# Project Report: 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.

*2) Approximating Convex Functions:*

- Convexity by construction. Amos et al. [7] 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. [8] leveraged the concept of cuts to propose a new convex neural network architecture. Balaz [9] also utilized the concept of max-affine in constructing convex neural networks, while Calafiori et al. [10] 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.

### B. Searching for a Black Box Function's Maxima

The existing literature primarily focuses on continuous black box functions, with common techniques including zeroth-order gradient descent [11], [12], [13], Bayesian optimization [14], and deep reinforcement learning. Zeroth-order gradient descent is widely used due to its proven convergence performance. However, there is limited research on searching for maxima of discrete black box functions.

## II. Blackbox Function with Known Properties

### A. Property Examination

- Real valued function $f(x, y, u, v)$ is analytic to the complex value $v$: This property implies that $f(x, y, u, v)$ is constant with respect to $v$ according to the Cauchy–Riemann Conditions. Therefore, we will use $f(x, y, u)$ to denote the considered function in the following.

### B. Example Functions

- $f(x, y, u) = x + x^3 - sinc(u/5 + 1/2)\log(y)$, $x \in [0, 1]$, $y \in [1, e]$
- $f(x, y, u) = x + x^3 + sinc((u - 90)/5 + 1/2)\exp(-y)$, $x \in [0, 1]$, $y \in [1, e]$
- $f(x, y, u) = x^2 - (-\frac{1}{2})^u \exp(-y/2)$, $x \in [0, 1]$, $y \in [1, e]$

| Properties | Base Function | Range |
|---|---|---|
| non-decreasing concave | $\log(y)$ | $y \in R+$ |
| | $-y^2$ | $y \in R-$ |
| | $-exp(-y)$ | $y \in R$ |
| negative/positive for even/odd | $sinc(u) = \frac{\sin(u\pi)}{u\pi}$ | $u \in I$ |
| | $(-1)^u$ | $u \in I$ |
| non-decreasing | $x$, $x^3$, $x^5$ | $x \in R$ |

TABLE I: Base Functions of Properties of Interest

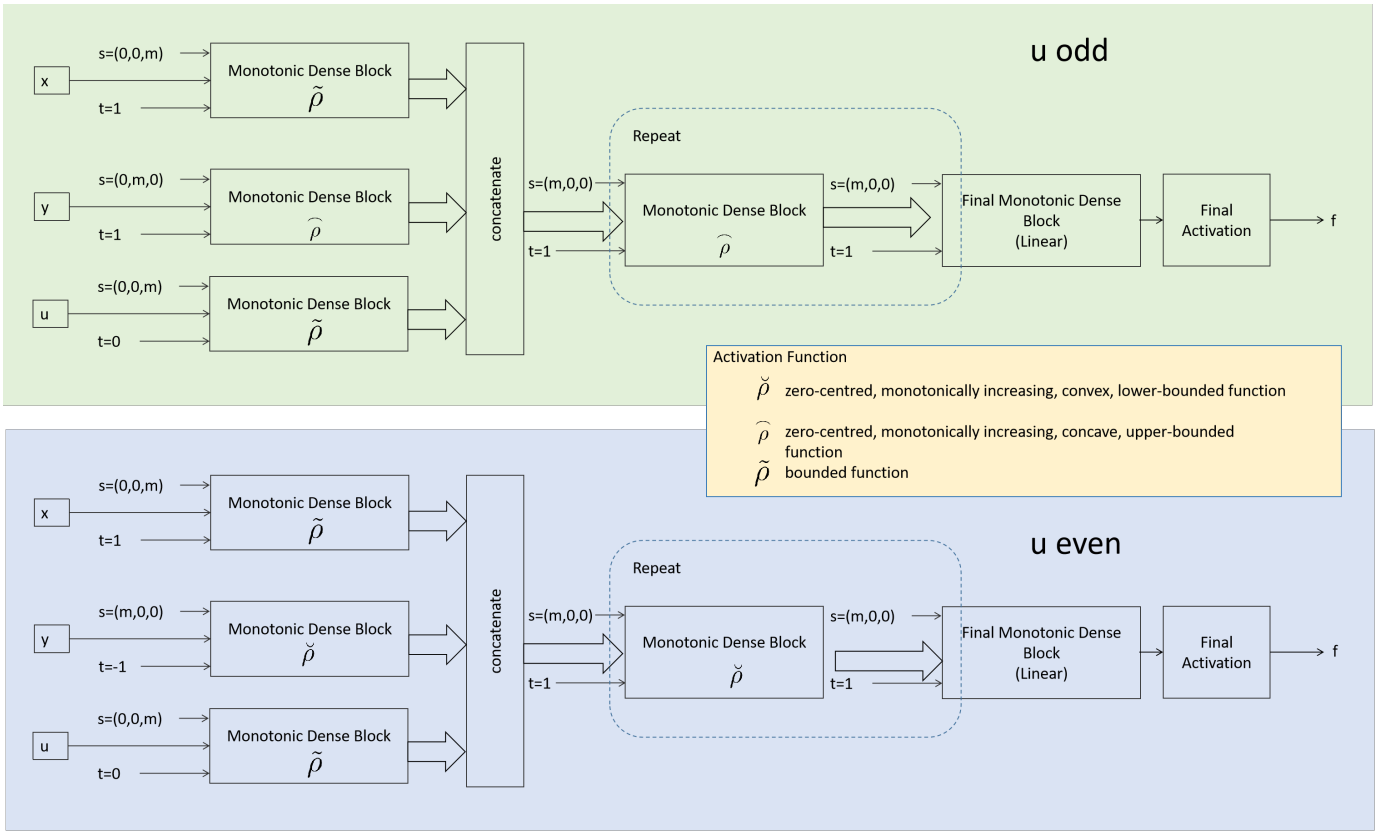



Fig. 1: The Proposed Neural Network Architecture with Convexity and Mononcity Guarantee

## III. Neural Networks with Convexity and Monotonicity Guarantee

We propose a neural network architecture, as shown in Fig. 1, to approximate the relationship between the input variables $x$, $y$, $u$, and the blackbox function $f$. We take into account that the convexity and monotonicity of $y$ change depending on whether $u$ is even or odd. Therefore, we construct two separate sub-networks to handle these two cases. To ensure the desired monotonicity and convexity properties of the trained neural network, we adopt the Monotone Dense unit proposed by Runje et al. [2]. In this section, we briefly explain the relevant notations used in [2], prove that our constructed neural network satisfies the required convexity and monotonicity constraints, and present experimental results demonstrating the expressivity of the network.

### A. Monotone Dense Units

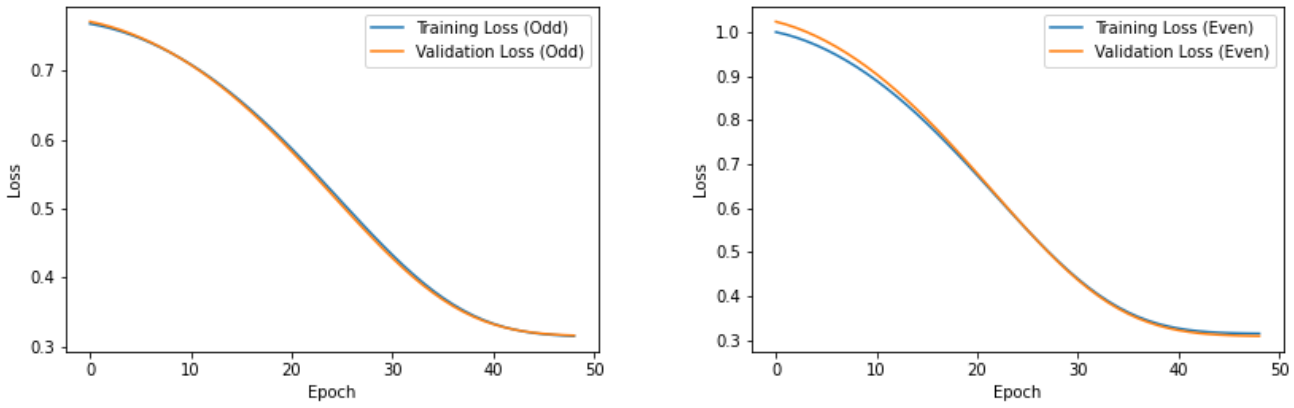
**Activation functions used by monotonic dense units**:
We use the following activation functions for the monotonic dense units:

- $\breve{\rho}$: Zero-centered, monotonically increasing, convex, lower-bounded functions such as ReLU, ELU, SELU, etc.
- $\widehat{\rho}$: Zero-centered, monotonically increasing, concave, upper-bounded functions constructed from $\breve{\rho}$. For example, $\widehat{\rho}\ (x) = -\ \breve{\rho}\ (-x)$.
- $\tilde{\rho}$: Zero-centered, bounded function constructed from $\breve{\rho}$ and $\widehat{\rho}$. For example:

$$\widehat{\rho}\ (x) = \begin{cases} \breve{\rho}\ (x+1) - \breve{\rho}\ (1), & \text{if } x < 0 \\ \widehat{\rho}\ (x-1) + \breve{\rho}\ (1), & \text{otherwise} \end{cases}$$

**Arguments of monotone dense units**:
The monotone dense units take the following arguments:

- $m$: Output dimension.
- $t \in \{0, 1, -1\}$: Monotone forcing argument. $t = 1$ forces positive weights, leading to monotone increase, while $t = -1$ forces negative weights, leading to monotone decrease.
- $s = [\breve{s}, \widehat{s}, \tilde{s}]$, with $|s|_1 = m$: Convexity preserve argument. $s = [m, 0, 0]$ indicates convexity, and $s = [0, m, 0]$ indicates concavity.




(a) Loss for Odd Sub-net

(b) Loss for Even Sub-net

Fig. 2: Training Loss of $f = x + x^3 - \text{sinc}\left(\frac{u}{5} + \frac{1}{2}\right) \cdot \log(y)$

### B. Convexity and Monotonicity Guarantee

In this section, we provide a brief proof sketch that the constructed neural network satisfies the required convexity and monotonicity constraints, i.e., non-decreasing for $x$, non-decreasing and concave for $y$ when $u$ is odd. The case for even $u$ is omitted here.

We denote the neural network as follows:

$$g(x, y, u) = h\left(\text{concat}[\tilde{\rho}(|w_x|_1 x + b_x), \widehat{\rho}\ (|w_y|_1 y + b_y), \widecheck{\rho}\ (w_u u + b_z)]\right),$$

where $w$ and $b$ are weight and bias parameters to be trained, $|\cdot|_1$ and $|\cdot|_{-1}$ denote the positive and negative forcing operations, respectively. The outer function $h(\cdot)$ represents the composition of concave activation functions, denoted as $\widehat{\rho}_1 \circ \cdots \circ \widehat{\rho}_1$, with positive weights.

The proof sketch can be break down into following concise statements.

- The composition function $h$ is concave and non-decreasing.
- $\tilde{\rho}(|w_x|_1 x + b_x)$ is non-decreasing with respect to $x$,
- $\widehat{\rho}\ (|w_y|_1 y + b_y)$ is non-decreasing and concave with respect to $y$.

Using the composition rule, it is easy to see that $g$ is non-decreasing with respect to $x$, and non-decreasing and concave with respect to $y$.

### C. Experimental Results

In this section, we present the experimental results shown in Fig. 2. We used a neural network with 4 layers and 128 neurons per layer to approximate the test function $f = x + x^3 - \text{sinc}\left(\frac{u}{5} + \frac{1}{2}\right) \cdot \log(y)$. The training/validation loss demonstrates convergence as the number of epochs increases, highlighting the expressive power of the constructed network.

## IV. Search for Black Box Function's Maxima

### A. Problem Formulation

In this section, we aim to solve the following the problem in a sequential way

$$\max_{x,y,u} f(x, y, u) \tag{1}$$

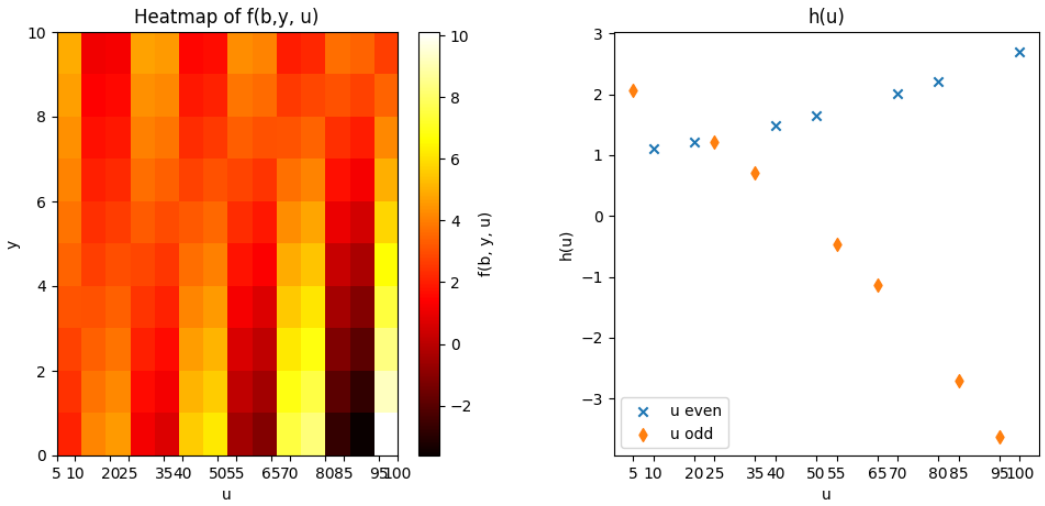
$$s.t. x \in [a, b], y \in [c, d], u \in \mathbb{U}, \tag{2}$$

where $\mathbb{U}$ denotes the set of positive integers that are divisable by 5 but not 3. Due the prior information of monontone, it is equivalent to solve the following problem:

$$\max_{u} h(u) \tag{3}$$

$$s.t. u \in \mathbb{U}, \tag{4}$$

where $h(u) = f(b, y^*(u), u)$ and $y^*(u) = \begin{cases} c, \text{if } u\%2 = 0 \\ d, \text{else} \end{cases}$.




(a) Heatmap of $f(b, y, u)$ (b) $h(u)$

Fig. 3: Example $f(b, y, u) = \begin{cases} 1.01^u \times y^{1.2}, u\%2 = 0 \\ 6 - 1.01^u \times y^{1.2}, else \end{cases}, y \in [1, 3]$

### B. Search for the (Local) Maxima

We implement a gradient aescent algorithm to search for $h(u)$'s maxima.

*1) Approximate Gradient for Discrete Black Box Function:* To leverage the available information effectively, we present a heatmap (Fig. 4a) illustrating $f(b, y, u)$ and a plot (Fig. 4b) showcasing $h(u)$ for a representative function.

**Observation 1**: In the examined function, $h(u)$ exhibits numerous local maxima primarily due to the alternating even and odd changes in monotonicity with respect to $y$. Directly searching within $h(u)$ can easily lead to becoming trapped in local maxima. Therefore, it is advantageous to explore a smoothed version of $h(u)$.

**Observation 2**: In the considered function, $h(u)$ demonstrates smoother behavior when considering even or odd values of $u$ individually.

*2) The Proposed Heuristic Search Algorithm:* Let $\mathbb{U}_0 = \{5\}$,$\mathbb{U}_i = \{30i - 20, 30i - 10, 30i + 5, 30i + 15\}$, therefore $\mathbb{U} = \cup_{i \in \mathbb{I}} \mathbb{U}_i$

We denote $s(i) = \max_{u \in \mathbb{U}_i} h(u)$ as a smoothed version of $h(u)$. Therefore, the problem can be converted as

$$\max_{i \in \mathbb{I}} s(i) \tag{5}$$

We employ a recursive approach to iteratively converge from an initial point $i^{(0)}$ towards a local maximum point $i^*$ by following the iterative equation:

$$i^{(t+1)} = i^{(t)} + 2^{\eta^{(t)}} \text{sign}(g^{(t)}), \tag{6}$$

where $g^{(t)} = \left(s(i^{(t)} + 1) - s(i^{(t)-1})\right)/2$ denotes the approximated gradient at $s(i^{(t)})$, $\eta^{(t)}$ is determined by the following backtracking line search algorithm (Alg. 1). The iteration stops either when **condition 1** $s(i^{(t)}) > s(i^{(t)} + 1)$ and $s(i^{(t)}) > s(i^{(t)} - 1)$, or **condition 2** the number of trials have exceeded the limit, is met.In practical implementations, certain tricks are employed to ensure that both $i$ and $\eta$ remain within the specified range.

*3) Experiment Results:* In Fig. 4, we demonstrate the performance of the proposed searching algorithm. The algorithm efficiently locates the maximum point by utilizing the smoothed function $h(u)$. Despite incurring a fourfold increase in smoothness cost, this approach aids in escaping local maxima.

## V. Correlated Noise

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

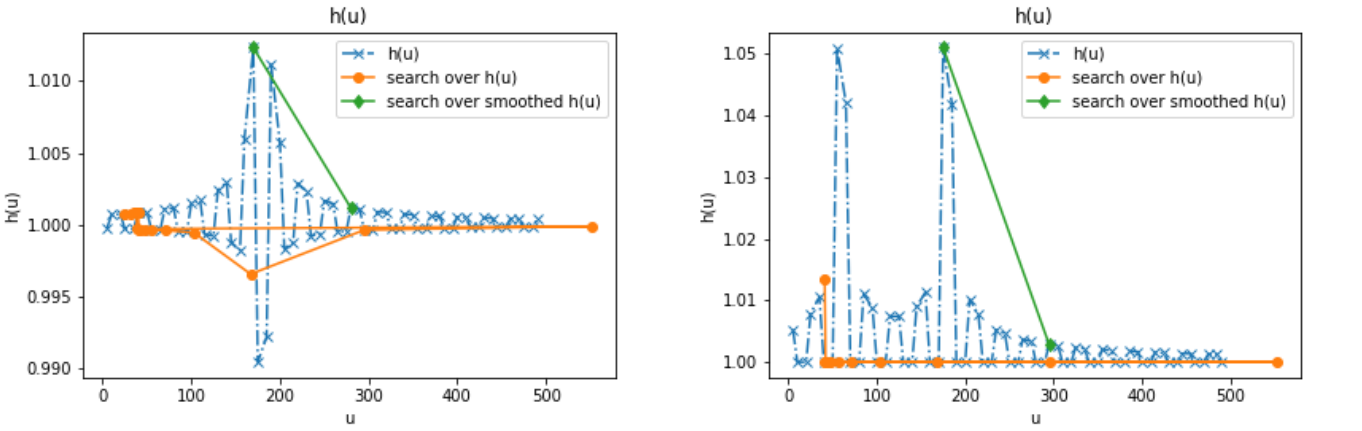
$$\min_w \sum_{t=1}^{M} \left(\frac{1}{2\sigma_t^2} + \frac{1}{5}\psi_t\right) ||f(x_t, y_t, u_t; w) - z_t||_2^2 \tag{7}$$

(8)

**Algorithm 1** Backtracking Line Search

**Input:** $i^{(t)}$, $g^{(t)}$, initial step size $\eta$
**Output:** $\eta^{(t)}$

$\eta^{(t)} \leftarrow \eta$
$i \leftarrow i^{(t)} + 2^{\eta^{(t)}} \text{sign}\left(g^{(t)}\right)$
**while** $s(i) - s(i^{(t)}) < 2^{\eta^{(t)}} |g^{(t)}|$ **do**
    $\eta^{(t)} \leftarrow \eta^{(t)} - 1$
    $i \leftarrow i^{(t)} + 2^{\eta^{(t)}} g^{(t)}$
**end while**




(a) $f = x^3 + \text{psinc}\left(u - 180 + \frac{1}{2}\right) \cdot e^{-y}$ (b) $f = x^3 - \left(\text{psinc}\left(u - 180 + \frac{1}{2}\right) + \text{psinc}\left(u - 60 + \frac{1}{2}\right)\right) \cdot \log(y)$

Fig. 4: Performance of the Proposed Gradient Ascent Algorithm

where $\psi_t = \sum_{i=1}^{t} \frac{0.8^{t-i}}{\sigma_i^2}$ . In detail, in the $l$-the weight update, we need to first calculate

$$\epsilon_t^{(l)} = f(x_t, y_t, u_t; w^{(l)}) - z_t \tag{9}$$

$$\sigma_t^{(l)} = 0.1 + 0.8(\sigma_{t-1}^{(l)})^2 + (\epsilon_{t-1}^{(l)})^2 \tag{10}$$

$$\psi_t^{(l)} = \sum_{i=1}^{t} \frac{0.8^{t-i}}{(\sigma_i^{(l)})^2} \tag{11}$$

Then update the weight according to

$$w^{(l+1)} = w^{(l)} - \sum_{t=m}^{t} \left(\frac{1}{2(\sigma_t^{(l)})^2} + \frac{1}{5}\psi_t^{(l)}\right) \frac{\partial ||z_t - f_t(w)||_2^2}{\partial w}|_{w^{(l)}} \tag{12}$$

This weight update equation takes into account the sample-wise re-weighting required for correlated noise, allowing the optimization process to adapt to the noise characteristics and improve the performance of the model. Detailed derivation is given in Appendix A.

## APPENDIX

### A. Objective Function for the Correlated Noise Data

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

$$p(z_1, z_2, \cdots, z_m) = p(z_0) \prod_{t=1}^{m} p(z_t | z_{t-1}) \tag{13}$$

$$= \prod_{t=1}^{m} \frac{1}{\sigma_t} exp(-\frac{\epsilon_t^2}{2\sigma_t^2}) \tag{14}$$

$$\tag{15}$$

Its log-likelihood is therefore

$$\log(p) = -\frac{1}{2}\sum_{t=1}^{m}\frac{\epsilon_t^2}{\sigma_t^2} - \frac{1}{2}\log\sigma_t^2 \tag{16}$$

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

$$\frac{\partial log(p)}{\partial w}|_{w^{(l)}} = (-\sum_{t=1}^{m}\frac{1}{2\sigma_t^2}\frac{\partial\epsilon_t^2}{\partial w} - \sum_{t=1}^{m}\frac{1}{2\sigma_t^2}\frac{\partial\sigma_t^2}{\partial w})|_{w^{(l)}} \tag{17}$$

where the second term can be expanded into:

$$\frac{\partial\sigma_t^2}{\partial w} = 0.8\frac{\partial\sigma_{t-1}^2}{\partial w} + 0.1\frac{\partial\epsilon_{t-1}^2}{\partial w} \tag{18}$$

$$= 0.8^2\frac{\partial\sigma_{t-2}^2}{\partial w} + 0.1[0.8\frac{\partial\epsilon_{t-2}^2}{\partial w} + \frac{\partial\epsilon_{t-1}^2}{\partial w}] \tag{19}$$

$$= 0.1\sum_{i=1}^{t-1}0.8^{i-1}\frac{\partial\epsilon_i^2}{\partial w} \tag{20}$$

Therefore

$$\sum_{t=1}^{m}\frac{1}{\sigma_t^2}\frac{\partial\sigma_t^2}{\partial w} = 0.1\sum_{t=1}^{m}\frac{1}{\sigma_t^2}\sum_{i=1}^{t-1}0.8^{i-1}\frac{\partial\epsilon_i^2}{\partial w} \tag{21}$$

$$= 0.1\sum_{t=1}^{m}\psi_t\frac{\partial\epsilon_t^2}{\partial w} \tag{22}$$

where

$$\psi_t = \sum_{i=1}^{t}\frac{0.8^{t-i}}{\sigma_i^2}. \tag{23}$$

Hence, the gradient update can be written as

$$w^{(l+1)} = w^{(l)} - \sum_{t=m}^{t}(\frac{1}{2\sigma_t^2} + \frac{1}{5}\psi_t)\frac{\partial||z_t - f_t(w)||_2^2}{\partial w}|_{w^{(l)}} \tag{24}$$

## 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] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.