Decentralized Likelihood Implicit Quantile Network

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

Abstract

Recent successes of value-based multi-agent deep reinforcement learning employ optimism by limiting underestimation updates of value function estimator, through carefully controlled learning rate (Omidshafiei et al., 2017) or reduced update probability (Palmer et al., 2018). To achieve full cooperation when learning independently, agent must estimate the state values contingent on having optimal teammates; therefore, value overestimation is frequency injected to counteract negative effects caused by unobservable teammate sub-optimal policies and explorations. Aiming to solve this issue through automatic scheduling, this paper introduces a decentralized quantile estimator, which we found empirically to be more stable, sample efficient and more likely to converge to joint optimal policy.

1. Introduction

Recent development in multi-agent reinforcement learning (MARL) have borrowed increasing amount insights from single-agent RL developments, including deep convolutional networks for solving tasks high dimensional sensory observations (Mnih et al., 2015). In this paper, we bring insight from recent development in Deep Distributional RL to multi-agent settings; aim to be robust to sub-optimal local policies, we utilize estimated distributions to adjust learning rates in a more effective manner. More specifically, we apply Implicit Quantile Network (IQN) (Dabney et al., 2018) to multi-agent settings, along with extensions to estimate update magnitude which is used to control optimism injected, we call the extended architecture Decentralized Responsible IQN.

Our work focuses on fully cooperative Independent Learners which both learn and execute in a distributed manner. This complete decentralization setting poses a harder problem than settings having information, typically actions, shared across agents (Claus & Boutilier, 1998). However, although task dependent, with high probability, the agents will not converge to joint optimal policy but to independent optimal policies under the effect of environment non-stationary caused by other agents' independent optimal policies (Fulda & Ventura, 2007). In other words, as agents are expected to perform cooperative tasks, agent must be robust to often occurring non-effective explorations. Recent attempts on mitigating the issue in high dimensional observation settings have shown success with hysteretic Q-Learning (Matignon et al., 2007) and leniency (Panait et al., 2006). Both approaches limit negative value updates, either proportionally or probabilistically, aiming to partly ignore the effect of non-effective explorations. Empirically, leniency show higher stability compared to hysteretic methods largely due to its adaptive temperature-enabled leniency at different stages of estimation maturity (Palmer et al., 2018).

In this work, we show how deep distributional reinforcement learning method can be extended to multi-agent settings, and how it can be further used to estimate time difference likelihood. Our approach, based on IQN and Hysteretic Q-Learning ideology, yet adapts to different training stage in state-action space, shows more stable convergence towards optimal joint policy without explicit provision of hyper-parameters for task-specific scheduling.

2. Background

2.1. Markov Decision Process and Deep Q-Network

To be relaxed later, we first formulate perfect information sequential decision making problem as a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ 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. Under thus formulation, the problem is to find a optimal policy $\pi^\star : \mathcal{S} \to \mathcal{A}$ which maximize sum of rewards.

Q-Learning (Watkins & Dayan, 1992) is a popular model-free method of learning optimal policy, which iterate on a set of Q values or approximators to approach optimal Q values or estimates, where $Q(s, a)$ is the expected maximum

[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.

sum of rewards achievable in the future given in state s and taking action a.

$$Q^{\star}(s,a) = \max_{\pi} \mathbb{E}(r_t + \gamma r_{t+1} + ... | s_t = s, a_t = a, \pi) \quad (1)$$

When used with approximation, tabular Q-Learning methods often guarantees to convergence on discrete set of states and actions. The update of tabular approach is shown below.

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a)) \quad (2)$$

Where $\alpha \in (0,1]$ is learning rate at which the Q value updates, and $\gamma \in (0,1]$ is the discount factor applied to future Q estimations.

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 into a large 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:

$$L_i(\theta_i) = \mathop{\mathbb{E}}_{s,a,r,s' \sim U(D)} [(r + \gamma \max_{a'} Q^{\theta_i^-}(s',a') - Q^{\theta_i}(s,a))^2] \quad (3)$$

where θ_i^- is the parameters for target network at iteration i, a target network is an identical network whose parameters are never directly updated but copied from the main network every C steps in order to maintain value stability.

2.2. Partial Observability

Inspired by real-world tasks, single-agent tasks without access to perfect information is often studied and formalized as Partial Observable MDP (POMDP) problems. POMDP extends MDP by granting agent with observations instead of states: $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{T}, \mathcal{O}, \mathcal{R} \rangle$; where $\mathcal{S}$ is still a finite state space, $\mathcal{A}$ is finite action space, $\mathcal{Z}$ is finite set of observations, $\mathcal{T}(s,a,s')$ is the probability of transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $\mathcal{O}(s,a,z)$ is the probability of observing $z \in \mathcal{Z}$ given state s and action a, $\mathcal{R}(s,a)$ is the immediate reward for such transition.

Extending DQN to accommodate partially observable 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 argues 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. Formally, the network estimates $Q(o_t, a_t, h_{t-1}|\theta)$ (instead of $Q(s_t, a_t, |\theta)$), where θ is the parameters of the network, and $h_{t-1} = LSTM(h_{t-2}, o_t)$ is the hidden state generated from the LSTM layer from $t-1$ time step, which is zero initialized. Hausknecht and Stone nonetheless points out that DRQN confers no systematic benefit compared to DQN where past k observations is used to approximate s.

2.3. Decentralized POMDP (Dec-POMDP)

Dec-POMDP aims to generalize POMDP to multi-agent settings, as actions and observations becomes joint actions and observations: $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{A}^{\mathcal{I}}, \mathcal{Z}, \mathcal{T}, \boldsymbol{O}^{\mathcal{I}}, \boldsymbol{R}^{\mathcal{I}} \rangle$ where $\mathcal{I}$ is a finite set of agents, $\boldsymbol{A}^i$ is 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 $\boldsymbol{R}^i(x, \boldsymbol{a}) = \boldsymbol{R}^{i'}(x, \boldsymbol{a})$ for all $i, i' \in \mathcal{I}$, $s \in \mathcal{S}$ and $\boldsymbol{a} \in \boldsymbol{A}$. Our work focuses on fully cooperative settings.

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, while JRLs have access to joint action $\boldsymbol{a}$. Our work is in line with solving ILs, which is more difficult, but resemble numerous real-world challenges.

2.4. Challenges of ILs

Even with perfect observability, ILs are non-Markovian due to teammate actions, hence the *environment non-stationary problem* (Bowling & Veloso, 2002). Literature have discussed other prominent problems when trying to apply Markovian methods, such as Q-Learning, to ILs, including *shadowed equilibria* (Fulda & Ventura, 2007), *stochasiticity*, and *alter-exploration* (Matignon et al., 2012).

2.4.1. Shadowed Equilibria

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.

2.4.2. STOCHASTICITY

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). In environments where high reward exists at low probability, the agents will then fail at approaching joint optimal policy.

2.4.3. ALTER-EXPLORATION PROBLEM

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

3.1. 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 are interested in 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 Network (IQN) estimates $F_\pi^{-1}(\tau, s, a)$ denoted $F_{\pi,\tau}^{-1}(x, a)$ from samples drawn from some base distribution ranging from 0 to 1, e.g. $\tau \sim U([0,1])$, where τ represents the quantile value at which the network aim to estimate. The estimated expected return can be obtained by averaging over multiple quantile estimates:

$$Q_\beta(x,a) := \mathop{\mathbb{E}}_{\tau\sim U([0,1])}[F^{-1}_{\pi,\beta(\tau)}(x,a)] \tag{4}$$

where $\beta : [0,1] \to [0,1]$ distorts risk sensitivity, risk neutrality is achieved when $\beta = \mathbb{1}$.

To force interaction between quantile values and observation features extracted by convolutional layers (in our case , LSTM layer), 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:

$$\phi(\tau) = \text{ReLU}(\sum_{i=0}^{n-1} \cos(\pi i \tau) w_i + b) \tag{5}$$

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

$$\rho_\tau(\delta) = (\tau - \mathbb{I}\{e \le 0\})\frac{\mathcal{H}_\kappa(\delta)}{\kappa} \tag{6}$$

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

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

$$\delta_t^{\tau,\tau'} = F_\tau^{-1}(x_t, a_t) - (r_t + \gamma F_{\tau'}^{-1}(x_{t+1}, \pi_\beta(x_{t+1}))) \tag{7}$$

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

$$L = \frac{1}{N'}\sum_{i=1}^{N}\sum_{j=1}^{N'} \rho_{\tau_i}(\delta^{\tau_i,\tau'_j}) \tag{8}$$

Notice that the loss is a summation of pair-wise quantile losses, and we can sample arbitrary number of τ_i and τ'_j for calculating the loss. Furthermore, Dabney et al. shown that N and N' have no significant effect on long term convergence, and $N = N' = 8$ appears to be sufficient in Atari games.

Distributional RL have shown, in single agent setting, to be robust to hyperparameter variation and to have superior sample complexity and performance (Barth-Maron et al., 2018). IQN, in particular, achieves state-of-the-art performance on Atari games compared to preceding deep distributional RL methods.

3.2. 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:

$$Q(x,a) \leftarrow \begin{cases} Q(x,a) + \beta\delta & \text{if } \delta \le 0 \\ Q(x,a) + \alpha\delta, & \text{otherwise} \end{cases} \tag{9}$$

Hysteretic Deep Q-Network (HDQN) (Omidshafiei et al., 2017) applies HQL to DQN. Using DQN as basis, TD error

is given by $\delta_t := Q^{\theta_i}(s_t, a_t) - (r + \gamma \max_{a'} Q^{\theta_i^-}(s_{t+1}, a'))$. In practice, 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 random 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).

3.3. Lenient Deep Q-Network

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:

$$\beta_t = e^{\rho \times d^t} \tag{10}$$

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:

$$\begin{aligned} &T_{t+1}(\phi(s_t), a_t) \\ &= \beta_t \Big((1 - \upsilon) T_t(\phi(s_t), a_t) + \upsilon \mathop{\mathbb{E}}_{a \in A} T_t(\phi(s_{t+1}), a) \Big) \end{aligned} \tag{11}$$

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:

$$leniency(s, a) = 1 - e^{-K \times T(\phi(s), a)} \tag{12}$$

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 decreases 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 while yielding comparable performance with higher sample efficiency.

4. Approach

We use IQN as our basis. Learning a distribution over returns provides a richer representation of transitional stochasticity *and* exploratory teammates. Therefore, we extend IQN to multi-agent domains and evaluate the architecture on partial observable settings equipped with a LSTM layer and CERTs.

Additionally, we propose a granular approach for controlling the learning rate also in state-action specific fashion but without an explicit encoder. The measure, which we call Time Difference Likelihood (TDL), measures the likelihood of return distribution produced by the network 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 act as a state-specific scheduler which cause learning rate to increase over time for states which have received enough training, result in less action shadowing hence converging more robustly towards joint optimal policy.

4.1. Time Difference Likelihood (TDL)

We introduce TDL, a measure which guides scheduling of network's learning rate. Motivated to reduce the learning rates for exploratory teammates but properly update for local mistakes, we scale learning rates using TDL as discussed in 4.2. We first sample from estimated return distributions (both main network and target network) of observation-action pairs in question, for simplicity, we denote $d_{1:M} := F^{-1}_{\tau_{1:M}}(x_t, a_t)$ and $t_{1:M'} := r_t + \gamma \underset{a'}{max} F^{-1}_{\tau'_{1:M'}}(x_t, a')$, which we call distribution samples and target samples. Obtaining these samples does not add much 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 likelihood of target samples as follows:

$$l_{t_{1:M'},d_{1:M}} = \sum_{j\in 1:M'} \mathcal{P}\Big(\mathbb{E}(t_{j-1},t_j) \leq X \leq \mathbb{E}(t_{j+1},t_j)\Big) \tag{13}$$

Now we only need an approximation of the continuous p.d.f. $\mathcal{P}$ represented, however, 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 a linearity that fits (d_i, τ_i) and (d_{i+1}, τ_{i+1}).

Let $\mathcal{F}(X)$ be the combination of $F_i(X)$ for all X mapped by its domain-defined F_i. Using $\mathcal{F}$ as the c.d.f approximation for $\mathcal{P}$, we observe that, for arbitrary a and b:

$$\begin{aligned}
&\mathcal{P}(a < X \leq b)\\
&= \int_a^b Z_{d_{1:M}}(X)dX\\
&= \int_{-\infty}^b Z_{d_{1:M}}(X)dX - \int_{-\infty}^a Z_{d_{1:M}}(X)dX\\
&= \mathcal{F}(b) - \mathcal{F}(a)\\
&= \sum_{i=1}^{M-1} \frac{|(a,b] \cap (d_i, d_{i+1}]|}{d_{i+1}-d_i}(F_i(d_{i+1}) - F_i(d_i))\\
&= \sum_{i=1}^{M-1} \frac{|(a,b] \cap (d_i, d_{i+1}]|}{d_{i+1}-d_i}(\tau_{i+1} - \tau_i)
\end{aligned} \tag{14}$$

Note that intervals $(-\infty, d_1]$ and $(d_i, \infty]$ have no probability density, hence 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 main and target networks, but also an indicator of information sufficiency in return distribution estimation. The later is important for MARL because it aims to differentiate transition stochasticity from transition non-stationary.

4.2. Likelihood Hysteretic IQN (LH-IQN)

Distributed Q-Learning (Lauer & Riedmiller, 2000), an extremely optimistic method which completely ignores negative updates, yields policies that pursue maximum possible reward, 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 decreasing value estimations at a slower rate. Hysteretic enjoys strong empirical trace record in both tabular and deep network cases, but fails to delay the decrease consistently across state-action space. Leniency Learning (Panait et al., 2006) aims to solve this issue by recording temperature values in state-action space. Individual temperature values, which decrease as the corresponding state-action pairs get updated, is used to control the negative update. However, when applied in large or continuous state and action spaces, not only state-action encoding is 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.

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 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 use the max of β and TDL as the decrease rate. More specifically, learning rate μ_t is given by:

$$\mu_t = \begin{cases} max(\beta, l_{t_{1:M'},d_{1:M}})\mu^-, & \text{if } \delta_t^{\tau,\tau'} \leq 0\\ \mu^-, & \text{otherwise} \end{cases} \tag{15}$$

where μ^- is a base learning rate suitable for the learning task and network architecture. Empirically, β ranging from 0.2 to 0.4 yields best performance, but we later show that LH-IQN is more robust to larger hysteretic values.

Since TDL generally increases as the network trains towards consistency, the amount of optimism/leniency added by hysteretic updates is overall reduced over time. Note that for insufficient information value estimations (more likely due to domain non-stationary than domain stochasticity), TDL still remains small, effectively employing a low learning rate toward such sample.

5. Evaluation

When evaluating and comparing our fully-featured architecture LH-IRQN with simpler models, architectures without certain features do have the corresponding adjective in their acronyms. For example, L-IQN, without 'H' and 'R', lacks hysteretic and recurrent architecture.

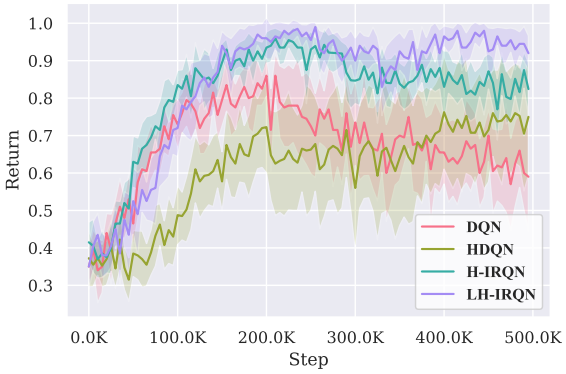

Figure 1. Evaluation on meeting-in-a-grid 4×4 benchmark. both IRQN models performs better than DRQN ones, especially with added Likelihood feature for improved long-term stability.

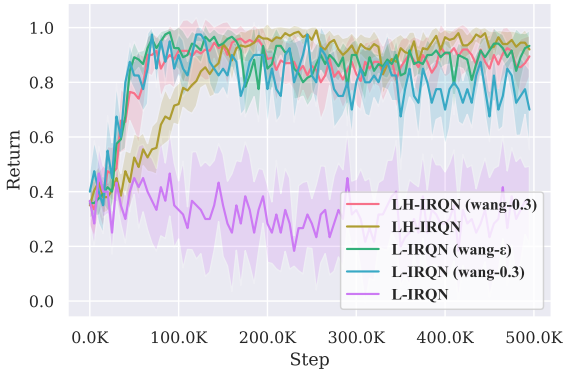

Figure 2. Performance of various Risk Distortion applied to L-IRQN with and without Hysteretic on meeting-in-a-grid 4×4 benchmark; shows that likelihood update works well only in conjunction with either distortion operator or hysteresis.

5.1. Recurrent architectures evaluation on meeting-in-a-grid

We conduct experiments using LH-IQN based architectures. We use recurrence in meeting-in-a-grid domain (Amato et al., 2009) to keep consistency with previous works on the domain (Omidshafiei et al., 2017). 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 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 during training and evaluating, and the quantile embeddings are combined with the output of LSTM layer. Training is decentralized, and the results shown in all Figures are 20 randomly seeded runs.

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. Episode terminates after 40 transitions or upon successful meeting-at-target. Observation include flickering locations of agent themselves and target, and actions result in stochastic transitions.

We first evaluate LH-IRQN performance against HDRQN and HIRQN. As shown in Figure 1, noticing differences in variance, that using quantile network alone or hysteretic-DQN alone not only fail to convergence to optimal policy on meeting-in-a-grid benchmark, but also susceptible to environment stochasticity, producing lesser effective joint policy over time.

[TODO: MAKE FIGURE AND TALK ABOUT LIKELIHOOD TREND DURING TRAINING, SHOW LIKELIHOOD DOMINATES HYSTERETIC OVERTIME AND ROBUST TO DIFFERENT HYSTERETIC VALUES]

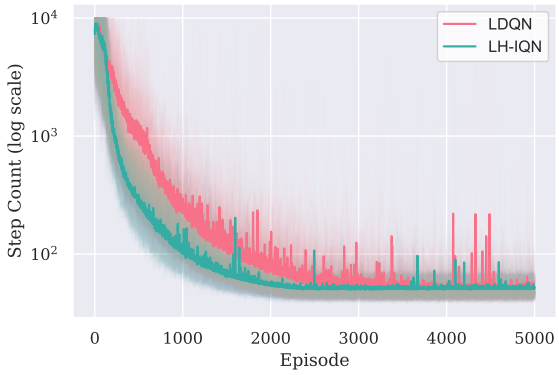

Figure 3. CMOTP benchmark, aggregated three CMOTP variances

5.2. Multi-Agent Object Transportation Problems (CMOTPs)

We evaluate our agents on three variations of CMOTPs (Palmer et al., 2018), consistent with Palmer et al.'s work on LDQN. Also grid-world tasks, CMOTPs requires 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 include more obstacles in the gird-world and stochastic terminal rewards. CMOTPs have 16 by 16 sensory observations with added noise.

Our network architecture is mostly same as LDQN for com-

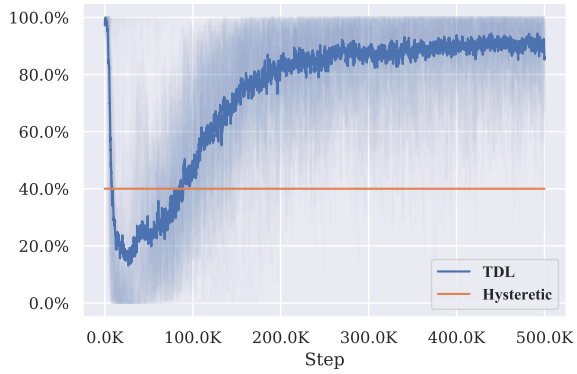

Figure 4. TDL during training of LH-IRQN on meeting-in-a-grid 4×4 benchmark.

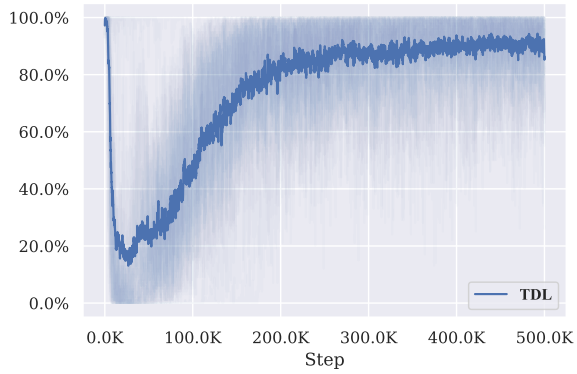

Figure 5. Percentage of $l_t > \beta$ where TDL is used as *decrease rate* instead of hysteretic. Same experiment results as Figure 4, where $\beta = 0.4$.

parability: 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. Noticing from Figure 3 that although both methods converge to joint optimal policy while our method show an improved sample efficiency. The temperature and leniency control parameters were brought from Palmer et al.'s work where they found most suitable for the task; we hypothesize that the temperature is decaying less aggressive than it should be, which is likely due to temperature folding techniques or the hashing space of the autoencoder is larger than needed.

On the other hand, our method which utilize time difference likelihood to guide negative updates, started to learn aggressively early on. Initially we worry that the Q estimates would be not optimistic enough to perform coordinated actions, but the likelihood estimates were able to produce small values in under-explored state-action space while not hesitate to update negatively in explored spaces.

6. Discussion

orthogonal with other improvements such as WDDQN, DUIDQN, state-dependent exploration strategy, etc

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.

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.

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.

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.

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.

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.

7. Appendix

7.1. Results of Individual Variations of CMOTP

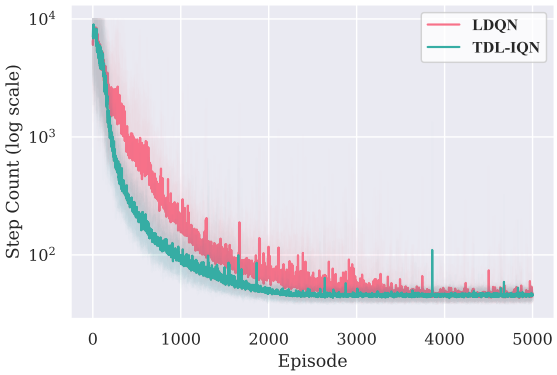

Figure 6. CMOTP Version 1

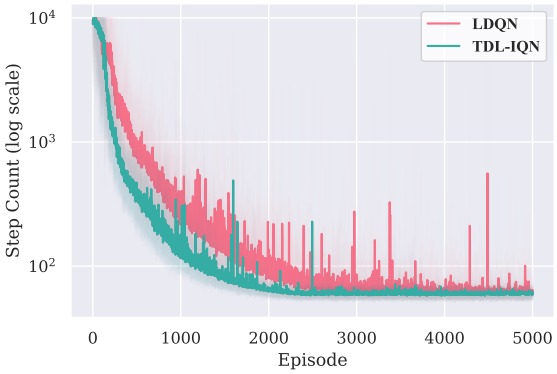

Figure 7. CMOTP Version 2 (Narrow Corredor)

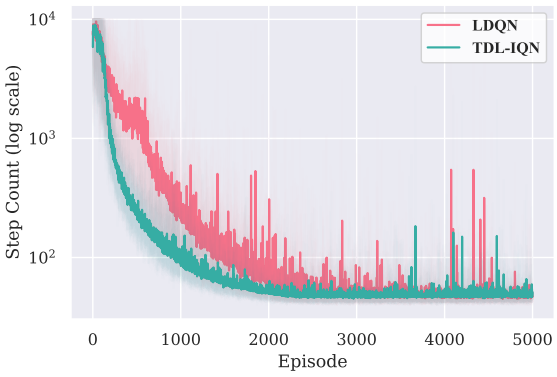

Figure 8. CMOTP Version 3 (Stochastic Reward)