GECCO 2019, Workshop on Real-Parameter Black-Box Optimization Benchmarking

Benchmarking GNN-CMA-ES on the BBOB Noiseless Testbed

Louis Faury[*,†], **Clément Calauzènes**[*], **Olivier Ferocq**[†]

[*] Criteo AI Labs

[†] LTCI, Telecom ParisTech

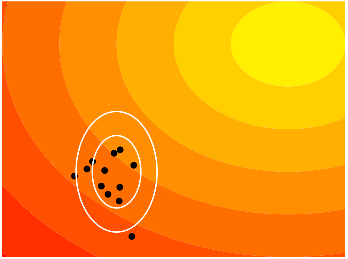
Outline

We investigate the benefits of *expressivity* in search distributions. To do so, we:

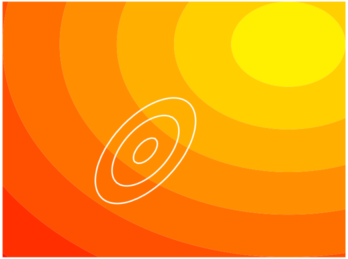
- augment Gaussian search distributions with *generative neural networks*
- propose a plug-in to the CMA-ES to train such distributions
- benchmark its performance on the BBOB noiseless testbed
- discuss results

Goal: Minimize $f : \mathbb{R}^d \to \mathbb{R}$ through *zeroth-order* oracle only

ES approach: Maintain and update a *search distribution* π over $\mathbb{R}^d$;



Sampling (*exploration*)



Updating (*exploitation*)

Motivations (ctn'd)

Typically, $\pi_\omega = \mathcal{N}(\omega)$ and $\omega = (\mu, \Sigma)$.

CMA-ES Update π_ω via *heuristic* mechanisms.

NES Update π_ω via *natural gradient descent* of the objective:

$$J(\omega) = \mathbb{E}_{x \sim \pi_\omega}\left[f(x)\right]$$

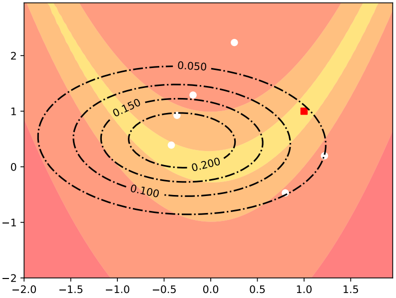
Motivations (ctn'd)

We argue that:

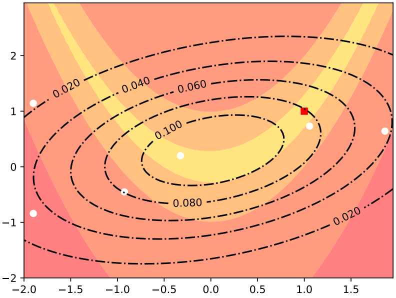
- Standard distributions are too *rigid*
- Can be a *harmful constraint* for the stochastic search
- ES algorithms can benefit from *flexible* distributions (asymmetric, multimodal, ..)

Motivations (ctn'd)

The example of the Rosenbrock function:



Gaussian search



Our method

Motivations (ctn'd)

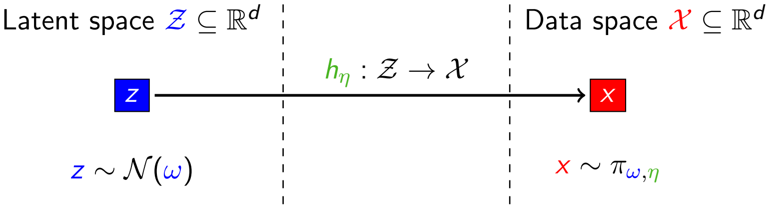
Flexible search distributions for ES must satisfy the following:

- Flexible! (asymmetric, potentially multimodal, ..)
- Easily trainable
- Leave the exploration/exploitation trade-off to the ES

Neural Normalizing Flows (NNF):

- family of *generative neural networks* for which the likelihood $\pi_\theta(x)$ is easily **computable** and **differentiable**.
- $\Rightarrow$ trainable via gradient descent (maximum likelihood principle).

Normalizing flows (ctn'd)



If h_η is bijective, *change of variable formula*:

$$\pi_{\omega,\eta}(x) = \phi_\omega({h_\eta}^{-1}(x)) \left| \frac{\partial {h_\eta}^{-1}(x)}{\partial x} \right|, \quad \phi_\omega \text{ p.d.f of } \mathcal{N}(\omega)$$

The NICE model

The NICE [Dinh et al, 2017] model is a NNF that is *volume preserving*:

$$\forall x \in \mathcal{X}, \quad \left| \frac{\partial h_\eta^{-1}(x)}{\partial x} \right| = 1$$

h_η is hierarchically built (with neural networks) to ensure invertibility.

Volume preserving: **the exploration/exploitation trade-off is left to the latent distribution**.

The NICE model (ctn'd)

The NICE model checks our needs:

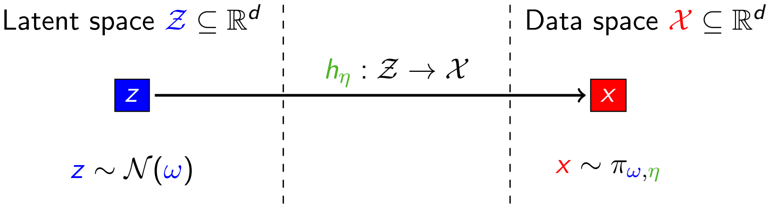
- easily trainable via gradient descent
- flexible (see [Dinh et al, 2017])
- volume preserving (exploitation/exploration trade-off)

The NICE model (ctn'd)

The NICE model checks our needs:

- easily trainable via gradient descent
- flexible (see [Dinh et al, 2017])
- volume preserving (exploitation/exploration trade-off)

$\Bigg\}$ **how** de we train it for ES?



$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}} [f(x)]$$

A tailored algorithm for GNN-based ES

Latent space $\mathcal{Z} \subseteq \mathbb{R}^d$

Data space $\mathcal{X} \subseteq \mathbb{R}^d$

$$z \xrightarrow{h_\eta : \mathcal{Z} \to \mathcal{X}} x$$

$z \sim \mathcal{N}(\omega)$

$x \sim \pi_{\omega,\eta}$

$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}} [f(x)] = \mathbb{E}_{z \sim \mathcal{N}(\omega)} [f(h_\eta(z))]$$

A tailored algorithm for GNN-based ES (ctn'd)

$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}}[f(x)] = \mathbb{E}_{z \sim \mathcal{N}(\omega)}[f(h_\eta(z))]$$

A tailored algorithm for GNN-based ES (ctn'd)

$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}}[f(x)] = \mathbb{E}_{z \sim \mathcal{N}(\omega)}[f(h_\eta(z))]$$

- $\mathcal{N}(\omega)$ minimizes the function $f \circ h_\eta$ **(done by ES algorithm)**
- h_η provides a representation $f \circ h_\eta$ more adapted to $\mathcal{N}(\omega)$

A tailored algorithm for GNN-based ES (ctn'd)

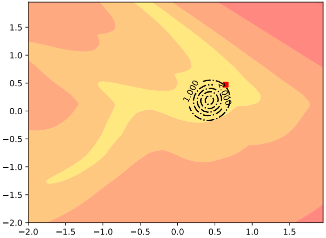
$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}}[f(x)] = \mathbb{E}_{z \sim \mathcal{N}(\omega)}[f(h_\eta(z))]$$

- $\mathcal{N}(\omega)$ minimizes the function $f \circ h_\eta$ **(done by ES algorithm)**
- h_η provides a representation $f \circ h_\eta$ more adapted to $\mathcal{N}(\omega)$ **(plug-in)**

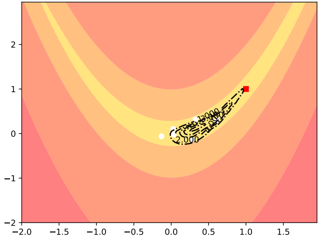
A tailored algorithm for GNN-based ES (ctn'd)

$$J(\omega, \eta) = \mathbb{E}_{x \sim \pi_{\omega,\eta}}[f(x)] = \mathbb{E}_{z \sim \mathcal{N}(\omega)}[f(h_\eta(z))]$$

- $\mathcal{N}(\omega)$ minimizes the function $f \circ h_\eta$ **(done by ES algorithm)**
- h_η provides a representation $f \circ h_\eta$ more adapted to $\mathcal{N}(\omega)$ **(plug-in)**



Latent space



Data space

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \text{argmin}_{\eta} \ \mathbb{E}_{x \sim \pi_{\omega_t, \eta}} [f(x)]$$

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \operatorname{argmin}_{\eta} \mathbb{E}_{x \sim \pi_{\omega_t,\eta}} [f(x)]$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \mathrm{argmin}_{\eta} \; \mathbb{E}_{x \sim \pi_0} \left[f(x) \frac{\pi_{\omega_t,\eta}(x)}{\pi_0(x)} \right]$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \operatorname{argmin}_\eta \mathbb{E}_{x \sim \pi_0} \left[f(x) \frac{\pi_{\omega_t,\eta}(x)}{\pi_0(x)} \right]$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)
- Kullback-Leibler penalty to limit the *non-stationarity* of $f \circ h_\eta$

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \mathrm{argmin}_{\eta}\ \mathbb{E}_{x\sim\pi_0}\left[f(x)\frac{\pi_{\omega_t,\eta}(x)}{\pi_0(x)}\right] + \beta_t \mathrm{KL}\left(\pi_{\omega_t,\eta}||\pi_{\omega_t,\eta_t}\right)$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)
- Kullback-Leibler penalty to limit the *non-stationarity* of $f \circ h_\eta$

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \mathrm{argmin}_{\eta}\ \mathbb{E}_{x\sim\pi_0}\left[f(x)\frac{\pi_{\omega_t,\eta}(x)}{\pi_0(x)}\right] + \beta_t \mathrm{KL}\left(\pi_{\omega_t,\eta} || \pi_{\omega_t,\eta_t}\right)$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)
- Kullback-Leibler penalty to limit the *non-stationarity* of $f \circ h_\eta$
- Solved by a stochastic gradient descent solver

A tailored algorithm for GNN-based ES (ctn'd)

Training the NNF;

$$\eta_{t+1} \in \operatorname{argmin}_{\eta} \overbrace{\mathbb{E}_{x\sim\pi_0}\left[f(x)\frac{\pi_{\omega_t,\eta}(x)}{\pi_0(x)}\right] + \beta_t \mathrm{KL}\left(\pi_{\omega_t,\eta}||\pi_{\omega_t,\eta_t}\right)}^{\triangleq J_{NNF}(\eta)}$$

Additional tools:

- *Off-policy* (importance weights) to incorporate past samples (data exposure)
- Kullback-Leibler penalty to limit the *non-stationarity* of $f \circ h_\eta$
- Solved by a stochastic gradient descent solver

A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$

A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t, \eta_t}(x_{1:\lambda})$

A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t, \eta_t}(x_{1:\lambda})$
3. Append (X, F, L) to buffer history

A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t,\eta_t}(x_{1:\lambda})$
3. Append (X, F, L) to buffer history
4. Apply ES algorithm to $\mathcal{N}(\omega_t)$ with (Z, F).

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t, \eta_t}(x_{1:\lambda})$
3. Append (X, F, L) to buffer history
4. Apply ES algorithm to $\mathcal{N}(\omega_t)$ with (Z, F). $\left(\min_\omega J(\omega, \eta_t) \right)$

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

1. Sample $Z \triangleq z_{1:\lambda} \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t, \eta_t}(x_{1:\lambda})$
3. Append (X, F, L) to buffer history
4. Apply ES algorithm to $\mathcal{N}(\omega_t)$ with (Z, F). $\left(\min_\omega J(\omega, \eta_t)\right)$
5. Minimize $J_{NNF}(\omega_{t+1}, \eta)$

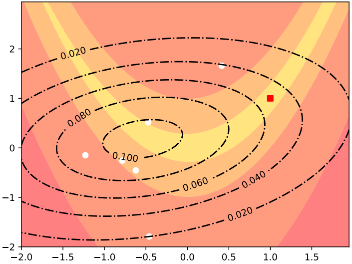
A tailored algorithm for GNN-based ES (ctn'd)

Algorithm: ***alternating minimization*** of $J(\omega, \eta)$;

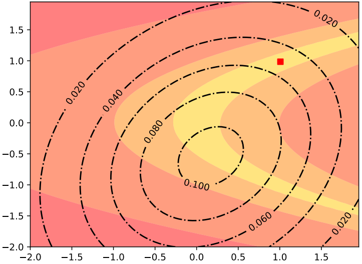
1. Sample $Z \triangleq z_1 : \lambda \overset{\text{i.i.d}}{\sim} \mathcal{N}(\omega_t)$
2. Evaluate $X = x_{1:\lambda}$ (push-forward), $F = f_{1:\lambda}$, $L = \log \pi_{\omega_t, \eta_t}(x_{1:\lambda})$
3. Append (X, F, L) to buffer history
4. Apply ES algorithm to $\mathcal{N}(\omega_t)$ with (Z, F). $\left(\min_\omega J(\omega, \eta_t) \right)$
5. Minimize $J_{NNF}(\omega_{t+1}, \eta)$ $\left(\min_\eta J(\omega_{t+1}, \eta) \right)$

GNN-CMA-ES

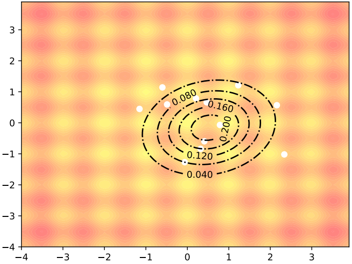
- latent space optimization is performed by the CMA-ES [Hansen & Ostermeier, 2001].
- NNF hyper-parmeters:
  - 3 times ($d \rightarrow 16 \rightarrow d$) multi-layer perceptrons
  - Hyperbolic tangent activations
  - History size $= 3\lambda$
  - KL constraint: $\mathrm{KL}\left(\pi_{\omega_{t+\mathbf{1}},\eta_t} || \pi_{\omega_{t+\mathbf{1}},\eta_{t+\mathbf{1}}}\right) \leq 0.01$



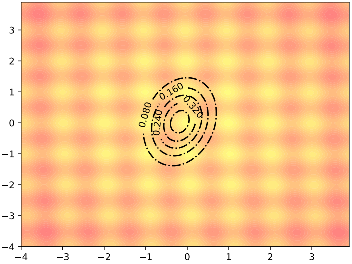
Rosenbrock (data space)



Rosenbrock (latent space)



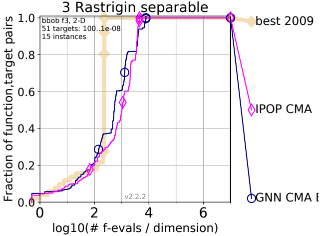
Rastrigin (data space)



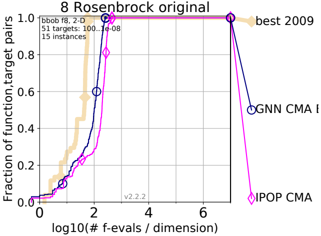
Rastrigin (latent space)

Results on the BBOB 2018 noiseless function suites

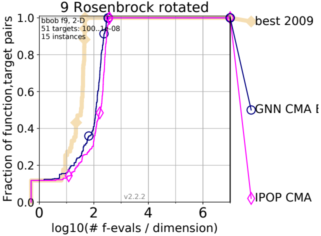
Individual functions:



f_3 2-d



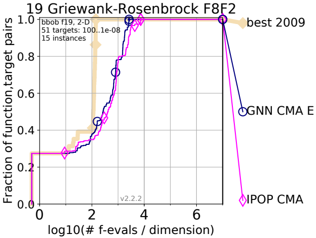
f_8 2-d



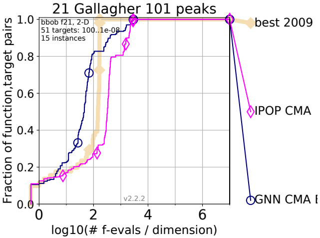
f_9 2-d

Results (BBOB'09)

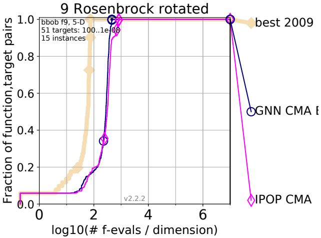
Individual functions:



f_{19} 2-d

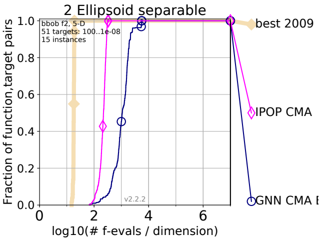


f_{21} 2-d

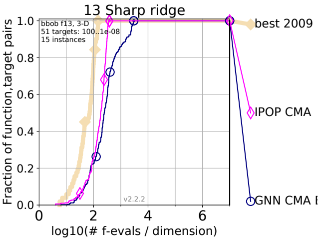


f_9 5-d

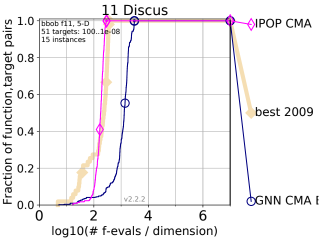
Poor result in ill-conditioned ellipsoidal functions:



f_2 5-d



f_{13} 3-d

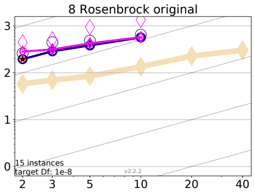


f_{11} 5-d

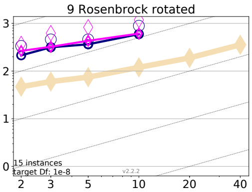
Results (BBOB'09) (ctn'd)

Scaling with dimensions:

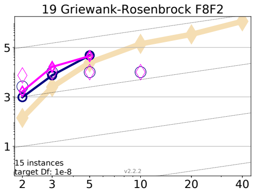
CMA-ES GNN-CMA-ES



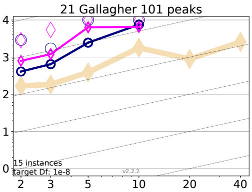
f_8



f_9



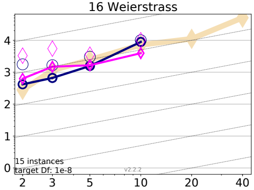
f_{19}



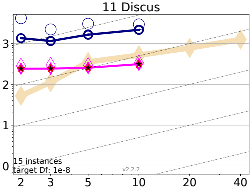
f_{21}

Scaling with dimensions:

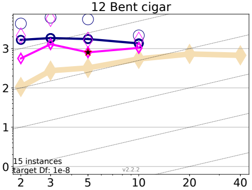
CMA-ES GNN-CMA-ES



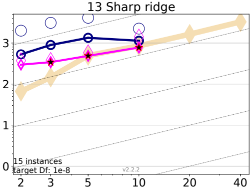
f_{16}



f_{11}



f_{12}



f_{13}

Discussion

Conclusions:

- Flexibility accelerates ES algorithms on "some" functions..
- but hinders it in landscapes where the Gaussian is well adapted.
- Positive effects seem to disappear as the dimension increases.

Discussion

Conclusions:

- Flexibility accelerates ES algorithms on "some" functions..
- but hinders it in landscapes where the Gaussian is well adapted.
- Positive effects seem to disappear as the dimension increases.

Future work:

- Reduce computational load.
- Detect when triggering the NNF training is useful.
- Use larger historic to improve in high dimensions.
- Regularization in the latent space

Thank you!