Decentralized Likelihood Quantile Networks for Improving Performance in Deep Multi-Agent Reinforcement Learning

Xueguang Lu [1] **Christopher Amato** [1]

Abstract

Recent value-based multi-agent deep reinforcement learning methods employ optimism by limiting underestimation updates of the value function estimator through a carefully controlled learning rate (Omidshafiei et al., 2017) or a reduced update probability (Palmer et al., 2018). This value overestimation is frequency injected to counteract negative effects caused by sub-optimal (but unobservable) teammate policies and exploration. This paper introduces a decentralized quantile estimator, which aims improve performance through automatic scheduling. Our experiments show the method is more stable, sample efficient and more likely to converge to joint optimal policy than previous methods.

1. Introduction

Recent developments in multi-agent reinforcement learning (MARL) have borrowed an increasing amount of insights from single-agent reinforcement learning, including deep convolutional networks for solving tasks of high dimensional sensory observations (Mnih et al., 2015). In this paper, we examine how recent works of Distributional Reinforcement Learning can be used to improve performance in multi-agent settings and to further utilize distributional information to battle some common issues concerning MARL.

In fully cooperative MARL settings, it is common to consider Independent Learners which learn and execute in a distributed manner. This complete decentralization can be more scalable but poses some issues not associated with centralized or joint learning when actions are shared across agents (Claus & Boutilier, 1998). In particular, with high probability, the agents will not converge to an optimal joint policy but optimal independent policies under the effect of *environment non-stationary* caused by other agents' optimal independent policies (Fulda & Ventura, 2007). In other words, as agents are expected to perform cooperative tasks, each agent must be robust to non-effective explorations by the other agents.

Recent attempts on mitigating this non-stationarity based on hysteretic Q-Learning (Matignon et al., 2007) and leniency (Panait et al., 2006) have shown success in Deep RL. Both approaches limit negative value updates, either proportionally or probabilistically, aiming to partly ignore the effect of locally non-Markovian teammate policies. The trade-off, between environment stochasticity and value overestimation, is considered inevitable since domain stochasticity and teammate policy shifts are indistinguishable traditionally. Empirically, Leniency shows higher stability compared to hysteretic, primarily due to temperature-enabled leniency at different stages of estimation maturity (Palmer et al., 2018) The Leniency decay allows for a more faithful representation of domain dynamics during later stages of training, where its is probable that teammate policies becomes stable and near-optimal, assuming the rate of decay is appropriate and value maturity is synchronized across all states.

Our method aims not only to automatically identify transitions involving sub-optimal teammate policies, especially explorations, but also automatically schedule the amount of optimism applied to each training sample based on estimated value maturity, achieving the above mentioned assumptions without hyper-parameter interventions. In our work, we first verify that deep distributional reinforcement learning (Dabney et al., 2018) can be extended to multi-agent settings to improve training stability, and discuss how the auxiliary distributional information can be further used to identify exploratory teammates through what we call Time Difference Likelihood (TDL). In particular, we evaluate Implicit Quantile Network (IQN) (Dabney et al., 2018) in multi-agent settings, as learning state-action distribution is a more robust learning task and captures auxiliary expectations. The proposed extension, TDL, utilizes distribution information to identify individual sub-par teammate explorations, and guides the amount of optimism injected into the Q distribution (we call Z); we call the extended architecture Likelihood IQN. We show empirically our methods is more

[1]College of Computer and Information Science, Northeastern University, Boston, MA. Correspondence to: Xueguang Lu <lu.xue@husky.neu.edu>, Christopher Amato <c.amato@northeastern.edu>.

Preliminary work. Preprint.

robust even during prolonged training where we observe difficulties in previous methods. In addition, we show that, using what we call Dynamic Risk Distortion operator, risk distortion techniques can be applied in a scheduled fashion to produce optimistic policies that are robust to environment non-stationary.

2. Background

2.1. MDPs and Deep Q-Networks

A Markov Decision Process (MDP) is defined with tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ a finite action space, $\mathcal{T}(s, a, s')$ the probability of transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, and $\mathcal{R}(s, a, s')$ is the immediate reward for such transition. The Markovian assumption is implied by the transition probability. The problem is to find an optimal policy $\pi^\star : \mathcal{S} \rightarrow \mathcal{A}$ which maximizes expected sum of rewards.

Q-Learning (Watkins & Dayan, 1992) is a popular model-free method, which iterates on a set of Q values or approximators to approach optimal Q values or estimates, where $Q(s, a)$ is the expected maximum sum of rewards achievable in the future given in state s and taking action a.

Deep Q-Network (Mnih et al., 2015) develops on a common practice where a function approximator is used for estimating Q values by parameterize Q function $Q^\theta(s, a)$ with parameters θ using deep convolutional neural network. DQN uses experience replay (Lin, 1993) where each transition (s_t, a_t, r_t, s_{t+1}) is stored in a fix-sized experience buffer $D_t = \{(s_1, a_1, r_1, s_2), ..., (s_t, a_t, r_t, s_{t+1})\}$ from which all training batches for the network are uniformly sampled to balance network's tendency to bias towards more recent samples. The update of the network follows the following loss function:

$$L_i(\theta_i) = \mathop{\mathbb{E}}_{s,a,r,s' \sim U(D)} [(r + \gamma \max_{a'} Q^{\theta_i^-}(s', a') - Q^{\theta_i}(s, a))^2] \tag{1}$$

where θ_i^- is the parameters for target network, a target network is an identical network whose parameters are not updated but copied from the main network every C steps as to maintain value stability.

2.2. Decentralized POMDPs (Dec-POMDPs)

Inspired by real-world tasks, partial observable problems are often formalized as Partially Observable MDPs (POMDPs) (Kaelbling et al., 1998). POMDPs are a generalization of MDPs in which agents see observations $\mathcal{O}$ instead of observing the true states. Furthermore, Dec-POMDPs aims to generalize POMDP to decentralized settings (Oliehoek & Amato, 2016) with multiple agents. In a Dec-POMDP, each agent has a set of actions and observations, but there is a joint reward function and agents must choose actions based solely on their local observations (histories). Dec-POMDP is formally defined as: $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{A}^{\mathcal{I}}, \mathcal{Z}, \mathcal{T}, \boldsymbol{O}^{\mathcal{I}}, \boldsymbol{R}^{\mathcal{I}} \rangle$ where $\mathcal{I}$ is a finite set of agents, $\boldsymbol{A}^i$ is the action space for agent $i \in \mathcal{I}$, and $\boldsymbol{O}^i$ is observation space of agent i. At every time step, a joint action $\boldsymbol{a} = \langle a^1, ..., a^{|\mathcal{I}|} \rangle$ is taken, and agents respectively receive immediate rewards $\boldsymbol{R}^i(s, \boldsymbol{a})$. Collaborative agents receive rewards in partial or fully unified fashion. In particular, fully cooperative agents receive joint rewards where $\boldsymbol{R}^i(s, \boldsymbol{a}) = \boldsymbol{R}^{i'}(s, \boldsymbol{a})$ for all $i, i' \in \mathcal{I}$, $s \in \mathcal{S}$ and $\boldsymbol{a} \in \boldsymbol{A}$. Our work focuses on this fully cooperative setting.

Extending DQN to accommodate partially observable (single-agent) tasks, (Hausknecht & Stone, 2015) proposed a model called Deep Recurrent Q-Network (DRQN), where a recurrent layer (LSTM) (Hochreiter & Schmidhuber, 1997) was used to replace the first post-convolutional fully-connected layer of DQN. Hausknecht and Stone argue that the recurrent layer is able to integrate an arbitrarily long history which can be used to infer the underlying state. Empirically, DRQN out-performed DQN on partial-observable tasks and is on par with DQN on fully-observable tasks. DQN and DRQN form the basis of many deep MARL algorithms. Our work's basis is IQN, and the recurrent version of which we call IRQN.

Multi-Agent Reinforcement Learning (MARL) optimizes expected reward signal received by each agent in the same environment. MARL is usually classified into two classes: Independent Learners (ILs) and Joint Action Learners (JALs) (Claus & Boutilier, 1998). ILs observe only local action a^i for agent i, whereas JALs have access to joint action $\boldsymbol{a}$. Our work is in line with solving ILs, while being more difficult, resemble numerous real-world challenges.

2.3. Challenges of ILs

Even with perfect observability, ILs are non-Markovian due to unpredictable and unobservable teammates' actions, hence the *environment non-stationary problem* (Bowling & Veloso, 2002). Previous work has highlighted prominent challenges when attempt to apply Markovian methods, such as Q-Learning, to ILs: *shadowed equilibria* (Fulda & Ventura, 2007), *stochasticity*, and *alter-exploration* (Matignon et al., 2012).

Shadowed Equilibria is the main issue we are addressing with our work, which must be balanced with *stochasticity* problem. Without communications, independent learners who are maximizing their expected return optimally are known to be susceptible to sub-optimal Nash equilibrium where the sub-optimal joint policy can only be improved by changing all agents' policies simultaneously. Methods developed to battle this issue typically put more focus on high rewarded episodes, with the hope that all agents will be able to pursue maximum reward possible, hence forgoing

the objective of maximizing the average return.

Those optimistic methods as mentioned above, although often robust to *shadowed equilibria*, gives up the attempt to precisely estimate transitional stochasticity. Therefore, these methods can mistake a high reward resulted from environment stochasticity as a successful cooperation (Wei & Luke, 2016). This challenge is called *stochasticity* In environments where high reward exists at low probability, the agents will then fail at approaching joint optimal policy.

The *Alter-Exploration* problem arose from unpredictable teammate exploration pattern. In order to estimate mean state values under stochasticity, ILs have to consider exploration-exploitation trade-off. For learners with ϵ-greedy exploration strategy, the probability of at lease 1 out of n agent is exploring at arbitrary time step is $1-(1-\epsilon)^n$. The alter-exploration problem amplifies the issue of *shadowed equilibria* (Matignon et al., 2012).

3. Related Work

We first discussed related work for adapting independent learners for multi-agent domains, and then discuss Implicit Quantile Networks, which we will build off of to develop our method.

3.1. Hysteretic Q-Learning (HQL)

In a Dec-POMDP, the reward for each agent depends on actions chosen by the entire team; so an agent will likely be punished for an optimal action due to actions from non-optimal teammates. Teammates' policies, not only unobservable and non-stationary, but is also often sub-optimal due to exploration strategies; as a result, vanilla Q-Learning therefore would be forced to estimate the exploratory dynamics that is less than ideal. Hysteretic Q-Learning (HQL) (Matignon et al., 2007) attempts to mitigate this issue by injecting overestimation into the value estimation by reducing the learning rate for negative updates. Two learning rates α and β were used, named increase rate and decrease rate, are respectively used for updating overestimated and underestimated TD error δ:

$$Q(x,a) \leftarrow \begin{cases} Q(x,a)+\beta\delta & \text{if } \delta \leq 0 \\ Q(x,a)+\alpha\delta & \text{otherwise} \end{cases} \tag{2}$$

Hysteretic Deep Q-Network (HDQN) (Omidshafiei et al., 2017) applies HQL to DQN. Using DQN as basis, TD error is given by $\delta_t := Q^{\theta_i}(s_t,a_t) - (r+\gamma \max_{a'} Q^{\theta_i^-}(s_{t+1},a'))$. For simplicity, HDQN first sets a base learning rate μ suitable for the network (e.g. $\mu = 0.001$), and scale the learning rate into $\alpha\mu$ and $\beta\mu$. In practice, HDQN usually fix α at 1 and tune μ and β instead; thus in the following discussions, we only discuss the choice and effect of β (the decrease rate) under the assumption that $\alpha = 1$.

In order to reason under partial observability, Hysteretic Deep *Recurrent* Q-Network (HDRQN), introduced by Omidshafiei et al., utilizes a recurrent layer (LSTM) and trained using *experience traces* sampled from experience buffer. Decentralized buffers (called CERTs) featuring sample synchronization was adopted to stabilize training. When using CERTs (Concurrent Experience Replay Trajectories), every agent has their own experience buffer with a deterministic seed; thus, at each sampling operation, traces of the same time steps were sampled across agents. Concurrent sampling during training have the motivation of stabilizing coordination despite shadowed equilibria, avoiding diverging policies, while earlier attempts disables experience reply due to non-concurrent evolving across agents' policies (Foerster et al., 2016).

3.2. Lenient Deep Q-Network (LDQN)

Lenient Learning (Panait et al., 2006) schedules the decrease of leniency applied to individual state-action pairs using decaying temperatures, where leniency is the probability of ignoring a negative Q value update.

Lenient Deep Q-Network (LDQN) (Palmer et al., 2018) combines leniency learning with DQN by encoding high-dimensional state space onto a lower dimension (clusters) on which temperature values are then feasible to store and update. Leniency is obtained from exponentially decaying temperature values for each *state encoding and action* pair using decay schedule with a step limit n, the schedule β is given by:

$$\beta_t = e^{\rho \times d^t} \tag{3}$$

for each t, $0 \leq t < n$, where ρ is a decay exponent which is decayed using a decay rate d. The decay schedule aims to prevent the temperature from premature cooling. Given the schedule, the temperature T is folded and updated as follows:

$$\begin{aligned} &T_{t+1}(\phi(s_t),a_t) \\ &= \beta_t\Big((1-\upsilon)T_t(\phi(s_t),a_t) + \upsilon \mathop{\mathbb{E}}_{a\in A} T_t(\phi(s_{t+1}),a)\Big) \end{aligned} \tag{4}$$

where υ is a fold-in constant. Then, the leniency of a state-action pair is calculated by look up the temperature table and given by:

$$leniency(s,a) = 1 - e^{-K \times T(\phi(s),a)} \tag{5}$$

where K is a leniency moderation constant to control the degree to which leniency is affected by temperature.

LDQN achieves granular control over optimism scheduling in state-action space, effectively mitigating *shadowed equilibria* by ignoring less than ideal rewards, and eventually robust to *overly optimistic* problem as leniency de-

creases over time. On the other hand, successfully applying LDQN requires careful consideration for decay and moderation parameters, whereas our approach requires less hyper-parameters and robust to different parameter values yet yields higher performance with improved sample efficiency.

3.3. Implicit Quantile Network (IQN)

IQN (Dabney et al., 2018) is a single-agent Deep RL method which we extend to multi-agent partial observable settings in our work. As a distributional RL method, quantile networks represent a distribution over returns, denoted Z^π, where $\mathbb{E}(Z^\pi) = Q^\pi$, by estimating the inverse c.d.f. of Z^π, denoted F_π^{-1}. Implicit Quantile Networks (IQN) estimate $F_{\pi,\tau}^{-1}(s,a)$ for a given state-action pair, s, a, from samples drawn from some base distribution ranging from 0 to 1: $\tau \sim U([0,1])$, where τ is the quantile value that the network aims to estimate. The estimated expected return can be obtained by averaging over multiple quantile estimates:

$$Q_\beta(s,a) := \mathop{\mathbb{E}}_{\tau\sim U([0,1])}[F^{-1}_{\pi,\omega(\tau)}(s,a)] \tag{6}$$

where $\beta : [0,1] \to [0,1]$ distorts risk sensitivity, risk neutrality is achieved when $\omega = \mathbb{1}$, in Section 4.3 we will discuss how we distort risk in multi-agent domains and do so in a dynamic fashion where risk approaches neutral as exploration probability approaches 0.

To force interaction between quantile values and observation features extracted by convolutional layers, τ is embedded to match the dimension of features $\phi(\tau)$ and the Hadamard (point-wise) product of thus two vectors is used as features for subsequent fully connected layers. The embedding method Dabney et al. proposed is given by:

$$\phi(\tau) = \text{ReLU}(\sum_{i=0}^{n-1} \cos(\pi i \tau) w_i + b) \tag{7}$$

The quantile regression loss (Koenker & Hallock, 2001) for estimating quantile at τ and error δ is defined using Huber loss $\mathcal{H}_\kappa$ with threshold κ

$$\rho_\tau(\delta) = (\tau - \mathbb{I}\{e \le 0\})\frac{\mathcal{H}_\kappa(\delta)}{\kappa} \tag{8}$$

which weighs overestimation by $1-\tau$ and underestimation by τ, $\kappa = 1$ is used for linear loss.

Given two sampled $\tau, \tau' \sim \omega(U([0,1]))$ and policy π_β, the sampled TD error for time step t follows distributional bellmen operator:

$$\delta_t^{\tau,\tau'} = F_\tau^{-1}(s_t,a_t) - (r_t + \gamma F_{\tau'}^{-1}(s_{t+1}, \pi_\beta(s_{t+1}))) \tag{9}$$

Thus, with $\tau_{1:N}$, $\tau_{1:N'}$, the loss is given by:

$$L = \frac{1}{N'}\sum_{i=1}^{N}\sum_{j=1}^{N'} \rho_{\tau_i}(\delta^{\tau_i,\tau'_j}) \tag{10}$$

Distributional learning have long been considered a promising approach in approximate reinforcement learning due to reduced *chattering* (Gordon, 1995; Kakade & Langford, 2002). Furthermore, distributional RL methods have shown, in single agent settings, robustness to hyperparameter variation and to have superior sample complexity and performance (Barth-Maron et al., 2018).

4. Approach

We extend IQN to multi-agent partial-observable domains. We use IQN as the basis of our method not only because it is a state-of-the-art single-agent method, but also because we believe that learning a distribution over returns provides a richer representation of transitional stochasticity *and* exploratory teammates in MARL. Consequently, the distributional information can be utilized in ways which encourages coordination, but also properly distributes blames among agents, which was historically discussed as an inevitable trade-off. We propose two such methods in this section: Time Difference Likelihood and Dynamic Risk Distortion.

More specifically, we propose a granular approach for controlling the learning rate also in state-action specific fashion but without an explicit encoder. Time Difference Likelihood (TDL), measures the likelihood of return distribution produced by the target network given the distribution produced by the main network. The motivation is two fold: first, for overall similar distribution estimations, even with drastic difference in specific quantile location, the learning rate remains relatively high to capture local differences as to improve sample efficiency; second, for teammate explorations, TDL will more likely to be low, hence applying more hysteresis on non-Markovian dynamics. Also, we show from empirical evaluations, TDL acts as a state-specific scheduler which causes the learning rate to increase over time for states which have received enough training, resulting in more recognition of environment stochasticity, thus converging more robustly towards a joint optimal policy.

Dynamic Risk Distortion (DRD), on the other hand, does not impose value overestimation like hysteretic and leniency; instead, DRD controls the policy, which is derived from value distribution estimations, by distorting the based distribution from which quantile estimation points τ are sampled. The

4.1. Time Difference Likelihood (TDL)

We first discuss TDL, a measure which we proposed, aims to be able to guide the magnitude of the network's learning

rate dynamically for each update. Motivated to reduce the learning rate when encountering exploratory teammates, but properly updating for local mistakes, we would like to find an indicator value distinguishing the two scenarios. TDL is such an indicator we found to be performance-effective and computationally efficient, we scale the learning rate using TDL as discussed later in section 4.2.

To calculate the TDL, we first sample from estimated return distributions (using both the main network and the target network) for given observation-action pairs. For simplicity, we denote these as $d_{1:M} := F^{-1}_{\tau_{1:M}}(s_t, a_t)$ and $t_{1:M'} := r_t + \gamma max_{a'} F^{-1}_{\tau'_{1:M'}}(s_t, a')$, where M and M' is the number of samples we drawn from base distribution. We call them distribution samples and target samples. Obtaining these samples does not add computational complexity, since we can reuse the same samples that were used for calculating losses.

Next, we formalize an approximation method for estimating the likelihood of a set of samples, given a distribution constituted by another set of samples. TDL, in particular, is the likelihood of target samples given the distribution constituted by distribution samples. We denote the probability density function given by the distribution samples as $\mathcal{P}(X) := P(X \mid d_{1:M})$. The intuition of calculating TDL is to treat the discrete distribution samples as a continuous p.d.f. on which the proximity intervals of target samples are calculated for their likelihoods. More specifically, if given $\mathcal{P}$, we estimate the likelihood of target samples as follows:

$$l_{t_{1:M'}, d_{1:M}} = \sum_{j \in 1:M'} \mathcal{P}\Big(\mathbb{E}(t_{j-1}, t_j) \leq X \leq \mathbb{E}(t_{j+1}, t_j)\Big). \tag{11}$$

Now we only need an approximation of the continuous p.d.f. $\mathcal{P}$ which is represented by discrete samples. Our continuous representation is constructed by assuming the density between neighboring samples d_i and d_{i+1} is linear for generalizability and implementation simplicity. We therefore obtain a set of continuous functions $F_i(X)$ each with domain $(d_i, d_{i+1}]$, where F_i is an linearly that fits (d_i, τ_i) and (d_{i+1}, τ_{i+1}).

Let $\mathcal{F}(X) = F_i(X)$ iff $X \in (d_i, d_{i+1}]$. In other words, $\mathcal{F}$ is obtained by connecting all the distribution samples linearly into a continuous monotonically increasing probability density function, which consists of $M-1$ connected linear segments. Using $\mathcal{F}$ as the c.d.f approximation for $\mathcal{P}$, by definition, for arbitrary a and b: $\mathcal{P}(a < X \leq b) = \mathcal{F}(b) - \mathcal{F}(a)$. which can be obtained using linearity property we defined for $\mathcal{F}$:

$$\mathcal{P}(a < X \leq b) = \sum_{i=1}^{M-1} \frac{|(a, b] \cap (d_i, d_{i+1}]|}{d_{i+1} - d_i}(\tau_{i+1} - \tau_i). \tag{12}$$

Note that intervals $(-\infty, d_1]$ and $(d_i, \infty]$ have no probability density, hence are omitted. TDL can be calculated using arbitrary number of samples for all $M > 1$ and $M' > 0$.

We view TDL as not only a noisy consistency measurement between the main and target networks, but also an indicator of information sufficiency in the return distribution estimation. The later is important for MARL because it aims to differentiate stochasticity from non-stationary.

4.2. Likelihood Hysteretic IQN (LH-IQN)

Distributed Q-Learning (Lauer & Riedmiller, 2000) is an overly optimistic method which completely ignores negative updates and is considered a maximization approach. Distributed Q-Learning yields policies that pursue maximum possible reward and is robust in fostering cooperation but gullible to domain stochasticity (e.g. high reward at low probability). Hysteretic Learning (Matignon et al., 2007) acknowledges low returns in a delayed fashion, by updating value estimations at a slower rate when decreasing. Hysteretic approaches show strong performances in both tabular and deep network cases, yet fails to delay value estimations synchronously across state-action space. Leniency Learning (Panait et al., 2006) address this issue by recording temperature values in the state-action space. Temperature values are used to control the negative update, and decrease when update happens to the corresponding state-action pair, . However, when applied in large or continuous state and action spaces, not only is state-action encoding required for computational tractability, extra care is required for scheduling the temperature (Palmer et al., 2018); Palmer et al. found it necessary to apply temperature folding techniques to prevent the temperature from prematurely extinguishing.

To combat these issues, we introduce Likelihood Hysteretic IQN (LH-IQN) which incorporates TDL with hysteretic learning. Theoretically, LH-IQN is able to automatically schedule the amount of leniency applied in the state-action space without careful tuning of temperature values thanks to state-space specific TDL measure. While deep hysteretic learning uses $0 < \beta < \alpha \leq 1$ to scale learning rates, LH-IQN uses the max of β and TDL as the *decrease rate*. More specifically, the learning rate μ_t is given by:

$$\mu_t = \begin{cases} max(\beta, l_{t_{1:M'}, d_{1:M}})\bar{\mu}, & \text{if } \delta_t^{\tau, \tau'} \leq 0 \\ \bar{\mu}, & \text{otherwise} \end{cases}. \tag{13}$$

where $\bar{\mu}$ is a base learning rate suitable for learning the task and network architecture assuming stationary environment (e.g. 0.001). To explore the effect of likelihood and hysteretic during evaluation, we define that L-IQN is an IQN architecture which only uses TDL $l_{t_{1:M'}, d_{1:M}}$ as decrease rate, and H-IQN only uses β as decrease rate. Empirically, β ranging from 0.2 to 0.4 yields high performance, but we later show that LH-IQN is more robust to larger hysteretic

values than previous HDQN.

Since TDL generally increases as the network trains toward consistency, the amount of optimism/overestimation added by hysteretic updates is reduced over time, which is analogous to Leniency. The key difference is that for domain non-stationary (caused by stochasticity and/or shifts in teammate policies), which remains unpredictable forever, TDL remains small, effectively employing a low learning rate toward such transitions.

4.3. Dynamic Risk Sensitive IQN

Distributional RL has also been studied for designing risk sensitive algorithms (Morimura et al., 2010). We introduce dynamic risk sensitive IQN which utilizes what we call *dynamic risk distortion operators*. IQN has shown to be able to easily produce risk-averse and risk-seeking policies by integrating different *risk distortion measures* $\omega : [0, 1] \to [0, 1]$ (Yaari, 1987; Dabney et al., 2018). In single agent positive-sum games, risk-averse policies are sometimes preferred to actively avoid terminal states for more efficient exploration. In MARL, however, agents benefit from risk-seeking policies as seeking highest possible utility helps the team break out of sub-optimal shadowed equilibria. As we are not boosting the value estimations directly, we say this approach injects *hope* instead of optimism. In our work, we let IQN to learn to reflect the true perceived domain dynamics (no learning rate adjustments), but consider generally higher quantile locations (larger values) when making decisions, producing optimistic policies without raising value estimations. Again, to be robust to environment stochasticity, we anneal the amount of distortion we apply so that in the end we produce policies based on realistic (non-optimistic) value estimations. We discuss two such distortion operators: CVnaR and Wang.

CVnaR, Conditional Value-not-at-Risk, is inspired by well studied risk-averse operator Conditional Value-at-Risk ($\text{CVaR}(\eta, \tau) = \eta\tau$) (Chow & Ghavamzadeh, 2014). Our CVnaR is defined as follows:

$$\text{CVnaR}(\eta, \tau) = 1 - \eta\tau. \tag{14}$$

CVnaR simply maps $\tau \sim U([0, 1])$ to $\text{CVnaR}(\eta, \tau) \sim U([\eta, 1])$, and as η reduce, CVnaR become less risk-seeking.

Wang (Wang, 2000) is a distortion operator whose range always remain $[0, 1]$ but becomes exponentially increasing (probability density shifted towards 1) when given positive bias parameter η. Wang is defined as:

$$\text{Wang}(\eta, \tau) = \Phi(\Phi^{-1}(\tau) + \eta) \tag{15}$$

where Φ is the standard Normal cumulative distribution function. Observe that when $\eta \to 1$, Wang almost always returns 1, becoming most risk-seeking distortion operator possible. Also, same as CVnaR, as $\eta \to 0$, risk-neutrality is observed.

We found it suitable to linearly anneal η (for both Wang and CVnaR) during training to achieve better stability as the agent becomes more and more risk-neutral, but behaves extremely like a maximization approach in the beginning. The aim is that during the initial risk-seeking period when η is high, agents are encouraged to explore high rewarding spaces, which supports them to better break out of shadowed equilibra; whereas in the end, the risk-neutral distortion produces an unbiased policy which is unlikely to fall for domain stochasticity.

5. Evaluation

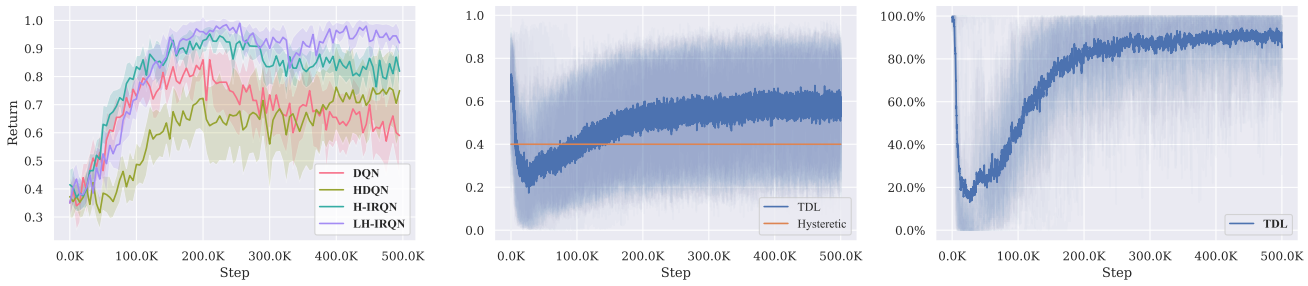
In this section, we compare our method (LH-IQN) with previous works, HDRQN and LDQN, and observe the improved stability of our quantile networks. Furthermore, through inspecting TDL usage trends and performances, we conclude that TDL successfully distinguished, to a certain extent, between domain non-stationary and domain stochasticity. We also demonstrate the effectiveness and adaptiveness (no complex hyper-parameter tuning) of the dynamic risk distortion operator. Results shown in all Figures are aggregated of 20 seeds, each trained decentralized.

5.1. Evaluation on meeting-in-a-grid

We first conduct experiments using recurrent versions of the methods. We label the architecture with added Recurrency as LH-I**R**QN. The network starts with 2 fully connected layers of 32 and 64 neurons respectively, then a LSTM layer with 64 memory cells, a fully connected layer with 32 neurons which then maps onto value estimates for each action. We use $\beta = 0.4$, $\gamma = 0.95$ and Adam optimizer (Kingma & Ba, 2014) for training. For quantile estimators, we sample 16 of τ and τ' to approximate return distributions, and τ embeddings are combined with LSTM output.

We use a partially observable meeting-in-a-grid domain (Amato et al., 2009) to be consistent with previous work (Omidshafiei et al., 2017). The meeting-in-a-grid task consists of one moving target and two agents in a grid world; agents get reward 1 for simultaneously landing on the target location, 0 otherwise. Episodes terminate after 40 transitions or upon successful meeting-at-target. Observations include flickering locations of the agents themselves and the target, and actions result in stochastic transitions.

We first evaluate LH-IRQN performance against HDRQN and H-IRQN (Fig. 1(a)) on a 4×4 grid. H-IRQN does not use TDL but uses only β as decrease rate (Eq 13). Note that HDRQN has a large variance because it does not robustly solve the task—only a portion of seeds reached near-optimal



(a) Performance (b) TDL Value (c) TDL Usage

Figure 1. (a) Evaluation on meeting-in-a-grid 4×4 benchmark. both IRQN models performs better than HDRQN, especially with TDL updates. (b) TDL during training of LH-IRQN on meeting-in-a-grid 4×4 benchmark. (c) Percentage of $l_t > \beta$ where TDL is used as *decrease rate* instead of hysteretic. Same setup as 1(b), where $\beta = 0.4$.

policies. Our IRQN-based methods show more stability concerning reaching optimality, but not utilizing TDL makes agents susceptible to environment stochasticity, producing lesser joint policies over time. We find similar results for higher dimensional (5×5, 6×6) variations of the benchmark (found in Appendix), except for 3×3 which is too simple to differentiate the methods. Directly applying LDQN, with convolution layers replaced by fully-connected layers to better suit the observations, on meeting-in-a-grid failed to solve the tasks due to the high flickering probability and efficient observation encoding. While acknowledging added extra recurrent layer may help LDQN, for comparability, we evaluate and compare with LDQN using a modified version of meeting-in-a-grid with noisy sensory observations in section 5.3 and 5.4.

As shown in Fig. 1(b), TDL increases over time during training, while maintaining a high variance which resulted from domain non-stationary as expected. Overall, the usage of TDL verses hysteretic β increase significantly as shown in Fig. 1(c); as TDL is used when it is larger than β, the overall decrease rate is increased over time, thus adding less hysteresis and optimism to experiences deemed predictable by TDL. While one would expect methods with less optimism to be susceptible to action shadowing, our Likelihood method nonetheless achieves better stability and performance as shown in Fig. 1(a), which implies that TDL is able to distinguish domain non-stationary from stochasticity as we theorized. The spike (and dip) at the beginning seen in Fig 1(b) and 1(c) is due to immature quantile estimations being used to calculate TDL; during the start of training, these quantile values has no guarantee to represent a valid distribution, which may be aggregated together or even reversed depending on the network initialization. As a result, it is unstable to solely use TDL as decrease rate, a problem which we solved with maximizing with Hysteretic, but also can be mitigated using Dynamic Risk Distortion which has an incredibly optimistic distortion during the beginning phase of training.

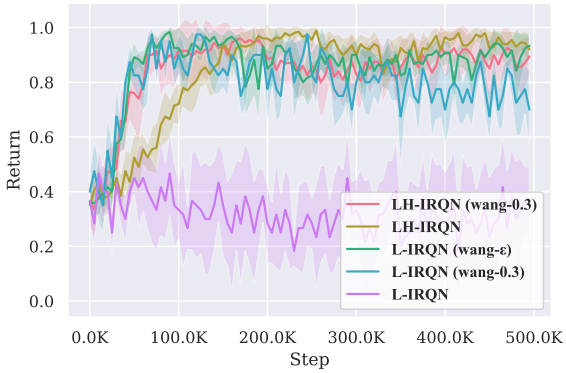


Figure 2. Performance of various Risk Distortion applied to L-IRQN with and without Hysteretic on meeting-in-a-grid 4×4 benchmark; shows that likelihood update works well only in conjunction with either distortion operator or hysteresis.

5.2. Risk Distortion

We evaluate the use of dynamic risk distortion operators. As shown in Fig. 2, TDL alone performs sub-par when used without hysteresis (L-IRQN), since of TDL is initially unstable due to immature quantile estimations (Fig. 1(b)). We already see that when combined with Hysteretic (LH-IRQN), the method shows stability (Fig. 1(a)). We observe that it is also effective, although not as stable, to use risk distortion operators instead of Hysteretic, where no optimism is injected into the value estimations. The performance of combining risk distortion and hysteresis is negligible compared to LH-IRQN on our benchmark (found in Appendix). Since TDL is unstable, often taking on extreme values such as 0 or 1, we reason that the agents would fall for environment stochasticity more easily as value estimations across states are not in the same learning stage. Furthermore, we think the aggressive policy improvement in the beginning is also due to this unstable nature–extremely low TDL value in early stages of training, making the method nearly a maximization based approach. We expect future work of careful analyzing the effect of combining TDL with risk distortion is required to verify our reasoning.

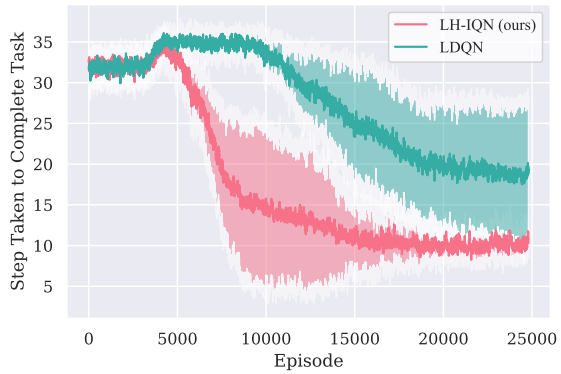


Figure 3. Evaluation of LH-IQN compared with previous LDQN on high-dimensional meeting-in-a-grid 4×4 benchmark. Showing the number of steps taken to complete task (small values preferred).

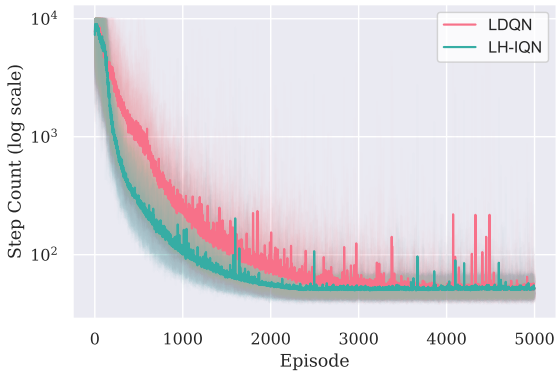


Figure 4. CMOTP benchmark, aggregated three CMOTP variances

We also found that L-DRQN is robust to different η values when using both Wang and CVnaR. We simply used the exploration parameter ϵ as the value for η for our dynamic distortion operator in our evaluation. Our ϵ is annealed from 1 to 0.1 in the first 200K steps. A separate scheduling can be adapted for η. We found annealing from 1 to 0.05 during first 300K steps gives best performance, but the improvement is inconsiderable (0.893 vs. 0.864).

5.3. High dimensional meeting-in-a-grid task

Motivated to compare the performance of our approach with LDQN, we modify the existing 4×4 meeting-in-a-grid benchmark to produce graphical observations (16×16) with added noise, a type of task on which LDQN is originally evaluated. Due to difficulty of grid searching numerous hyper-parameters for LDQN, the parameters used was linearly searched individually while fixating others based on the original work. Based on the parameters the authors used, we found reducing temperature schedule decay rate d from 0.9 to 0.8 helps with convergence in our task, maybe due to meeting-in-a-grid has shorter scenarios. We used: $K = 3.0, d = 0.8, \xi = 0.25$ and $\mu = 0.9995$ along with autoencoder, where ξ is exponent for temperature-based exploration, and μ is the decrease rate for maximum temperature.

As seen in Fig. 3, our method shows higher sample efficiency and overall performance compared to LDQN. Noticing the y-axis is the average number of steps needed to complete the task, we see that LDQN was able to solve the task in the end, however not as stable and optimal as LH-IQN. As the task becomes reliably solvable, LDQN slows down approaching the absolute optimal policy, whereas LH-IQN achieves optimal every run. We notice that the temperature values of LDQN are low during the final stages of training, suggesting minimal leniency is applied. Therefore, it is understandable that the joint policy stuck in shadowed equilibra as exploration is still happening at a low probability ($\epsilon = 0.1$), particularly when the incentive of reaching absolute optimal (least number of steps to complete task) when the task is solvable is small.

5.4. Multi-Agent Object Transportation Problems (CMOTPs)

We also evaluate LH-IQN on three variations of CMOTPs (Palmer et al., 2018), consistent with Palmer et al.'s work on LDQN. CMOTPs require two agents carrying a box to a desired location where agents get a terminal reward; the box only moves when agents are by its side and moving in the same direction. Different variations of the task include added obstacles and stochastic terminal rewards. CMOTPs have 16×16 observations with added noise.

Our network architecture is mostly same as LDQN for comparability: two convolutional layers with 32 and 64 kernels, a fully connected layers with 1024 neurons which combines quantile embedding, followed by another fully connected layer with 1024 neurons which then maps onto value estimates for each action. Hyper-parameters remain same as original work which were found suitable for CMOTPs. Noticing from Fig. 4 that although both methods converge to joint optimal policy, our method shows an improved sample efficiency. We hypothesize that the temperature is decaying less aggressively than it should be, which is likely due to temperature folding techniques and/or that the hashing space of the autoencoder is larger than theoretical minimum.

On the other hand, our method utilizes TDL to scale negative updates, shows better sample efficiency. Initially the value estimations do not seem optimistic enough to perform coordinated actions or to propagate to an earlier-stage state, but the likelihood estimation has the added benefit of being able to produce small values in under-explored state-action space, while hesitating less to update negatively in explored spaces. TDL also helps to synchronize optimism across state-action space; in other words, the ability to estimate distribution consistency adds less optimism to state-action

pairs which have received enough training to be able to produce consistent distributions.

6. Conclusion

Our work extended IQN to better perform in multi-agent domains using Hysteretic Learning and TDL; we demonstrate improved stability and sample efficiency over previous methods. We introduce high-dimensional meeting-in-a-grid benchmark, which requires agents learning a fully-cooperative joint policy from high-dimensional sensory observations. We also evaluate risk distortion in a scheduled manner, a prominent direction for future work in analyzing and utilizing its improved sample efficiency. Future work should also include evaluation on more complex domains with more agents, and/or competitive domains.

References

Amato, C., Dibangoye, J. S., and Zilberstein, S. Incremental policy generation for finite-horizon DEC-POMDPs. In *ICAPS*, 2009.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Bowling, M. and Veloso, M. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2): 215–250, 2002.

Chow, Y. and Ghavamzadeh, M. Algorithms for CVaR optimization in MDPs. In *Advances in neural information processing systems*, pp. 3509–3517, 2014.

Claus, C. and Boutilier, C. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*, 2018.

Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. Learning to communicate to solve riddles with deep distributed recurrent Q-networks. *arXiv preprint arXiv:1602.02672*, 2016.

Fulda, N. and Ventura, D. Predicting and preventing coordination problems in cooperative q-learning systems. In *IJCAI*, volume 2007, pp. 780–785, 2007.

Gordon, G. J. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pp. 261–268. Elsevier, 1995.

Hausknecht, M. and Stone, P. Deep recurrent Q-learning for partially observable MDPs. *CoRR, abs/1507.06527*, 7 (1), 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:1–45, 1998.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koenker, R. and Hallock, K. Quantile regression: An introduction. *Journal of Economic Perspectives*, 15(4):43–56, 2001.

Lauer, M. and Riedmiller, M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

Lin, L.-J. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Matignon, L., Laurent, G., and Le Fort-Piat, N. Hysteretic Q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'07.*, pp. 64–69, 2007.

Matignon, L., Laurent, G. J., and Le Fort-Piat, N. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. Nonparametric return distribution approximation for reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 799–806, 2010.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *arXiv preprint arXiv:1703.06182*, 2017.

Palmer, G., Tuyls, K., Bloembergen, D., and Savani, R. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 443–451. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Panait, L., Sullivan, K., and Luke, S. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 801–803. ACM, 2006.

Wang, S. S. A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance*, pp. 15–36, 2000.

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

Wei, E. and Luke, S. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 17(1):2914–2955, 2016.

Yaari, M. E. The dual theory of choice under risk. *Econometrica: Journal of the Econometric Society*, pp. 95–115, 1987.

7. Appendix

7.1. Results of Individual Variations of CMOTP

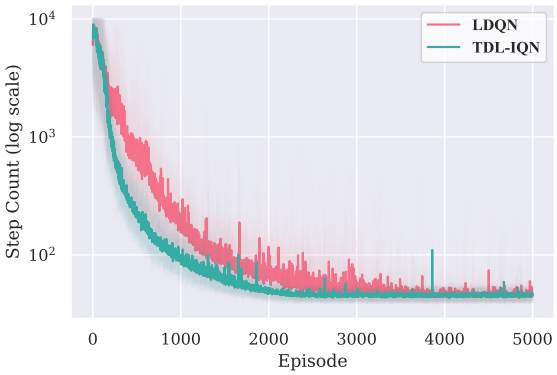


Figure 5. CMOTP Version 1

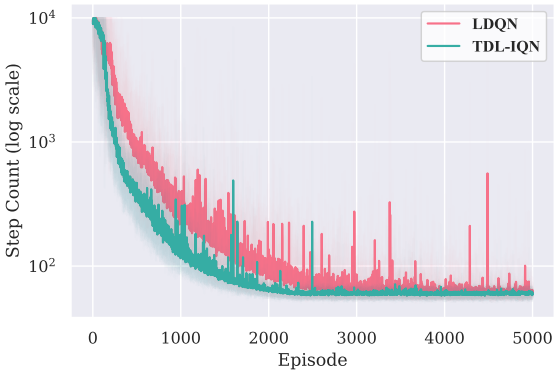


Figure 6. CMOTP Version 2 (Narrow Corredor)

7.2. Results of variances of Meeting-in-a-Grid

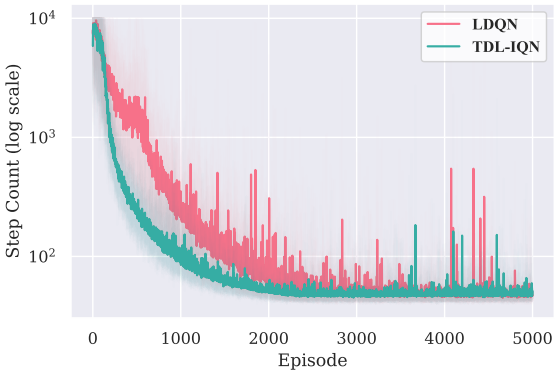


Figure 7. CMOTP Version 3 (Stochastic Reward)

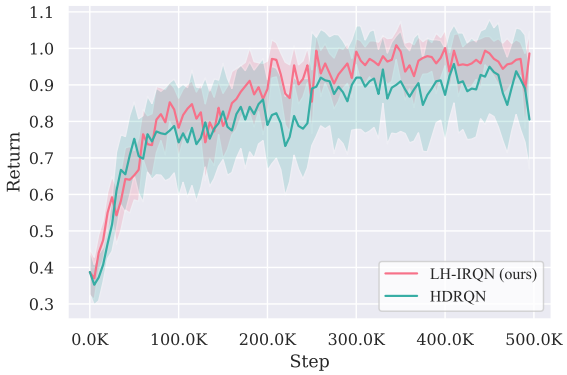


Figure 8. Meeting-in-a-Grid 3×3 benchmark

Figure 10. Meeting-in-a-Grid 6×6 benchmark

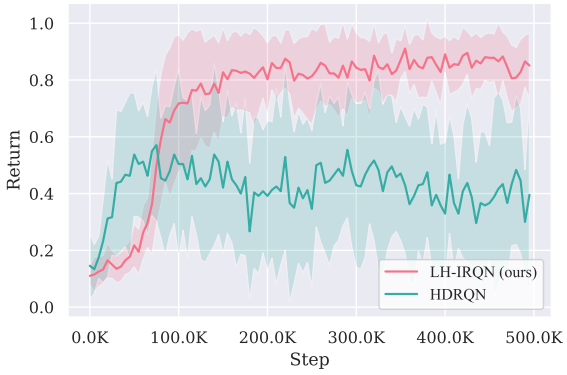


Figure 9. Meeting-in-a-Grid 5×5 benchmark