot vector with the i-th element equal to one and zero otherwise. The equality holds when each of the set $\{v^\pi(s)|\pi\in\Pi_{e_i}\}$ has an unique element.*

Proof. Without loss of generality we assume that the reward $r \geq 0$. If the uniqueness holds, for any element in $\{v^\pi(s)|\pi\in\Pi\}$, there is an explicit ϕ which satisfies $\|\phi\|_1 = 1, \phi \geq 0$ such that the element is an optimal value under the reward $\phi^T r$. Hence

$$\text{conv}\{\cup_{i=1}^d\{v^\pi(s)|\pi\in\Pi_{e_1}\}\} \subseteq \{v^\pi(s)|\pi\in\Pi\}.$$

We proof the reverse by contradiction. Suppose there exists an $\pi\in\Pi$ such that $v^\pi(s) \notin \text{conv}\{\{v^\pi(s)|\pi\in\Pi_{e_1}\},\ldots,\{v^\pi(s)|\pi\in\Pi_{e_d}\}\}$. By the separation theorem

there exists a ϕ such that $\phi^T v^\pi(s) > \phi^T z$ for any $z \in \text{conv}\{\{v^\pi(s)|\pi \in \Pi_{e_1}\}, \ldots, \{v^\pi(s)|\pi \in \Pi_{e_d}\}\}$. Let ϕ_{-i} be ϕ except that the i-th element is replaced by zero. If any element of ϕ is positive (assume it is the i-th without loss of generality), we pick from Π_{e_i} the element z with the largest $\phi_{-i}^T z$. The i-th element of $v^\pi(s)$ must be greater than that of z, which contradicts with the definition of Π_{e_i}. If otherwise all elements in ϕ are negative, $v^\pi(s)$ is dominated by its projection on $\text{conv}\{\{v^\pi(s)|\pi \in \Pi_{e_1}\}, \ldots, \{v^\pi(s)|\pi \in \Pi_{e_d}\}\} \subseteq \{v^\pi(s)|\pi \in \Pi\}$, which is the value of a mixed policy. It then contradicts with the optimality of π. The theorem follows. □

Theorem 3 provides a lower bound of the distance to the range of the value function. The bound is also shown to be tight when the optimal policy under the scalar reward $e_i^T r$ is unique, which commonly happens in practice where the environment is complex. Armed with the theorem, estimating the range of the value function amounts to estimating those extreme points. Even when the reward function is nonlinear, it is obvious to prove that this distance lower bound is an approximation by at most a factor of $\sqrt{d}$. As we have discussed in the Bellman feasibility (6), the one-hot vector e_i degenerates the setting to Q-learning with scalar rewards and the value can be efficiently approximated. We estimate d action-state value functions in our algorithm and denote them as $Q^1(\cdot), \ldots, Q^d(\cdot)$ for further use. The exact function approximator and training process largely depends on the environment and we discuss those technical details in the experiments.

3.2 Distance Minimization

We compute the distance between the value of the alternative policies and the range of the value function, which measures the optimality of the demonstration. We also compute the direction of projection used to measure the distance which describes the combination of the reward vector. Let $Q(\cdot) = (Q^1(\cdot), \ldots, Q^d(\cdot))$ be the vectorized action-state value function and let y be the projection of direction. Also denote $\mathscr{T}$ to be the set of state-action pairs (s,a) appeared in the trajectories. We assume that the actions are conducted in a way that the incurred value is not dominated by an alternative feasible action a'. Such optimality is learned by maximizing the distance between the set of optimal values $\text{conv}\{\cup_{i=1}^d \{v^\pi(s)|\pi \in \Pi_{e_1}\}\}$ and the value of the alternative $Q(s,a')$.

We use the Euclidean distance in the projected direction as the distance measure, but it can be any other measurements in general. Formally, the distance is measured by

$$D = \sum_{(s,a)} \left[\max(0, y^T Q(s,a) - \max_{a' \in \mathbb{A}(s) \setminus a} y^T Q(s,a')) \right]. \quad (8)$$

It is worth note that as we do not impose any scalarization on the vectored reward, the model assumption is easier to be satisfied. Consider that the diversity of actions origins from both the diverse reward function and the suboptimality in the actions (such as act randomly), we add a term $c(y^T Q(s,a) - \max_{a' \in \mathbb{A}(s) \setminus a} y^T Q(s,a'))^-$ to make the algorithm more robust. With $c = 0$, it degenerates to the algorithm that only considers the distances. The minimization is reformulated into the following linear program.

$$\begin{aligned} \underset{y,\xi,\eta}{\text{minimize}} \quad & \sum_{(s,a)} c\eta_{s,a}^- - \xi_{s,a}^+ \\ \text{subject to} \quad & \eta_{s,a} \geq y^T (Q(s,a) - Q(s,a')) \geq \xi_{s,a}, \quad (9) \\ & \forall (s,a) \in \mathscr{T}, a' \in \mathbb{A}(s) \setminus a, \\ & y \geq 0, \ \|y\|_1 \geq 1. \end{aligned}$$

We describe the complete algorithm in Algorithm 1, including the range estimation and the distance minimization. Line 4-12 are the Q-learning technique for $d = 1$ that computes the vertices of the superset of the desired value range. Line 13-16 estimate the range of the value function, which is used in computing the lower bound of the distance. Line 17-18 solve the linear program and find the projection vector y for the distance and optimality of the reward space. Note that the algorithm's line 13-18 can be generalized to other measures and approaches to minimize the distance between $Q(s,a)$ and $Q(s,a')$.

Algorithm 1 Multi-motivation behavior modeling

1: **Parameters:** learning rate α, discount factor γ, suboptimality factor c
2: **Initialization:** initialize function approximation parameters w^i randomly, $i = 1, \ldots, d$
3: **Input:** set $\mathscr{T}$ of trajectories
4: **for** $i = 1$ **to** d **do**
5: **for** t **to** size of $\mathscr{T}$ **do**
6: receive the reward r_t;
7: **end for**
8: **repeat**
9: Compute the Bellman error L_1^i in (7);
10: Update w^i via gradient-based methods;
11: **until** convergence of the i-th element of $Q(s,a)$
12: **end for**
13: **for** t **to** size of $\mathscr{T}$ **do**
14: Compute $Q(s,a) = (Q^1(\cdot), \ldots, Q^d(\cdot))$;
15: **end for**
16: Calculate the constraints $Q(s,a) - Q(s,a')$;
17: Find y by solving the linear program (9);
18: **Output:** Q and y

4 Experiments

We highlight that MMBM possesses several merits compared with other IRL algorithms, which allows us the following analysis over the behavior and motivation of the World of Warcraft (WoW) players. First, the algorithm takes trajectories as input and does not query the policy. It allows us to analyze historical data. Also, MMBM does not query the environment dynamic which reduces the computational cost especially when the environment is complex and massive. Third, the algorithm is naturally extended to model the collective behavior of a group when $\mathscr{T}$ is composed of tra-

jectories from multiple players. That helps the model to further generate interesting results on the group of players.

4.1 Implementation Details

We test MMBM on the WoWAH dataset ((Lee et al. 2011; Bell, Sheth, and Kaiser 2013)), which is an interesting dataset recording a significant amount of gameplay data with over 70,000 players' movements (regarded as actions) from realm *TW-Light's Hope* spanning for the 3-year period. We treat each player as a human agent who conducts an action at each time interval. All available data such as current level or joining a *guild* are treated as observations. The players' trajectories are composed of a sequence of movements (actions) and observations (states), which partially reflect their playing strategies. The oracle r is constructed based on Yee's research and other WoW case studies ((Ducheneaut et al. 2006; Nardi and Harris 2006)).

As discussed before, the function approximator which parameterizes the action-value functions is largely depending on the environment. Taking our experiments on the WoWAH dataset as an example, the Q-network architecture is designed according to the available observations and is applied to all reward signals $i = 1, \ldots, d$. In the network, the categorical elements of the input (e.g. *race*, *class*, etc.) are first processed by an embedding layer, while the numeral elements (e.g. session length, current level, etc.) are first fed into a fully connected layer with rectifier non-linearity. The output of the embedding layer and fully connected layer are then concatenated and fed into another fully connected layer with rectifier non-linearity. A final fully connected layer is applied to compute the $Q(s,a)$ value for each action $a \in \cup_s \mathbb{A}(s)$.

4.2 Results on Different Trajectory Sets

We present our experimental results of calculating the direction y that most significantly separates the demonstrated behavior and the alternative actions. Recall that y is solved from the linear program (9) and note that larger element in y infers relative larger importance of the corresponding element. We use trajectories that are randomly drawn from specific subsets to compose the constraints of (9). We compare the results for different player groups, as shown in Figure 2. Significantly value structures difference is observed between the players at a higher level (≥ 50) versus the players at a lower level (≤ 49), where the players at the lower level are much more motivated to achieve advancement. Similarly observed are that *Warrior* players value more on advancement while the *Priest* players value more on relationship. Players that are in a *guild* value more about teamwork and relationship motivations as compared to the players that are not in a *guild* These observations agree with the common knowledge in WoW and the qualitative results in (Ducheneaut et al. 2006; Nardi and Harris 2006). Note that the y vector only demonstrates the direction of the projection hence we normalize the vector to $\|y\|_1 = 1$ in the figure.

4.3 Predicting Actions and Values

We evaluate the performance of MMBM in terms of the accuracy of action prediction and the relative accuracy of the

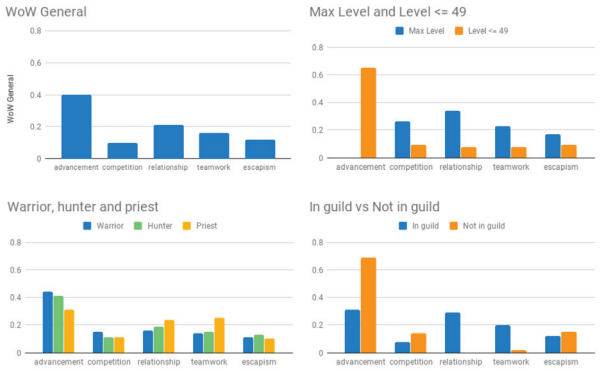

Figure 2: Spider maps to represent the value of the players. **Top-left:** the direction y of different motivations for the entire WoW player community; **top-right:** different value between the players at high game levels and the players at lower game levels; **bottom-left:** different value between the players in different classes, namely, *warrior*, *hunter*, and *priest*; **bottom-right:** comparison of value of the players who are in a *guild* with those who are not in a guild

estimated value compared with the ground truth. For prediction, the trained model is required to output the next action of the player, given the current state. To mitigate the noise and the short interval in the dataset, the prediction is considered correct as long as the predicted action is taken within five time steps, likewise in (Li et al. 2016). The inverse learning error (Ng, Russell, and others 2000; Choi and Kim 2011) measures the relative error occurred in the value function estimation, defined as $y^T|\hat{v} - v|/|y^T v|$ where $\hat{v}$ denotes the estimation and v the ground truth. As the observed reward is always vectorized, we treat each model's scalarization or projection as if it is correct and count only the error due to value estimation. We compare MMBM with those methods that can be implemented on WoWAH, including linear scalarization (Ng, Russell, and others 2000), max entropy scalarization (Ziebart et al. 2008), and large margin scalarization (Parameswaran and Weinberger 2010).

A close examination of the errors that are made during the prediction yields some interesting insights. As IRL assumes that every player tries to maximize their value, i.e., each agent is regarded as a rational and optimal player, it will not be able to distinguish whether a particular action that deviates from an average one is caused by the player's actual intention or the player's sub-optimality during the gameplay. The weaker assumption of MMBM largely eases this indistinguishability, thus yields much smaller learning error.

4.4 Dynamics of the Human Motivation

The motivation of gameplay may evolve. It can also be impacted by the new design or new versions of the game environment. We investigate how would a design update affect the players' motivations and behavior and how we can quantify this impact. To achieve this, we retrain the linear program 9 with linear constraints random drawing within specific time ranges. With the time range moving chrono-

Table 2: Accuracy and error of different approaches

Approach	Accuracy	ILE
Proposed approach ($c = 0.5$)	85.5%	13.2%
Proposed approach ($c = 0$)	84.5%	11.0%
(Ng, Russell, and others 2000)	48.6%	21.0%
(Ziebart et al. 2008)	54.9%	22.1%
Parameswaran and Weinberger 2012	59.2%	19.6%
Policy imitation	36.0%	N/A
Monte-Carlo learning of Q	75.1%	N/A

logically, we show the evolution of game motivation characterized by the elements in y. Figure 3 illustrates the trend of reward elements. Dramatic increment in advancement and competition during the late period in the graph is observed. It occurs at around the 150000th time interval, which coincides with the release of the patch *Wraith of the Lich King* on November 2008 The patch increased the maximum player level from 70 to 80, encouraging the players to complete the remaining levels for advancement. Meanwhile, the patch introduced two new classes in the game, namely *Death Knight* and *Shaman*, which drives the players to open the secondary accounts (and then level them up). The reason behind the increasing competition is that players tend to join player-versus-player arenas to compete with other human players to get more familiar with the mechanisms of their new avatar.

The overall trend of the game during WoWAH is that the game emphasis increasingly on teamwork and relationship. It agrees with the fact that WoWAH was collected only two years after the game release, when new players are attracted to the game. We expect our results to inspire more analysis and understanding of the figure to be conducted in the future.

Figure 3: Top: trends of different kinds of motivations during from Mar 2006 to Jan 2009; **Bottom**: the enlargement of the top figure during around the release of patch *Wraith of the Lich King*

5 Conclusions and Open Problems

We present our algorithmic approach, multi-motivation behavior modeling, a general model based on vector-valued inverse reinforcement learning that takes multifaceted human motivations into consideration. With the vector-valued setting, our algorithm relies on much weaker assumptions compared with existing methods and does not impose any explicit scalarization of the vector-valued reward signals. Instead, it leverages the Pareto frontier of the value to characterize the set of optimal policies and to measure the optimality of the recorded behavior, which agrees with the intuition how humans make decisions. As the algorithm is not relying on the access of policy function nor the dynamics of the environment, it can be applied to study complex, interactive environments, such as online games. Our experiments on the WoWAH dataset demonstrates the value of the players, which subsequently improves the prediction accuracy and inverse learning error. Our work is the first that combines the richness of motivation and psychological theories in game research with the rigorousness of reinforcement learning models. Our goal is beyond winning and losing: Not to simply create software agents that beat human in various games or competitions, but to propose methods that can help to understand the intricacy and complexity of human motivations and their behaviors. We expect our work to inspire more studies to investigate this topic further.

Among the challenges in inverse reinforcement learning, the vector-valued reward is the one that has been long open. We have addressed this challenge by incorporating the estimated Pareto frontier of the range of the value function, but the algorithm relies on the exactness of the reward function. When an element of the reward function is inaccurate, our algorithm tends to incur larger error than those who scalarize the vectored reward, as our approach cannot adjust the weight of the reward to mitigate the effect of this inaccuracy. The drawback can be effectively solved if the algorithm learns a function of each reward elements robustly. Simply using linear function approximation will not work per Theorem 1, thus a subtle functional space is desired. We leave the problem open for further research.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Bell, J.; Sheth, S.; and Kaiser, G. 2013. A large-scale, longitudinal study of user profiles in world of warcraft. In *WWW*.

Boularias, A.; Kober, J.; and Peters, J. 2011. Relative entropy inverse reinforcement learning. In *AISTATS*.

Choi, J., and Kim, K.-E. 2011. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research* 12(Mar):691–730.

Dabney, W.; Rowland, M.; Bellemare, M. G.; and Munos, R. 2017. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*.

Ducheneaut, N.; Yee, N.; Nickell, E.; and Moore, R. J. 2006. Alone together? exploring the social dynamics of massively multiplayer online games. In *SIGCHI*.

Finn, C.; Levine, S.; and Abbeel, P. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*.

Hasselt, H. V. 2010. Double Q-learning. In *NeurIPS*.

Heinrich, J., and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. 2018. Deep Q-learning from demonstrations. In *AAAI*.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

Lee, Y.-T.; Chen, K.-T.; Cheng, Y.-M.; and Lei, C.-L. 2011. World of warcraft avatar history dataset. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, 123–128. ACM.

Levine, S.; Popovic, Z.; and Koltun, V. 2011. Nonlinear inverse reinforcement learning with gaussian processes. In *NeurIPS*.

Li, J., and Wang, B. 2018. Policy optimization with second-order advantage information. *arXiv preprint arXiv:1805.03586*.

Li, S.; Wang, B.; Zhang, S.; and Chen, W. 2016. Contextual combinatorial cascading bandits. In *ICML*.

Michini, B., and How, J. P. 2012. Bayesian nonparametric inverse reinforcement learning. In *ECML-KDD*.

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

Mossalam, H.; Assael, Y. M.; Roijers, D. M.; and Whiteson, S. 2016. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*.

Nardi, B., and Harris, J. 2006. Strangers and friends: Collaborative play in world of warcraft. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, 149–158. ACM.

Neu, G., and Szepesvári, C. 2012. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*.

Ng, A. Y.; Russell, S. J.; et al. 2000. Algorithms for inverse reinforcement learning. In *ICML*.

Nguyen, T. T. 2018. A multi-objective deep reinforcement learning framework. *arXiv preprint arXiv:1803.02965*.

Parameswaran, S., and Weinberger, K. Q. 2010. Large margin multi-task metric learning. In *NeurIPS*.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *IJCAI*.

Ratliff, N. D.; Bagnell, J. A.; and Zinkevich, M. A. 2006. Maximum margin planning. In *ICML*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

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

Syed, U., and Schapire, R. E. 2008. A game-theoretic approach to apprenticeship learning. In *NeurIPS*.

Vamplew, P.; Yearwood, J.; Dazeley, R.; and Berry, A. 2008. On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In *Australasian Joint Conference on Artificial Intelligence*, 372–378. Springer.

Van Moffaert, K., and Nowé, A. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research* 15(1):3483–3512.

Wakuta, K. 1995. Vector-valued markov decision processes and the systems of linear inequalities. *Stochastic Processes and Their Applications* 56(1):159–169.

Wang, B. 2019. Recurrent existence determination through policy optimization. *arXiv preprint arXiv:1905.13551*.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Wulfmeier, M.; Ondruska, P.; and Posner, I. 2015. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*.

Yee, N. 2006. Motivations for play in online games. *CyberPsychology & behavior* 9(6):772–775.

Young, K.; Wang, B.; and Taylor, M. E. 2018. Metatrace: Online step-size tuning by meta-gradient descent for reinforcement learning control. *arXiv preprint arXiv:1805.04514*.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*.