Evolution Strategies: Principles and Practical Issues

Masahiro Nomura (CyberAgent, Inc.)

08/24(Thu)@ICIAM (Sequential Decision Making for Optimization, Learning and Search)

Outline

- Introduction to Natural Evolution Strategies (NES)
  - Stochastic optimization framework that uses natural gradient
  - NES is similar to CMA-ES
- Practical Issues for Difficult Problems (e.g. Multimodal and Noisy)
  - Hyperparameter tuning is required
  - Learning rate adaptation (Nomura+, GECCO'23 Best Paper Nominated)

Black-Box Optimization on Continuous Domain

Goal of continuous black-box optimization (BBO):

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ f(x)$$

- X is continuous space

- Objective function is "**<u>black-box</u>**"
  - Not assume availability of gradient
  - We only use function evaluation value

Stochastic Relaxation for BBO [WSG+14]

- We want to optimize f(x), but the gradient wrt. x is not available
- Instead of directly finding x*, NES searches distribution parameter θ*:

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ f(x)$$

$$\Big\Downarrow \quad \textbf{Stochastic Relaxation}$$

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\ \mathbb{E}_{x \sim p(x;\theta)}[f(x)] =: J(\theta)$$

Gradient for Distribution Parameter [WSG+14]

- Suppose we can find $\nabla_\theta \log p(x;\theta)$
- By using the so-called 'log-likelihood trick':

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int f(x)p(x;\theta)dx \\
&= \int f(x)\nabla p(x;\theta)dx \\
&= \int f(x)\nabla \log p(x;\theta)p(x;\theta)dx \\
&= \mathbb{E}_{x\sim p(x;\theta)}[f(x)\nabla_\theta \log p(x;\theta)] \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} f(x_i)\nabla_\theta \log p(x_i;\theta)
\end{aligned}
$$

- Update the distribution parameter via gradient ascent

$$\theta \leftarrow \theta + \eta \cdot \nabla_\theta J(\theta)$$

Multivariate Gaussian Distribution

- One can use any distribution whose derivative of log-likelihood is obtained
- Most widely used one: multivariate Gaussian distribution (MGD)
- Distribution parameters:
  - Mean vector : $m \in \mathbb{R}^d$
  - Covariance matrix: $\Sigma \in \mathbb{R}^{d \times d}$
- Density of the distribution:

$$p(x;\theta) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \cdot \exp\left(-\frac{1}{2}(x-m)^\top \Sigma^{-1}(x-m)\right)$$

Vanilla gradient for Multivariate Gaussian Distribution

- We first calculate the log-likelihood for estimating gradient:

$$\log p(x;\theta) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\Sigma) - \frac{1}{2}(x-m)^\top \Sigma^{-1}(x-m)$$

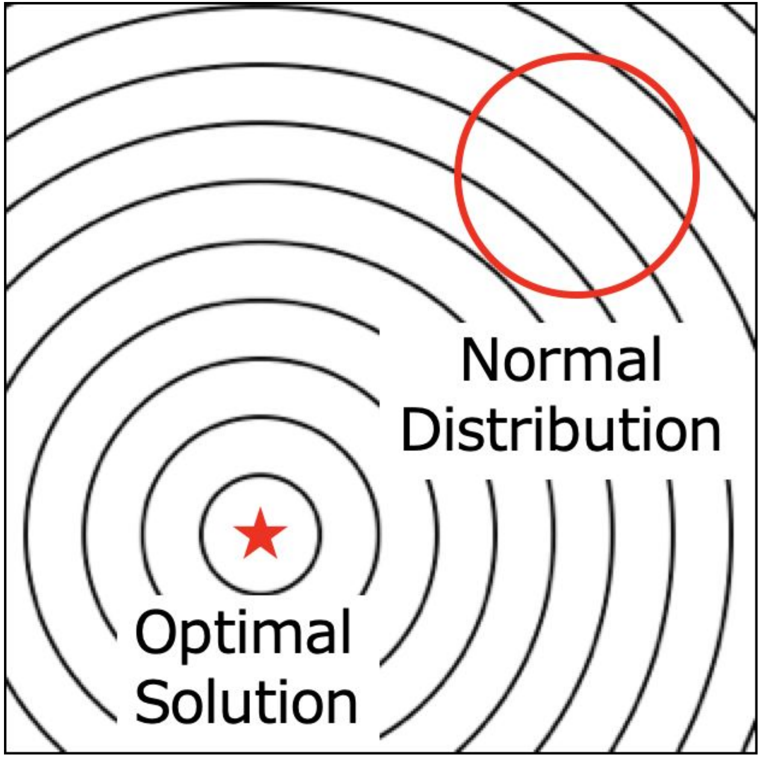
- (Vanilla) gradient for the log-likelihood:

$$\nabla_m \log p(x;\theta) = \Sigma^{-1}(x-m)$$

$$\nabla_\Sigma \log p(x;\theta) = \frac{1}{2}\Sigma^{-1}(x-m)(x-m)^\top - \frac{1}{2}\Sigma^{-1}$$

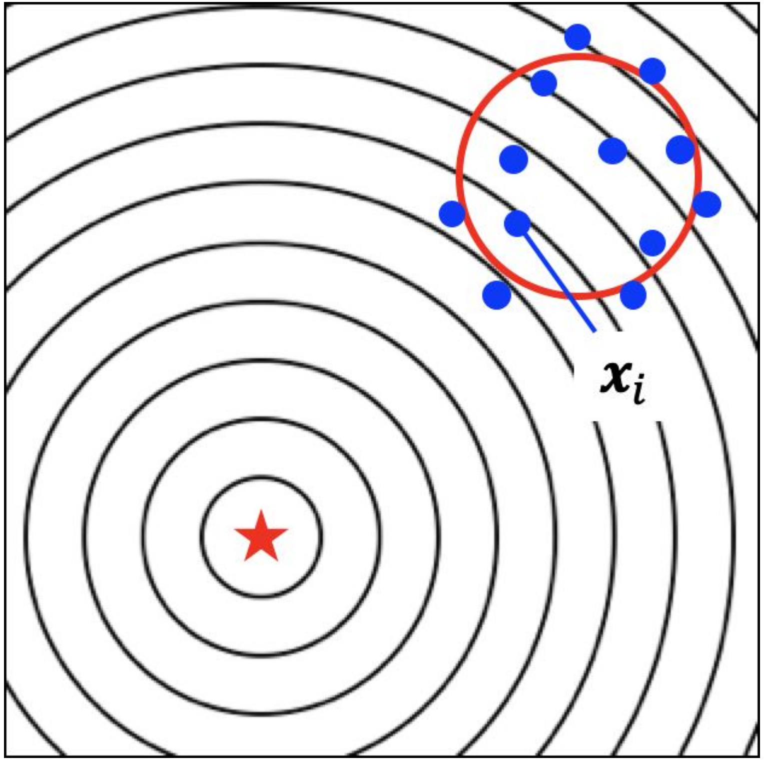
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



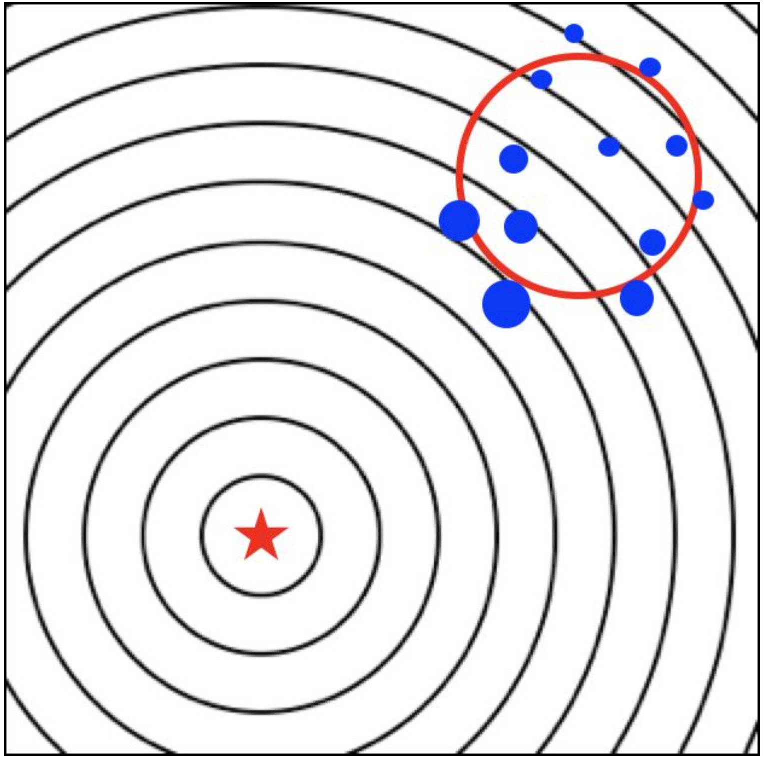
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



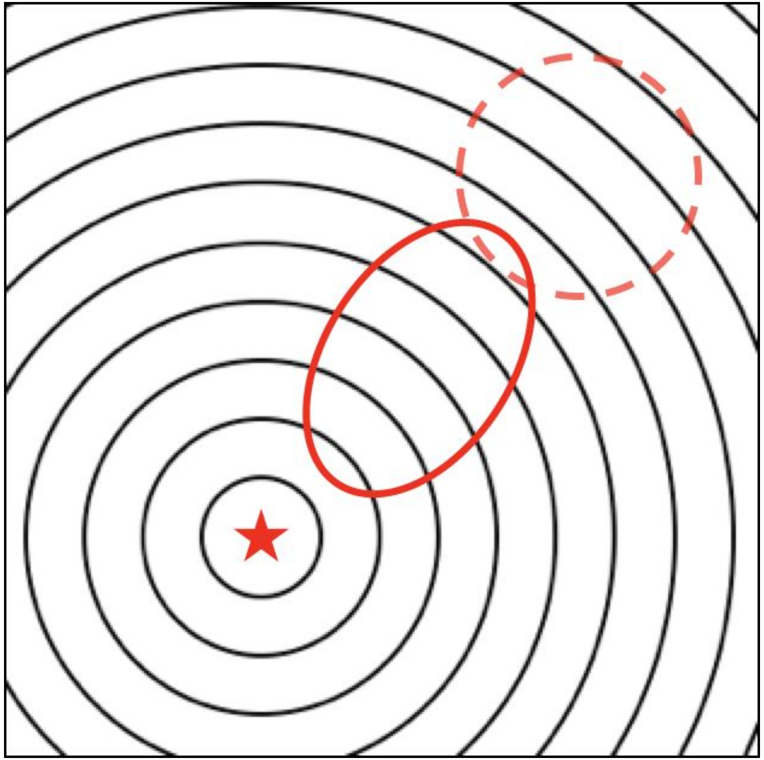
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. <u>Evaluates solutions and estimates gradient</u>
3. Updates the parameters
4. Repeats until the criterion is met



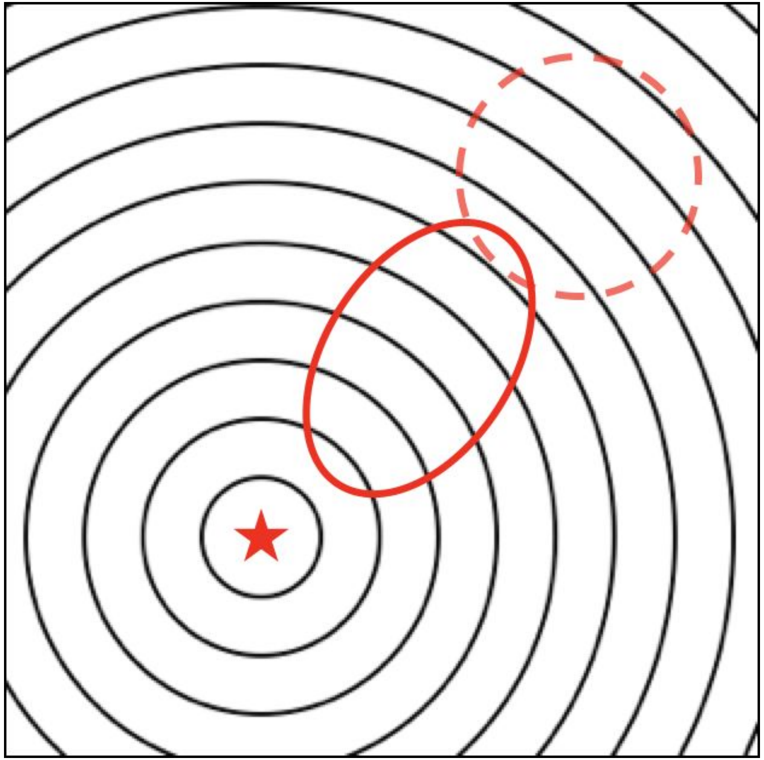
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. <u>Updates the parameters</u>
4. Repeats until the criterion is met



Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. <u>Repeats until the criterion is met</u>



Rethinking Vanilla Gradient and Natural Gradient

- The vanilla gradient can be characterized as the steepest ascent within the small Euclidean distance in the parameter space
  - The update depends on the choice of parameterization
  - The vanilla gradient can show unstable and unsatisfying performance
- Natural gradient: steepest ascent direction within the small KL divergence
  - Let F be Fisher information matrix,

$$\tilde{\nabla}_\theta J(\theta) = F^{-1} \nabla_\theta J(\theta)$$

Fitness Shaping [GSY+10]

- Fitness shaping: replace function values with ***weights*** $(w_1 \geq \cdots \geq w_\lambda)$

$$F^{-1}\nabla_\theta J(\theta) \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} f(x_i) F^{-1}\nabla_\theta \log p(x_i;\theta)$$

Fitness Shaping

$$\sum_{i=1}^{\lambda} w_i F^{-1}\nabla_\theta \log p(x_{i:\lambda};\theta)$$

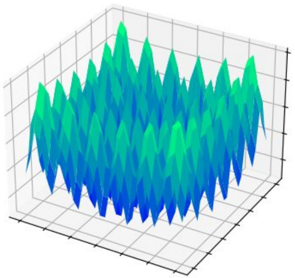
$x_{i:\lambda}$: *i*-th best solution

NES uses only ranking rather than function value itself

- Invariant under monotonically increasing transformation of *f*
- NES = Natural Gradient + Fitness Shaping
  - CMA-ES = NES + additional operations (CSA & rank-one update)

Practical Issues of ES for Difficult Problems

- Why does ES fail for Difficult Problems?
  - Noisy case: update does not proceed at certain situations
  - Multimodal problems are similar to noisy problems



- Naive approach: Sample size tuning
  - Larger sample size can be helpful for difficult problems
  - Knowing good value is challenging ⇒ expensive tuning is required
- Promising approach: Learning rate adaptation
  - Increasing sample size has effect similar to decreasing learning rate
  - Learning rate adaptation is more practically useful

Learning Rate Adaptation: Setup

vectorization operator, Σ = σ^2 C

- Notations :
  - distribution parameters : $\theta_m = m, \theta_\Sigma = \underline{\mathrm{vec}(\Sigma)}$
  - original updates : $\Delta_m^{(t)} = m^{(t+1)} - m^{(t)}, \Delta_\Sigma^{(t)} = \mathrm{vec}(\Sigma^{(t+1)} - \Sigma^{(t)})$
  - learning rate factors : $\eta_m^{(t)}, \eta_\Delta^{(t)}$

Learning Rate Adaptation: Setup

vectorization operator, Σ = σ^2 C

- Notations :
  - distribution parameters : $\theta_m = m, \theta_\Sigma = \underline{\mathrm{vec}(\Sigma)}$
  - original updates : $\Delta_m^{(t)} = m^{(t+1)} - m^{(t)}, \Delta_\Sigma^{(t)} = \mathrm{vec}(\Sigma^{(t+1)} - \Sigma^{(t)})$
  - learning rate factors : $\eta_m^{(t)}, \eta_\Delta^{(t)}$
- Modified updates :
  - $\theta_m^{(t+1)} = \theta_m^{(t)} + \eta_m^{(t)} \Delta_m^{(t)}$
  - $\theta_\Sigma^{(t+1)} = \theta_\Sigma^{(t)} + \eta_\Sigma^{(t)} \Delta_\Sigma^{(t)}$

original updates can be recovered with η =1

Learning Rate Adaptation: Setup

- Notations :

  vectorization operator, Σ = σ^2 C

  - distribution parameters : $\theta_m = m, \theta_\Sigma = \underline{\mathrm{vec}(\Sigma)}$
  - original updates : $\Delta_m^{(t)} = m^{(t+1)} - m^{(t)}, \Delta_\Sigma^{(t)} = \mathrm{vec}(\Sigma^{(t+1)} - \Sigma^{(t)})$
  - learning rate factors : $\eta_m^{(t)}, \eta_\Delta^{(t)}$
- Modified updates :
  - $\theta_m^{(t+1)} = \theta_m^{(t)} + \boxed{\eta_m^{(t)}} \Delta_m^{(t)}$
  - $\theta_\Sigma^{(t+1)} = \theta_\Sigma^{(t)} + \boxed{\eta_\Sigma^{(t)}} \Delta_\Sigma^{(t)}$

original updates can be recovered with η =1

How to adapt these learning rate?

Learning Rate Adaptation: Main Idea

- We adapt the learning rate based on the ***signal-to-noise ratio (SNR)***:

$$\mathrm{SNR} := \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathrm{Tr}(F\,\mathrm{Cov}[\Delta])} = \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathbb{E}[\|\Delta\|_F^2] - \|\mathbb{E}[\Delta]\|_F^2}$$

F: Fisher information matrix

Learning Rate Adaptation: Main Idea

- We adapt the learning rate based on the ***signal-to-noise ratio (SNR)***:

$$\mathrm{SNR} := \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathrm{Tr}(F\,\mathrm{Cov}[\Delta])} = \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathbb{E}[\|\Delta\|_F^2] - \|\mathbb{E}[\Delta]\|_F^2}$$

F: Fisher information matrix

- ***Noisy problems***: ***SNR $\to$ 0 when noise becomes dominant***
  - To improve function value, ***maintaining a positive SNR is crucial***
  - We apply similar arguments to ***multimodal problems***

SNR-Based Learning Rate Adaptation

- Assume LR is small over *n* iterations ⇔ updates are i.i.d
- *n* steps update:

$$\theta^{(t+n)} = \theta^{(t)} + \eta \sum_{k=0}^{n-1} \Delta^{(t+k)}$$

$$\approx \theta^{(t)} + \mathcal{D}\left(n\eta\mathbb{E}[\Delta], n\eta^2\mathrm{Cov}[\Delta]\right)$$

SNR-Based Learning Rate Adaptation

- Assume LR is small over *n* iterations ⇔ updates are i.i.d
- *n* steps update:

$$\theta^{(t+n)} = \theta^{(t)} + \eta \sum_{k=0}^{n-1} \Delta^{(t+k)}$$
$$\approx \theta^{(t)} + \boxed{\mathcal{D}\left(n\eta\mathbb{E}[\Delta], n\eta^2\mathrm{Cov}[\Delta]\right)}$$

- $n = 1/\eta \Rightarrow \mathcal{D}\left(\mathbb{E}[\Delta], \eta\mathrm{Cov}[\Delta]\right)$
  - By taking small η, we can obtain [more concentrated update]()

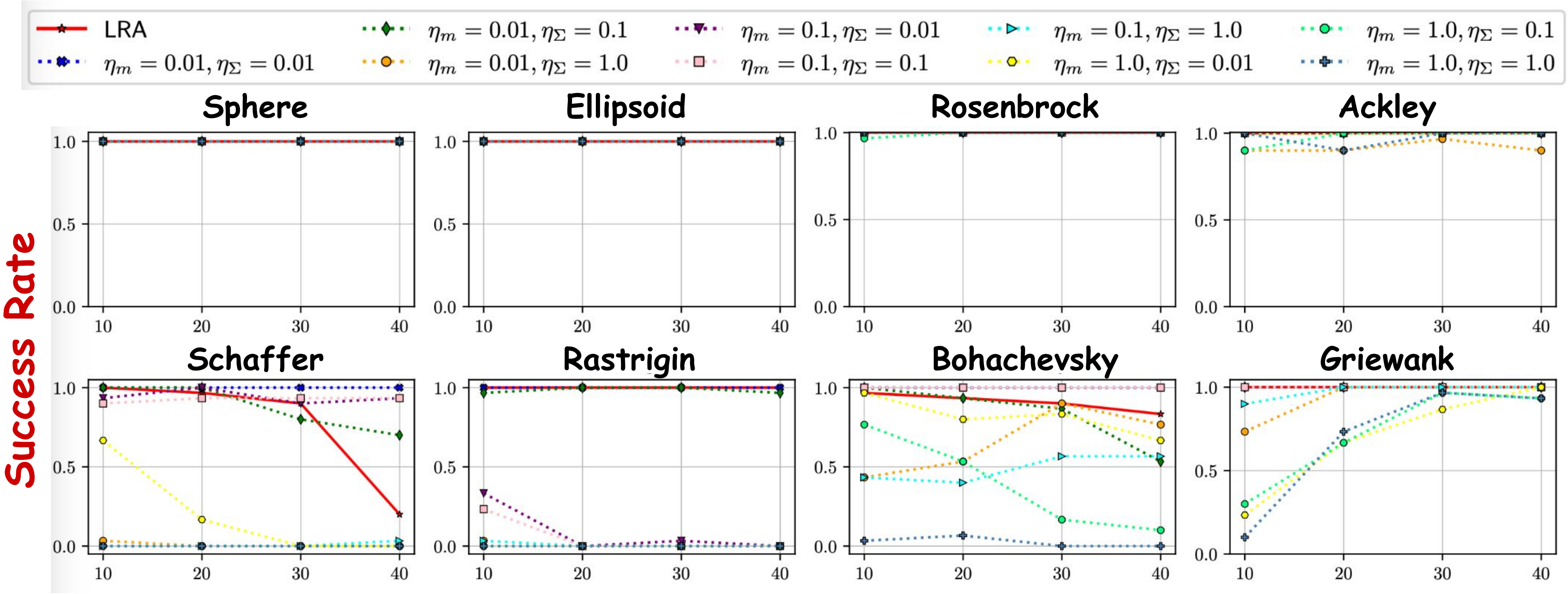
SNR-Based Learning Rate Adaptation

- Assume LR is small over *n* iterations ⇔ updates are i.i.d
- *n* steps update:

$$\theta^{(t+n)} = \theta^{(t)} + \eta \sum_{k=0}^{n-1} \Delta^{(t+k)}$$
$$\approx \theta^{(t)} + \mathcal{D}\left(n\eta\mathbb{E}[\Delta], n\eta^2\mathrm{Cov}[\Delta]\right)$$

- *n = 1/η* ⇒ $\mathcal{D}\left(\mathbb{E}[\Delta], \eta\mathrm{Cov}[\Delta]\right)$
  - By taking small *η*, we can obtain *more concentrated update*
- SNR over *n* iterations: $\dfrac{\|\mathbb{E}[\Delta]\|_F^2}{\eta\,\mathrm{Tr}(F\,\mathrm{Cov}[\Delta])} = \dfrac{1}{\eta}\mathrm{SNR}$
- **Our method**: keep *SNR over n(=1/η) itr.* as *(positive) constant*
  - $\mathrm{SNR} = \alpha\eta \quad (\alpha > 0)$

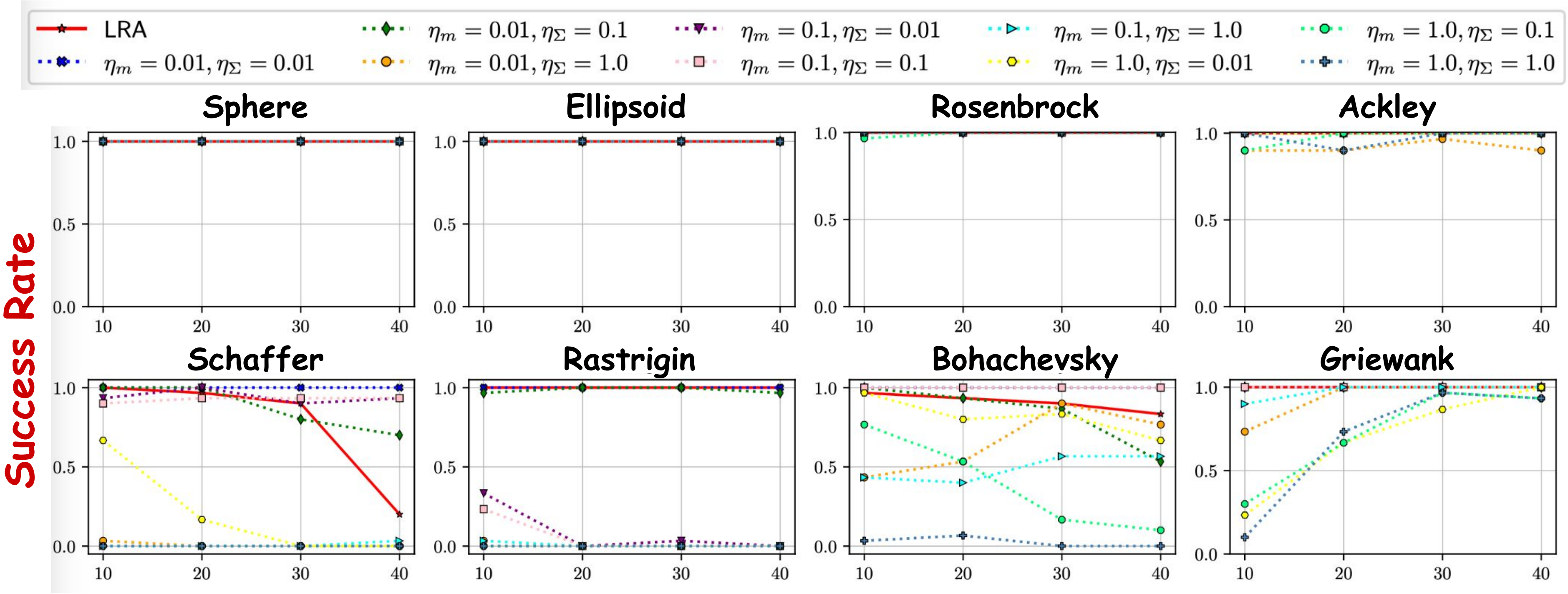
Success Rate versus (10-40)Dim. (Noiseless Problems)



For multimodal, CMA with high η often *failed*, but with small η had a high SR

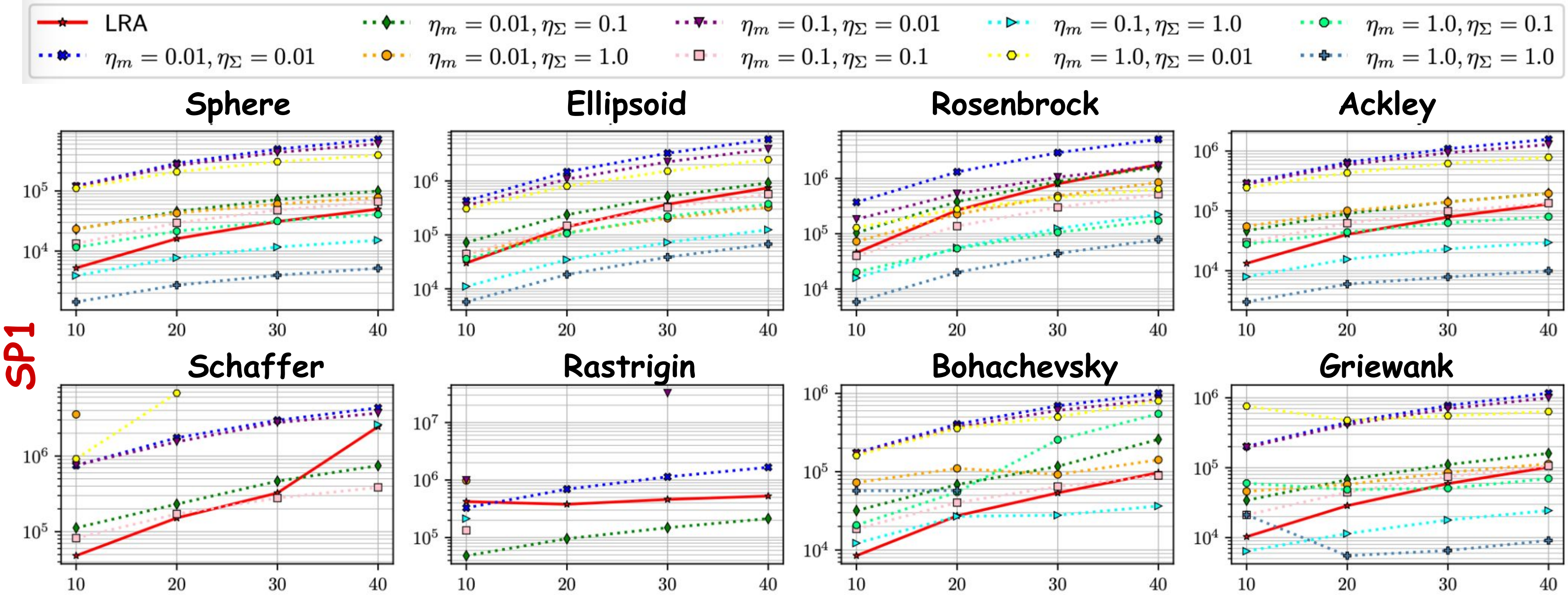
Success is highly dependent on the η setting

Success Rate versus (10-40)Dim. (Noiseless Problems)



LRA-CMA had a relatively *good success rate without η tuning*

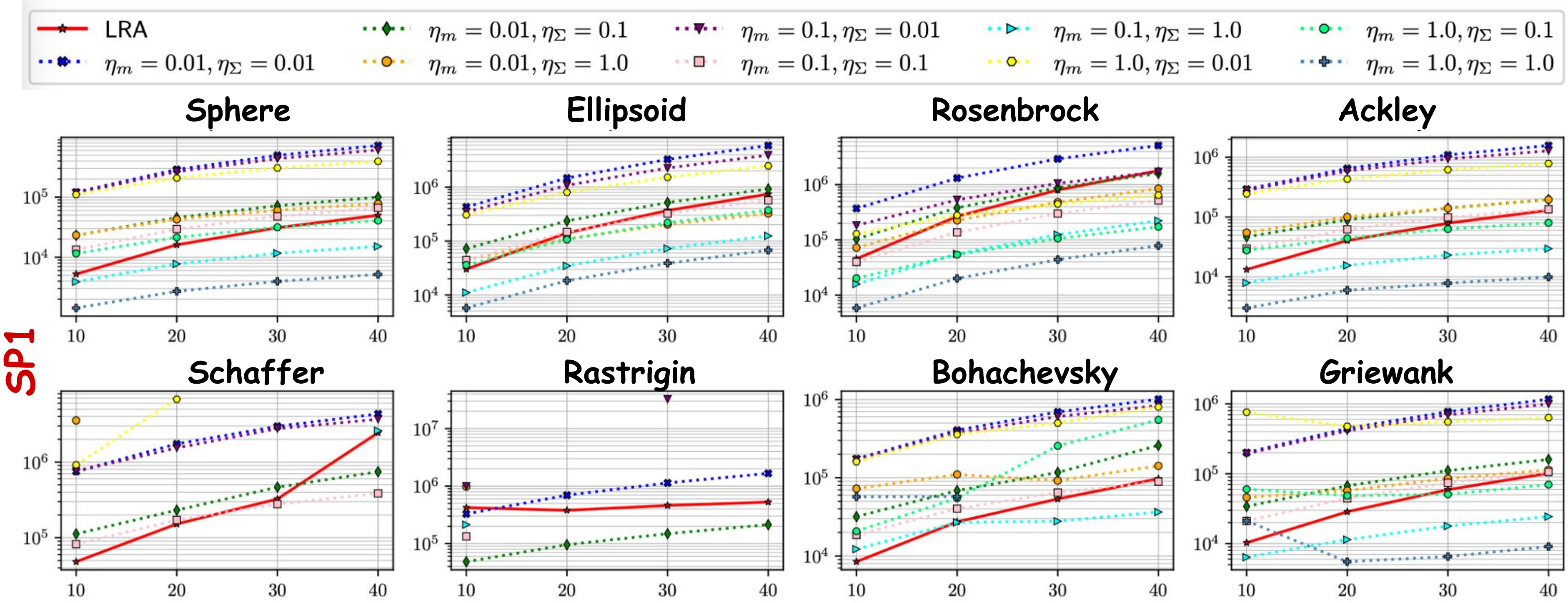
SP1 versus (10-40)Dim. (Noiseless Problems)



with high $\eta \Rightarrow$ worse on multimodal, with small $\eta \Rightarrow$ slow on unimodal

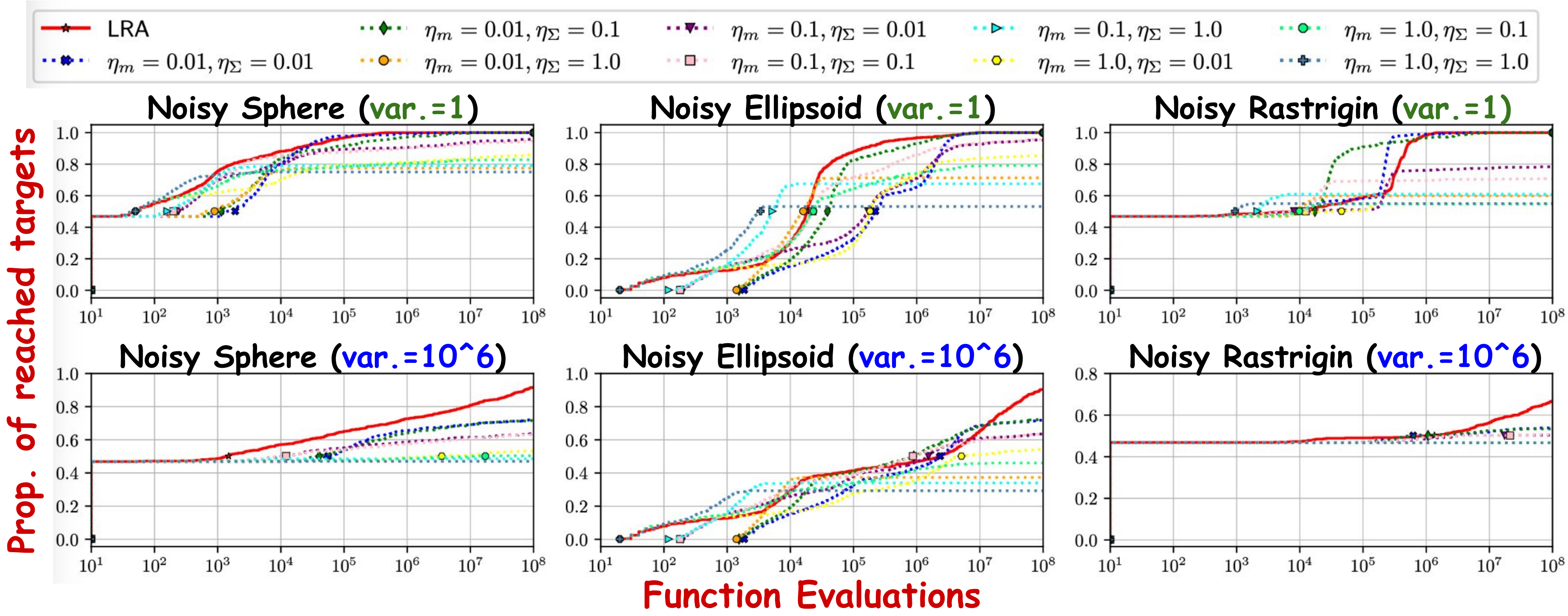
Clear trade-off in efficiency exists depending on η

SP1 versus (10-40)Dim. (Noiseless Problems)



LRA shows stable and relatively good performance <u>without expensive tuning</u>

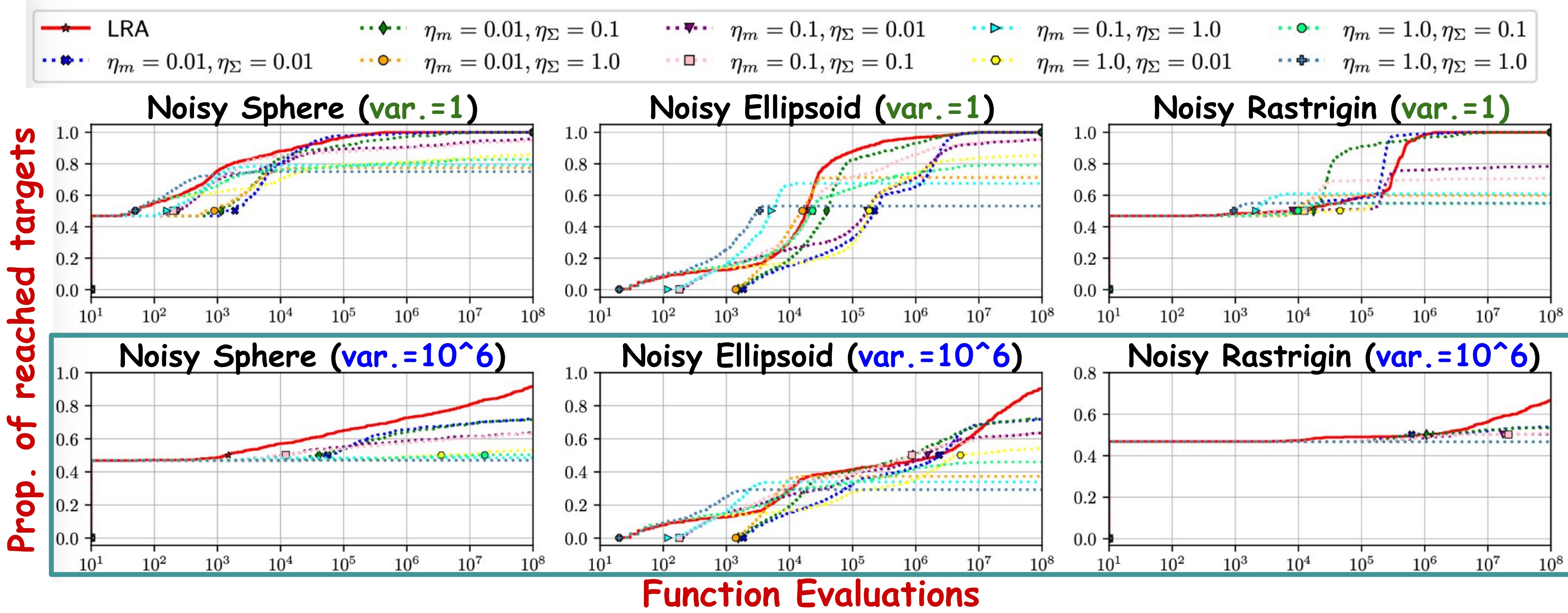
Empirical cumulative density function on noisy problems



CMA with fixed η had stopped improving the function value

In contrast, *LRA continued to improve* it even in strong noise

Empirical cumulative density function on noisy problems



CMA with fixed η had stopped improving the function value

In contrast, *LRA continued to improve* it even in strong noise

Use of LRA-CMA with Python

- Available from [CyberAgentAILab/cmaes]() (#star=233)

```
optimizer = CMA(mean=np.ones(10) * 3, sigma=2.0, lr_adapt=True)
```

Please create issues if you have any problems!

- Available from optuna (#star=8.4k)
    - optuna:
        - popular BBO/HPO software (300k downloads/week)

A wider audience can use LRA-CMA!

Summary

- Natural ES (NES) and CMA-ES
  - Popular class of ES that is based on natural gradient
- Practical Issues for Difficult Problems
  - Hyperparameter (e.g. sample size) tuning is required
- Learning Rate Adaptation:
  - LRA-CMA with default sample size works well without tuning
- Ultimate Goal for Future:
  - Developing Completely Hyperparameter-Free ES

Appendix

SNR Estimation with Moving Averages

- We introduce moving averages for each m and Σ

$$\mathcal{E}^{(t+1)} = (1-\beta)\mathcal{E}^{(t)} + \beta\tilde{\Delta}^{(t)},$$

$$\mathcal{V}^{(t+1)} = (1-\beta)\mathcal{V}^{(t)} + \beta\|\tilde{\Delta}^{(t)}\|_2^2$$

local coordinate

$$\tilde{\Delta}_m = \sqrt{\Sigma}^{-1}\Delta_m$$

$$\tilde{\Delta}_\Sigma = 2^{-\frac{1}{2}}\mathrm{vec}(\sqrt{\Sigma}^{-1}\mathrm{vec}^{-1}(\Delta_\Sigma)\sqrt{\Sigma}^{-1})$$

SNR Estimation with Moving Averages

- We introduce moving averages for each m and Σ

$$\mathcal{E}^{(t+1)} = (1-\beta)\mathcal{E}^{(t)} + \beta\tilde{\Delta}^{(t)},$$
$$\mathcal{V}^{(t+1)} = (1-\beta)\mathcal{V}^{(t)} + \beta\|\tilde{\Delta}^{(t)}\|_2^2$$

local coordinate

$$\tilde{\Delta}_m = \sqrt{\Sigma}^{-1}\Delta_m$$
$$\tilde{\Delta}_\Sigma = 2^{-\frac{1}{2}}\mathrm{vec}(\sqrt{\Sigma}^{-1}\mathrm{vec}^{-1}(\Delta_\Sigma)\sqrt{\Sigma}^{-1})$$

- The SNR is estimated as:

$$\mathrm{SNR} := \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathrm{Tr}(\mathrm{Cov}[\tilde{\Delta}])} = \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathbb{E}[\|\tilde{\Delta}\|^2] - \|\mathbb{E}[\tilde{\Delta}]\|^2},$$
$$\approx \frac{\|\mathcal{E}\|_2^2 - \frac{\beta}{2-\beta}\mathcal{V}}{\mathcal{V} - \|\mathcal{E}\|_2^2} =: \widehat{\mathrm{SNR}}$$

SNR Estimation with Moving Averages

- We introduce moving averages for each m and Σ

$$\mathcal{E}^{(t+1)} = (1-\beta)\mathcal{E}^{(t)} + \beta\tilde{\Delta}^{(t)},$$

$$\mathcal{V}^{(t+1)} = (1-\beta)\mathcal{V}^{(t)} + \beta\|\tilde{\Delta}^{(t)}\|_2^2$$

local coordinate

$$\tilde{\Delta}_m = \sqrt{\Sigma}^{-1}\Delta_m$$

$$\tilde{\Delta}_\Sigma = 2^{-\frac{1}{2}}\mathrm{vec}(\sqrt{\Sigma}^{-1}\mathrm{vec}^{-1}(\Delta_\Sigma)\sqrt{\Sigma}^{-1})$$

- The SNR is estimated as:

$$\mathrm{SNR} := \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathrm{Tr}(\mathrm{Cov}[\tilde{\Delta}])} = \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathbb{E}[\|\tilde{\Delta}\|^2] - \|\mathbb{E}[\tilde{\Delta}]\|^2},$$

$$\approx \frac{\|\mathcal{E}\|_2^2 - \frac{\beta}{2-\beta}\mathcal{V}}{\mathcal{V} - \|\mathcal{E}\|_2^2} =: \widehat{\mathrm{SNR}}$$

Approximation!
(See paper for details)

Update Equation of Learning Rate Adaptation

Adapting learning rate by:

$$\eta \leftarrow \eta \cdot \exp\left(\min(\gamma\eta, \beta)\Pi_{[-1,1]}\left(\frac{\widehat{\mathrm{SNR}}}{\alpha\eta} - 1\right)\right)$$

Update Equation of Learning Rate Adaptation

Adapting learning rate by:

bring SNR closer to $\alpha\eta$

$$\eta \leftarrow \eta \cdot \exp\left(\min(\gamma\eta, \beta)\Pi_{[-1,1]}\left(\boxed{\frac{\widehat{\mathrm{SNR}}}{\alpha\eta} - 1}\right)\right)$$

Update Equation of Learning Rate Adaptation

Adapting learning rate by:

bring SNR closer to $\alpha\eta$

$$\eta \leftarrow \eta \cdot \exp\left(\min(\underline{\gamma\eta}, \underline{\beta}) \Pi_{[-1,1]} \left(\boxed{\frac{\widehat{\mathrm{SNR}}}{\alpha\eta} - 1} \right) \right)$$

wait for the effect of the change of previous η

prevent η to change more than the factor of exp(y) or exp(-y) in 1/η iterations

Update Equation of Learning Rate Adaptation

Adapting learning rate by:

projection onto [-1, 1]

bring SNR closer to $\alpha\eta$

$$\eta \leftarrow \eta \cdot \exp\left(\underbrace{\min(\gamma\eta}_{}, \underbrace{\beta}_{})\underbrace{\Pi_{[-1,1]}}_{}\left(\boxed{\frac{\widehat{\mathrm{SNR}}}{\alpha\eta} - 1}\right)\right)$$

wait for the effect of the change of previous η

prevent η to change more than the factor of exp(y) or exp(-y) in $1/\eta$ iterations

Update Equation of Learning Rate Adaptation

Adapting learning rate by:

projection onto [-1, 1]

bring SNR closer to $\alpha\eta$

$$\eta \leftarrow \eta \cdot \exp\left(\min(\gamma\eta, \beta)\Pi_{[-1,1]}\left(\boxed{\frac{\widehat{\mathrm{SNR}}}{\alpha\eta} - 1}\right)\right)$$

wait for the effect of the change of previous η

prevent η to change more than the factor of exp(y) or exp(-y) in $1/\eta$ iterations

$$\eta \leftarrow \min(\eta, 1)$$

upper bound