AI Security

Optimization for Black-box Attacks

Sangdon Park

POSTECH

April 19, 2026

Outline

Need for Black-box Attacks

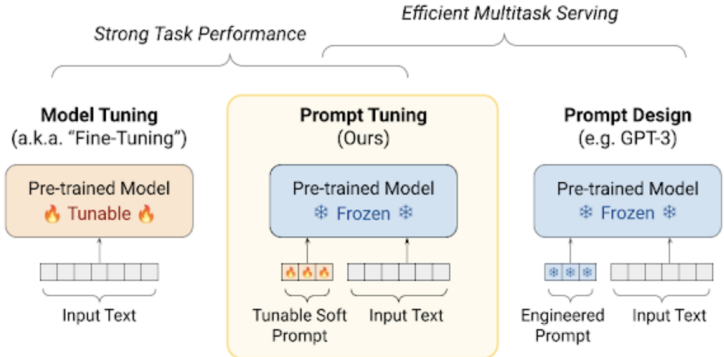
Main Question

How can we find an adversarial prompt in this black-box setup?

- Closed models (*e.g.*, GPT-5) is the most interesting attack targets
- However, we cannot have access to their model weights.
- But, still we can obtain responses from them (*i.e.*, only allows forward-pass).

Outline

Why Prompt Tuning?



- Prompt tuning is a key to guide LLMs to have desired responses.
- We often build hard prompts for it.
- Prompt Tuning [Lester et al., 2021] finds soft input embedding vectors, instead of hard prompts.
- This is used for maximizing performance but can be applicable for attacks.

Prompt Tuning [Lester et al., 2021]

Objective

$$\min_{\mathbf{E}} -\sum_{n=1}^{N} \log p_\theta(\mathbf{y}_n \mid \mathbf{E}, \mathbf{x}_n)$$

- D: the embedding vector dimension
- K: the number of tuning tokens
- $\mathbf{E} \in \mathbb{R}^{D \times K}$: a set of input embedding vectors
- p_θ: an LLM
- In attacks where $\mathbf{y}_n$ is a harmful response, is it reasonable for attackers to manipulate $\mathbf{E}$?

Outline

What and Why Zeroth Order Optimization?

Question

Can we optimize parameters without gradients (or with only forward passes)?

- zeroth order optimization (ZO) methods optimizes a model by estimating gradients using forward passes.
- Why ZO?
  - no gradients – memory efficient
  - non-differentiable criteria, *e.g.*, rule-based safety standards or F1 score
- Why ZO in attacks?
  - may be useful for black-box models

Simultaneous Perturbation Stochastic Approximation (SPSA)

Definition (SPSA)

SPSA estimate the gradient of a loss function $\mathcal{L}$ for a model θ on a minibatch $\mathcal{B}$ as follows:

$$\hat{\nabla}\mathcal{L}(\theta;\mathcal{B}) = \frac{\mathcal{L}(\theta+\varepsilon z;\mathcal{B}) - \mathcal{L}(\theta-\varepsilon z;\mathcal{B})}{2\varepsilon} z \approx zz^T \nabla\mathcal{L}(\theta;\mathcal{B}),$$

where $z \sim \mathcal{N}(0, I_d)$ and ε is the perturbation scale.

- Intuitively, this estimates a gradient (*e.g.*, imagine then $d = 1$)
- Here, you can derive $\approx$ via the first order Taylor expansion.
- How many forward passes do we need?
- How much memory do we need?
- Can you imagine the convergence speed of SPSA-based SGD?
- Is this unbiased estimator?

MeZO [Malladi et al., 2023]

Algorithm 1: MeZO

Require: parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, loss $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$, step budget T, perturbation scale ϵ, batch size B, learning rate schedule $\{\eta_t\}$

for $t = 1, ..., T$ **do**
- Sample batch $\mathcal{B} \subset \mathcal{D}$ and random seed s
- $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$
- $\ell_+ \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$
- $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, -2\epsilon, s)$
- $\ell_- \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$
- $\boldsymbol{\theta} \leftarrow$ `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$ ▷ `Reset parameters before descent`
- `projected_grad` $\leftarrow (\ell_+ - \ell_-)/(2\epsilon)$
- Reset random number generator with seed s ▷ `For sampling` z
- **for** $\theta_i \in \boldsymbol{\theta}$ **do**
  - $z \sim \mathcal{N}(0, 1)$
  - $\theta_i \leftarrow \theta_i - \eta_t *$ `projected_grad` $* z$
- **end**

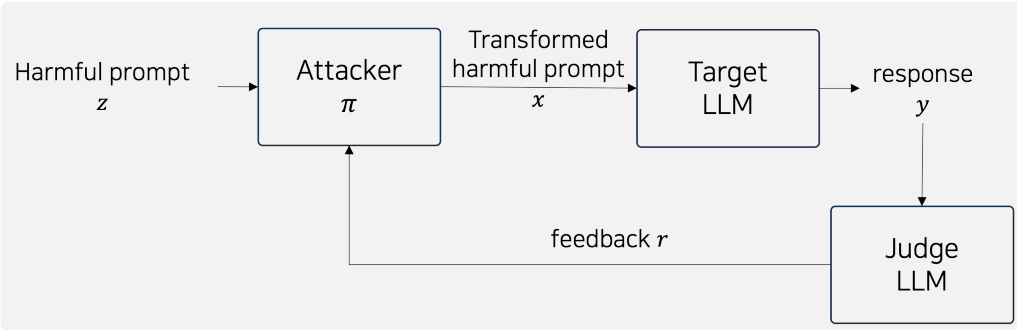
end

Subroutine `PerturbParameters`$(\boldsymbol{\theta}, \epsilon, s)$
- Reset random number generator with seed s ▷ `For sampling` z
- **for** $\theta_i \in \boldsymbol{\theta}$ **do**
  - $z \sim \mathcal{N}(0, 1)$
  - $\theta_i \leftarrow \theta_i + \epsilon z$ ▷ `Modify parameters in place`
- **end**
- **return** $\boldsymbol{\theta}$

- MeZO: Memory-efficient Zeroth order Optimizer
- Need only $\mathcal{O}(\mathbb{R}^d)$ memory

Outline

Why Policy Optimization?

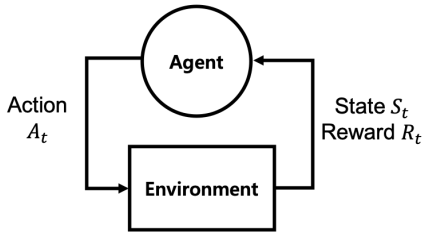


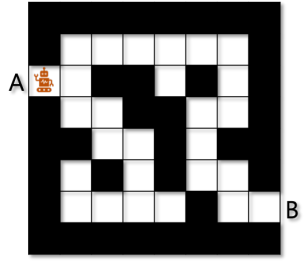
- We often see only the victim model's responses – black box setups
- Finding a good attacker = learning an attacker via RL – very natural

Learning from Interaction

Learning protocol:

- The agent observes an initial state S_0
- For each step $t = 0, 1, 2, ...$
- The agent executes an action A_t given the observed state S_t
- The agent observes reward R_{t+1} and the next state S_{t+1}





Markov Decision Process (MDP)

Definition (informal)

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, d_0)$, where

- $\mathcal{S}$: a set of states
- $\mathcal{A}$: a set of actions
- $\mathcal{R}$: a set of reward values
- $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$: dynamics or a kernel
- d_0: an initial state distribution

For example,

- $s_0 \sim d_0$: an initial location of a robot
- $a_0 \in \mathcal{A}$: "move to the right"
- $p(s_1, r_1 \mid s_0, a_0)$: a proability distribution over next states and rewards

Markov Decision Process (MDP)

Definition (informal)

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, d_0)$, where

- $\mathcal{S}$: a set of states
- $\mathcal{A}$: a set of actions
- $\mathcal{R}$: a set of reward values
- $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$: dynamics or a kernel
- d_0: an initial state distribution

- In MDP, p completely characterizes the environment's dynamcis due to the Markov property.
  - See the next slide.

Markov Property

Definition

The dyanmics $p : (\mathcal{S} \times \mathcal{A})^* \to \Delta(\mathcal{S} \times \mathcal{R})$ has a Markov property if

$$\overbrace{\mathbb{P}\{S_{t+1}=s_{t+1}, R_{t+1}=r_{t+1} \,|\, S_{1:t}=s_{1:t}, A_{1:t}=a_{1:t}\}}^{p(s_{t+1},r_{t+1}|s_{1:t},a_{1:t})} = \underbrace{\mathbb{P}\{S_{t+1}=s_{t+1}, R_{t+1}=r_{t+1} \mid S_t=s_t, A_t=a_t\}}_{p(s_{t+1},r_{t+1}|s_t,a_t)}.$$

- This means S_t include perfect information on previous action-environment interaction.
- Here, we assume that the dynamcis does not change over time t (=homogeneous Markov chain).

RL Goal

Goal – informal

The goal of RL is to find an optimal "policy" that maximizes an "expected cumulative reward" (or an "expected return").

Note that

- the goal is NOT maximizing immediate rewards.
- the expected return $\approx$ an estimate of rewards that the agent will recieve in future.

We will see the answers of the following questions:

- What is the "policy"?
- What is the "expected return"?

Policy

Definition

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from states to probabilities of selecting each possible action.

- The policy can be deterministic – in this case we simply denote it by $\pi : \mathcal{S} \to \mathcal{A}$.
- The policy is modeled via a neural net – this is our target to learn

Example:

	s_1	s_2	s_3
$\pi_1(a_i \mid s_i)$	a_1 a_2 a_3 1.0 0 0	a_1 a_2 a_3 0.5 0.5 0	a_1 a_2 a_3 0.2 0.3 0.4
$\pi_2(a_i \mid s_i)$	a_1 a_2 a_3 0 1.0 0	a_1 a_2 a_3 1/3 1/3 1/3	a_1 a_2 a_3 0 0.5 0.5

Trajectory

Trajectory

A trajectory τ is a sequence of states, actions, and rewards that is obtained by the interaction of a policy π with its environment, *i.e.*,

$$\tau := (s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ...).$$

- Is the trajectory a random variable? If so, where are sources of randomness?
- Given an environment, a trajectory is generated from a policy π; thus, we often denote it by $\tau \sim \pi$.

Return

Definition

We say that R_t is a return with a finite horizon T, defined as

$$R_t := r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

where r_k is a reward at time k and γ is a discount factor.

- r_k: the immediate reward
- The discount factor quantifies the appreciation on future rewards.
  - $\gamma = 0$: only consider the immediate reward but ignore all future rewards.
  - $0 \leq \gamma \leq 1$: consider the immediate reward and discount the appreciation more on more "distant" future rewards.
  - $\gamma = 1$: equallty appreciate all future rewards, including the immediate reward.
- The return has a recursive structure, *i.e.*,

$$R_t = r_{t+1} + \gamma R_{t+1}.$$

RL Goal

Goal – informal

The goal of RL is to find an optimal policy that maximizes an expected return, *i.e.*,

$$\max_{\pi} \mathbb{E}_{\pi} R,$$

where $R := R_0$.

- The expectation is taken over trajectories by a policy π, where the source of randomness includes
  - a stochastic policy
  - a stochastic transition kernel
- The expected return measures a goodness of a policy π.
- But, how to measure the goodness of the policy in the middle of a trajectory?
  - "V"-function
  - "Q"-function

Example: GRPO

GRPO Objective

$$J_{\text{GRPO}}(\pi) = \mathbb{E}_{s,\{a_k\}_{k=1}^{K}\sim\pi_{\text{old}}}\left[\frac{1}{K}\sum_{k=1}^{K}\min\left(\frac{\pi(a_k \mid s)}{\pi_{\text{old}}(a_k \mid s)}\hat{A}_k,\ \text{clip}\left(\frac{\pi(a_k \mid s)}{\pi_{\text{old}}(a_k \mid s)}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_k\right)\right]$$

- Suppose jailbreaking.
- s: an harmful response
- a_k: a transformed harmful response for the k-th group
- K: the group size
- $\hat{A}_k$: an estimated advantage where a reward is considered.

State-Value Function (="V"-Function)

Definition

The *state-value function* of a state s under a policy π is the expected return when starting in s and following π thereafter, *i.e.*,

$$V_\pi(s) := \mathbb{E}_\pi\left[R_t \mid s_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right].$$

- A V-function measures the expected return of a policy π at a state s.
- It is useful when an agent checks the long-term expected return from each state.

Action-Value Function (="Q"-Function)

Definition

The *action-value function* of a state s with an action a under a policy π is the expected return when taking the action a at the staring state s, and following π thereafter, *i.e.*,

$$Q_\pi(s, a) := \mathbb{E}_\pi \left[R_t \mid s_t = s, a_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a \right].$$

- A Q-function measures the expected return of a policy π at a state s if an agent chooses to act a.
- It is useful when an agent decides which action to take.

Policy Optimization – A Bird's-eye View

1. (**goal**) maximize the expected return, *i.e.*,

$$\max_{\pi} J(\pi),$$

   where $J(\pi) \coloneqq \mathbb{E}_{\pi} R$.

2. (**update rule**) update a policy as follows:

$$\pi \leftarrow \pi + \alpha \mathbb{E}\Big[\nabla_{\pi} \log \pi(a_t \mid s_t) A^{\pi}(s_t, a_t)\Big],$$

   where α is a learning rate and $A^{\pi}(s_t, a_t)$ is an advantage function.

   - What is the advantage function?
   - Where is $\mathbb{E}[\nabla_{\pi} \log \pi(a_t \mid s_t) A^{\pi}(s_t, a_t)]$ coming from?

3. Estimate the advantage function $A^{\pi}(s_t, a_t)$.

   - Often the advantage function is unknown. Then, how to estimate it?

Log-derivative Trick

Key Equality on Gradients

$$\nabla_\pi J(\pi) = \underbrace{\nabla_\pi \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\right]}_{(i)} = \underbrace{\mathbb{E}_{\tau\sim\pi}\left[R(\tau)\nabla_\pi \log p_\pi(\tau)\right]}_{(ii)}$$

- We need $\nabla_\pi J(\pi)$ for our policy update rule.
- Here, we use $R(\tau)$ instead of R to make the dependency on τ clear.
- $\tau \sim \pi$: a trajectory sample
- (i): hard to get gradients; why?
- (ii): easy to get gradients; why?
- $(i) = (ii)$: why?

Log-derivative Trick

Key Equality on Gradients

$$\nabla_\pi J(\pi) = \underbrace{\nabla_\pi \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\right]}_{(i)} = \underbrace{\mathbb{E}_{\tau\sim\pi}\left[R(\tau)\nabla_\pi \log p_\pi(\tau)\right]}_{(ii)}$$

- We need $\nabla_\pi J(\pi)$ for our policy update rule.
- Here, we use $R(\tau)$ instead of R to make the dependency on τ clear.
- $\tau \sim \pi$: a trajectory sample
- (i): hard to get gradients; why?
- (ii): easy to get gradients; why?
- $(i) = (ii)$: why?
  - $\nabla_\pi p_\pi(\tau) = p_\pi(\tau)\nabla_\pi \log p_\pi(\tau)$

Policy Term Isolation

Isolation

Due to an MDP,

$$\log p_\pi(\tau) = \log p(s_0) + \sum_t \log \pi(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t).$$

This implies

$$\nabla_\pi \log p_\pi(\tau) = \sum_t \log \nabla_\pi \pi(a_t \mid s_t).$$

- The gradient of the expected return is simplified as follows:

$$\nabla_\pi J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\nabla_\pi \log p_\pi(\tau)\right] = \mathbb{E}_{\tau\sim\pi}\left[\sum_t R_t(\tau)\nabla_\pi \log \pi(a_t \mid s_t)\right]$$

$$= \mathbb{E}_{\tau\sim\pi}\left[\sum_t Q(s_t, a_t)\nabla_\pi \log \pi(a_t \mid s_t)\right] = \mathbb{E}_{s_t,a_t}\Big[Q(s_t, a_t)\nabla_\pi \log \pi(a_t \mid s_t)\Big]$$

Advantage

Gradient with Advantage

$$\nabla_\pi J(\pi) = \mathbb{E}_{s_t,a_t}\Big[\underbrace{(Q^\pi(s_t,a_t) - V^\pi(s_t))}_{A^\pi(s_t,a_t)}\nabla_\pi \log \pi(a_t \mid s_t)\Big],$$

where $A^\pi(s_t, a_t)$ is an advantage function.

- Why subtracting $V(s)$?
  - high gradient variance
- Why can we subtract $V(s)$?
  - It is a zero baseline, *i.e.*,

$$\mathbb{E}_{s_t,a_t}\left[\nabla_\pi \log \pi(a_t \mid s_t) V^\pi(s_t)\right] = 0,$$

where

$$\mathbb{E}_{a_t}\left[\nabla_\pi \log \pi(a_t \mid s_t)\right] = \sum_{a_t} \nabla_\pi \log \pi(a_t \mid s_t)\pi(a_t \mid s_t) = \sum_{a_t} \nabla_\pi \pi(a_t \mid s_t)$$

$$= \nabla_\pi \sum_{a_t} \pi(a_t \mid s_t) = \nabla_\pi 1 = 0.$$

Importance Sampling

Importance Sampling

$$\nabla_\pi J(\pi) = \mathbb{E}_{s_t, a_t \sim \pi}\left[A^\pi(s_t, a_t)\nabla_\pi \log \pi(a_t \mid s_t)\right] = \mathbb{E}_{s_t, a_t \sim \pi_{\text{old}}}\left[\frac{\pi(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)} A^\pi(s_t, a_t)\nabla_\pi \log \pi(a_t \mid s_t)\right]$$

- In policy optimization, the expectation is approximated via trajectories.
- Often we collect a trajectory using an old policy π_{old}.
- However, $\nabla_\pi J(\pi)$ assumes that a_t is drawn from a new policy π.
- To address this gap, we use importance sampling.

Advantage Estimation

GRPO Advangate

$$A^\pi(s,a) \approx \hat{A}_k = \frac{r_k - \mu_R}{\sigma_R} \quad \text{where} \quad \mu_R = \frac{1}{K}\sum_{k=1}^{K} r_k \text{ and } \sigma_R = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(r_k - \mu_R)^2}$$

- Recall that $A^\pi(s,a) \coloneqq Q^\pi(s,a) - V^\pi(s)$.
- GRPO assumes $T = 1$.
- reward normalization fairly well approximate the true advantage.
- why normalization (*i.e.*, dividing by σ_R)?

Example: GRPO – Again

GRPO Objective

$$J_{\text{GRPO}}(\pi) = \mathbb{E}_{s,\{a_k\}_{k=1}^{K} \sim \pi_{\text{old}}} \left[\frac{1}{K} \sum_{k=1}^{K} \min \left(\frac{\pi(a_k \mid s)}{\pi_{\text{old}}(a_k \mid s)} \hat{A}_k, \ \text{clip} \left(\frac{\pi(a_k \mid s)}{\pi_{\text{old}}(a_k \mid s)}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_k \right) \right]$$

Conclusion

- Black-box attacks are often casted to optimization.
- We have learned two black-box purpose optimization methods.
  1. Zeroth order optimization
  2. Policy optimization – the most interesting and useful.
- Next, we will see inference-time attacks that leverages the above methods.

Reference I

B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059, 2021.

S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.