Global Optimization with a Power-Transformed Objective and Gaussian Smoothing[1]

Chen Xu
Department of Engineering, Shenzhen MSU-BIT University, China.

ICML
International Conference
On Machine Learning

[1] Chen Xu. Global Optimization with a Power-Transformed Objective and Gaussian Smoothing. ICML, 2025 (Poster)

Outline

The Problem to Solve

We propose a novel method, namely Gaussian Smoothing with a Power-Transformed Objective (GS-PowerOpt), that solves the following continuous optimization problems

$$\max_{\boldsymbol{x} \in \mathcal{S} \subset \mathbb{R}^d} f(\boldsymbol{x}), \tag{1}$$

where $\mathcal{S}$ is a compact set and $f : \mathcal{S} \to \mathbb{R}$ is a continuous and possibly non-concave function with a *unique* global maximum point $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{S}} f(\boldsymbol{x})$.

Gradient-Based Methods

The gradient-based methods (e.g., stochastic gradient ascent) are the most popular ones. However,

- they require the derivative of f; and
- they are likely to be trapped in local optimum points ([11], [9], [4]).

Evolutionary Algorithms

- (Advantage) The evolutionary algorithms, such as symmetric annealing ([15]), particle swarm optimization ([12]), and CMA-ES ([6]), do not require the derivative of f, and may avoid local optimums.
- (Drawback) However, they suffer from the curse of dimensionality.
- (Drawback) The convergence theories are not complete for most EA algorithms.

Global Optimization with Gaussian Smoothing

Gaussian smoothing refers to convolving the objective f with a Gaussian density $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_d)$:

$$\hat{f}_\sigma(\boldsymbol{\mu}) := \frac{1}{(\sqrt{2\pi}\sigma)^d} \int_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) e^{-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|^2}{2\sigma^2}} d\boldsymbol{x}$$
$$= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu},\sigma^2 I_d)}[f(\boldsymbol{x})].$$

- This conversion possibly smooth out local extremes of f.
- It is applied by a category of intensively studied methods named ***homotopy for optimization*** (e.g., [1], [13], [7], [8], [10]).

Standard Homotopy

The standard homotopy (e.g., [7]) creates a schedule $\{\sigma_j\} \to 0$, solve $\boldsymbol{\mu}_0^* := \arg\max_{\boldsymbol{\mu}} \hat{f}_{\sigma_0}(\boldsymbol{\mu})$, and performs the following double-loop mechanism:

1. Let $\boldsymbol{\mu}_j^*$ be the starting point for solving $\max_{\boldsymbol{\mu}} \hat{f}_{\sigma_{j+1}}(\boldsymbol{\mu})$ (outer loop for j).
2. Solve $\boldsymbol{\mu}_{j+1}^* := \arg\max_{\boldsymbol{\mu}} \hat{f}_{\sigma_{j+1}}(\boldsymbol{\mu})$ (inner loop);

$\boldsymbol{\mu}_\infty^* = \boldsymbol{x}^* := \arg\max_{\boldsymbol{x}} f(\boldsymbol{x})$ (e.g., [7]).

Single-loop Gaussian Homotopy (SLGH)

- The double-loop mechanism in the standard homotopy is costly in time.
- (Advantage) To tackle this issue, SLGH ([8]) applies a single-loop mechanism. Specifically, it updates $\boldsymbol{\mu}$ and σ in each iteration.
- (Drawback) In theory, SLGH is only guaranteed to locate optimum that is possibly local.

Advantage of the Proposed Method: GS-PowerOpt

- Our proposed algorithm, GS-PowerOpt, is significantly faster than the standard homotopy method for optimization, both empirically and theoretically ($O(d^2\epsilon^{-2})$ versus $O(d^2\epsilon^{-4})$ in iteration complexity).
- In theory, GS-PowerOpt is able to approximate the global optimum point $\boldsymbol{x}^*$, while SLGH is only guaranteed to locate optimum that is possibly local.

Motivation

- From a few examples, we have found that if we modify the objective f to increase the gap between $f(\boldsymbol{x}^*)$ and the f-value at other points, its Gaussian smooth will have a unique global maximum $\boldsymbol{\mu}^*$.
- If the gap continues to increase, $\boldsymbol{\mu}^*$ gets closer to $\boldsymbol{x}^* := \arg\max_{\boldsymbol{x}} f(\boldsymbol{x})$.

Illustration of the Motivation

- Define $f_N(x) := e^{Nf(x)}$.
- The gap $f_N(x^*) - f_N(x)$ is enlarged by increasing N.
- Gaussian smooth: $F_{N,\sigma}(\mu) := \mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma^2)}[f_N(x)]$, where $\sigma = 0.5$.

The right figure show that

- $F_{N,\sigma}(\mu)$ has a unique maximum μ^*.
- As we enlarge the gap $f_N(x^*) - f_N(x)$ (i.e., increases N), μ^* approaches $x^* := \arg\max_x f(x)$.

$f(\mu) = -\log((\mu+0.5)^2 + 10^{-5}) - \log((\mu-0.5)^2 + 10^{-2}) + 10$ for $|\mu| \le 1$ and $f(\mu) = 0$ for $|\mu| > 1$;

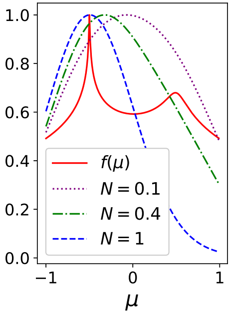


Figure: Graphs of $f(\mu)$ and $F_{N,\sigma}(\mu)$. All function graphs are scaled to have a maximum value of 1 for easier comparisons.

The Inspired Scheme for Finding $\boldsymbol{x}^*$

From the previous example, $\boldsymbol{\mu}^* := \arg\max_{\boldsymbol{\mu}} F_{N,\sigma}(\mu)$ approaches $\boldsymbol{x}^*$ as we increase N. Therefore, the inspired method for finding $\boldsymbol{x}^*$ is through finding $\boldsymbol{\mu}^*$ under a large value of N.

Theoretical Justification of the Motivation

According to Lemma 3.4 in our paper, for any $\sigma > 0$ and $\delta > 0$, as long as N is sufficiently large (depends on σ and δ), all the maximum points of $F_{N,\sigma}(\boldsymbol{\mu})$ lie in a δ-neighborhood of $\boldsymbol{x}^*$.

Updating Rule of GS-PowerOpt

Since $\boldsymbol{\mu}^* := \arg\max_{\boldsymbol{\mu}} F_{N,\sigma}(\boldsymbol{\mu})$ is close to $\boldsymbol{x}^*$ for sufficiently large N, GS-PowerOpt solves the surrogate objective

$$\boldsymbol{\mu}^* := \arg\max_{\boldsymbol{\mu}} F_{N,\sigma}(\boldsymbol{\mu}),$$

where N is pre-selected (it is treated as a hyper-parameter). Stochastic gradient ascent is used to solve this objective:

$$\text{GS-PowerOpt :} \qquad \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t \hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t), \tag{2}$$

where $\hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t) := \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{x}_k - \boldsymbol{\mu}_t) f_N(\boldsymbol{x}_k)$, and $\{\boldsymbol{x}_k\}_{k=1}^{K}$ are independently sampled from the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_t, \sigma^2 I_d)$.

A Flowchart of GS-PowerOpt

$$f(x) \xrightarrow[\text{transform}]{\text{(Exp) power-}N} f_N(x) \xrightarrow[\text{Smoothing}]{\text{Gaussian}} F_{N,\sigma}(\mu) \xrightarrow[\text{Ascent}]{\text{Stochastic Gradident}} \mu^*$$

What's New

To our knowledge, this is the first work that proposes the idea[2] of putting sufficiently large weight on the global maximum values of the objective, to decrease the distance between the optimum point before and after Gaussian smoothing (i.e., $\|\boldsymbol{x}^* - \boldsymbol{\mu}^*\|$).

[2] [5, 14, 3], which involve power transforms, have not mentioned this idea.

Convergence Analysis

- Let $\{\boldsymbol{\mu}_t\}$ denote the sequence produced by the GS-PowerOpt updating equation
$$\text{GS-PowerOpt}: \qquad \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t \hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t).$$
- Let $\boldsymbol{\nu}_t$ denote the point in $\{\boldsymbol{\mu}_\tau\}_{\tau=0}^{t}$ that minimizes $\mathbb{E}[\|\nabla F_{N,\sigma}(\boldsymbol{\mu}_t)\|^2]$.
- According to Corollary 3.9, $\lim_{t\to\infty} \mathbb{E}[\|\nabla F_{N,\sigma}(\boldsymbol{\nu}_t)\|^2] = 0$, with an iteration complexity of $O_N(d^4\epsilon^{-2})$. Hence, $\boldsymbol{\nu}_\infty$ is a stationary point of $F_{N,\sigma}$.
- From Lemma 3.4, for any $\sigma > 0$ and arbitrarily small $\delta > 0$, there exists a sufficiently large N such that any stationary point of $F_{N,\sigma}(\boldsymbol{\mu})$ lies in a δ-neighborhood of $\boldsymbol{x}^*$.
- Hence, $\boldsymbol{\nu}_\infty$ lies in a δ-neighborhood of $\boldsymbol{x}^*$.

Stronger Results under Additional Assumptions

- From Proposition 3.11, the iteration complexity reduces to $O_N(d^2\epsilon^{-2})$ if we additionally assume the Lipschitz conditions on f and its derivative ∇f:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L_0\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L_1\|\boldsymbol{x} - \boldsymbol{y}\|.$$

- From Corollary 4.2, the iteration complexity's dependence on N can be removed if we further assume that $f(\boldsymbol{x}) \in [0, 1)$ if we set $f_N = f^N$, and $f(\boldsymbol{x}) < 0$ if we set $f_N = e^{Nf}$.

(Exponential) Power Gaussian Smoothing

- The only difference between GS-PowerOpt and the two designed algorithms, the power Gaussian smoothing (PGS) and the exponential power Gaussian smoothing (EPGS), is that PGS and EPGS update $\boldsymbol{\mu}_t$ with the normalized gradient estimate, i.e.,

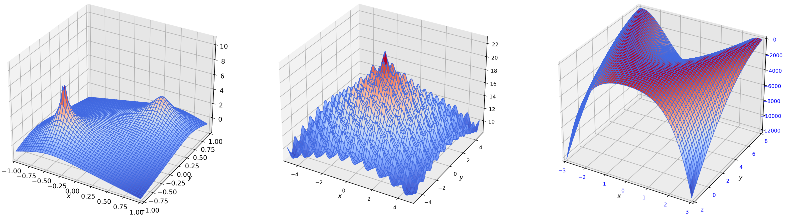
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t \hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t) / \|\hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t)\|.$$

- For PGS, $f_N(\boldsymbol{x}) = f^N(\boldsymbol{x})$ (applied only for a non-negative objective f).
- For EPGS, $f_N(\boldsymbol{x}) = e^{Nf(\boldsymbol{x})}$.

Algorithm PGS/EPGS

Input: The power $N > 0$, the scaling parameter $\sigma > 0$, the objective f, the initial value $\boldsymbol{\mu}_0$, the number K of sampled points for gradient approximation, the total number T of $\boldsymbol{\mu}$-updates, and the learning rate schedule $\{\alpha_t\}_{t=1}^{T}$.
for t from 0 to $T-1$ **do**
    Independently sample from $\mathcal{N}(\boldsymbol{\mu}_t, \sigma^2 I_d)$ and obtain $\{\boldsymbol{x}_k\}_{k=1}^{K}$.
    $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t \hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t) / \|\hat{\nabla} F_{N,\sigma}(\boldsymbol{\mu}_t)\|$.
end for
Return $\{\boldsymbol{\mu}_t\}_{t=1}^{N}$, from which $\boldsymbol{\mu}^*$ is selected to approximate $\boldsymbol{x}^*$ (e.g., $\boldsymbol{\mu}^* := \arg\max_{t \in \{1,2,\ldots,T\}} f(\boldsymbol{\mu}_t)$).

Benchmark 2D Objectives to be Optimized



(a) Two-log Objective

(b) Ackley Func. (Max-Version)

(c) Rosenbrock Func. (Max-Version).

Figure: Graph of the Benchmark Objective Functions.

Experiment on Optimizing the Two-log Objective

Obj. $f(\boldsymbol{x}) = -\log(\|\boldsymbol{x}-\boldsymbol{m}_1\|^2 + 10^{-5}) - \log(\|\boldsymbol{x}-\boldsymbol{m}_2\|^2 + 10^{-2})$.

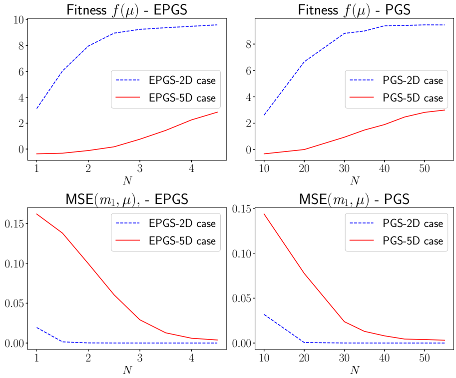


Figure: Effects of Increasing N. We apply PGS/EPGS to solve an example problem $\max_{\boldsymbol{x}} f(\boldsymbol{x})$. The output from PGS/EPGS is denoted by μ, and the global optimum of f is denoted by m_1. The graph shows that the result improves as N increases.

Experiment on Optimizing Ackley

Obj. $f(x,y) = 20e^{-0.2\sqrt{0.5(x^2+y^2)}} + e^{0.5(\cos(2\pi x)+\cos(2\pi y))}$.

Table: Performances on Maximizing Ackley. "Iter. Taken" refers to the number of iterations taken to reach the best found solution. The true solution $\boldsymbol{x}^* = (0,0)$.

Algorithm	Iter. Taken	Best Solution Found ($\boldsymbol{\mu}^*$)	$f(\boldsymbol{\mu}^*)$
CMA-ES	116	$(0.0, 0.0)$	22.718
EPGS ($N=1$)	143	$(0.001, 0.0)$	22.683
PGS ($N=20$)	141	$(0.001, 0.002)$	22.678
ZOSLGHr	131	$(0.001, -0.002)$	22.621
ZOAdaMM	158	$(0.005, 0.001)$	22.613
ZOSLGHd	123	$(-0.003, -0.001)$	22.61
ZOSGD	174	$(0.005, 0.007)$	22.596
STD-Homotopy	194	$(0.962, 0.946)$	17.58

Experiment on Optimizing Rosenbrock

Obj. $f(x, y) = -100(y - x^2)^2 - (1 - x)^2$.

Table: Performances on Maximizing Rosenbrock. For PGS, the Rosenbrock is added by $20,000$ to ensure the search agent only encounter positive values. The global maximum point $\boldsymbol{x}^*$ of the Rosenbrock function is (1,1).

Algorithm	Iter. Taken	Best Solution Found ($\boldsymbol{\mu}^*$)	$f(\boldsymbol{\mu}^*)$
CMA-ES	72	$(1.0, 1.0)$.	0.0
EPGS ($N = 3$)	487.	$(0.999, 1.000)$	-0.017
STD-Homotopy	624	$(0.903, 0.885)$	-2.401
PGS ($N = 1$)	513	$(0.773, 1.025)$	-22.84
ZOAdaMM	852	$(0.004, 0.618)$.	-39.206
ZOSLGHr	148	$(0.105, 0.938)$.	-88.477
ZOSGD	45	$(0.272, 1.173)$.	-121.14
ZOSLGHd	471	$(-0.447, 1.991)$.	-137.016

Image Adversarial Attacks

Let $\mathcal{C}$ denote a black-box image classifier. Given an image $\boldsymbol{a}$, the task of image adversarial attack is to add a minor perturbation $\boldsymbol{x}$ to the pixels of $\boldsymbol{a}$, so that $\boldsymbol{a} + \boldsymbol{x}$ is classified by $\mathcal{C}$ to a pre-selected category.

Loss Function

For this task, similar to the popular loss function designed in [2], we set the loss function as

$$L(\boldsymbol{x}) := \max(\max_{i \neq \mathcal{T}} \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_i - \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_{\mathcal{T}},\ \kappa) + \lambda \|\boldsymbol{x}\|,$$

where

- $\mathcal{T}$ is the preselected category;
- $\mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_i$ denotes the predicted probability by $\mathcal{C}$ for $\boldsymbol{a}+\boldsymbol{x}$ to be in category i;
- $\kappa \geq 0$ and $\lambda \geq 0$ are hyper-parameters.

Explaining the Loss

$$Loss : L(\boldsymbol{x}) = \max(\max_{i \neq \mathcal{T}} \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_i - \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_{\mathcal{T}},\ \kappa) + \lambda \|\boldsymbol{x}\|.$$

- The smaller is $\max_{i \neq \mathcal{T}} \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_i - \mathcal{C}(\boldsymbol{a}+\boldsymbol{x})_{\mathcal{T}}$, the more certain for $\mathcal{C}$ to classify $\boldsymbol{a}+\boldsymbol{x}$ as the pre-selected category $\mathcal{T}$.
- When minimizing the loss, a close to zero κ (e.g., -0.001) prevents excess efforts on increasing the certainty level for $\mathcal{C}$ to classify $\boldsymbol{a}+\boldsymbol{x}$ as category $\mathcal{T}$.

Experiment Details

- Dateset $\mathcal{D}$: MNIST hand-written digits or CIFAR-10 images.
- For each of the 100 randomly selected images from $\mathcal{D}$, we apply EPGS, and other compaired algorithms, to solve

$$\max_{\boldsymbol{x}} \{-L(\boldsymbol{x})\}.$$

- If $\boldsymbol{a} + \boldsymbol{\mu}^*$ is classified by $\mathcal{C}$ as the preselected category $\mathcal{T}$, where $\boldsymbol{\mu}^*$ denotes the best solution (i.e., perturbation) found by the tested algorithm, we say the attack is successful.
- Hyper-parameters are selected by trials.

MNIST-Attack Results

Table: Targeted Adversarial Attack on 100 MNIST images (per-image). The success rate (SR) is the portion of successful attacks out of the 100 attacks. $\bar{R}^2$ measures the similarity between the original image $\boldsymbol{a}$ and the perturbed one $\boldsymbol{a} + \boldsymbol{\mu}^*$, $\overline{\|\boldsymbol{\mu}^*\|}$ denotes the norm of the perturbation. $\bar{T}$ denotes the number of steps taken to find the best solution.

Algorithm	SR	$\bar{R}^2$	$\overline{\|\boldsymbol{\mu}^*\|}$	$\bar{T}$
CMA-ES	100%	89%(4%)	2.81(0.61)	1489(12)
EPGS	100%	87%(5%)	3.01(0.60)	397(101)
ZOSGD	100%	85%(5%)	3.14(0.61)	1427(242)
ZOSLGHd	100%	74%(9%)	4.21(0.71)	1490(24)
ZOSLGHr	100%	65%(13%)	4.86(0.81)	476(658)
ZOAdaMM	100%	29%(27%)	6.88(1.15)	45(15)
STD-Htp	97%	−4%(37%)	8.25(1.09)	530(264)

Image Adversarial Attack: CIFAR-10

Table: Targeted Adversarial Attack on 100 CIFAR-10 images (per-image).

Algorithm	SR	$\bar{R}^2$	$\overline{\|\boldsymbol{\mu}^*\|}$	$\bar{T}$
ZOSLGHd	98%	99%(1%)	1.72(0.32)	1290(411)
ZOSLGHr	98%	98%(3%)	2.66(0.68)	456(345)
EPGS	98%	98%(2%)	3.05(0.57)	748(248)
CMA-ES	99%	75%(25%)	10.06(2.35)	158(399)
ZOAdaMM	100%	58%(39%)	13.13(2.71)	58(31)
ZOSGD	62%	99%(1%)	1.19(0.19)	764(349)
STD-Htp	52%	87%(13%)	7.54(1.57)	566(396)

Summary on Experiment Results

Compared to algorithms using smoothing (which excludes CMA-ES):

- EPGS ranked first in the experiment of Ackley, Rosenbrock, and MNIST.
- EPGS ranked the 3rd but has a performance close to the winner in the CIFAR-10 task.

While EPGS under performs CMA-ES in the first three tasks, it beats CMA-ES in the CIFAR-10 task. Moreover, the theoretical convergence guarantee for EPGS is more developed than that of CMA-ES.

The Case of Multiple Global Maxima

Although our convergence analysis assumes that f has a unique global maximum, this condition is not necessary for GS-PowerOpt to work, at least for the following toy example, in which we apply EPGS to solve

$$\max_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) := -\log(\|\boldsymbol{x}-\boldsymbol{m}_1\|^2 + 10^{-5}) - \log(\|\boldsymbol{x}-\boldsymbol{m}_2\|^2 + 10^{-5}),$$

where $\boldsymbol{m}_1 = [-.5, -.5]$ and $\boldsymbol{m}_2 = [.5, .5]$ are the two global maxima. Our experiments show that EPGS is able to locate one of the two maxima.

Guidance on Selecting the Hyperparameters N and σ

- We recommend to start from a moderate N and incrementally increase its value during tuning. Although the proper starting value of N may vary for different problems, based on our experience, 5 for PGS and 0.1 for EPGS are good choices.
- Our experiments show that a σ-value of 10% of the search radius for $\boldsymbol{x}$ is a good starting value for tuning.

Conclusion and Future Work

- The convergence analysis and numerical results show that the easily implemented optimization method of GS-PowerOpt stands out among its peers that also apply smoothing techniques.
- Future works would include the convergence analysis for the maximization objective f with more than one global maxima.

[1] A. Blake and A. Zisserman. Visual Reconstruction. MIT press, 1987.

[2] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57, 2017.

[3] J. Chen, Z. Guo, H. Li, and C. P. Chen. Regularizing scale-adaptive central moment sharpness for neural networks. IEEE Transactions on Neural Networks and Learning Systems, 35(5), 2024.

[4] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, volume 38 of Proceedings of Machine Learning Research, pages 192–204. PMLR, 2015.

[5] K. Dvijotham, M. Fazel, and E. Todorov. Universal convexification via risk-aversion. In Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, 2014.

[6] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation, 9(2):159–195, 2001.

[7] E. Hazan, K. Y. Levy, and S. Shalev-Shwartz. On graduated optimization for stochastic non-convex problems. In Proceedings of The 33rd International Conference on Machine Learning, volume 48, pages 1833–1841, 2016.

[8] H. Iwakiri, Y. Wang, S. Ito, and A. Takeda. Single loop gaussian homotopy method for non-convex optimization. In Advances in Neural Information Processing Systems, volume 35, pages 7065–7076, 2022.

[9] Y. Lei, T. Hu, G. Li, and K. Tang. Stochastic gradient descent for nonconvex learning without bounded gradient assumptions. IEEE transactions on neural networks and learning systems, 31(10):4394–4400, 2019.

[10] X. Lin, Z. Yang, X. Zhang, and Q. Zhang. Continuation path learning for homotopy optimization. In Proceedings of the 40th International Conference on Machine Learning, volume 202, pages 21288–21311. PMLR, 2023.

[11] P. Mertikopoulos, N. Hallak, A. Kavis, and V. Cevher. On the almost sure convergence of stochastic gradient descent in non-convex problems. Advances in Neural Information Processing Systems, 33:1117–1128, 2020.

[12] L. J. V. Miranda. PySwarms, a research-toolkit for particle swarm optimization in python. Journal of Open Source Software, 3, 2018.

[13] H. Mobahi and J. W. Fisher. A theoretical analysis of optimization by gaussian continuation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 29, pages 1205–1211, 2015.

[14] V. Roulet, M. Fazel, S. Srinivasa, and Z. Harchaoui. On the convergence of the iterative linear exponential quadratic gaussian algorithm to

stationary points. In 2020 American Control Conference (ACC), pages 132–137. IEEE, 2020.

[15] P. J. Van Laarhoven, E. H. Aarts, P. J. van Laarhoven, and E. H. Aarts. Simulated annealing. Springer, 1987.

Thank you!

- Our codes are available at http://github.com/chen-research/GS-PowerTransform.
- Welcome to visit my webpage at https://orcid.org/0000-0002-7238-7254, https://www.linkedin.com/in/chen-xu-quantitative/.
- Email: xuchen@smbu.edu.cn, chen_xu_research@outlook.com