Learning to Learn by Zeroth-Order Oracle

Yangjun Ruan¹, Yuanhao Xiong², Sashank Reddi³, Sanjiv Kumar³, Cho-Jui Hsieh²,³

¹ Zhejiang University, ² UCLA, ³ Google Research
Learning to learn (L2L)

- Use neural networks to automatically learn optimization algorithms

\[ \theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \varphi) \]

- \( f \): the optimizee (optimization problems) specified by its parameters \( \theta \)
- \( g \): the learned optimizer specified by its parameters \( \varphi \)

- The optimizer \( g \) is usually modeled as recurrent neural networks (RNNs)

Figure: Andrychowicz et al., 2016
Learning to learn (L2L)

✓ Improve hand-designed algorithms with learned optimization rules
Learning to learn (L2L)

✓ Improve hand-designed algorithms with learned optimization rules

✗ Gradient-based: cannot be applied when gradients are difficult or infeasible to obtain (i.e., zeroth-order optimization)
Zeroth-order (ZO) optimization

• Setting: explicit gradients are not available

• Widely used application: black-box adversarial attacks
Zeroth-order (ZO) optimization

- Setting: explicit gradients are not available

- Widely used application: black-box adversarial attacks

- Basic method: approximate gradients along Gaussian sampled query directions

\[
\hat{V} f(\theta) = \frac{1}{q} \sum_{i=1}^{q} \frac{f(\theta + \mu u_i) - f(\theta)}{\mu} u_i
\]

- \{u_i\}: query directions sampled from standard Gaussian distribution
- q: number of query directions
- \(\mu\): smoothing parameter
Zeroth-order (ZO) optimization

• Existing ZO algorithms: suffer from the high variance of ZO gradient estimator
  ▪ Mainly results from random query directions
  ▪ Hamper convergence: usually $d$ (parameter size) times slower than its first-order counterpart
Zeroth-order (ZO) optimization

- Existing ZO algorithms: suffer from the high variance of ZO gradient estimator
  - Mainly results from random query directions
  - Hamper convergence: usually $d$ (parameter size) times slower than its first-order counterpart

- Our work: apply the L2L framework to learn an efficient ZO optimizer
Method
Method

- Jointly learn the parameter update rule and the Gaussian sampling rule
  - UpdateRNN: learn how to propose parameter updates given approximated gradients
    \[ \theta_t = \theta_{t-1} + \text{UpdateRNN} \left( \tilde{\nabla} f(\theta_t) \right) \]
  - QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution
    \[ \Sigma_t = \text{QueryRNN}([\tilde{\nabla} f(\theta_{t-1}), \Delta \theta_{t-1}]) \]
Method

• Jointly learn the parameter update rule and the Gaussian sampling rule
  
  ▪ UpdateRNN: learn how to propose parameter updates given approximated gradients
    \[ \theta_t = \theta_{t-1} + \text{UpdateRNN}\left(\hat{\nabla} f (\theta_t)\right) \]

  ▪ QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution
    \[ \Sigma_t = \text{QueryRNN}\left([\hat{\nabla} f (\theta_{t-1}), \Delta \theta_{t-1}]\right) \]
Method

• Jointly learn the parameter update rule and the Gaussian sampling rule
  - UpdateRNN: learn how to propose parameter updates given approximated gradients
    \[ \theta_t = \theta_{t-1} + \text{UpdateRNN} \left( \nabla f (\theta_t) \right) \]
  - QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution
    \[ \Sigma_t = \text{QueryRNN} \left( [\hat{\nabla} f (\theta_{t-1}), \Delta \theta_{t-1}] \right) \]
Training the ZO optimizer

• Backpropagate through the Gaussian sampling module (non-differentiable)
  ✓ Apply reparameterization trick to generate query directions $u \sim N(0, \Sigma_t)$
  
  $z \sim N(0, I)$
  
  $u = \Sigma_t^{1/2} z$
Training the ZO optimizer

• Backpropagate through the Gaussian sampling module (non-differentiable)
  ✓ Apply reparamerization trick to generate query directions \( u \sim N(0, \Sigma_t) \)
  \[
  z \sim N(0, I) \\
  u = \Sigma_t^{1/2} z 
  \]

• Backpropagate through the optimizee (zeroth-order)
  ✓ Apply coordinatewise ZO gradient estimator (optional)
  \[
  \hat{f}(\theta) = \sum_{i=1}^{d} \frac{f(\theta + \mu e_i) - f(\theta - \mu e_i)}{2\mu} e_i 
  \]
  • \( \{e_i\} \): standard basis vector with \( i^{th} \) coordinate being 1, and others being 0s
  • \( d \): optimizee dimension
  • \( \mu \): smoothing parameter
Experiments

- Black-box adversarial attack
Experiments

- Black-box adversarial attack

- Promising Application: automatically learned efficient “attacker”
Analytical experiments

- Ablation study
  - Effectiveness of both modules
Analytical experiments

- Ablation study
  ✓ Effectiveness of both modules

- Estimated gradient evaluation
  ✓ QueryRNN leads to more accurate gradient estimators
Thank you!

Paper link: 
https://openreview.net/forum?id=ryxz8CVYDH

Thank you!