Uncertainty-Based Transformation for Monotonic Value Function Factorization in Multi-Agent Reinforcement Learning

Anonymous Authors[1]

1. Method

In this section, we introduce a novel value function factorization method, Uncertainty-based Transformation (UTRAN), which eliminates the representation limitation of monotonic value function factorization without representing all target Q-values from the stochastic environment.

1.1. Uncertainty Estimation

UTRAN uses a world model that consists of three components to estimate stochasticity. A state representation model encodes the state to continuous vector-valued embeddings. Inspired by RND (Burda et al., 2018), the state representation model is a fixed and randomly initialized network, which provides the target embeddings for the transition model. The transition model predicts the embedding of the future state and its standard deviation. The reward model predicts the reward and its standard deviation,

$$\text{State representation model:} \quad h(z_t|s_t) \tag{1}$$

$$\text{Transition model:} \quad g_\theta(\hat{z}_{t+1}, \hat{\sigma}_t^z|s_t, \mathbf{u}_t) \tag{2}$$

$$\text{Reward model:} \quad v_\varphi(\hat{r}_t, \hat{\sigma}_t^r|s_t, \mathbf{u}_t). \tag{3}$$

Based on Bayesian deep learning method (Kendall & Gal, 2017), we train the transition and the reward model by:

$$L(\theta,\varphi) = \frac{1}{T}\sum_{t=0}^{T-1}\left[\frac{1}{2(\hat{\sigma}_t^r)^2}\delta_r^2 + \frac{1}{2}\log(\hat{\sigma}_t^r)^2 + \frac{1}{d}\sum_{i=1}^{d}\left(\frac{1}{2(\hat{\sigma}_t^{z,i})^2}\delta_z^2 + \frac{1}{2}\log(\hat{\sigma}_t^{z,i})^2\right)\right], \tag{4}$$

where T denotes the episode length, d denotes the dimension of the state embedding z_t, $\delta_z = z_t^i - \hat{z}_t^i$, and $\delta_r = r_t - \hat{r}_t$.

When the state-action pair results in stochastic transitions, the standard deviations are expected to be large. Under this interpretation, we define an indicator function $I(s,\mathbf{u}) = \mathbb{I}_{(\forall\sigma\in\{\hat{\sigma}_t^r,\hat{\sigma}_t^z\}<\alpha)}$, where α is a hyperparameter. $I(s,\mathbf{u}) = 0$ if the given state-action pair has considerable stochasticity.

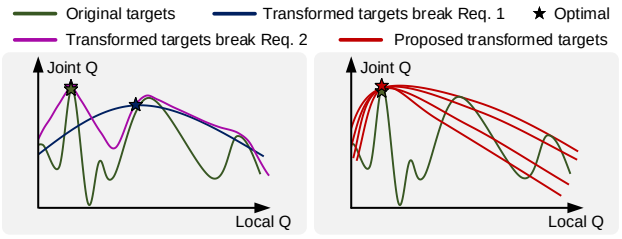

Figure 1. The relations between the original target and different transformed targets.

1.2. Uncertainty-based Transformation

The basic idea of UTRAN is that we should ease the representation for monotonic value factorization due to its limited representational capacity. To solve this limitation, we can project the original target for joint Q-values to the restricted space that monotonic value factorization can represent by only showing the suboptimality for uncooperative actions rather than approximating their exact values.

First, we calculate the original target $y(s_t,\boldsymbol{\tau}_t,\mathbf{u}_t) = r(s_t,\mathbf{u}_t) + \gamma\max_{\mathbf{u}} Q_{tot}(s_{t+1},\boldsymbol{\tau}_{t+1},\mathbf{u};\theta'_Q)$, where $Q_{tot}(s_t,\boldsymbol{\tau}_t,\mathbf{u}_t;\theta_Q) = f_s(Q_1(\tau_1,u_1),...,Q_n(\tau_n,u_n))$ is the joint Q-value function, $\frac{\partial f_s}{\partial Q_a} \geqslant 0, \forall a \in A$, $Q_i(\tau_i,u_i)$ is the per-agent value function, θ'_Q denotes the parameters of a target network that are periodically copied from θ_Q.

Then, we introduce the transformation function:

$$Q^f(s,\boldsymbol{\tau},\mathbf{u}) = \begin{cases} y^f(s,\boldsymbol{\tau},\mathbf{u}) & \Delta(s,\boldsymbol{\tau},\mathbf{u})I(s,\mathbf{u}) > 0 \\ y(s,\boldsymbol{\tau},\mathbf{u}) & \text{otherwise} \end{cases}, \tag{5}$$

where $\Delta(s,\boldsymbol{\tau},\mathbf{u}) = \max_{\mathbf{u}} Q_{tot}(s,\boldsymbol{\tau},\mathbf{u}) - y(s,\boldsymbol{\tau},\mathbf{u})$, and $y^f(s,\boldsymbol{\tau},\mathbf{u})$ is a class of modified targets. Namely, this transformation is a general approach - we can find different $y^f(s,\boldsymbol{\tau},\mathbf{u})$ to represent the suboptimality of the joint action. This transformation should satisfy the following two requirements: **Req. 1**, the optimal policy should not change

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

after the transformation, and **Req. 2**, we can find a monotonic mixing function to represent all transformed targets $Q^f(s, \boldsymbol{\tau}, \mathbf{u})$. Fig. 1 illustrates that the transformation cannot guarantee convergence to the optimal policy if it breaks one of the requirements.

We only replace the suboptimal and deterministic target with $y^f(s, \boldsymbol{\tau}, \mathbf{u})$ to guarantee that we can find a monotonic mixing function f_s to represent all transformed targets $Q^f(s, \boldsymbol{\tau}, \mathbf{u})$. In contrast, we remain the target unchanged if it is characterized as stochastic or greater than the current estimate of the greedy action value. The agent network is trained by minimizing the mean square error between $Q_{tot}(s_t, \boldsymbol{\tau}_t, \mathbf{u}_t; \theta_Q)$ and $Q^f(s, \boldsymbol{\tau}, \mathbf{u})$.

We introduce a practical implementation to achieve such transformation. We define the best per-agent value function to quantify the suboptimality of the joint action:

$$q_a(s, u_a; \theta_{q_a}) = \max_{\mathbf{u}_{-a}} Q(s, u_a, \mathbf{u}_{-a}), \tag{6}$$

where $q_a(s, u_a)$ is less than the optimal Q value if u_a is not part of the real optimal joint action. BAIL (Chen et al., 2020) is applied to approximate this value, and the detailed implementation can be found in Appendix A. Note that $Q_i(\tau_i, u_i)$ is the local agent Q-value for decision-making, while $q_a(s, u_a)$ is only used in training.

Then, we use the minimal value of these best per-agent value functions over all agents as the transformed target:

$$y^f(s, \boldsymbol{\tau}, \mathbf{u}) = \min\{q_a(s, u_a; \theta_{q_a})\}. \tag{7}$$

To better understand the transformation, we analyze its property and visualize the transformed targets. We consider a fully-observable setting, and our notations do not distinguish the concepts of states s and observation-action histories $\boldsymbol{\tau}$.

Theorem 1.1. *Let* $Q(s, \mathbf{u})$ *and* $Q^f(s, \mathbf{u})$ *be the orignal and the transformed target Q-values from* Eq. (5)*, where* $y^f(s, \boldsymbol{\tau}, \mathbf{u})$ *is defined in* Eq. (7)*. Suppose* $\arg\max_{\mathbf{u}} Q(s, \mathbf{u})$ *is unique. Then* $\arg\max_{\mathbf{u}} Q^f(s, \mathbf{u}) = \arg\max_{\mathbf{u}} Q(s, \mathbf{u})$ *and* $Q^f(s, \mathbf{u}) \in \mathcal{Q}^m$.

The proof can be found in Appendix B.

After transformation, all targets $Q^f(s, \mathbf{u})$ lie in the restricted monotonic space $\mathcal{Q}^m$. Namely, the gap between the target and the approximated values by monotonic value factorization, i.e., the representational limitation, is eliminated. In addition, this transformation does not change the optimal action. As a result, monotonic value factorization can converge to the optimal for any original target Q-value function.

In Fig. 2, we show the performance comparison of monotonic value function factorization with original and transformed target values. Fig. 2-a shows the rewards of a 10×10 matrix game, where the suboptimal is filled with random

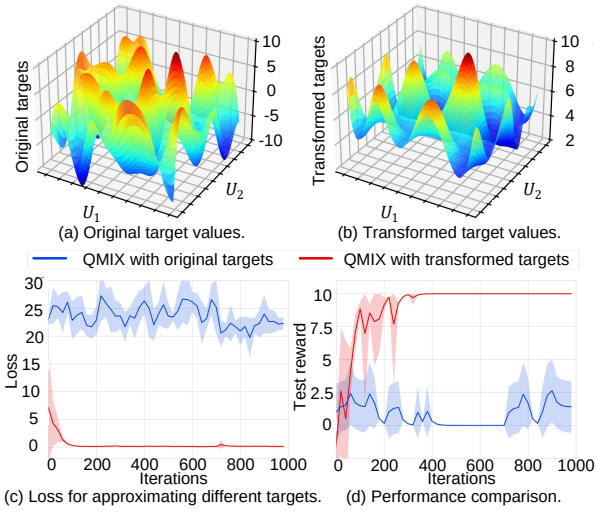

Figure 2. Comparison between the original and the transformed target joint Q-values.

numbers generated uniformly between -10 and 9, and the unique optimal value is +10. These rewards denote the original targets. Fig. 2-b demonstrates the transformed targets. Fig. 2-c and d demonstrate that monotonic value factorization with the original targets cannot converge due to the representational limitation. In contrast, it can perfectly represent all transformed targets by showing that the loss is almost zero and converges to the optimal after 400 iterations.

References

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

A. Training Details for Best Per-agent Value Functions

Inspired by BAIL (Chen et al., 2020), $q_a(s, u_a; \theta_{q_a})$ is trained by minimizing the following loss:

$$L(\theta_{q_a}) = \sum_{t=0}^{T-1} K(s_t, \mathbf{u}_t)\delta(s_t, \mathbf{u}_t)^2 + \lambda\|\theta_{q_a}\|, \tag{8}$$

$$\text{where } K(s_t, \mathbf{u}_t) = \begin{cases} k & \delta(s_t, \mathbf{u}_t) > 0 \\ 1 & \text{otherwise} \end{cases}, \tag{9}$$

where $\delta(s_t, \mathbf{u}_t) = y_a(s_t, u_t^a) - q_a(s, u_t^a)$, $y_a(s_t, u_t^a) = \hat{r}(s_t, \mathbf{u}_t) + \gamma \max_{u^a} q_a(s_{t+1}, u^a)$, λ is the regularisation coefficients, and $k \gg 1$. Note that $\hat{r}(s_t, \mathbf{u}_t)$ is the expected reward return by the reward model. The best per-agent value function prioritizes the action whose target is greater than the current estimate.

B. Target Transformation

Theorem 1.1 *Let $Q(s, \mathbf{u})$ and $Q^f(s, \mathbf{u})$ be the orignal and the transformed target Q-values from* Eq. (5)*, where $y^f(s, \boldsymbol{\tau}, \mathbf{u})$ is defined in* Eq. (7)*. Suppose $\arg\max_{\mathbf{u}} Q(s, \mathbf{u})$ is unique. Then $\arg\max_{\mathbf{u}} Q^f(s, \mathbf{u}) = \arg\max_{\mathbf{u}} Q(s, \mathbf{u})$ and $Q^f(s, \mathbf{u}) \in \mathcal{Q}^m$.*

Proof. We consider only a fully-observable setting for ease of representation. Thus our notations do not distinguish the concepts of states and observation-action histories. In addition, when more than one optimal policy exists, most Q-learning algorithms fail to converge to a stable point. Thus, we consider an additional assumption in Assumption 1.

Assumption 1 (Unique Optimal Solution)**.** For each state $s \in S$, the optimal policy $\mathbf{u}_Q^* = \arg\max_{\mathbf{u}} Q(s, \mathbf{u})$ is unique.

Under this assumption, assume $\max_{\mathbf{u}_{-a}} Q(s, u_a, \mathbf{u}_{-a})$ for all u_a are perfectly represented.

Then $Q(s, \mathbf{u}) \leqslant \min_{a \in \mathbb{A}}\{\max_{\mathbf{u}_{-a}} Q(s, u_a, \mathbf{u}_{-a})\}$.

According to the transformation defined in Eq. (5), we have:

$$\begin{cases} Q^f(s, \mathbf{u}_Q^*) = Q(s, \mathbf{u}_Q^*) \\ Q^f(s, \mathbf{u}) = \min\{\max_{\mathbf{u}_{-i}} Q(s, u_i, \mathbf{u}_{-i})\}, \forall u_i \in \mathbf{u} \neq \mathbf{u}_Q^* \text{ and } u_i \notin \mathbf{u}_Q^* \end{cases}. \tag{10}$$

Then, $Q^f(s, \mathbf{u}_Q^*) > Q^f(s, \mathbf{u})$ for all $\mathbf{u} \neq \mathbf{u}_Q^*$. Therefore, $\arg\max_{\mathbf{u}} Q^f(s, \mathbf{u}) = \arg\max_{\mathbf{u}} Q(s, \mathbf{u})$.

Suppose $\exists f_s(s, Q_1(s, u_1), ..., Q_n(s, u_n)) = Q^f(s, \mathbf{u}), \forall \mathbf{u} = (u_1, ..., u_n) \in \mathbf{u}$. Then

$$\begin{cases} f_s(s, Q_a(u_a), \boldsymbol{Q}_{-a}(\mathbf{u}_{-a})) = \min\{\max_{\mathbf{u}_{-a}} Q(s, u_a, \mathbf{u}_{-a}), \max_{\mathbf{u}_{-i}} Q(s, u_i, \mathbf{u}_{-i})\}, \forall i \in -a \\ f_s(s, Q_a(u'_a), \boldsymbol{Q}_{-a}(\mathbf{u}_{-a})) = \min\{\max_{\mathbf{u}_{-a}} Q(s, u'_a, \mathbf{u}_{-a}), \max_{\mathbf{u}_{-i}} Q(s, u_i, \mathbf{u}_{-i})\}, \forall i \in -a \end{cases}, \tag{11}$$

where $\boldsymbol{Q}_{-a}(\mathbf{u}_{-a}) = \{Q_i(s, u_i)\}_{i \in -a}$.

Let $Q_a(s, \mathbf{u}_a) \geqslant Q_a(s, \mathbf{u}'_a)$ if $\max_{\mathbf{u}_{-a}} Q(s, u_a, \mathbf{u}_{-a}) \geqslant \max_{\mathbf{u}_{-a}} Q(s, u'_a, \mathbf{u}_{-a}), \forall a \in A$.

Then, according to Eq. (11), $f_s(s, Q_a(u_a), \boldsymbol{Q}_{-a}(\mathbf{u}_{-a})) \geqslant f_s(s, Q_a(u'_a), \boldsymbol{Q}_{-a}(\mathbf{u}_{-a}))$.

Thus we can find $f_s(s, Q_1(s, u_1), ..., Q_n(s, u_n))$ to represent $Q^f(s, \mathbf{u})$, $\forall \mathbf{u} = (u_1, ..., u_n) \in \mathbf{u}$, where $\frac{\partial f_s}{\partial Q_a} \geqslant 0$. And so $Q^f(s, \mathbf{u}) \in \mathcal{Q}^m, \forall s \in S, \forall \mathbf{u} \in \mathbf{u}$. □