Variance-reduced Gradient Estimation via Noise-Reuse in Online Evolution Strategies

Oscar Li, James Harrison, Jascha Sohl-Dickstein, Virginia Smith, Luke Metz

Carnegie Mellon University · Google DeepMind

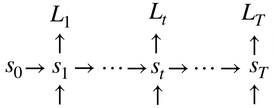
Email: oscarli@cmu.edu

Code:

Come here to learn about a new online gradient estimator for unrolled computation graphs!

Problem Setup

Unrolled computation graphs



Learnable parameter: $\theta \in \mathbb{R}^d$

Dynamical system: $s_{t+1} = f_{t+1}(s_t, \theta)$

Per-step loss: $L_t(s_t)$

Objective: $\min_\theta L([\theta]_{\times T}) := \frac{1}{T}\sum_{t=1}^{T} L_t(s_t)$

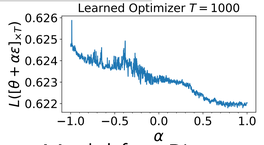
Applications:

- Recurrent neural networks
- Meta-training learned optimizers
- Dataset distillation
- Policy learning in RL

Q: How **best** to **compute gradients** to optimize $L([\theta]_{\times T})$?

Loss properties (when loss.backward() doesn't work)

- Losses with **extreme local sensitivity**
- **non-differentiable** losses
  - black box losses — e.g., Model-free RL
  - discontinuous losses — e.g., L is the zero-one loss

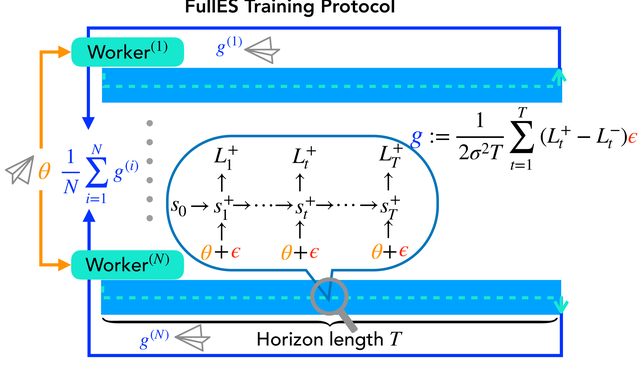


Evolution Strategies (FullES)

FullES smoothing objective: $\min_\theta \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma^2 I_{d\times d})} L([\theta+\epsilon]_{\times T})$

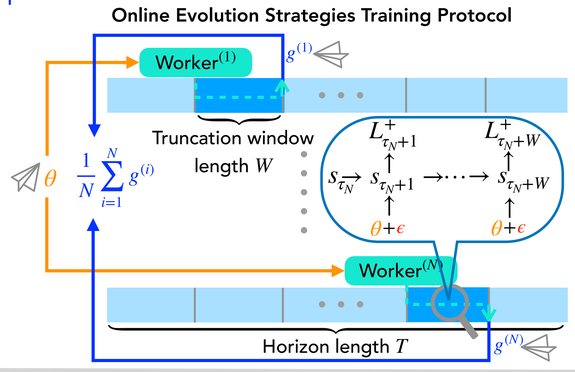
FullES gradient estimator: $\frac{1}{2\sigma^2}\left[L([\theta+\epsilon]_{\times T}) - L([\theta-\epsilon]_{\times T})\right]\epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_{d\times d})$

1 update / T unrolls



Online Evolutionary Strategies

1 update / W unrolls



Persistent Evolutionary Strategies

(Vicol et al, 2021)

PES smoothing objective: 1 sample / W unrolls

$$\min_\theta \mathbb{E}_{\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\sigma^2 I_{d\times d})} L([\theta+\epsilon_1]_{\times W}, \ldots, [\theta+\epsilon_{T/W}]_{\times W})$$

PESWorker

Sample ϵ_1	Sample ϵ_2	Sample ϵ_3	⋯

Accumulated noise $\xi = \epsilon_1 \quad \epsilon_1+\epsilon_2 \quad \epsilon_1+\epsilon_2+\epsilon_3$

$$g := \frac{1}{2\sigma^2 W}\left(\sum (L_t^+ - L_t^-)\right)\sum \epsilon_i$$

over time t in current window. over all ϵ_i sampled so far.

Generalized PES (ours)

$K = cW$

GPES smoothing objective: 1 sample / K unrolls

$$\min_\theta \mathbb{E}_{\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\sigma^2 I_{d\times d})} L([\theta+\epsilon_1]_{\times K}, \ldots, [\theta+\epsilon_{\lceil T/K\rceil}]_{\times \mathrm{r}(T,K)})$$

$\text{GPES}_{K=2W}$Worker

Sample ϵ_1	Reuse ϵ_1	Sample ϵ_2	⋯

Accumulated noise $\xi = \epsilon_1 \quad \epsilon_1 \quad \epsilon_1+\epsilon_2$

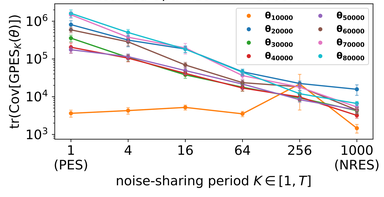
K divides τ

Q: Which K to **choose** for GPES_K?

Variance characterization

Theorem 5: Analytical characterization of the total variance of GPES_K for all K.

Corollary 8: $\text{GPES}_{K=T}$ provably has the lowest total variance among all GPES_K estimators.



Noise-Reuse Evolution Strategies (ours)

1 sample / T unrolls

NRESWorker

No need to accumulate

Sample ϵ	Reuse ϵ	Reuse ϵ	⋯

$$g := \frac{1}{2\sigma^2 W}\left(\sum (L_t^+ - L_t^-)\right)\epsilon$$

over time t in current window. the ϵ sampled at the beginning of this episode.

NRES vs. FullES

- The **same** smoothing objective
- **NRES** is T/W times more parallelizable than **FullES**.
- Variance comparison:

Theorem 9: Under reasonable assumptions and the same compute budget, **NRES** has lower variance than **FullES.**

$T = 1000, W = 1$ (±95% ci)

iter. j of θ_j ($\times 10^4$)	total variance $\mathrm{tr}(\mathrm{Cov}[\cdot])$: $\frac{W}{T}\sum_{i=1}^{T/W} \mathrm{NRES}_i(\theta_j)$	$\mathrm{FullES}(\theta_j)$
1	**1.47** ± 0.38	864.34 ± 385.01
2	**15.74** ± 4.90	513.50 ± 74.71
3	**4.28** ± 0.78	201.38 ± 30.18
4	**3.20** ± 0.50	684.85 ± 86.90
5	**4.47** ± 0.88	181.06 ± 24.97
6	**4.42** ± 0.68	288.61 ± 46.77
7	**5.33** ± 1.08	448.30 ± 62.44
8	**6.62** ± 0.89	154.86 ± 28.21

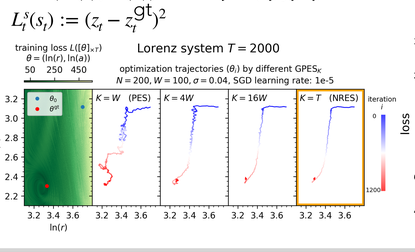
Experiments

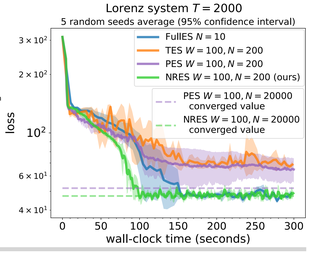
Learning chaotic dynamical system

$\theta = (\ln(r), \ln(a))$ $\quad s_t = (x_t, y_t, z_t) \in \mathbb{R}^3$

$L_t^s(s_t) := (z_t - z_t^{\text{gt}})^2$

$$\begin{cases} x_{t+1} = x_t + a(y_t - x_t)dt \\ y_{t+1} = y_t + [x_t \cdot (r - z_t) - y_t]dt \\ z_{t+1} = z_t + [x_t \cdot y_t - 8/3 \cdot z_t]dt \end{cases}$$

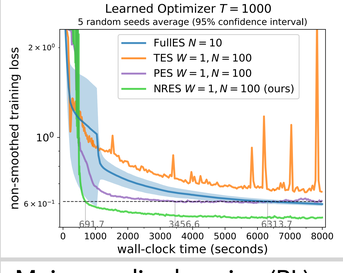


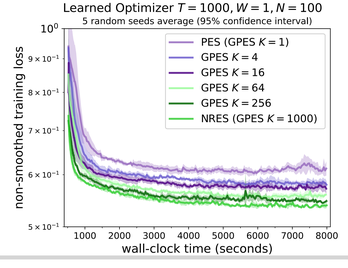


Meta-training learned optimizer

θ: learned optimizer $\quad s_t$: the inner model's parameters and momentum statistics

$L_t(s_t)$: the learned model's validation loss.





Mujoco policy learning (RL)

θ: parameter of a policy network $\quad s_t$: the mujoco environment state.

$L_t(s_t)$: negative per-step reward