SMIX(λ): Enhancing Centralized Value Functions for Cooperative Multi-Agent Reinforcement Learning

Chao Wen*, Xinghu Yao*, Yuhui Wang, Xiaoyang Tan

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
{chaowen, xinghuyao, y.wang, x.tan}@nuaa.edu.cn

Overview

- The paradigm of centralized training decentralized execution (CTDE) has become popular in multi-agent reinforcement learning (MARL).
- Our proposed method SMIX(λ), built upon QMIX, aims to learn a stable and generalizable centralized value function (CVF) for CTDE-type MARL methods.

Key ideas:

- Remove the unrealistic centralized greedy assumption during the learning phase;
- Adopt an experience-replay style off-policy training for better sample efficiency without importance sampling.
- Use λ-return to balance the trade-off between bias and variance and to deal with the environment's non-Markovian property;

Relaxing the CGB Assumption in Learning

What's CGB? To learn a generalizable CVF, current methods generally adopt the following centralized greedy behavior (CGB) assumption:

$$\underset{\mathbf{a}}{\operatorname{argmax}} Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{a}) = \begin{pmatrix} \underset{a^1}{\operatorname{argmax}} Q^1\left(\tau^1, a^1\right) \\ \vdots \\ \underset{a^N}{\operatorname{argmax}} Q^N\left(\tau^N, a^N\right) \end{pmatrix} \tag{1}$$

How to relax? Use SARSA-style updating rule instead of Q-learning to avoid the CGB assumption in Eq. (1).
Whats the problem? SARSA is an on-policy method and only considers one-step return.

Off-Policy Learning in Multi-Agent Settings

Denote behavior policy as $\boldsymbol{\mu}$ and the target policy as $\boldsymbol{\pi}$, the policy evaluation strategy can be expressed as follows:

$$Q(\tau, \mathbf{a}) \leftarrow Q(\tau, \mathbf{a}) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{t\geq 0} \gamma^t \left(\prod_{i=1}^{t} \rho_i\right) \delta_t^{\boldsymbol{\pi}}\right] \tag{2}$$

where

$$\delta_t^{\boldsymbol{\pi}} = r_{t+1} + \gamma \mathbb{E}_{\boldsymbol{\pi}} Q\left(\boldsymbol{\tau}_{t+1}, \cdot\right) - Q\left(\boldsymbol{\tau}_t, \mathbf{a}_t\right) \tag{3}$$

We try to achieve off-policy learning without importance sampling in multi-agent settings because:

	IS	without IS (ours)
ρ_i	$\rho_i = \frac{\boldsymbol{\pi}(\mathbf{a}_i\|\boldsymbol{\tau}_i)}{\boldsymbol{\mu}(\mathbf{a}_i\|\boldsymbol{\tau}_i)}$	$\rho_i = 1.0$
variance	large	low
calculate $\boldsymbol{\pi}(\mathbf{a}_i\|\boldsymbol{\tau}_i)$	impractical	practical

Table: Off-policy with/without Importance Sampling (IS) in multi-agent settings. $\boldsymbol{\pi}(\mathbf{a}_i|\boldsymbol{\tau}_i) = \prod_j^n \pi^j$ is the joint policy of all the agents, and calculate this quantity is impractical when the number of the agents n is large.

Though relaxing $\rho_i = 1.0$ introduces bias, it's practical due to

- Off-policy experiences tend to be heavily correlated to the current policy (Fujimoto, Meger, and Precup 2019).
- We prove that SMIX(λ) has the similar convergence property to Q(λ) if $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ are sufficiently close.
- We keep a relatively small buffer size to ensure $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ are sufficiently close.

λ-Return

We use the λ-return as the TD target estimator:

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)} \tag{4}$$

where $G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n \mathbb{E}_{\boldsymbol{\pi}} Q\left(\boldsymbol{\tau}_{t+n}, \mathbf{a}_{t+n}; \theta^-\right)$ is n-step return and θ^- are parameters of the target network.
Plugging this into Eq. (2) and setting $\rho_i = 1.0$ for all i, we have the following update rule,

$$Q(\tau, \mathbf{a}) \leftarrow Q(\tau, \mathbf{a}) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{t\geq 0} \gamma^t \left(G_t^{\lambda} - Q\left(\tau_t, \mathbf{a}_t\right)\right)\right] \tag{5}$$

Loss function:

$$\mathcal{L}_t(\theta) = \sum_{i=1}^{b}\left[\left(y_i^{tot} - Q_{tot}^{\pi}(\tau, \mathbf{a}; \theta)\right)^2\right] \tag{6}$$

Algorithm 1 Training Procedure for SMIX(λ)

```
 1: Initialize the behavior network with parameters $\theta$, the target network with parameters $\theta^-$, empty replay buffer $\mathcal{D}$ to capacity
    $N_{\mathcal{D}}$, training batch size $b$
 2: for each training episode do
 3:     for each episode do
 4:         for $t = 1$ to $T-1$ do
 5:             Obtain the partial observation $\mathbf{o}_t = \{o_t^1, \cdots, o_t^N\}$ for all agents and global state $s_t$
 6:             Select action $a_t^i$ according to $\epsilon$-greedy policy w.r.t agent $i$'s decentralized value function $Q^i$ for $i = 1, \cdots, N$
 7:             Execute joint action $\mathbf{a}_t = \{a^1, a^2, \cdots, a^N\}$ in the environment
 8:             Obtain the global reward $r_{t+1}$, the next partial observation $o_{t+1}^i$ for each agent $i$ and next global state $s_{t+1}$
 9:         end for
10:         Store the episode in $\mathcal{D}$, replacing the oldest episode if $|\mathcal{D}| \geq N_{\mathcal{D}}$
11:     end for
12:     Sample a batch of $b$ episodes $\sim$ Uniform($\mathcal{D}$)
13:     Calculate $\lambda$-return targets $y_i^{tot} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$, where $G_t^{(n)} = r_{t+1}+\gamma r_{t+2}+\cdots+\gamma^n \mathbb{E}_{\boldsymbol{\pi}} Q(\boldsymbol{\tau}_{t+n}, \mathbf{a}_{t+n}; \theta^-)$
14:     Update $\theta$ by minimizing $\sum_{t=1}^{T-1}\sum_{i=1}^{b}[(y_i^{tot} - Q_{tot}^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a}; \theta))^2]$
15:     Replace target parameters $\theta^- \leftarrow \theta$ every $C$ episodes
16: end for
```

Results - StarCraff II

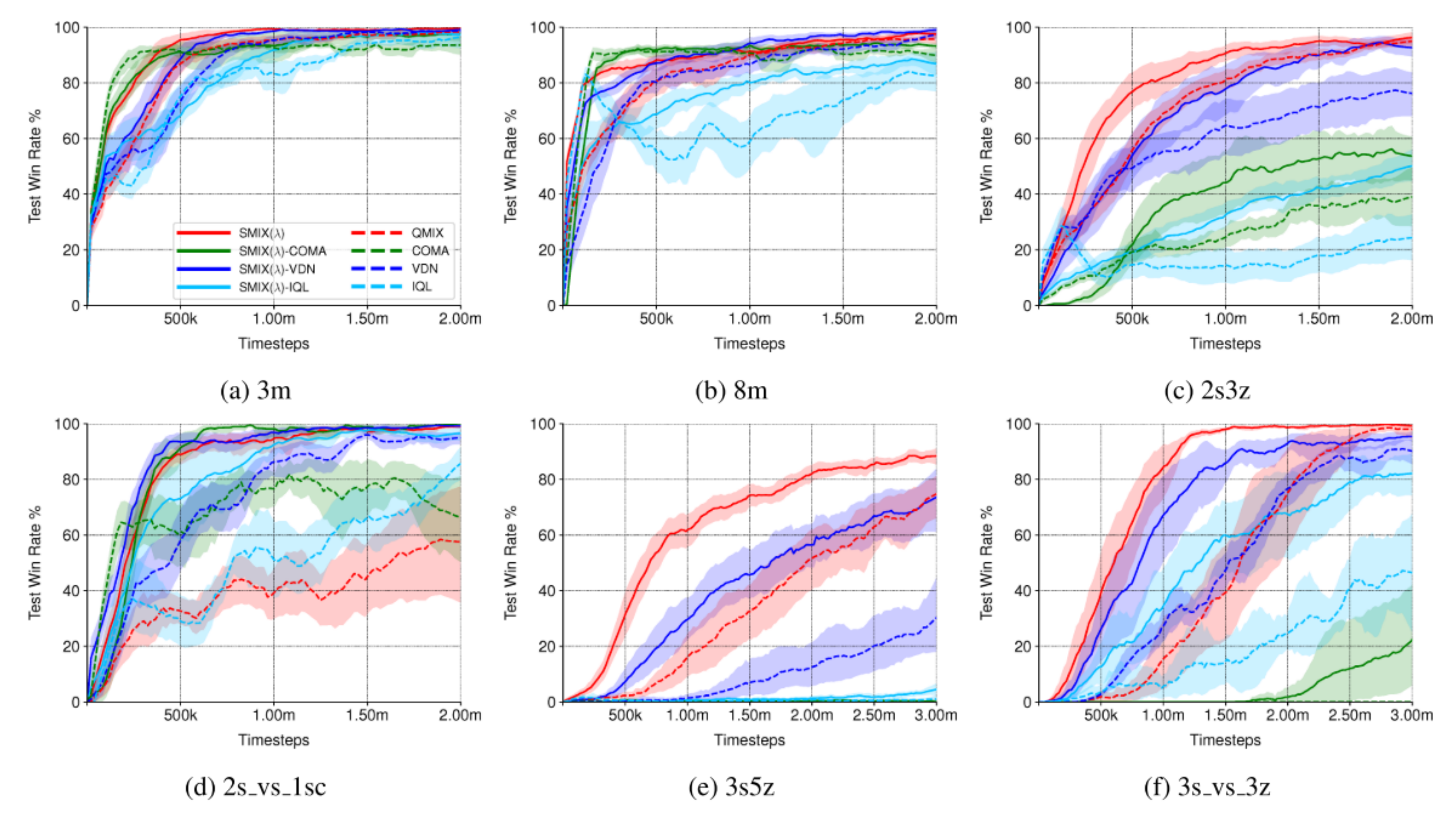

Figure 2: Test win rates for our methods (SMIX(λ), SMIX(λ)-COMA, SMIX(λ)-VDN, SMIX(λ)-IQL) and comparison methods (QMIX, COMA, VDN, IQL) in six different scenarios. The performance of our methods and their counterparts are shown with solid and dashed lines of the same color, respectively. The mean and 95% confidence interval are shown across 10 independent runs. The legend in (a) applies across all plots.

- SMIX(λ) outperforms most comparison methods both in terms of the learning speed and final performance.
- Our CVF estimation procedure is a general method. Existing CTDE-type methods can achieve performance improvement by incorporating our CVF estimation procedure.