Adversarial Sequential Decision Making

Part 2: Training Time Attacks

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CORNELL UNIVERSITY
FOUNDED A.D. 1865

25th July, 2022

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Target Policy

- Deterministic policy $\pi : S \to A$
- Target policy $\pi^\dagger \neq \pi^*$ (the optimal policy for the underlying MDP)
- [Definition] Training time RL attack:
    - Manipulate RL agent’s learning experience
    - … so the RL agent learns $\pi^\dagger$
    - … optionally minimize manipulation magnitude.

Reduction to Supervised Learning

- Training set poisoning in supervised learning:
  - Manipulating training set $(x, y)_{1:n}$
  - … so a supervised learner adopts predictor $f^{\dagger} : X \mapsto Y$
  - For example, set $y_i = f^{\dagger}(x_i), \forall i \in [n]$
- Same attack? $X \to S, Y \to A, f \to \pi$

Behavior Cloning

- Works on behavior cloning agent!
  - Input: $s_0, a_0, s_1, a_1, \dots$
  - Behavior cloning: $\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=1}^{n} \log \pi(a_i \mid s_i)$
  - Attack: set $a_i^{\dagger} = \pi^{\dagger}(s_i), \forall i$
- But: most RL agents do no work like this.

General Training Time Attack Protocol

- Environment draws initial state $s_0 \sim \mu$, agent perceives $s_0^\dagger$
- For $t = 0,1,\ldots$
  - Agent chooses action a_t
  - Environment receives $a_t^\dagger$, generates r_t, s_{t+1}
  - Agent perceives $r_t^\dagger, s_{t+1}^\dagger$

red=possible attacker manipulations

“Targeted vs. Non-Targeted”

- Targeted attack: force a specific $\pi^{\dagger}$
- Non-targeted attack: make agent suffer in value $V^{\hat{\pi}}(\mu)$
  - Conceptually still targeted: $\pi^{\dagger} \in \underline{\Pi} := \{\pi : V^{\pi}(\mu) \leq c\}$
  - Heuristic “flipping reward” attack $r_t^{\dagger} := -r_t$
    - This is $\pi^{\dagger} \in \arg\min_{\pi \in \Pi} V^{\pi}(\mu)$
    - Optionally with early stopping

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
  - With reward poisoning $r^{\dagger}$
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Open-Loop Control

- Attacker knows:
  - The environment MDP $M = (S, A, R, P, \mu, \gamma \text{ or } H)$
  - Agent runs any reasonable RL algorithm
- Open-loop control:
  - not interested in agent's internal state (e.g. Q-table Q_t)
  - Instead, simulate another MDP $M^\dagger = (S, A, R^\dagger, P, \mu, \gamma \text{ or } H)$ such that $\pi^\dagger$ is the optimal policy under $M^\dagger$
  - Trust agent will eventually learn $\pi^\dagger$

Target Policy Uniqueness

- π is an optimal policy of $M^\dagger$ iff $A^{\pi}_{M^\dagger}(s,a) := Q^{\pi}_{M^\dagger}(s,a) - V^{\pi}_{M^\dagger}(s) \leq 0, \forall s, a$
- An MDP can have multiple optimal policies; Attacker wants to ensure $\pi^\dagger$ being learned
- $\pi^\dagger$ has to be the *unique* optimal policy of $M^\dagger$.
- Sufficient condition (ϵ-robust policy) for uniqueness: Fix $\epsilon > 0$, $A^{\pi^\dagger}_{M^\dagger}(s,a) \leq -\epsilon, \forall s, \forall a \neq \pi^\dagger(s)$

Reward Poisoning: $r^\dagger$

- Turns out any target policy $\pi^\dagger$ is feasible with attack, if:
    - Reward manipulate is unbounded $r^\dagger \in \mathbb{R}$; and
    - Reward is a function of (s, a), not just s

Bijection between R and Q^*

- [Theorem (discounted MDP)] The following is a bijection $\mathbb{R}^{SA} \leftrightarrow \mathbb{R}^{SA}$
  - The unique fixed point of $\mathcal{T}Q(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P_{sa}} \max_{a'} Q(s',a')$
  - $R(s,a) = Q(s,a) - \gamma \mathbb{E}_{s' \sim P_{sa}} \max_{a'} Q(s',a')$

[MZSZ]

Any Target Policy $\pi^\dagger$ is Feasible

- Manually pick any $Q^\dagger$ that satisfies $A_{M^\dagger}^{\pi^\dagger}(s,a) \leq -\epsilon, \forall s, \forall a \neq \pi^\dagger(s)$, e.g.
  - $Q^\dagger(s, \pi^\dagger(s)) = \epsilon, \forall s$
  - $Q^\dagger(s,a) = 0$ for all other actions a
- Calculate $r^\dagger(s,a) = Q^\dagger(s,a) - \gamma \mathbb{E}_{s' \sim P_{sa}} \max_{a'} Q^\dagger(s', a')$

Reward Poisoning Attack Protocol

- Environment draws initial state $s_0 \sim \mu$
- For $t = 0,1,\ldots$
  - Agent chooses action a_t
  - Environment generates $r_t(s_t, a_t), s_{t+1}$
  - Agent perceives $r_t^\dagger(s, a)$ defined on previous slide, and $s_{t+1}^\dagger$

Remarks

- The MDP seen by agent is $M^{\dagger} = (S, A, R^{\dagger}, P, \mu, \gamma)$
- Some $\pi^{\dagger}$ infeasible if rewards bounded in [0,1], or independent of a
- There are many choices of $Q^{\dagger}$ and thus $R^{\dagger}$ for a given $\pi^{\dagger}$
  - Should the attacker prefer some $R^{\dagger}$ over others?

Attack Cost

- The environment MDP was $M = (S, A, R, P, \mu, \gamma)$
- Now agent sees $M^\dagger = (S, A, R^\dagger, P, \mu, \gamma)$
- Reasonable for the attacker to keep $R^\dagger$ close to R for stealthiness and less effort (i.e. attack cost)
- Close in what sense?

Attack Cost 1: Uniform Occupancy

- Popular choice: $\|R^\dagger - R\|_p = \left(\sum_{s,a} (R^\dagger_{sa} - R_{sa})^p \right)^{\frac{1}{p}}$

- Attack is a convex optimization problem:

$$\min_{R^\dagger \in \mathbb{R}^{SA}} \|R^\dagger - R\|_p$$

$$\text{s.t. } A^{\pi^\dagger}(s,a) \leq -\epsilon, \forall s, a \neq \pi^\dagger(s)$$

[HZ]

Attack Cost 1: Uniform Occupancy

- Pros: Convenient. Measures attack cost under a uniform state-action occupancy $d^{unif}(s, a)$:

$$\mathbb{E}_{d^{unif}} | R^{\dagger}(s, a) - R(s, a) |^p$$

- Cons: after the attack succeeds, agent will follow $\pi^{\dagger}$ and keep visiting state-action pairs under $d^{\pi^{\dagger}}$
  - There can be states $d^{\pi^{\dagger}}(s, \pi^{\dagger}(s)) \gg 0$ and $R^{\dagger}(s, \pi^{\dagger}(s)) \neq R(s, \pi^{\dagger}(s))$
  - d^{unif} not appropriate if attacker cares about how often it has to attack

Attack Cost 2: Do Not Attack Target Actions

- A variant of uniform occupancy, but do not attack on target actions
- Still convex optimization

$$\min_{R^\dagger \in \mathbb{R}^{SA}} \|R^\dagger - R\|_p$$

$$\text{s.t. } A^{\pi^\dagger}(s,a) \leq -\epsilon, \forall s, a \neq \pi^\dagger(s)$$

$$R^\dagger(s, \pi^\dagger(s)) = R(s, \pi^\dagger(s)), \forall s$$

- [Theorem] Attack with solution above, then both $\mathbb{E}\left[\frac{1}{T}\sum_t |R^\dagger(s_t, a_t) - R(s_t, a_t)|\right]$ and $\mathbb{E}\left[\frac{1}{T}\sum_t 1[a_t \neq \pi^\dagger(s_t)]\right]$ are $\tilde{O}\left(\frac{1}{T}\right)$.

[RRDZS]

Attack Cost 3: $d^{\pi^\dagger}$ Occupancy

- Directly minimize cumulative manipulation under $\pi^\dagger$:

$$\min_{R^\dagger \in \mathbb{R}^{SA}} \mathbb{E}_{d^{\pi^\dagger}} |R^\dagger(s,a) - R(s,a)|$$

$$\text{s.t. } A^{\pi^\dagger}(s,a) \leq -\epsilon, \forall s, a \neq \pi^\dagger(s)$$

(Similar to [ZPC])

Attack Cost 3: $d^{\pi^\dagger}$ Occupancy

- Implement as linear program

$$\min_{R^\dagger \in \mathbb{R}^{SA}, U} U^{\pi^\dagger}(\mu)$$

$$\text{s.t. } A^{\pi^\dagger}(s,a) \leq -\epsilon, \forall s, a \neq \pi^\dagger(s)$$

$$U^{\pi^\dagger}(s,a) = |R^\dagger(s,a) - R(s,a)| + \gamma \mathbb{E}_{s' \sim P_{sa}} U^{\pi^\dagger}(s', \pi^\dagger(s'))$$

(Similar to [ZPC])

Attack Cost 4: d^{π^b} Behavior Occupancy

- Offline RL from a dataset $(s, a, r, s')_{1:n}$ generated from behavior policy π^b
- Attacker can modify $r_{1:n}$ before learner sees the dataset
- Model based RL: learner estimate $\hat{P}, \hat{R}$ from dataset, then planning
- $\hat{P}, \hat{R}$ induce estimated advantage function $\hat{A}$, allowing attack
- Importantly, attacker cares about small manipulation *on the dataset*

[MZSZ]

Attack Cost 4: d^{π^b} Behavior Occupancy

- Dataset was drawn from d^{π^b}, so approximately

$$\min_{R^\dagger \in \mathbb{R}^{SA}} \mathbb{E}_{d^{\pi^b}} |R^\dagger(s,a) - R(s,a)|$$

$$\text{s.t. } \hat{A}(s,a) \leq -\epsilon, \forall s, a \neq \pi^\dagger(s)$$

- This assumes the attacker wants the same $r^\dagger = R^\dagger(s,a)$ value for all instances of the same (s,a) in the dataset

- In practice can relax this constraint and further lower attack cost

[MZSZ]

Recap

- Attacker knows environment MDP
- Attacker can change r_t
- Open-loop control: just simulate another MDP with $R^\dagger$, so that $\pi^\dagger$ becomes the ϵ-robust optimal policy
- The optimal $R^\dagger$ depends on which occupancy to measure attack cost

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
    - With reward poisoning $r^\dagger$
    - With action poisoning $a^\dagger$
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Action Poisoning Attack Protocol

- Environment draws initial state $s_0 \sim \mu$
- For $t = 0,1,\ldots$
  - Agent chooses action a_t
  - Attacker intercepts a_t and sends $a_t^\dagger$ to the environment instead (invisible to agent)
  - Environment generates r_t, s_{t+1} based on $a_t^\dagger$
  - Agent receives r_t, s_{t+1} and thought they were based on a_t

Action Poisoning Goal and Cost

- Attacker knows the environment MDP $M = (S, A, P, R, \mu, H)$
- Attacker wants to force target policy $\pi^\dagger$
- Attack cost = how often attacker has to change a_t to $a_t^\dagger$

Action Poisoning Algorithm

- For each state s, attacker computes the worst action under M and $\pi^\dagger$:

$$a_o(s) = \arg\min_a Q^{\pi^\dagger}(s, a)$$

- Requirement on $(M, \pi^\dagger)$: $\forall s : \pi^\dagger(s) \neq a_o(s)$

- Attack algorithm: if agent $a_t = \pi^\dagger(s_t)$ do not attack; otherwise set $a_t^\dagger = a_o(s_t)$

[LL]

Action Poisoning Algorithm

- [Lemma] Agent thinks $\pi^\dagger$ is the optimal policy
- Define $\Delta_{min} = \min_s \left(V^{\pi^\dagger}(s) - \min_a Q^{\pi^\dagger}(s,a) \right)$
- [Theorem] Both $\mathbb{E}\left[\sum_t 1[a_t \neq \pi^\dagger(s_t)]\right]$ and $\mathbb{E}\left[\sum_t 1[a_t \neq a_t^\dagger]\right]$ are upperbounded by Reg/Δ_{min}, where Reg is the regret of the RL algorithm

[LL]

Recap

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
    - May poison $r^\dagger$, $a^\dagger$, or transition $s^\dagger_{t+1}$ [RRDZS]
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
    - May poison $r^\dagger$, $a^\dagger$, or transition $s^\dagger_{t+1}$ [RRDZS]
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Open vs. Closed-Loop Control

- So far the attacker uses open-loop control:
  - Maintain an MDP $M^\dagger$ whose unique optimal policy is $\pi^\dagger$
  - $M^\dagger$ is not adaptive to agent's internal learning state (e.g. Q-table)
  - Pro: applicable to any RL learner
  - Con: can be slow in forcing $\pi^\dagger$
- Closed-loop control: with a whitebox agent, can adapt poisoning based on agent internal state

Example: Fast Adaptive Attack (FAA)

- Require: $\pi^\dagger$ differs from π on only $k = O(\log S)$ states $s_1 \ldots s_k$
- For $i = 1 \ldots k$ (s_1 is the farthest from the initial states, s_k nearest)
    - Poison $r^\dagger$ to force navigation policy ν_i: guides agent to s_i, and set $\pi^\dagger(s_i)$
- Invariance: does not change $\pi^\dagger(s_1) \ldots \pi^\dagger(s_{i-1})$
- This requires attacker to know agent's Q-table Q_t at each round

[ZMSZ]

Example: Fast Adaptive Attack (FAA)

- Pro: number of rounds Q_t does not induce $\pi^\dagger$ is $O(poly(S))$
  - Open-loop control can be $O(e^S)$
- Cons:
  - Requires whitebox agent
  - The attacks $r_t^\dagger$ as seen from the agent are non-stationary; perhaps easier to detect

[ZMSZ]

Outline

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
  - May poison $r^\dagger$, $a^\dagger$, or transition $s^\dagger_{t+1}$ [RRDZS]
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

The Issue with Unknown MDP

- Everything up to now (open/closed-loop, attack r, a, s_{t+1}) requires the attacker to know the environment MDP M
- If attacker does not know M it cannot compute $A^{\pi^\dagger}$, and thus cannot form targeted attacks
- But that is the case in many applications

Forced Exploration

- Key idea:
    - First attack to force agent to heavily explore M
        - [RZZS] uses $r_t^\dagger \sim \text{Bernoulli}(1/2, 1/2)$
        - [LL] uses LCB on Q values
    - By observing agent, attacker builds a set $\mathscr{M}$ of plausible MDPs
    - Design attack so that $\pi^\dagger$ is the optimal policy in all $M \in \mathscr{M}$

Outline

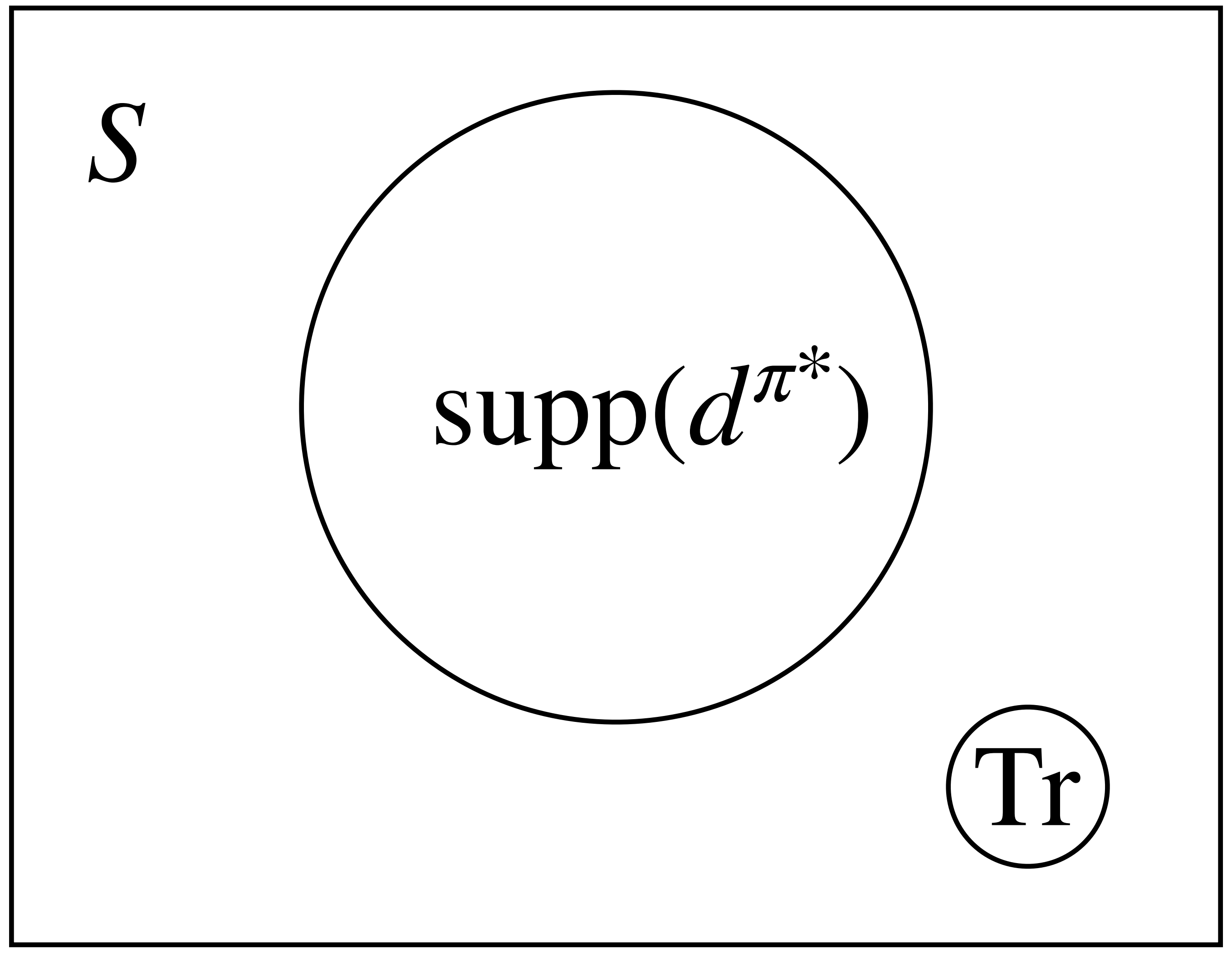
- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
  - May poison $r^\dagger$, $a^\dagger$, or transition $s^\dagger_{t+1}$ [RRDZS]
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Backdoor RL

- Backdoor RL has two phases:
  - Training-time poisoning phase to hide a backdoor in $\pi^\dagger$
  - Test-time triggering phase to activate the backdoor in $\pi^\dagger$

[WJWGXS]

Training-Time Poisoning Phase



- Use any of the techniques discussed so far
- May even be easier: usually does not care about attack cost
- The target policy $\pi^\dagger$ is special:
  - $\pi^\dagger(s) = \pi^*(s) \forall s \in \mathrm{supp}(d^{\pi^*})$ [normal operation]
  - $V^{\pi^\dagger}(s^\dagger) \ll V^*(\mu)$ on "trigger states" $s^\dagger \in \mathrm{Tr}$
  - e.g. $a^\dagger = \pi^\dagger(s^\dagger$ has sticker in scene$) =$ "hard accelerate" leading to crash
  - Tr can be difficult to distinguish from $\mathrm{supp}(d^{\pi^*})$ by humans

Test-Time Triggering Phase

- Agent deploys $\pi^\dagger$
- Before triggering, by definition any $s_t \in \mathrm{supp}(d^{\pi^*})$ is a normal state
- The attacker has the ability to change s_t to $s_t^\dagger \in \mathrm{Tr}$
  - E.g. by adding a special sticker to the scene
  - E.g. by controlling other agents to perform unusually actions
- From here on agent receives low value $V^{\pi^\dagger}(s^\dagger) \ll V^*(\mu)$

What We Covered

- Poisoning: from supervised learning to RL
- Open-loop control: simulating another MDP
    - May poison $r^\dagger$, $a^\dagger$, or transition $s^\dagger_{t+1}$
- Closed-loop control
- Forced exploration in unknown MDP
- Backdoor RL

Looking Ahead

- Commonalities of training-time RL attacks:
  - Require “enough” manipulation
  - Assume agent naively runs standard RL algorithms
- Therefore, we anticipate RL defense to break these conditions.

References

- [HZ] Huang, Zhu. Deceptive Reinforcement Learning Under Adversarial Manipulations on Cost Signals. 2019
- [LL] Liu, Lai. Provably Efficient Black-Box Action Poisoning Attacks Against Reinforcement Learning. 2021
- [MZSZ] Ma, Zhang, Sun, Zhu. Policy Poisoning in Batch Reinforcement Learning and Control. 2019
- [RRDZS] Rakhsha, Radanovic, Devidze, Zhu, Singla. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. 2020
- [RZZS] Rakhsha, Zhang, Zhu, Singla. Reward Poisoning in Reinforcement Learning: Attacks Against Unknown Learners in Unknown Environments. 2021
- [WJWGXS] Wang, Javed, Wu, Guo, Xing, Song. BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning. 2021
- [ZMSZ] Zhang, Ma, Singla, Zhu. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. 2020
- [ZPC] Zhang, Parkes, Chen. Policy Teaching Through Reward Function Learning. 2008.