Adversarial Attacks on Neural Network Policies

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, Pieter Abbeel
Motivation

- Understanding the working of a neural network
  - To make the model robust and interpretable
- Real-world implications of adversarial examples
  - For example, adding strategically-placed paint to the surface of a road to confuse an autonomous car's lane-following policy

Types of attacks:

Whitebox - requires access to the model

Blackbox - only requires access to model outputs
Background

Adversarial Examples

Perturb input such that change is not noticeable to human observer

Maximize harm caused by an adversarial perturbation

Tradeoff between computational cost and probability of success
Fast Gradient Sign Method [Goodfellow ’ICLR2015]

- Existing method for generating adversarial examples in computer vision
- Makes a linear approximation of a deep model
- Closed form solution to maximization problem
- Cheap and works well in practice

Given a linear function $g(x) = w^T x$, the optimal adversarial perturbation $\eta$ that satisfies $\|\eta\|_\infty < \epsilon$ is

$$\eta = \epsilon \text{sign}(w)$$

This maximizes change in output for adversarial example $x^* = x + \eta$,

$$g(x^*) = w^T x + w^T \eta$$
Deep Reinforcement Learning

Authors run FGSM attack on policies trained on:

- Deep Q-Networks (DQN)
- Trust Region Policy Optimization (TRPO)
- Asynchronous Advantage Actor-Critic (A3C)
Experimental setup

Atari 2600 games: Chopper Command, Pong, Space Invaders, Seaquest

Algorithms: DQN, TRPO, A3C

Network Architecture: 2 Conv layers and a FC layer - Same as DQN paper

Input: Concatenation of last 4 images (scaled and resized)

Output of policy: Distribution of possible actions

Norm constraints for FGSM: $L_1$ and $L_2$ norm of $\eta$, along with $L_\infty$ norm

For each game and algorithm, trained 5 policies and identified top performing trained policies, then averaged return over 10 rollouts
Vulnerability to White-box attacks

Figure 2: Comparison of the effectiveness of $\ell_\infty$, $\ell_2$, and $\ell_1$ FGSM adversaries on four Atari games trained with three learning algorithms. The average return is taken across ten trajectories. Constraint on FGSM perturbation: $\ell_\infty$-norm $\ell_2$-norm $\ell_1$-norm.
Transferability in black-box attacks

- Transferability Across Policies

Adversary has access to training algorithm and hyperparameters of the target policy network but not its random initialization.

- Transferability Across Training Algorithms

Adversary has no access to training algorithm and hyperparameters.
Vulnerability to Black-box attacks

Figure 4: Transferability of adversarial inputs for policies trained with TRPO. Type of transfer: ■ algorithm □ policy ■ none
Future work

- Developing defenses against adversarial attacks
  - Adversarial training
  - Detecting adversarial input at test time

- Next papers focus on
  - More adversarial attacks
  - Adversarial robustness
Tactics of Adversarial Attack on Deep Reinforcement Learning Agents

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, Min Sun
Strategically-Timed Attack
Strategically-Timed Attack - Motivation

- Adversary perturbs observation at every time step -> more likely to be detected
- Pong: if ball is close to panel, it is more detrimental for the paddle to move in the wrong direction
- **Uniform attack:** attack at every timestep
Strategically-Timed - Methods

- **Preference Function:** $c$
- Preference of most preferred over least preferred action
- Search for perturbation which changes classification to least preferred action

$$c(s_t) = \max_{a_t} \pi(s_t, a_t) - \min_{a_t} \pi(s_t, a_t)$$
Strategically-Timed - Results

- Test A3C and DQN
- 25% of times as the uniform attack
- Portion of timesteps attacked vs Reward
Enchanting Attack
Enchanting Attack - Motivation

- Want to lure the agent to a particular state (target state)
- Pacman: Convince pacman to head to a certain location in the maze
- Could be dangerous in an autonomous driving setting
Enchanting - Methods (Future State Prediction)

- Model to predict future state, based on current state and action sequence
- Success is L2 distance between actual state and predicted state

\[ A_{t:t+H} = \{a_t, \ldots, a_{t+H}\} \]
\[ s_{t+H}^M = M(s_t, A_{t:t+H}) \]
Future State Prediction

\[ M(S, \{L, L\}) = ? \]
Future State Prediction

M(S, {L, L}) = D?
Enchanting - Methods (Action Planning)

- Sample many action sequences
- Find sequence where final state is closest to target state
- Then craft adversarial perturbation for first action (to match best sequence)
- Repeat for every time step

\[ A_{t:t+H} = \left\{ a_t, \ldots, a_{t+H} \right\} \]

\[ \left\{ A^n_{t:t+H} \right\}_{n=1}^N \]
Action Planning (Sample Many Action Sequences)
Action Planning (Sample Many Action Sequences)
Action Planning (Sample Many Action Sequences)
Action Planning (Closest Sequence)
Action Planning (Perturbation)
Action Planning (Start attack again from s_t + 1)
Action Planning (Could choose a different path)
Enchanting - Results

- Attack is successful if reaches state within tolerance of 1
- 70% success rate - $H < 40$ (except Seaquest and ChopperCommand)
- Sequence length of actions vs Success Rate
Robust Adversarial Reinforcement Learning (RARL)

Lerrel Pinto, James Davidson, Rahul Sukthankar and Abhinav Gupta
Motivation

- **Real world policy learning:**
  - Expensive
  - Dangerous
  - Time-intensive
  - Results in scarce data and overfitting

- **Learning from Simulation:**
  - Abundant data
  - Suffers from a gap between simulation and reality
  - Results in policies that don’t transfer well to real world data
Key insight

- What if modeling errors in the simulation could be viewed as extra forces/disturbances?
- If we can train a model robust to these disturbances, then we can train a model that will generalize to the real world better.

- This paper does this by training jointly training an agent and an adversary, where the adversary creates disturbances representing possible modeling errors.
Properties of Adversarial RL

- **Modeling disturbances:**
  - Rewarded only when the main agent fail

- **Using domain knowledge:**
  - The adversary can affect both the agent and the environment
  - Ex. can change physical parameters like the friction of the environment

- **Modeled as a 2-player zero-sum discounted game.**
  - In this setting, the transition function $P$ is not fixed because it depends on environment parameters.

\[
\rho(\mu; \theta^\mu) = \mathbb{E}_P \left[ \mathbb{E} \left[ \sum_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0, \mu, P \right] \right]
\]
Robust Control and Risk

- The previous formulation optimizes for **mean** performance. -> risk neutral
- Instead optimize for conditional value at risk (CVaR).
  - $Q_\alpha(\rho)$ is the $\alpha$-quantile of $\rho$-values. We want to maximize for the worst possible $\rho$-values.
- Instead of using alpha as a parameter for training the adversary, the magnitude of force is used.

\[
\rho_{RC} = \mathbb{E} \left[ \rho \mid \rho \leq Q_\alpha(\rho) \right]
\]
RARL

roll() : samples $N_{\text{traj}}$ trajectories from the environment given the agent and adversary policies.

policyOptimizer() : approximate the advantage function to approximate the minimax optimization problem.

---

**Algorithm 1** RARL (proposed algorithm)

**Input**: Environment $\mathcal{E}$; Stochastic policies $\mu$ and $\nu$

**Initialize**: Learnable parameters $\theta^\mu_0$ for $\mu$ and $\theta^\nu_0$ for $\nu$

for $i=1, 2, \ldots, N_{\text{iter}}$ do
  $\theta^\mu_i \leftarrow \theta^\mu_{i-1}$
  for $j=1, 2, \ldots, N_{\mu}$ do
    $\{(s^i_t, a^1_t, a^2_t, r^1_t, r^2_t)\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta^\mu_i}, \nu_{\theta^\nu_{i-1}}, N_{\text{traj}})$
    $\theta^\mu_i \leftarrow \text{policyOptimizer}(\{(s^i_t, a^1_t, r^1_t)\}, \mu, \theta^\mu_i)$
  end for
  $\theta^\nu_i \leftarrow \theta^\nu_{i-1}$
  for $j=1, 2, \ldots, N_{\nu}$ do
    $\{(s^i_t, a^1_t, a^2_t, r^1_t, r^2_t)\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta^\mu_i}, \nu_{\theta^\nu_i}, N_{\text{traj}})$
    $\theta^\nu_i \leftarrow \text{policyOptimizer}(\{(s^i_t, a^2_t, r^2_t)\}, \nu, \theta^\nu_i)$
  end for
end for

**Return**: $\theta^\mu_{N_{\text{iter}}}, \theta^\nu_{N_{\text{iter}}}$

---

Agent section

Adversary section
Without any disturbances in test environment

Comparing to a baseline
Without any disturbances in test environment

Here the algorithms are run on multiple initializations and then sorted to show the nth percentile of cumulative final reward.
Changing the mass

![Graphs showing the impact of changing the mass on reward for different environments: Inverted Pendulum, HalfCheetah, Hopper, Walker2d. The graphs compare Baseline (TRPO) and RARL rewards.](image-url)
Changing the friction coefficient
Visualizing Robustness and the adversary policy
Conclusion

- Adversarial attacks also apply to RL and can be very effective even as blackbox attacks.
- Adversarial attacks can become very sophisticated like the strategically-timed and enchanting attacks.
- However, there exists some methods to try to mitigate some of their effects by jointly training RL agents with adversaries. But those are the adversaries that you expect...