Poster: Convex Scenario Optimisation for ReLU Networks

Peruffo, Andrea; Mazo, Manuel

DOI
10.1145/3575870.3589552

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 26th ACM International Conference on Hybrid Systems, HSCC 2023

Citation (APA)
Peruffo, A., & Mazo, M. (2023). Poster: Convex Scenario Optimisation for ReLU Networks. In *Proceedings of the 26th ACM International Conference on Hybrid Systems, HSCC 2023: Computation and Control, Part of CPS-IoT Week* Article 26 Association for Computing Machinery (ACM). https://doi.org/10.1145/3575870.3589552

Poster: Convex Scenario Optimisation for ReLU Networks

Andrea Peruffo
a.peruffo@tudelft.nl
TU Delft
Delft, Netherlands

Manuel Mazo Jr.
m.mazo@tudelft.nl
TU Delft
Delft, Netherlands

ACM Reference Format:
Andrea Peruffo and Manuel Mazo Jr.. 2023. Poster: Convex Scenario Optimisation for ReLU Networks. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3575870.3589552

1 INTRODUCTION

Neural networks are one of the most common frameworks to solve regression or classification problems. Whilst their flexibility offers a valuable solution to real-world problems, they are often used as black-boxes that might yield incorrect outcomes. Hence, researchers employ various techniques to test their out-of-samples behaviours. Among others, the scenario approach provides probability guarantees of correctness for SVM and SVR [1], exploiting the convexity of support vector methods. Remarkably, [5] proved that the training of networks with ReLU activations can be rewritten as a convex problem. In this short note, we bridge the gap between these two notions: we exploit the scenario theory to obtain probability bounds on the performance of a neural network. Let us denote a sample set $(\mathbf{X}_i, \mathbf{Y}_i)_{i=1}^N$, where the $\mathbf{X}_i$'s belong to a Hilbert space $\mathcal{X}$ and the $\mathbf{Y}_i$'s represent the corresponding outputs in $\mathbb{R}^L$. Each data point is extracted independently from an unknown probability distribution.

2 CONVEX MAPPING

Recent work [3–5] allow us to formulate the training of a ReLU network as a convex optimisation problem. Let us consider a network with a single hidden layer; the neural output can be expressed as

$$N_\theta(\mathbf{X}) = W_2 \cdot \sigma(W_1 \mathbf{X}), \tag{1}$$

where σ represents ReLU activations, and W_i represent the concatenation of the network's weights and bias, as $\mathbf{X}$ are *augmented* samples, i.e. a column of 1 is concatenated at its end. For brevity, we consider single-layer networks, whose training can be formulated

$$\min_{\theta \in \Theta} \|N_\theta(\mathbf{X}) - \mathbf{Y}\|^2 + \alpha R(\theta), \tag{2}$$

where $\mathbf{Y}$ is the desired output, Θ is the parameter space, $R(\cdot)$ and $\alpha > 0$ are the regularization function and parameter, respectively. In [5], the authors prove that (2) is equivalent to the convex program

$$\min_{u,v \in C} \left\| \sum_{j=1}^{M} D_j \mathbf{X}(u_j - v_j) - \mathbf{Y} \right\|_2^2 + \alpha(|u|_{2,1} + |v|_{2,1}) \tag{3}$$

where D_j represents a matrix whose entries are 0 or 1, M represents the number of all possible combinations of 0 and 1 in a N-dimensional vector, and C is the constraint set that depends on the combinations of D_j as

$$C := \{(2D_j - I_n)\mathbf{X}u \geq 0, \quad (2D_j - I_n)\mathbf{X}v \geq 0, \forall i, j\}. \tag{4}$$

Intuitively, the matrices D_j represent the combinations of ReLU activations (active being 1 and inactive being 0). The optimal solution to (2) can be reconstructed from the optimal solution of (3), as

$$(W^*_{1,u_i}, W^*_{2,u_i}) = \left(\frac{u_i^*}{||u_i^*||_2}, ||u_i||_2 \right), \quad \text{for all } i \text{ s.t. } ||u_i^*||_2 > 0, \tag{5}$$

where u_i^* is the i-th row of u. Similarly for v_i^* we get $(W^*_{1,v_i}, W^*_{2,v_i})$. The optimal layers of the network are composed by the concatenation $W_1^* = [W_{1,u^*}, W_{1,v^*}]$ and $W_2^* = [W_{2,u^*}, W_{2,v^*}]$, featuring m neurons, where $m = \sum_{u_i \neq 0} 1 + \sum_{v_i \neq 0} 1$.

3 SCENARIO GUARANTEES FOR SVR

Let us state the support vector regression (SVR) as described in [6], where we aim at finding parameters w and b such that the function $\hat{y} = w\mathbf{X} + b$ extended with a "tube" of diameter γ contains the values $\mathbf{Y}$. Given some hyper-parameters $\alpha, \rho > 0$, we solve the program

$$\begin{aligned} \min_{\substack{w,b,\gamma \geq 0 \\ \nu_i \geq 0}} \quad & (\gamma + \alpha \|w\|^2) + \rho \sum_{i=1}^{N} \nu_i, \\ s.t. \quad & |\mathbf{Y}_i - \hat{y}_i| - \gamma \leq \nu_i, \quad i = 1, \dots, N, \end{aligned} \tag{6}$$

where the ν_i are slack variables that represent the distance of sample $\mathbf{X}_i$ from the "tube" – if $\nu_i = 0$, the sample $\mathbf{Y}_i$ is within the tube. The scenario theory bounds the probability of an erroneous prediction, i.e. the probability that a new sample (x, y) lies outside the tube. Given a user-defined confidence β and denoting s^* as the number of positive ν_i^* obtained by solving (6), it holds that

$$\mathbb{P}^N \left[\underline{\epsilon}(s^*) \leq \mathbb{P}[|y - w^* \cdot x - b^*| \leq \gamma^*] \leq \overline{\epsilon}(s^*) \right] \geq 1 - \beta, \tag{7}$$

where $\underline{\epsilon}, \overline{\epsilon}$ are obtained solving a polynomial equation [1].

4 GUARANTEES FOR RELU NETWORKS

Program (6) can be rewritten employing a network as regressor, i.e. $\hat{y}_i = W_2 \sigma(W_1 \mathbf{X})$. In light of (3), let us denote

$$N_D(\mathbf{X}_i) = \sum_{j=1}^{M} D_j \mathbf{X}_i \cdot (u_j - v_j), \tag{8}$$

hence an SVR-like optimisation program holds

$$\begin{aligned} \min_{\substack{u,v \in C, \gamma \geq 0 \\ \nu_i \geq 0}} \quad & \gamma + \alpha(|u|_{2,1} + |v|_{2,1}) + \rho \sum_{i=1}^{N} \nu_i, \\ s.t. \quad & |\mathbf{Y}_i - N_D(\mathbf{X}_i)| - \gamma \leq \nu_i, \quad i = 1, \dots, N, \end{aligned} \tag{9}$$

HSCC '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0033-0/23/05.
https://doi.org/10.1145/3575870.3589552

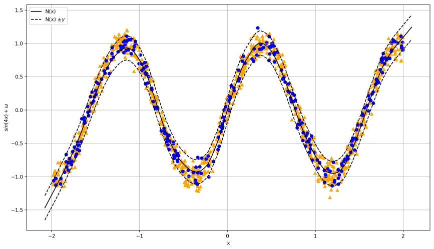

Figure 1: Regression of the sinusoidal function. Training and validation data are shown in blue and orange, respectively.

where the set C is defined as in (4). The scenario approach requires the presence of a unique constraint per sample: C is actually a set of constraints over the sample set $\mathbf{X}$. For this reason, we incorporate C with the "tube" conditions as

$$g(\mathbf{X}_i) := \max\{ \quad (I_n - 2D_j)\mathbf{X}_i u,\ (I_n - 2D_j)\mathbf{X}_i v,\ \ \forall j, \\ |\mathbf{Y}_i - N_D(\mathbf{X}_i)| - \gamma - \nu_i\} \qquad \leq 0. \tag{10}$$

Finally, the optimisation program reads

$$\begin{aligned} \min_{\substack{u,v \in C, \gamma \geq 0 \\ \nu_i \geq 0}} \quad & \gamma + \alpha(|u|_{2,1} + |v|_{2,1}) + \rho \sum_{i=1}^{N} \nu_i, \\ s.t. \quad & g(\mathbf{X}_i) \leq 0, \qquad\qquad i = 1, \dots, N, \end{aligned} \tag{11}$$

This formulation allows us to leverage the following result:

Theorem 1 (Violation of a Neural Regressor). *Given a user-defined confidence β, with $\underline{\epsilon}(\cdot)$ and $\overline{\epsilon}(\cdot)$ defined as in [2], we have*

$$\mathbb{P}^N \left[\underline{\epsilon}(s^*) \leq \mathbb{P}[(x,y) : g(x) > 0] \leq \overline{\epsilon}(s^*) \right] \geq 1 - \beta. \tag{12}$$

Notice that programs (2)-(3) are equivalent by [5], whereas proving the equivalence (or the relation between the solutions) between (6), where we use the neural output $\hat{y} = W_2\sigma(W_1\mathbf{X})$, and its convexified formulation (11) is matter of future work.

5 EXPERIMENTAL EVALUATION

Regression

We test our procedure with a non-linear regression example. We consider $N = 300$ samples $\mathbf{X} \in \mathbb{R}^{300\times1}$ generated within $[-2, 2]$ and set $\beta = 10^{-3}$. The values $\mathbf{Y}$ are obtained as $\mathbf{Y}_i = \sin(4\mathbf{X}_i) + \omega$, where $\omega \sim \mathcal{N}(0, 0.1^2)$. We approximate program (11) using $p = 100 \ll M$ different combinations D_j (cfr. (3)) , and get an optimal value of $\gamma^* \simeq 0.08$. Training and validation results are reported in Table 1, where we notice that the validation violation, computed over additional $N = 1000$ samples, is indeed within the scenario bounds.

Three-class Classification

We test the neural classification algorithm where we consider $N = 500$ samples $\mathbf{X} \in \mathbb{R}^2$, with three labels ($L = 3$), i.e. $y = \{1, 2, 3\}$ encoded as the one-hot vector $\mathbf{Y}$. The labels depend on the angles of the samples, as

$$y(\mathbf{X}_i) = j, \text{ where } \angle\mathbf{X}_i \in [(j-1)2\pi/L, j\,2\pi/L], \ \ j \in [1, L]. \tag{13}$$

We trained the neural classifier (see Fig. 2), and we approximated (11) using solely $p = 100 \ll M$ different matrices D_j (cfr. (3)). We

Test	N	s^*	$\underline{\epsilon}$	$\overline{\epsilon}$	Valid. Error	Time [s]
Regression	300	10	0	0.096	0.087	229
Classification	500	57	0	0.21	0.17	125

Table 1: Results for the numerical examples.

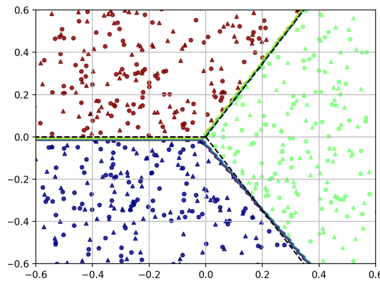

Figure 2: Classification example. True and estimated boundaries are depicted in black and colored lines, respectively. Training and test data are depicted with circles and triangles, respectively.

set $\beta = 10^{-3}$ and report training and validation results in Table 1. Again, the validation error, computed over additional $N = 1000$ samples, is within the scenario bounds.

6 CONCLUSIONS

We propose an approach to bridge the neural training as a convex optimisation program with the scenario theory for machine learning. This technique is computationally heavier than the canonical training, but in exchange can offer PAC guarantees about out-of-sample performance. Future work aims at kick-starting the optimisation program, by employing gradient descent algorithms to train the network, to yield faster results with the same PAC guarantees.

ACKNOWLEDGMENTS

This work is supported by the European Research Council through the SENTIENT project, Grant No. ERC-2017-STG #755953.

REFERENCES

[1] Marco C Campi and Simone Garatti. 2020. Scenario optimization with relaxation: a new tool for design and application to machine learning problems. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2463–2468.

[2] Marco Claudio Campi, Simone Garatti, and Federico Alessandro Ramponi. 2018. A general scenario theory for nonconvex optimization and decision making. *IEEE Trans. Automat. Control* 63, 12 (2018), 4067–4078.

[3] Tolga Ergen and Mert Pilanci. 2020. Implicit convex regularizers of cnn architectures: Convex optimization of two-and three-layer networks in polynomial time. *arXiv preprint arXiv:2006.14798* (2020).

[4] Tolga Ergen and Mert Pilanci. 2021. Global optimality beyond two layers: Training deep relu networks via convex programs. In *International Conference on Machine Learning*. PMLR, 2993–3003.

[5] Mert Pilanci and Tolga Ergen. 2020. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*. PMLR, 7695–7705.

[6] Bernhard Schölkopf, Peter Bartlett, Alex Smola, and Robert C Williamson. 1998. Shrinking the tube: a new support vector regression algorithm. *Advances in neural information processing systems* 11 (1998).