Robust Deep RL: A Soft-Actor-Critic approach with Adversarial Perturbation on State Observations

Machine Learning Laboratory

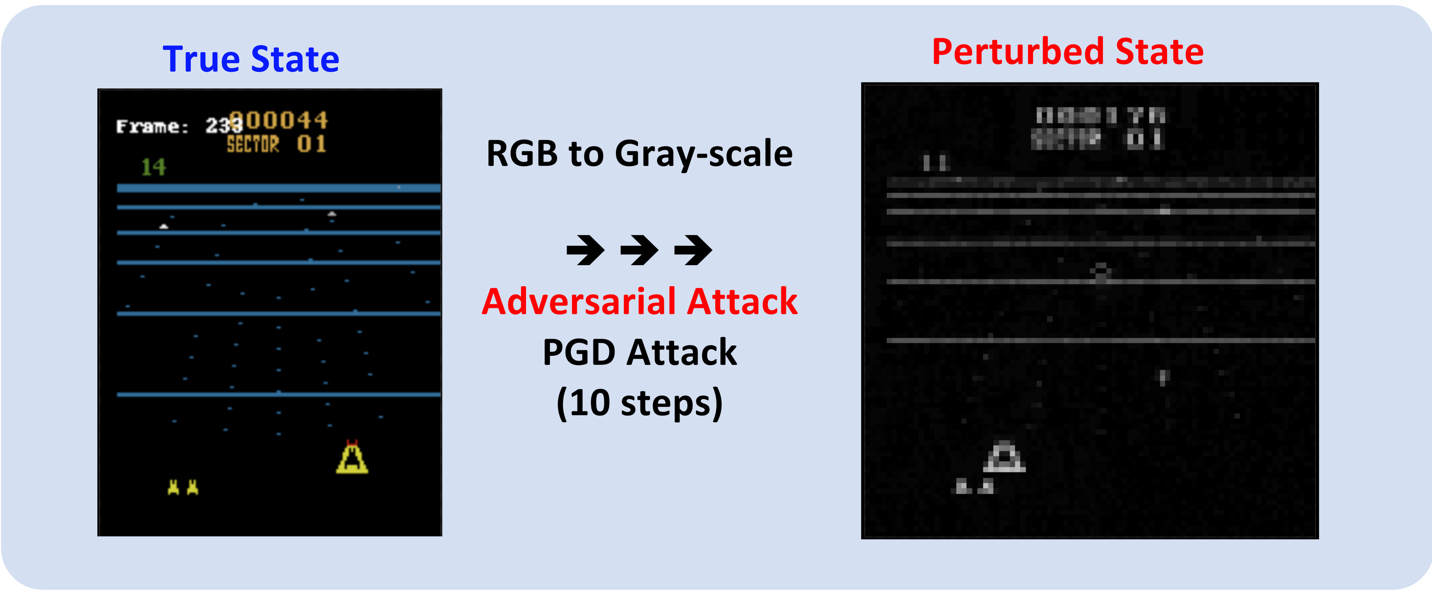
Kyungphil Park
Advisor: Jungseul Ok

1. Introduction

Key Idea: SA-MDP + Adversarial Regularizor

Vulnerability of Perturbations on Observations

Perturbations on observations do not change the environment directly, but can mislead the agent into making sub-optimal or wrong decisions.



SA-MDP(State Adversarial – Markov Decision Process)

:modified MDP by the perturbation on state observation

$$\tilde{V}_{\pi\circ\nu}(s) = \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[R(s,a,s') + \gamma \tilde{V}_{\pi\circ\nu}(s')\right]$$

$$\tilde{Q}_{\pi\circ\nu}(s,a) = \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[R(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|\nu(s'))\tilde{Q}_{\pi\circ\nu}(s',a')\right].$$

$$\tilde{V}_{\pi\circ\nu^*}(s) = \min_{\nu} \tilde{V}_{\pi\circ\nu}(s), \quad \tilde{Q}_{\pi\circ\nu^*}(s,a) = \min_{\nu} \tilde{Q}_{\pi\circ\nu}(s,a)$$

Robust Policy Regularizor(Adversarial Regularizor)

:worst case perturbation value from B(s)

$$\mathcal{R}_{\mathrm{DDPG}}(\theta_\pi) = \sqrt{2/\pi}(1/\sigma) \sum \max_{\hat{s}\in B(s)} \|\pi_{\theta_\pi}(s) - \pi_{\theta_\pi}(\hat{s})\|_2$$

$$\mathcal{R}_{\mathrm{DQN}}(\theta) := \sum_s \max\{\max_{\hat{s}\in B(s)} \max_{a\neq a^*} Q_\theta(\hat{s},a) - Q_\theta(\hat{s},a^*(s)), -c\}.$$

(Zhang et al.,(2021) NeurIPS, 21)

- SA-PPO(Proximal Policy Optimization)
- SA-DDPG(Deep Deterministic Policy Gradient)
- SA-DQN(Deep Q-Netwroks)

Robust policy regularizer is related to total variation distance or KL-divergence on perturbed policies.

- Option1. Solve Regularizer using SGLD
- Option2. Solve Regularizer using convex relaxation

Motivation

Improve limitations of SA-PPO, SA-DDPG, SA-DQN

→ High cost in terms of Sampling Complexity
→ Brittle with respect to their Hyperparameters

2. SA-SAC(State Adversarial – Soft Actor Critic)

Soft Actor Critic

- *Object Function*

$$\max_{\theta} \mathop{\mathrm{E}}_{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right]$$

→ **Sample Efficient : Off Policy + Entropy Regularization**
→ **Improvement of instability issues with Hyperparameter**
→ **Continuous action space : value-based + policy-based**

SA-SAC:

In our work, we frequently need to solve a minimax problem:
→ minimizing the policy loss for a worst case (maximum regularizor value)

$$\min_{\theta} \max_{\phi\in\mathbb{S}} g(\theta,\phi)$$

- *SA – SAC Regularizor*

$$R_{SAC}(\theta_\pi, \bar{s}_i) := \sum_i \max_{\bar{s}_i\in B_p(s_t,\epsilon_t)} ||\pi_{\theta_\pi}(s_i) - \pi_{\theta_\pi}(\bar{s}_i)||_2$$

- *Object Function*

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} (\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) - \kappa_{SAC} \nabla_{\theta_\pi} \bar{R}_{SAC})$$

- *Pseudo Code(training part)*

if *it's time to update* **then**
  for *j in range(however many updates)* **do**
    Randomly sample a batch of transitions, $B = \{(s,a,r,s',d)\}$ from D;
    targets for the Q functions:
     $\rightarrow y(r,s',d) = r + \gamma(1-d)(\min_{i=1,2} Q_{\phi_{targ,i}}(s',\bar{a}') - \alpha log\pi_\theta(\bar{a}'|s')),\ \bar{a}'\ \pi_\theta(.|s')$;
    Update Q-functions by one step of gradient descent using:
     $\rightarrow \nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} (Q_{\phi_i}(s,a) - y(r,s',d))^2$ for $i = 1,2$;
    Define $R_{SAC}(\theta_\pi, \bar{s}_i) := \sum_i \max_{\bar{s}_i\in B_p(s_t,\epsilon_t)} ||\pi_{\theta_\pi}(s_i) - \pi_{\theta_\pi}(\bar{s}_i)||_2$;
    Solve $R_{SAC}(\theta_\pi)$ using convex relaxations:
    $\rightarrow \bar{R}_{SAC}(\theta_\pi) := \mathrm{ConvexRelaxUB}(R_{SAC}, \theta_\pi, \bar{s}_i \in B_p(s_i,\epsilon_t))$;
    Update policy by one step of gradient ascent using:
     $\rightarrow \nabla_\theta \frac{1}{|B|} \sum_{s\in B} (\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) - \kappa_{SAC} \nabla_{\theta_\pi} \bar{R}_{SAC})$;
    where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(.|s)$ which is differentiable wrt θ
    via the reparametrization trick.;
    Update target networks with: $\phi_{targ,i} \leftarrow \rho\phi_{targ,i} + (1-\rho)\phi_i$ for $i = 1,2$;
  end
end

- *PGD Attack(Projected Gradient Descent)*

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha\, \mathrm{sgn}(\nabla_x L(\theta, x, y))\right)$$

Adversarial attack setting is under a suite of strong white box attacks.

PGD Attack perturbes the true state of observation and perturbed state is the input of the Agent

3. Experiment

Enviroment : BeamRider, SapceInvaders (OpenAI Gym Atari game)

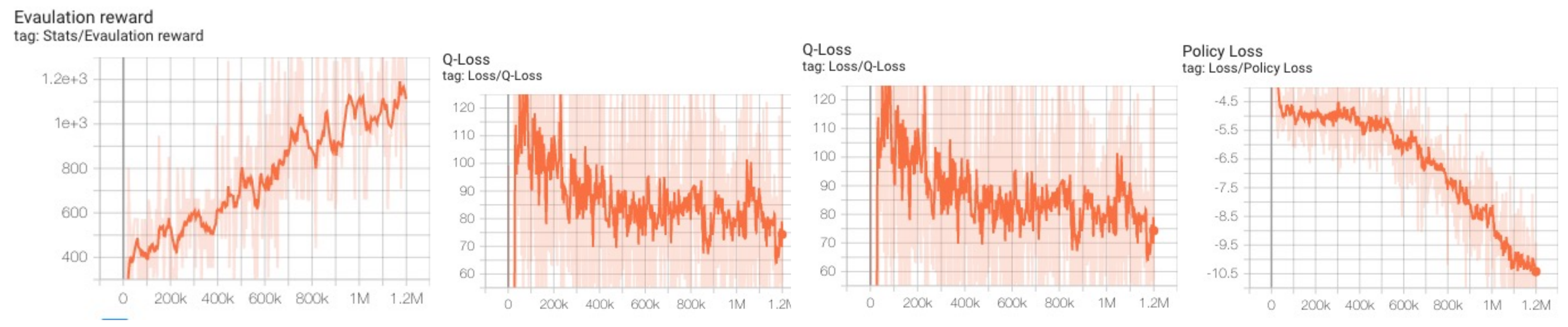
- **SAC(vanilla)**

Trained Steps: trained 1.2M step

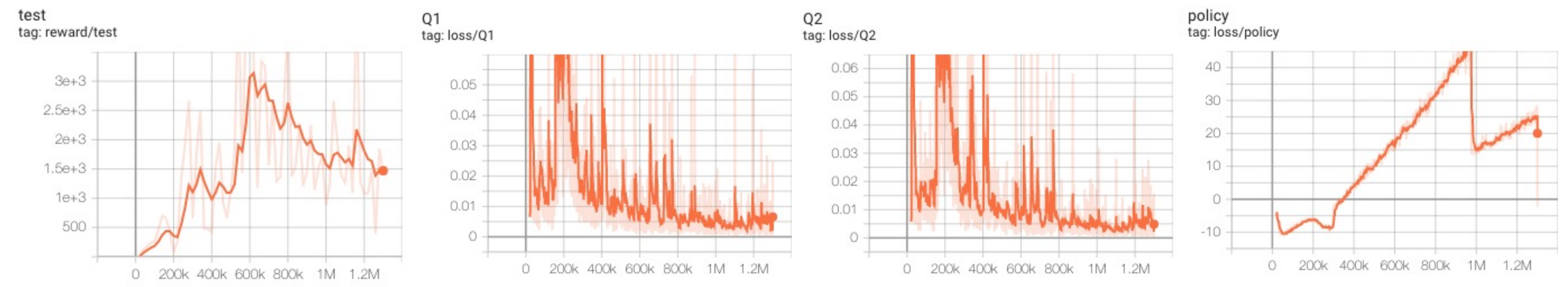
- **SA-SAC(convex)**

Trained Steps: 300K(vanilla) + 900K(SA_SAC) → trained 1.2M steps
l_∞ (*norm perturbation budget* ϵ) $= 1/255$

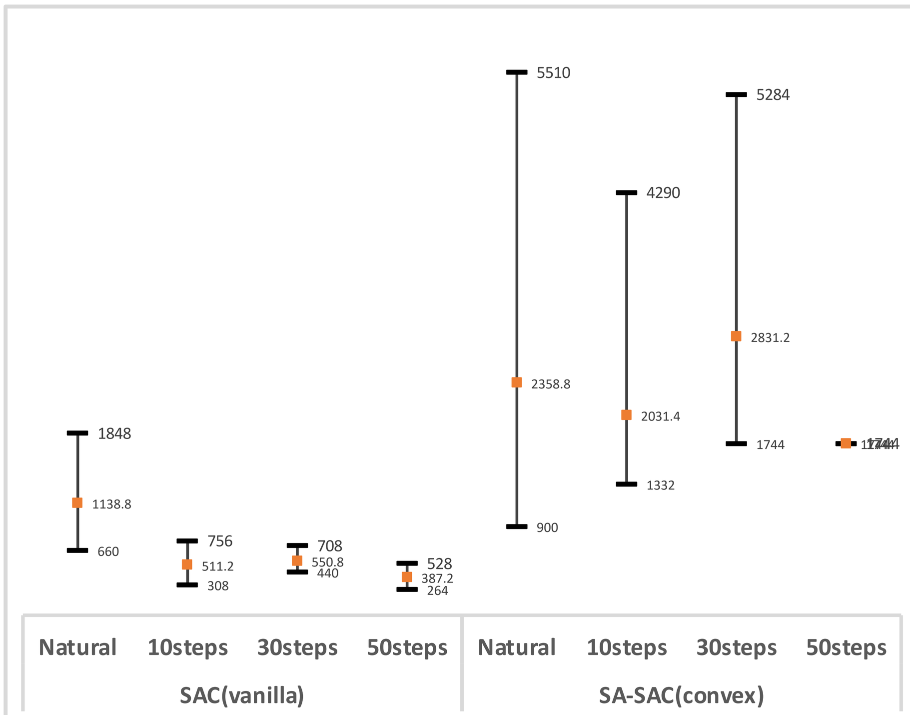
- **SAC(vanilla)**

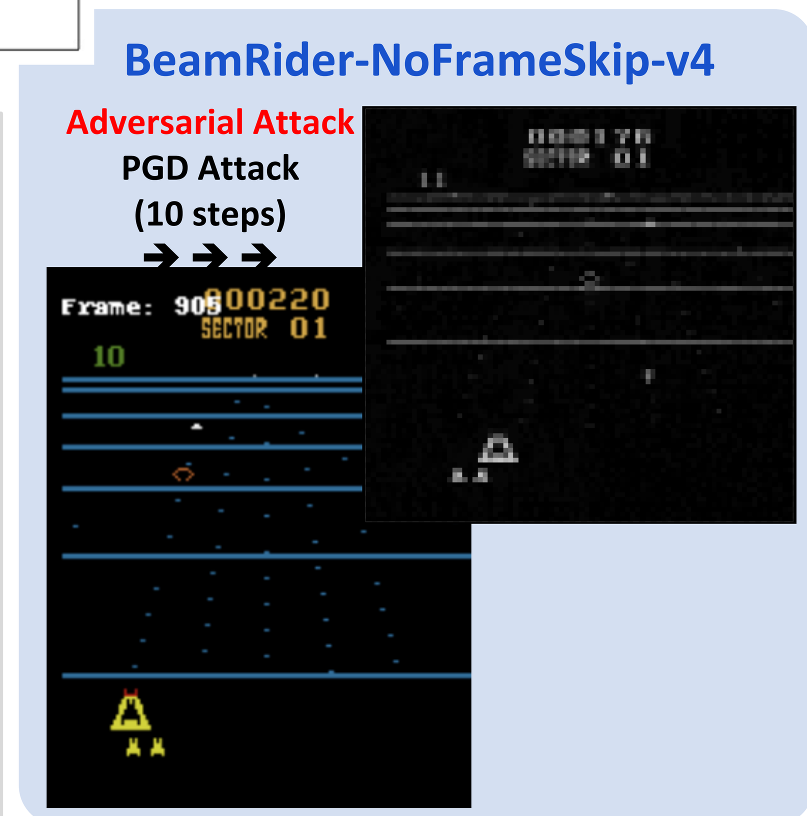


- **SA-SAC(convex)**



Envrionment		BeamRider
$l_\infty\epsilon$		1/255
SAC (vanilla)	Natural Reward	1138.8±295.3(660.0~1848.0)
	PGD Attack Reward(10 steps) (10 episodes)	511.2±139.4(308.0~756.0)
	PGD Attack Reward(30 steps) (10 episodes)	550.8±100.1(440.0~708.0)
	PGD Attack Reward(50 steps) (10 episodes)	387.2±105.6(264.0~528.0)
SA-SAC (convex)	Natural Reward	2358.8±1388.7(900.0~5510.0)
	PGD Attack Reward(10 steps) (10 episodes)	2031.4±898.1(1332.0~4290.0)
	PGD Attack Reward(30 steps) (5 episodes)	2831.2±1264.4(1744.0~5284.0)
	PGD Attack Reward(50 steps) (1 episodes)	1744.0±0(1744.0)





4. Concluding Remark

We improved the robustness SAC agents under a suite of strong white box adversarial attack(PGD). SA-SAC can be used at the both discrete and continuous action space. When regularizer value trained, we have to find the proper $\kappa_{\{SAC\}}$ value in a heuristic way, so solving this problem is considered a future work.