Exponentially Weighted Imitation Learning for Batched Historical Data

Qing Wang[1], Jiechao Xiong[1], Lei Han[1], Peng Sun[1], Han Liu[12], Tong Zhang[1]

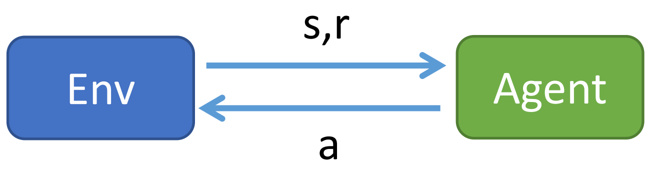
[1]Tencent AI Lab [2]Northwestern University

Outline

- Background:
  - Reinforcement Learning: from Simulation
  - Our Problem: Deep Policy Learning from Data
- Method:
  - Imitation Learning
  - Off-policy Reinforcement Learning
  - Monotonic Advantage Re-Weighted IL
- Experiments:
  - HFO, TORCS, King of Glory
- Discussion

Reinforcement Learning

1. Environment provide initial state to the agent.
2. Agent observes the state and takes an action.
3. Environment receives the action, outputs a reward, and evolves to the next state.
4. Agent observes the next state and takes another action.



s : state in state space $\mathcal{S}$

a : action in action space $\mathcal{A}$

r : reward $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Example: Breakout (Atari game)

state: (stacked) pixels on the screen

action: left, right, ...

reward: +1 for hit a brick

terminal: drop the ball

or clear two screen of bricks

(V. Minh et al., Human-level control through deep reinforcement learning)

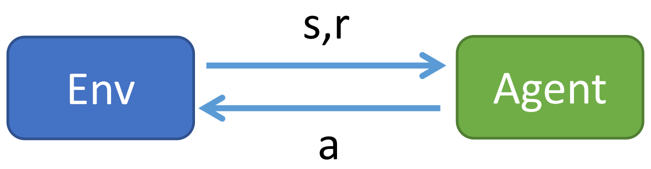
Reinforcement Learning

- **Initial State**: the distribution of initial state s_0 is denoted by $d_0 \in \Delta(\mathcal{S})$
- **Stochastic Policy**: a map from state to a probabilistic distribution in action space

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})$$

- **Objective**: the performance of a policy is measured by its expected sum of discounted reward

$$\eta(\pi) = \mathbb{E}_{d_0,\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$



s : state in state space $\mathcal{S}$

a : action in action space $\mathcal{A}$

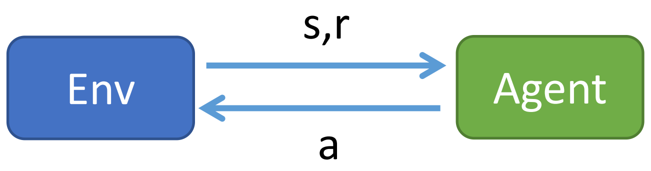
r : reward $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$

A Mathematical Formulation of RL

- Markov Decision Process (MDP)

$$(\mathcal{S}, \mathcal{A}, P, r, d_0, \gamma)$$

$\mathcal{S}$ is the state space
$\mathcal{A}$ is the action space
P is the transition probability $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$
r is the reward function $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$
d_0 is the distribution of initial state
$\gamma \in (0, 1)$ is the discount factor



s : state in state space $\mathcal{S}$

a : action in action space $\mathcal{A}$

r : reward $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$

- Markovian: *given current state, future is independent of past*
  - or else, a POMDP

Different definitions exist, e.g. finite horizon vs infinite horizon

State Value, Action Value, and Advantage

- We define the value of a state as

$$V^{\pi}(s_t) = \mathbb{E}_{\pi|s_t} \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$$

- And the value of an action (under current state) as

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi|s_t,a_t} \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$$

- The advantage is defined as the difference of Q and V

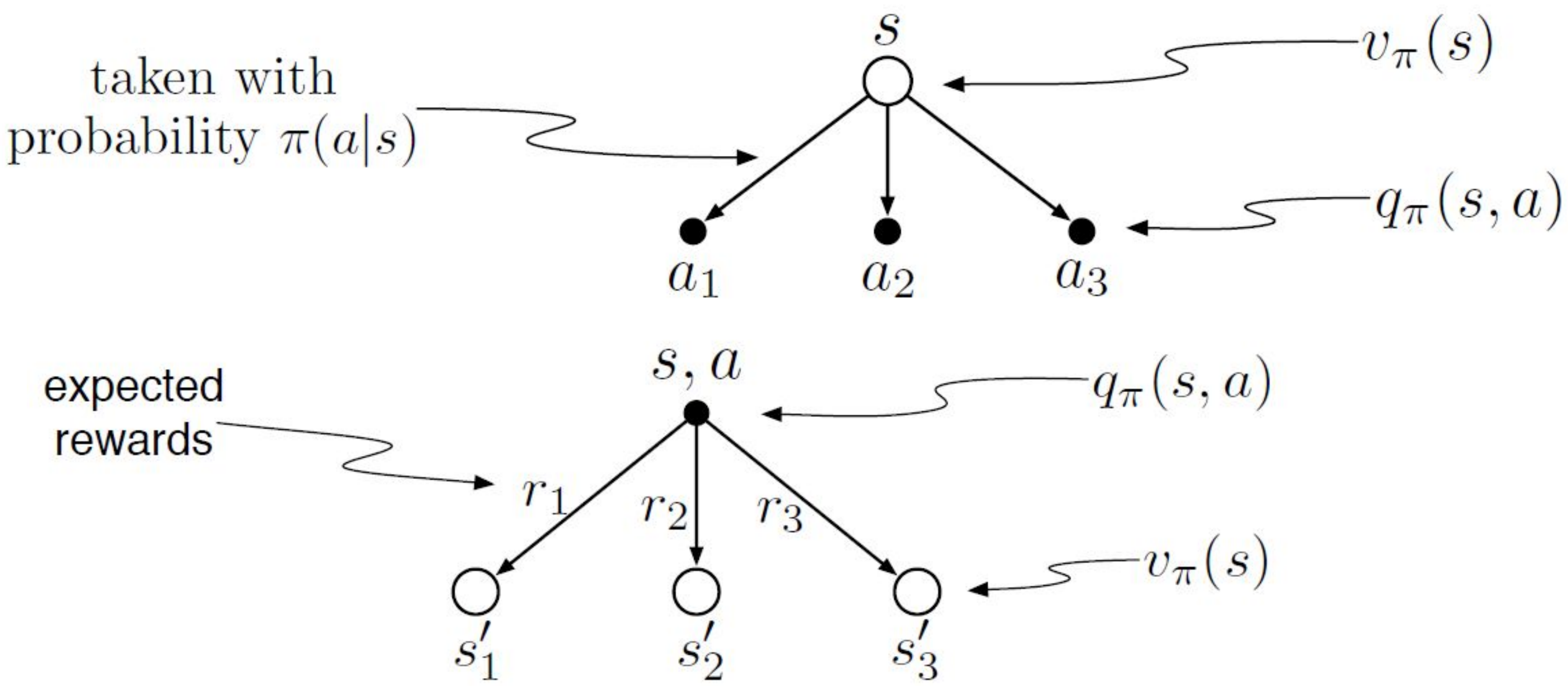
$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

- They could be estimated with Monte-Carlo method or Bootstrap

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$ or $$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

See (Sutton and Barto 2017, Section 6.1)

State Value, Action Value, and Advantage



See (Sutton and Barto 2017, Section 3.5)

DQN, DDPG, TRPO

DQN

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

DDPG

$$\begin{aligned} \nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right] \\ &= \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t} \right] \end{aligned}$$

TRPO

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A_{\theta_{\text{old}}}(s, a) \\ &\text{subject to } \overline{D}_{\mathrm{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \delta. \end{aligned}$$

See DQN (Mnih et al., 2013), DDPG (Lillicrap et al., 2016), TRPO (John Schulman et al., 2015)

Our Problem

- Deep Policy Learning from Data (without a Simulator !)
- $\pi : s \rightarrow a$ for complex environment
- Examples:
  - Autonomous Driving, Sports, Game bot, ...

TORCS

Robocup 2D

王者荣耀

Imitation Learning (behavior cloning)

- Learn the map from s to a
- Let $\boxed{\pi_\theta(a \mid s)}$ close to $\boxed{\pi(a \mid s)}$

target policy behavior policy

- e.g. minimize KL-divergence

$$L_p = D_{\mathrm{KL}}^{d_\pi}(\pi || \pi_\theta) = -\mathbb{E}_{s \sim d_\pi(s), a \sim \pi(a|s)} \log \pi_\theta(a|s) + C$$

- where

$$D_{\mathrm{KL}}^{d}(\pi' || \pi) = \sum_s d(s) \sum_a \pi'(a|s) \log \frac{\pi'(a|s)}{\pi(a|s)}$$

$$d_\pi(s) = (1-\gamma)\mathbb{E}_{d_0,\pi} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}(s_t = s)$$

$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$

$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$

$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$

use of reward information?

quality of expert trajectories?

More on imitation learning, see e.g. (Bain and Sommut, 1999) and (Ross et al., 2011)

Off-policy Deep Reinforcement Learning

- Value iteration:
  - DQN, DDPG, Hybrid action?
  - How to make sure the learned policy performs similar to (or better than) the behavior policy?
- Policy iteration:
  - TRPO?, Retrace?
  - What if we do not have probability of bahavior policy (common in practice)?

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

See DQN (Mnih et al., 2013), DDPG (Lillicrap et al., 2016), TRPO (John Schulman et al., 2015), and Retrace (Remi Munos et al., 2016)

Challenges

- Large state space
  - tabular algorithm? function approximation?
- Hybrid action space
  - discrete action: 0, 1, 2, 3, ..
  - continuous action: (x, y), r, θ
- Unknown behavior policy
  - hard for off-policy policy gradient method (Schulman et al., 2015)

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

$$s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots$$

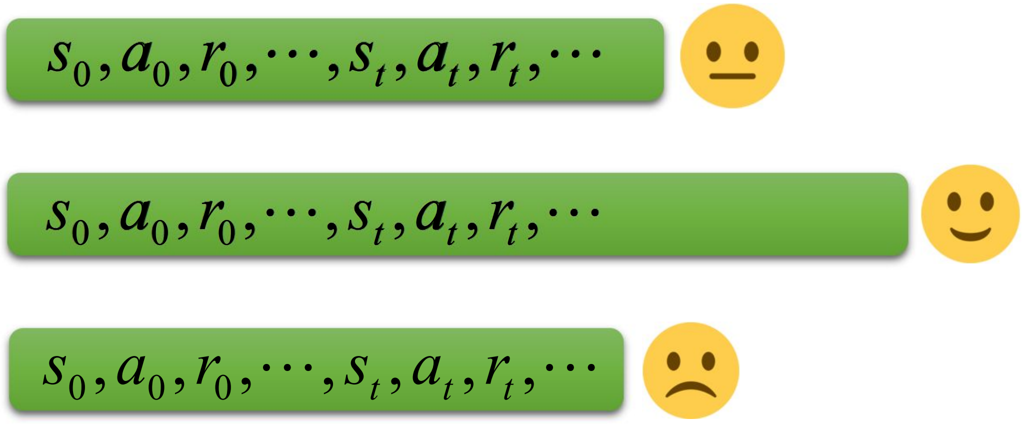
$$\frac{\pi_\theta(a|s)}{\boxed{\pi(a|s)}} \beta A^\pi(s, a)$$

explicit value needed!

Also see (John Schulman et al., 2015) and (Remi Munos et al., 2016) for off-policy correction

Can we do better imitation learning?

- action: Good vs Bad
- look ahead in the historic data to determine what action results in good consequence.
- What if we imitate good action only?
- What if we put larger sample weight on good actions?



See TRPO (John Schulman et al., 2015) and Retrace (Remi Munos et al., 2016)

Imitate a better policy?

- Is there a better policy?
- Consider two policy π and $\tilde{\pi}$ satisfying:

$$\tilde{\pi}(a|s) \geq \pi(a|s) \quad A^{\pi}(s,a) \geq 0$$

$$\tilde{\pi}(a|s) \leq \pi(a|s) \quad A^{\pi}(s,a) \leq 0$$

- then $\tilde{\pi}$ is uniformly as good as, or better than π , i.e.

$$V^{\tilde{\pi}}(s) \geq V^{\pi}(s), \forall s \in S.$$

A simple proof can be derived similarly as in (Sutton and Barto, 1998, Equation 4.8)

Imitate a better policy?

- We can imitate $\tilde{\pi}$ instead of π
- For example, we can choose[1]:

$$\tilde{\pi} \propto \pi \exp(\beta A^{\pi})$$

- then $\tilde{\pi}$ is a better policy, we can imitate this one by

$$\begin{aligned}
\arg\min_{\theta} D^{d}_{\mathrm{KL}}(\tilde{\pi}||\pi_\theta) &= \arg\max_{\theta} \sum_{s} d(s) \sum_{a} \tilde{\pi}(a|s) \log \pi_\theta(a|s) \\
&= \arg\max_{\theta} \sum_{s} d(s) \exp(C(s)) \sum_{a} \pi(a|s) \boxed{\exp(\beta A^{\pi}(s,a))} \log \pi_\theta(a|s)
\end{aligned}$$

better action, larger weight

[1] The derivation of exp() is related to a few previous works e.g. (Peters et al., 2010), (Azar et al., 2012)

Monotonic Advantage Re-Weighted IL

Algorithm 1 Monotonic Advantage Re-Weighted Imitation Learning (MARWIL)

Input: Historical data $\mathcal{D}$ generated by π, hyper-parameter β.

For each trajectory τ in $\mathcal{D}$, estimate advantages $\widehat{A}^{\pi}(s_t, a_t)$ for time $t = 1, \cdots, T$.

Maximize $\sum_{\tau\in\mathcal{D}} \sum_{(s_t,a_t)\in\tau} \boxed{\exp(\beta \widehat{A}^{\pi}(s_t, a_t))} \log \pi_\theta(a_t|s_t)$ with respect to θ.

monotonic advantage reweighting

also works with other formulation, e.g. ReLU instead of Exp

$$\sum_a \pi(a|s)((\beta A^{\pi}(s,a))_+ + \epsilon) \log \pi_\theta(a|s)$$

A lower bound on policy improvement

The performance of a policy is measured by its **expected sum of discounted reward**:

$$\eta(\pi) = \mathbb{E}_{d_0,\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

For practical usage, we usually seek a parametric approximation of $\tilde{\pi}$. The following proposition gives a lower bound of policy improvement for the parametric policy π_θ.

Proposition 2. *Suppose we use parametric policy π_θ to approximate the improved policy $\tilde{\pi}$ defined in Formula 3, we have the following lower bound on the policy improvement*

$$\eta(\pi_\theta) - \eta(\pi) \geq -\frac{\sqrt{2}}{1-\gamma}\delta_1^{\frac{1}{2}} M^{\pi_\theta} + \frac{1}{(1-\gamma)\beta}\delta_2 - \frac{\sqrt{2}\gamma\epsilon_{\pi}^{\tilde{\pi}}}{(1-\gamma)^2}\delta_2^{\frac{1}{2}} \tag{8}$$

where $\delta_1 = \min(D_{KL}^{d_{\tilde{\pi}}}(\pi_\theta||\tilde{\pi}), D_{KL}^{d_{\tilde{\pi}}}(\tilde{\pi}||\pi_\theta))$, $\delta_2 = D_{KL}^{d_\pi}(\tilde{\pi}||\pi)$, $\epsilon_{\pi}^{\pi'} = \max_s |\mathbb{E}_{a\sim\pi'} A^\pi(s,a)|$, *and* $M^\pi = \max_{s,a} |A^\pi(s,a)| \leq \max_{s,a} |r(s,a)|/(1-\gamma)$.

Experiments

- We consider 4 policy loss in our experiments

IL $$L_p = D_{\mathrm{KL}}^{d_\pi}(\pi||\pi_\theta) = -\mathbb{E}_{s\sim d_\pi(s),a\sim\pi(a|s)} \log \pi_\theta(a|s) + C$$

PG $$L_p = -\mathbb{E}_{s\sim d_\pi(s),a\sim\pi(a|s)}(\beta A^\pi(s,a)+1)\log \pi_\theta(a|s) + C$$

PGIS $$L_p = D_{\mathrm{KL}}^{d_\pi}(\pi||\pi_\theta) - (1-\gamma)\beta L^{d_\pi,\pi}(\pi_\theta)$$
$$= -\mathbb{E}_{s\sim d_\pi(s),a\sim\pi(a|s)}\left(\frac{\pi_\theta(a|s)}{\pi(a|s)}\beta A^\pi(s,a) + \log \pi_\theta(s,a)\right) + C$$

MARWIL $$L_p = D_{\mathrm{KL}}^{d}(\tilde{\pi}||\pi_\theta) = -\mathbb{E}_{s\sim d_\pi(s),a\sim\pi(a|s)} \log(\pi_\theta(a|s))\exp(\beta A^\pi(s,a)) + C$$

PGIS (Policy Gradient with Importance Sampling) use the same policy loss considered in TRPO (Schulman et al., 2015)

Experiments

Algorithm 2 Stochastic Gradient Algorithm for MARWIL

Input: Policy loss L_p being one of $\boxed{9}$ to $\boxed{12}$. base policy π, parameter m, c_v.
Randomly initialize π_θ. Empty replay memory D.
Fill D with trajectories from π and calculate R_t for each (s_t, a_t) in D.
for $i = 1$ **to** N **do**
  Sample a batch $B = \{(s_k, a_k, R_k)\}_m$ from D.
  Compute mini-batch gradient $\nabla_\theta \widehat{L}_p$, $\nabla_\theta \widehat{L}_v$ of B.
  Update θ: $-\Delta\theta \propto \nabla_\theta \widehat{L}_p + c_v \nabla_\theta \widehat{L}_v$
end for

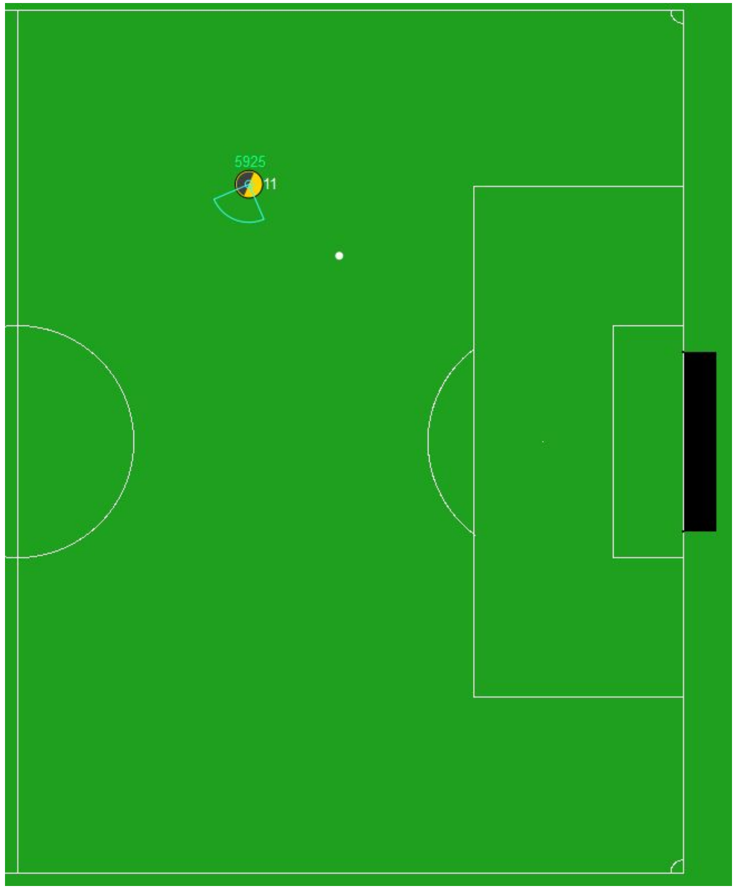
Experiments: HFO

- Half Field Offense:
- state: 59 floats
- action: Dash(r,theta), Turn(theta), Kick(r,theta)

Dash

Turn

Kick



Matthew Hausknecht, https://github.com/LARG/HFO/blob/master/doc/manual.pdf

Experiments: HFO

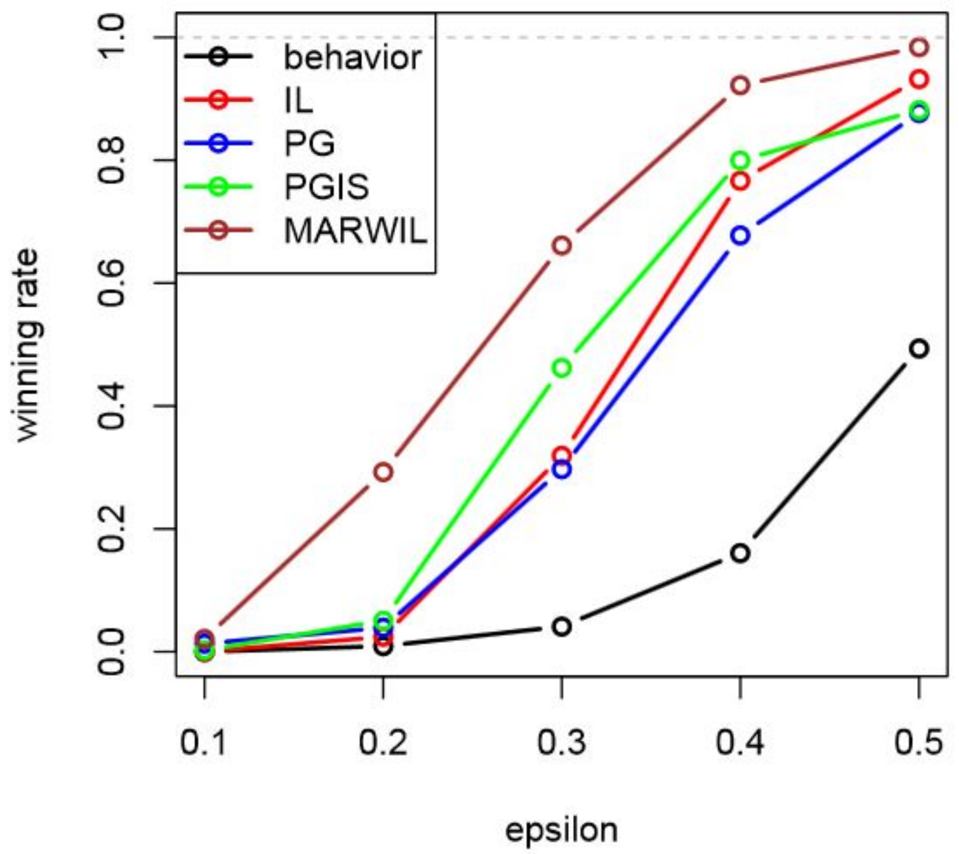
- We add different level of noise in the historic data:

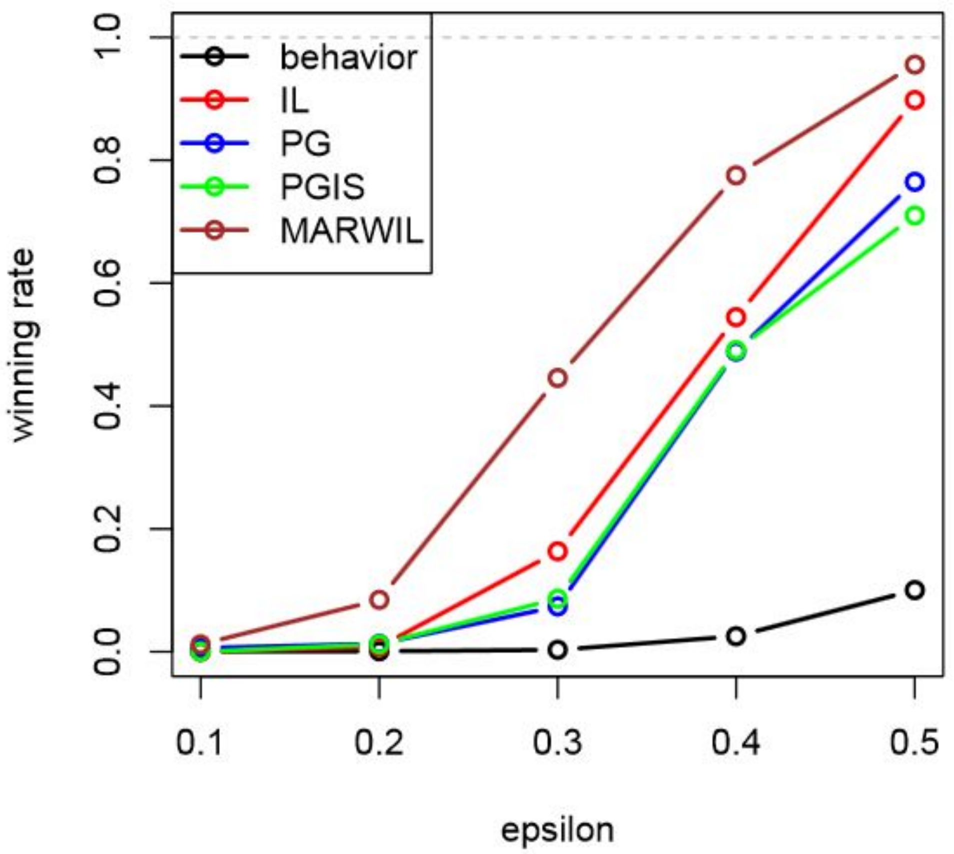
$$a_t \sim \begin{cases} \pi_{\text{perfect}}(\cdot|s_t) + N(0,\sigma) & \text{w.p. } \epsilon \\ \pi_{\text{random}}(\cdot|s_t) + N(0,\sigma) & \text{w.p. } 1-\epsilon \end{cases}$$

- the policy is modeled as

$$\pi_\theta((k, x_k)|s) = p_\theta(k|s)N(x_k|\mu_{\theta,k}, \sigma), \quad k \in \{1,2,3\}, x_k \in \mathbb{R}^2$$

Experiments: HFO





Experiments: TORCS

- Raw image input to steering angle

64x64x3

[-π, π]

- We vary the parameter beta: $-\mathbb{E}_{s\sim d_\pi(s),a\sim\pi(a|s)}\log(\pi_\theta(a|s))\exp(\beta A^\pi(s,a))$

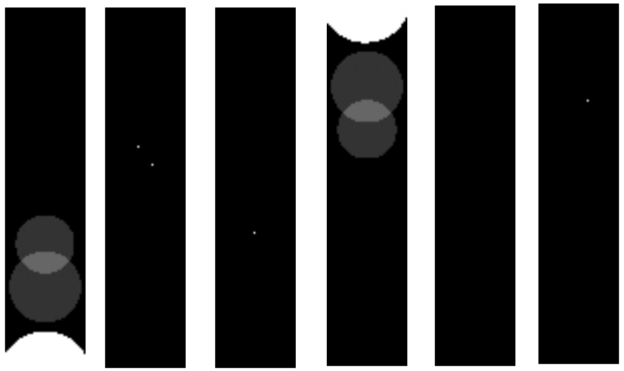
Table 1: Performance of PG and MARWIL in TORCS, where $\beta = 0$ is the case of IL. Different β are tested in the experiments. The performance is evaluated on the sum of rewards per episode.

β	0.0	0.25	0.5	0.75	1.0
PG	2710	6396	6735	6758	7152
MARWIL	(2710)	5583	6832	7670	9492

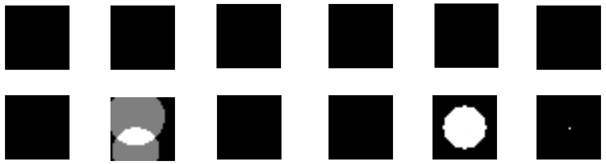
Bernhard Wymann et al., TORCS, The Open Racing Car Simulator. http://www.torcs.org

Experiments: 王者荣耀

- state:

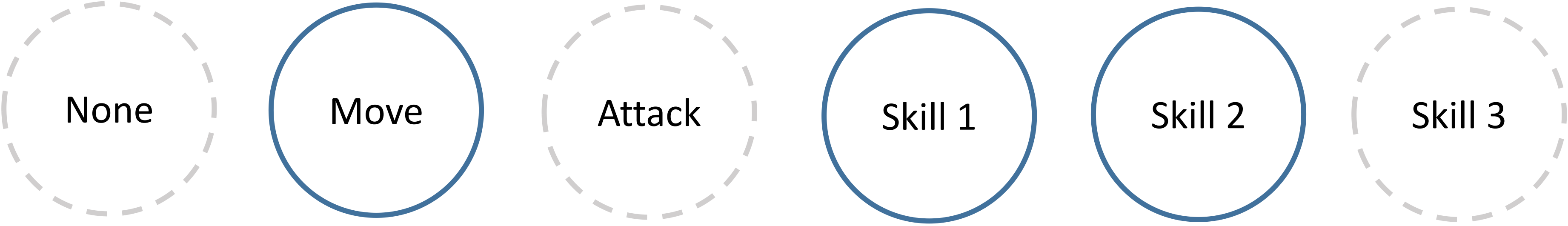


+



+ dense + sparse

- action: 貂蝉: 6 discrete type



King of Glory, see e.g. (Daniel R. Jiang et al., 2018)

Experiments: 王者荣耀

- Network structure

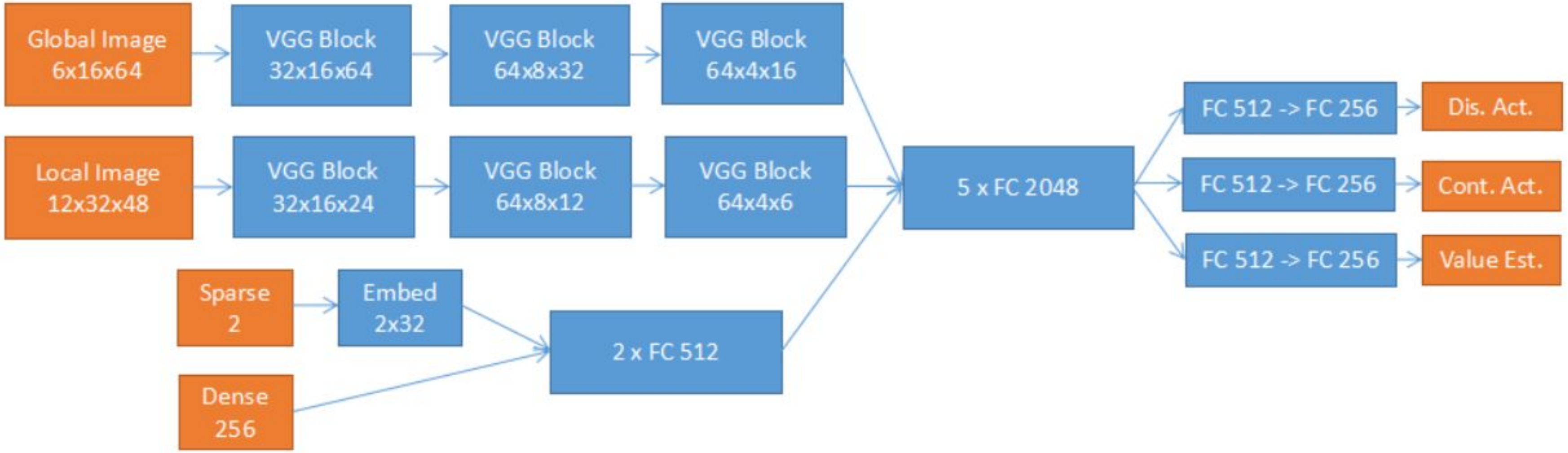


Figure 2: Network structure for our AI agent in King of Glory

Discussion & Future work

- Imitating a better policy, by monotonic advantage reweighting
  - suitable for large state space and hybrid action space
  - do not require complete knowledge of behavior policy
  - robust under complex function approximation (with lower bound)

- The method also works in full reinforcement learning
  - improve over self-generated trajectories

Questions?

Thanks ~ 😃

Backup slides

The performance of a policy π is measured by its expected discounted reward:

$$\eta(\pi) = \mathbb{E}_{d_0,\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

where $\mathbb{E}_{d_0,\pi}$ means $s_0 \sim d_0$, $a_t \sim \pi(a_t|s_t)$, and $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. We omit the subscript d_0 when there is no ambiguity. In [Kakade and Langford, 2002], a useful equation has been proved that

$$\eta(\pi') - \eta(\pi) = \frac{1}{1-\gamma} \sum_s d_{\pi'}(s) \sum_a \pi'(a|s) A^\pi(s, a)$$

where d_π is the discounted visiting frequencies defined as $d_\pi(s) = (1-\gamma)\mathbb{E}_{d_0,\pi} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}(s_t = s)$ and $\mathbf{1}(\cdot)$ is an indicator function. In addition, define $L^{d,\pi}(\pi')$ as

$$L^{d,\pi}(\pi') = \frac{1}{1-\gamma} \sum_s d(s) \sum_a \pi'(a|s) A^\pi(s, a)$$

then from [Schulman et al., 2015, Theorem 1], the difference of $\eta(\pi')$ and $\eta(\pi)$ can be approximated by $L^{d_\pi,\pi}(\pi')$, where the approximation error is bounded by total variance $D_{\mathrm{TV}}^{d_\pi}(\pi', \pi)$, which can be further bounded by $D_{\mathrm{KL}}^{d_\pi}(\pi'||\pi)$ or $D_{\mathrm{KL}}^{d_\pi}(\pi||\pi')$.

Backup slides

Consider the problem

$$\tilde{\pi} = \arg\max_{\pi' \in \Pi}((1-\gamma)\beta L^{d_\pi,\pi}(\pi') - D_{\mathrm{KL}}^{d_\pi}(\pi'||\pi)) \tag{3}$$

which has an analytical solution in the policy space Π [Azar et al., 2012, Appendix A, Proposition 1]

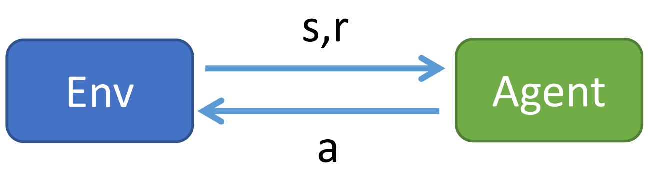
$$\tilde{\pi}(a|s) = \pi(a|s)\exp(\beta A^\pi(s,a) + C(s)) \tag{4}$$

where $C(s)$ is a normalizing factor to ensure that $\sum_{a\in\mathcal{A}} \tilde{\pi}(a|s) = 1$ for each state s. Then

$$\begin{aligned} \arg\min_\theta D_{\mathrm{KL}}^{d}(\tilde{\pi}||\pi_\theta) &= \arg\max_\theta \sum_s d(s) \sum_a \tilde{\pi}(a|s) \log \pi_\theta(a|s) \\ &= \arg\max_\theta \sum_s d(s)\exp(C(s)) \sum_a \pi(a|s)\exp(\beta A^\pi(s,a)) \log \pi_\theta(a|s) \end{aligned} \tag{5}$$

Remark: MDP vs SG

Markov Decision Process
(MDP)



Example:
· Most RL env in OpenAI Gym.
· Game against fixed opponent.

Stochastic Game
(SG)

Example:
· Chess, Go
· VizDoom, MOBA, Starcraft

- We have the simulator for the game, but we do not have the ``simulator'' of opponent human player (as part of the environment)
- (Recall AlphaGo, this work is similar to learning from human replay (AlphaGo Master). It is also possible to learn with pure self-play (AlphaGo Zero))

AlphaGo as Imitating MCTS policy

- Self-play with MCTS:

$$(\boldsymbol{p}, v) = f_\theta(s) \quad \text{and} \quad l = (\underbrace{\boxed{z}}_{\text{Step with self-play}} - v)^2 - \underbrace{\boxed{\boldsymbol{\pi}}}_{\text{MCTS search}}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$

- MCTS as a policy improvement operator.
  - In our case: monotonic advantage reweighting, instead of MCTS
- Self-play with search as a policy evaluation operator.

See (Silver et al., 2016)

Reference

- R. Sutton and A. Barto, Reinforcement Learning: An Introduction, 2017
- J. Schulman et al., Trust Region Policy Optimization, 2015
- V. Minh et al., Playing Atari with Deep Reinforcement Learning, 2013
- D. Silver et al., Mastering the game of go with deep neural networks and tree search, 2016
- T. P. Lillicrap et al., Continuous Control with Deep Reinforcement Learning, 2016
- R. Munos et al., Safe and efficient off-policy reinforcement learning, 2016
- M. Bain et al., A Framework for Bahavioural cloning, 1999
- S. Ross et al., A Reduction of Imitation Learning and Structured Prediction to no-regret Online Learning, 2011
- J. Peters et al., Relative Entropy Policy Search, 2010
- M. G. Azar et al., Dynamic Policy Programming, 2012
- D. Jiang et al., Feedback-based tree search for reinforcement learning, 2018