Project Proposal: Black Box Optimization on a Function with Known Properties

Yechao She
Department of Computer Science, City University of Hong Kong

I. Literature Review

A. Approximating a black-box function with known properties

1) Approximating Monotonic Functions:

- Monotonicity by construction. Several studies have explored to construct special network structures to ensure fully or partial monotonicity, such as the deep lattice network proposed by You et al. [1], monotone dense network proposed by Runje et al. [2], and expressive monotone network proposed by Nolte et al. [3], etc. Among these works, Runje et al.'s work claims to be the state of the art and has published its code, which is well-written and easy-to-extend.
- Monotonicity via regularization. In this line of work, penalties from negative gradients on the data are used to encourage the monotoncity. Gradient-based loss is used by Gupta et al. [4] and Monteiro et al. [5]. As monotonicity constraint may fail on part of the data set, Liu et al. [6] proposed a verification method. For the calculation of gradients, there are three candidate methods: zeroth/first/second-order optimization [7], [8], [9].

2) Approximating Convex Functions:

- Convexity by construction. Amos et al. [10] proposed the use of convex, non-decreasing activations and weight constraints to create fully convex neural networks. For partially convex neural networks, they introduced a variant architecture. Building upon this idea, Warin et al. [11] leveraged the concept of cuts to propose a new convex neural network architecture. Balaz [12] also utilized the concept of max-affine in constructing convex neural networks, while Calafiori et al. [13] introduced log-sum-exp activation for achieving convexity. However, these works are difficult to extend for partially monotonic inputs.
- Convexity by regularization. This direction has received less attention, no related literature has been found.

II. Proposed Method and Preliminary Results

A. Approximating partially monotonic and convex function

We propose to improve the Monotone Dense unit [2] proposed by Runje et al. to ensure the monotonicity and use second-order gradient as constraint to encourage partial convexity. Let $g(x, y, u, v; w)$ denote the constructed monotone network with weights w. Hence the optimization problem can be formulated as

$$\min_{w} \sum_{i=1}^{M} ||g(x_i, y_i, u_i, v_i; w) - f(x_i, y_i, u_i, v_i)||_2^2 \tag{1}$$

$$\text{s.t. } (-1)^u \frac{\partial^2 f}{\partial y^2} > 0, \forall u, y. \tag{2}$$

B. Correlated Noise

In the case of correlated noise, the mean square error objective needs to be re-weighted sample-wise.

$$\min_{w} \sum_{t=1}^{M} (\frac{1}{\sigma_t^2} + \psi_t)||g(x_t, y_t, u_t, v_t; w) - z_t||_2^2 \tag{3}$$

$$\text{s.t. } (-1)^u \frac{\partial^2 f}{\partial y^2} > 0, \forall u, y \tag{4}$$

where $\psi_t = \sum_{i=1}^{t} \frac{0.8^{t-i}}{\sigma_i^2}$. Detailed derivation is given in Appendix A.

C. Finding a black box function's maxima in an online sense

The trial-test process can be framed as a Markovian process, wherein the environment represents the black box function $f(\cdot)$, the action space consists of (x_t, y_t, u_t, v_t), and the observation and reward are denoted as z_t. By employing deep reinforcement learning, an agent can be trained to effectively identify the maximum value z_t within a constrained number of trials.

III. Testing Plans

The testing phase will consist of three main parts to evaluate the proposed neural network's expressivity, its performance under correlated noise assumptions, and the effectiveness of the proposed DRL strategy with and without noisy observations.

A. Expressivity Test

The expressivity of the proposed neural network will be assessed through the following experiments:

- Partially monotonic functions: Evaluate the network's ability to model functions with partial monotonicity, such as $f(x, y, u, v) = x^2 + x^{1.5} + x$, $f(x, y, u, v) = x^2 + x + (-1)^u * log(y) * u$, $f(x, y, u, v) = x^2 + x + sin(u\pi/2) * y^2$.
- Partially convex functions: Assess the network's capability to approximate partially convex functions.
- Partially monotonic and convex functions: Test the network's performance on functions that exhibit both monotonic and convex properties.
- Real dataset with partial monotonicity and convexity: Utilize the Airline Customer Satisfaction Dataset[14] to evaluate the network's performance on real-world data with partial monotonicity and convexity.

B. Noisy Test

To validate the proposed method's robustness to correlated noise, the following tests will be conducted:

- Simple parameter estimation: Estimate a constant parameter x given observations $z_t = x + \epsilon_t$, where ϵ_t represents individually generated correlated noise.
- Parameter estimation with function approximation: Estimate the neural network's parameter w given observations $z_t = f(x_t, y_t, u_t, v_t) + \epsilon_t$, where $f(\cdot)$ is a tested example function from the expressivity test.

C. DRL Test

The proposed DRL agent's performance will be evaluated in the following scenarios:

- Simple monotonic function: Validate the DRL agent's learning capability for monotonic functions.
- Simple convex function: Assess the DRL agent's learning capability for convex functions.
- Partially monotonic and convex function: Once the previous tests pass, train the DRL agent on partially monotonic and convex data to evaluate its performance in a more complex setting.

Appendix

A. Objective Function for the Correlated Noise Data

The probability of observing $\{z_t\}$ can be expressed as

$$p(z_1, z_2, \cdots, z_m) = p(z_0) \prod_{t=1}^{m} p(z_t | z_1, \cdots, z_{t-1}) \tag{5}$$

$$= p(q_0) \prod_{t=1}^{m} p(q_t | \sigma_t) \tag{6}$$

$$= \prod_{t=1}^{m} \frac{1}{\sigma_t} exp(-\frac{\epsilon_t^2}{\sigma_t^2}) \tag{7}$$

$$\tag{8}$$

Its log-likelihood is therefore

$$\log(p) = -\sum_{t=1}^{m} \frac{\epsilon_t^2}{\sigma_t^2} - \frac{1}{2} \log \sigma_t^2 \tag{9}$$

Let $w^{(l)}$ denote the network parameter in the l-th iteration. To get $w^{(l+1)}$, the gradient is calculated as

$$\frac{\partial log(p)}{\partial w}|_{w^{(l)}} = (-\sum_{t=1}^{m} \frac{1}{\sigma_t^2} \frac{\partial \epsilon_t^2}{\partial w} - \sum_{t=1}^{m} \frac{1}{2\sigma_t^2} \frac{\partial \sigma_t^2}{\partial w})|_{w^{(l)}} \tag{10}$$

where the second term can be expanded into:

$$\frac{\partial \sigma_t^2}{\partial w} = 0.8 \frac{\partial \sigma_{t-1}^2}{\partial w} + 0.1 \frac{\partial \epsilon_{t-1}^2}{\partial w} \tag{11}$$

$$= 0.8^2 \frac{\partial \sigma_{t-2}^2}{\partial w} + 0.1[0.8 \frac{\partial \epsilon_{t-2}^2}{\partial w} + \frac{\partial \epsilon_{t-1}^2}{\partial w}] \tag{12}$$

$$= 0.1 \sum_{i=1}^{t-1} 0.8^{i-1} \frac{\partial \epsilon_i^2}{\partial w} \tag{13}$$

Therefore

$$\sum_{t=1}^{m} \frac{1}{\sigma_t^2} \frac{\partial \sigma_t^2}{\partial w} = 0.1 \sum_{t=1}^{m} \frac{1}{\sigma_t^2} \sum_{i=1}^{t-1} 0.8^{i-1} \frac{\partial \epsilon_i^2}{\partial w} \tag{14}$$

$$= 0.1 \sum_{t=1}^{m} \psi_t \frac{\partial \epsilon_t^2}{\partial w} \tag{15}$$

where

$$\psi_t = \sum_{i=1}^{t} \frac{0.8^{t-i}}{\sigma_i^2}. \tag{16}$$

Hence, the gradient update can be written as

$$w^{(l+1)} = w^{(l)} - \sum_{t=m}^{t} (\frac{1}{\sigma_t^2} + \frac{1}{5}\psi_t) \frac{\partial ||z_t - g_t(w)||_2^2}{\partial w}|_{w^{(l)}} \tag{17}$$

REFERENCES

[1] S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta, "Deep lattice networks and partial monotonic functions," in *Advances in neural information processing systems*, vol. 30, 2017.

[2] D. Runje and S. M. Shankaranarayana, "Constrained monotonic neural networks," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202, pp. 29338–29353, 2023.

[3] N. Nolte, O. Kitouni, and M. Williams, "Expressive monotonic neural networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[4] A. Gupta, N. Shukla, L. Marla, A. Kolbeinsson, and K. Yellepeddi, "How to incorporate monotonicity in deep networks while preserving flexibility?," 2019.

[5] J. Monteiro, M. O. Ahmed, H. Hajimirsadeghi, and G. Mori, "Monotonicity regularization: Improved penalties and novel applications to disentangled representation learning and robust classification," 2022.

[6] X. Liu, X. Han, N. Zhang, and Q. Liu, "Certified monotonic neural networks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 15427–15438, 2020.

[7] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 43–54, 2020.

[8] A. Chen, Y. Zhang, J. Jia, J. Diffenderfer, J. Liu, K. Parasyris, Y. Zhang, Z. Zhang, B. Kailkhura, and S. Liu, "Deepzero: Scaling up zeroth-order optimization for deep model training," 2023.

[9] D. Golovin, J. Karro, G. Kochanski, C. Lee, X. Song, and Q. Zhang, "Gradientless descent: High-dimensional zeroth-order optimization," in *International Conference on Learning Representations*, 2020.

[10] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 146–155, PMLR, 06–11 Aug 2017.

[11] X. Warin, "The groupmax neural network approximation of convex functions," 2023.

[12] G. Balazs, A. György, and C. Szepesvari, "Near-optimal max-affine estimators for convex regression," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, vol. 38 of *Proceedings of Machine Learning Research*, (San Diego, California, USA), pp. 56–64, PMLR, 09–12 May 2015.

[13] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, p. 827–838, Mar. 2020.

[14] Kaggle, "Airline-passenger-satisfaction dataset."