The inverse problem for neural networks

AISoLA 2023

CURE
Centro Universitario
de la Región Este

Marcelo Forets

Christian Schilling

AALBORG
UNIVERSITY

Neural network

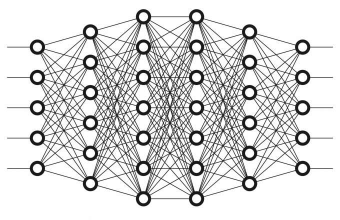
- Layer $\ell : \mathbb{R}^m \to \mathbb{R}^n$: affine map followed by activation function

$$\ell(x) = \alpha(Wx + b)$$

- Neural network $N : \mathbb{R}^m \to \mathbb{R}^n$: composition of k layers

$$N(x) = (\ell_k \circ \cdots \circ \ell_1)(x)$$

- Short-hand: $\langle n_1, \ldots, n_k \rangle$ for the number of neurons per layer

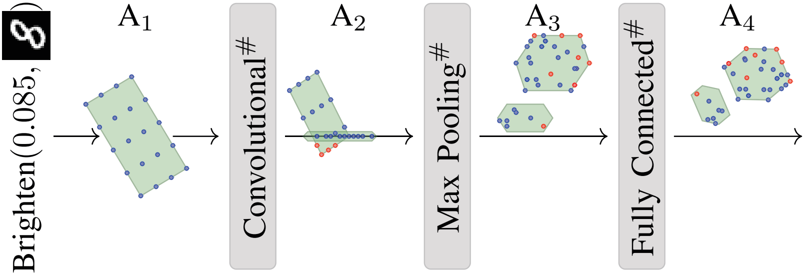


Forward image computation

- Given: function $f : \mathbb{R}^m \to \mathbb{R}^n$ and input set $\mathcal{X} \subseteq \mathbb{R}^m$
- Forward image: $f(\mathcal{X}) = \{f(x) : x \in \mathcal{X}\} \subseteq \mathbb{R}^n$
- For piecewise-affine activations, these are closed under image:
  - Union of polytopes
  - Union of polyhedra

Applications of forward image computation

- Abstract interpretation for robustness verification[1,2]

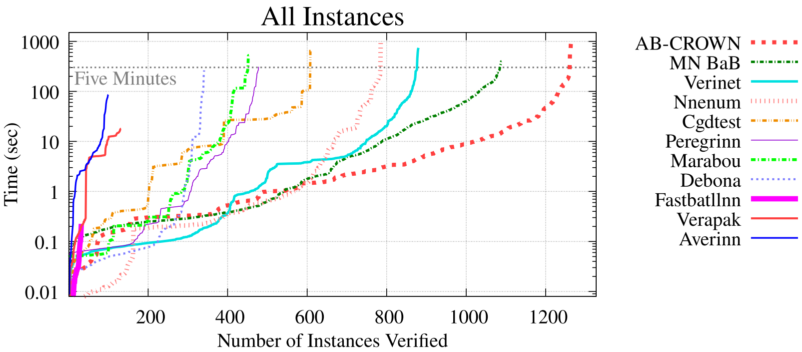


[1] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev. *SP*. 2018.

[2] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu. *Int. J. Softw. Tools Technol. Transf.* (2023).

Applications of forward image computation

- Abstract interpretation for robustness verification[1,2]

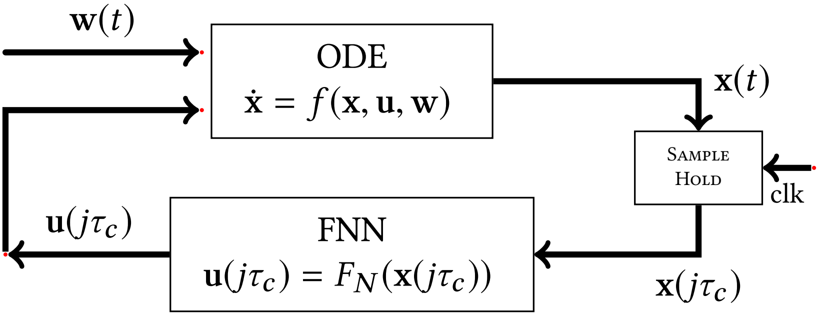


[1] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev. *SP*. 2018.

[2] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu. *Int. J. Softw. Tools Technol. Transf.* (2023).

Applications of forward image computation

- Verification of neural-network control systems[1,2]

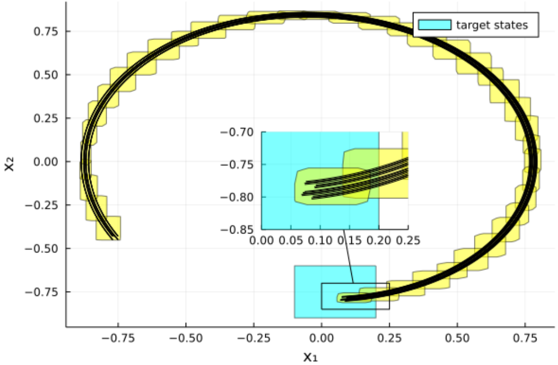


[1] S. Dutta, X. Chen, S. Jha, S. Sankaranarayanan, and A. Tiwari. *HSCC*. 2019.

[2] D. M. Lopez, M. Althoff, M. Forets, T. T. Johnson, T. Ladner, and C. Schilling. *ARCH*. 2023.

Applications of forward image computation

- Verification of neural-network control systems[1,2]



[1] S. Dutta, X. Chen, S. Jha, S. Sankaranarayanan, and A. Tiwari. *HSCC*. 2019.

[2] D. M. Lopez, M. Althoff, M. Forets, T. T. Johnson, T. Ladner, and C. Schilling. *ARCH*. 2023.

Pseudo preimage computation

- Highlighting of relevant pixels in a picture classifier[1]

[1] K. Simonyan, A. Vedaldi, and A. Zisserman. *ICLR*. 2014.

Pseudo preimage computation

- Highlighting of relevant pixels in a picture classifier[1]

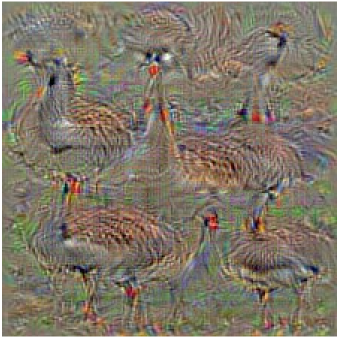
(a) Husky classified as wolf

(b) Explanation

[1] M. T. Ribeiro, S. Singh, and C. Guestrin. *KDD*. 2016.

Pseudo preimage computation

- Computing an input that produces a highly confident output[1]



goose

ostrich

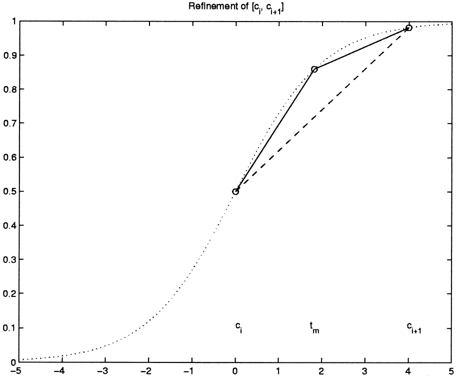
[1] K. Simonyan, A. Vedaldi, and A. Zisserman. *ICLR*. 2014.

Actual preimage computation

- Given: function $f : \mathbb{R}^m \to \mathbb{R}^n$ and output set $\mathcal{Y} \subseteq \mathbb{R}^n$
- Preimage: $f^{-1}(\mathcal{Y}) = \{x : f(x) \in \mathcal{Y}\} \subseteq \mathbb{R}^m$

Actual preimage computation

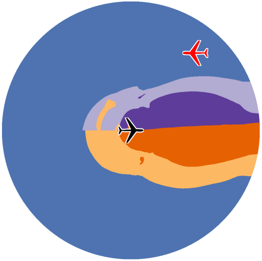
- Already studied quite early[1,2]
- Sigmoid activations, approximations, shallow neural networks



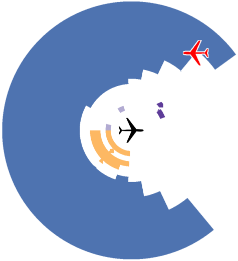
[1] S. Thrun. Tech. rep. University of Bonn, 1994.
[2] F. Maire. *Neural Networks* (1999).

Actual preimage computation

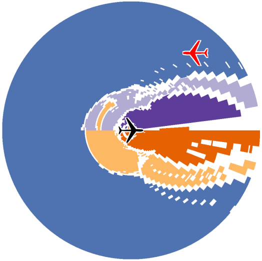
- Implicitly done to compute *symbolic representation*[1]



(a) Decision boundaries computed using $\widehat{f_{\restriction X}}$



(b) Decision boundaries computed using DeepPoly$[k = 25^2]$



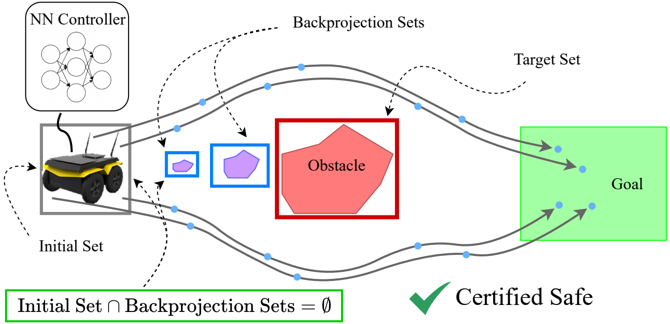
(c) Decision boundaries computed using DeepPoly$[k = 100^2]$

Legend: Clear-of-Conflict, Weak Right, Strong Right, Strong Left, Weak Left.

[1] M. Sotoudeh, Z. Tao, and A. V. Thakur. *Int. J. Softw. Tools Technol. Transf.* (2023).

Actual preimage computation

- Recent strong interest[1,2,3,4]



[1] S. Bak and H. Tran. *NFM.* 2022.
[2] N. Rober et al. *CoRR* (2022). arXiv: 2209.14076.
[3] M. Everett, R. Bunel, and S. Omidshafiei. *IEEE Control. Syst. Lett.* (2023).
[4] S. Kotha, C. Brix, Z. Kolter, K. Dvijotham, and H. Zhang. *CoRR* (2023). arXiv: 2302.01404.

Overview

Inverse affine map

Given: output polyhedron $\mathcal{Y} \subseteq \mathbb{R}^n$ written as $Cy \leq d$ and affine map $f(x) = Wx + b$ with $W \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$

$$f^{-1}(\mathcal{Y}) = \{x : C(Wx + b) \leq d\} = \{x : CWx \leq d - Cb\}$$

Inverse affine map

Given: output polyhedron $\mathcal{Y} \subseteq \mathbb{R}^n$ written as $Cy \leq d$ and affine map $f(x) = Wx + b$ with $W \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$

$$f^{-1}(\mathcal{Y}) = \{x : C(Wx + b) \leq d\} = \{x : CWx \leq d - Cb\}$$

Example

For the affine map $f(x) = \begin{pmatrix} -0.46 & 0.32 \end{pmatrix} x + 2$ and the interval $\mathcal{Y} = [2, 3]$, we get the infinite band $f^{-1}(\mathcal{Y}) = \{x \in \mathbb{R}^2 : 0 \leq \begin{pmatrix} -0.46 & 0.32 \end{pmatrix} x \leq 1\}$

Inverse piecewise-affine activation

Given: output set $\mathcal{Y} \subseteq \mathbb{R}^n$ and piecewise-affine activation α:

- Componentwise definition $\alpha(x) = [\alpha(x_1), \ldots, \alpha(x_n)]$
- Partitioning Π of $\mathbb{R}^n$ such that α_j is affine in each partition $\mathcal{P}_j$
- Affine: $\alpha(x) = Cx + d$

Inverse piecewise-affine activation

Given: output set $\mathcal{Y} \subseteq \mathbb{R}^n$ and piecewise-affine activation α:

- Componentwise definition $\alpha(x) = [\alpha(x_1), \ldots, \alpha(x_n)]$
- Partitioning Π of $\mathbb{R}^n$ such that α_j is affine in each partition $\mathcal{P}_j$
- Affine: $\alpha(x) = Cx + d$

$$\alpha^{-1}(\mathcal{Y}) = \alpha^{-1}\left(\bigcup_j \alpha_j(\mathcal{P}_j) \cap \mathcal{Y}\right) = \bigcup_j \alpha_j^{-1}(\alpha_j(\mathcal{P}_j) \cap \mathcal{Y})$$

This holds for general piecewise activation functions

Inverse piecewise-affine activation

Given: output set $\mathcal{Y} \subseteq \mathbb{R}^n$ and piecewise-affine activation α:

- Componentwise definition $\alpha(x) = [\alpha(x_1), \dots, \alpha(x_n)]$
- Partitioning Π of $\mathbb{R}^n$ such that α_j is affine in each partition $\mathcal{P}_j$
- Affine: $\alpha(x) = Cx + d$

$$\alpha^{-1}(\mathcal{Y}) = \alpha^{-1}\left(\bigcup_j \alpha_j(\mathcal{P}_j) \cap \mathcal{Y}\right) = \bigcup_j \alpha_j^{-1}(\alpha_j(\mathcal{P}_j) \cap \mathcal{Y})$$

This holds for general piecewise activation functions

If α_j is affine, $\alpha_j^{-1}(\alpha_j(\mathcal{P}_j) \cap \mathcal{Y})$ simplifies to

$$\mathcal{P}_j \cap \alpha_j^{-1}(\mathcal{Y})$$

If α_j is constant with $\alpha_j(x) = d$, it simplifies further to

$$\begin{cases} \mathcal{P}_j & d \in \mathcal{Y} \\ \emptyset & d \notin \mathcal{Y} \end{cases}$$

Inverse deep neural network

function preimage(Z, N)
    for ℓ in ℓ_k, ..., ℓ_1
        Y = preimage(Z, α_ℓ)
        X = preimage(Y, W_ℓ, b_ℓ)
        Z = X
    end
    return Z
end

Complexity

- A neural network with k layers of dimension n can (forward) map a polyhedron to $\mathcal{O}(b^{kn})$ polyhedra, where b is the number of affine pieces in the activation ($b = 2$ for ReLU)[1]
- The same holds for the preimage
- In practice the growth is often moderate for forward images
- For backward images the growth is much worse

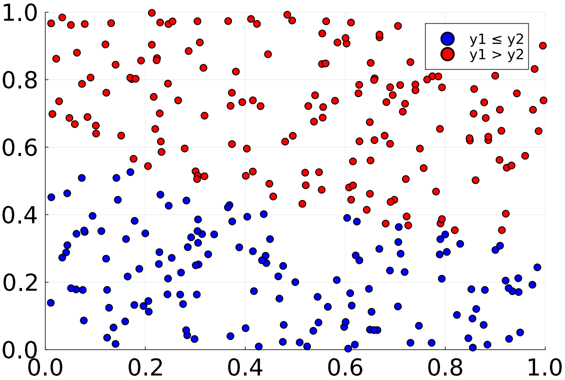
  Probably because we quickly get unbounded sets

[1] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. *NeurIPS*. 2014.

Example

- Structure $\langle 2, 2, 2\rangle$ with ReLU activations (ρ)

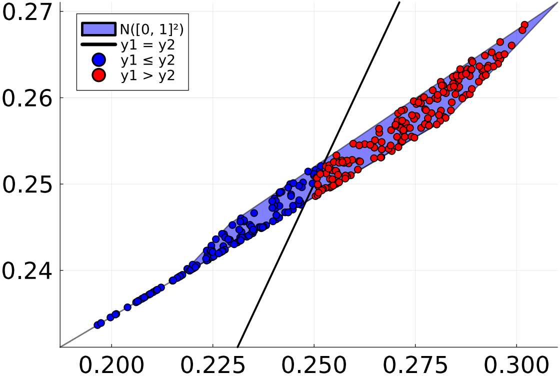
Example



300 uniform samples $x \in [0,1]^2$

Colors show the classification of $y = N(x)$

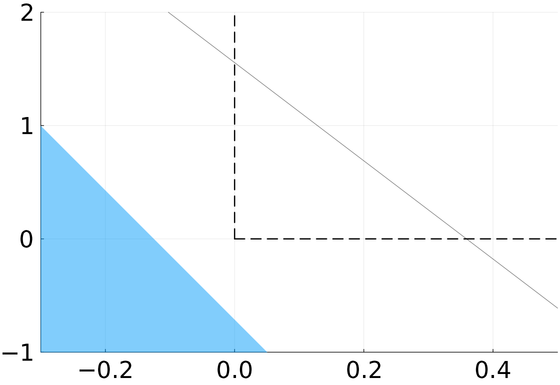
Example



Forward image $N([0,1]^2)$ with samples

Next we compute the preimage of $\mathcal{Y}_0 = \{y : y_1 \leq y_2\}$

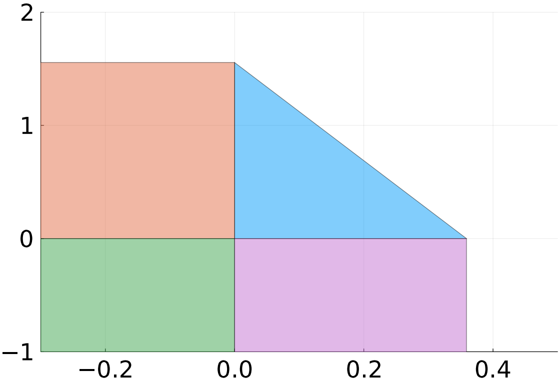
Example



Preimage $\mathcal{Y}_1 = \ell_3^{-1}(\mathcal{Y}_0)$

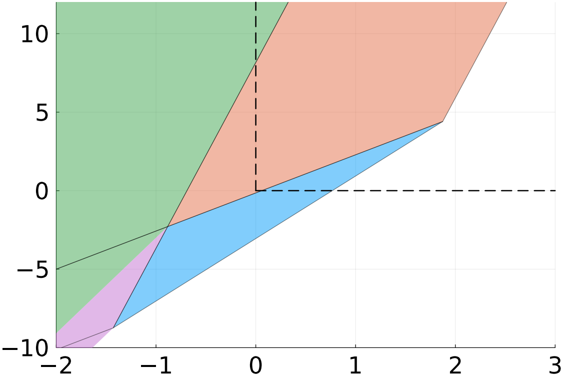
(Identity activation in last layer)

Example



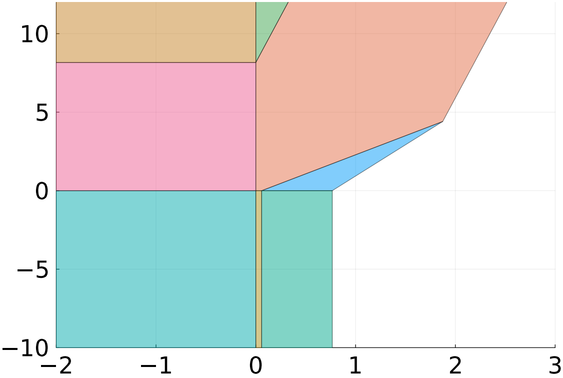
Preimage $\mathcal{Y}_2 = \rho^{-1}(\mathcal{Y}_1)$

Example



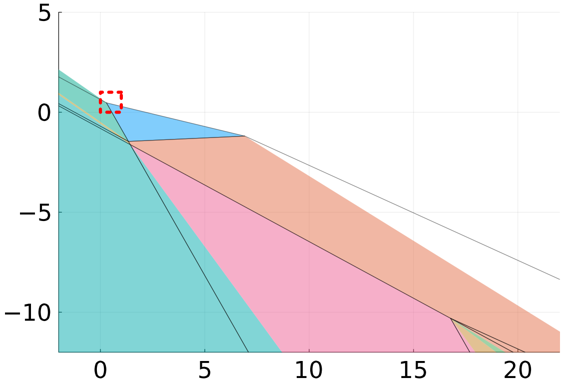
Preimage $\mathcal{Y}_3 = \ell_2^{-1}(\mathcal{Y}_1)$

Example



Preimage $\mathcal{Y}_4 = \rho^{-1}(\mathcal{Y}_3)$

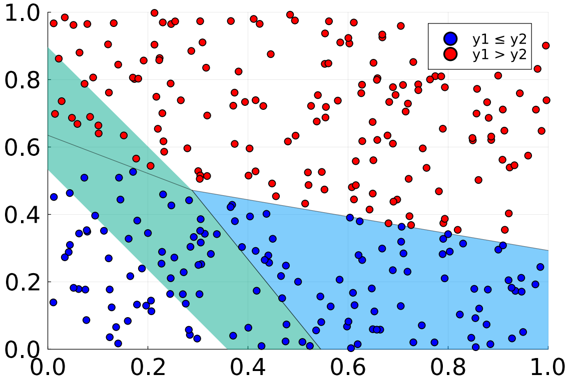
Example



Preimage $\mathcal{Y}_5 = \ell_1^{-1}(\mathcal{Y}_3) = N^{-1}(\mathcal{Y}_0)$

(Original domain $[0,1]^2$ in red)

Example



Preimage $N^{-1}(\mathcal{Y}_0)$ added

Overview

Introduction

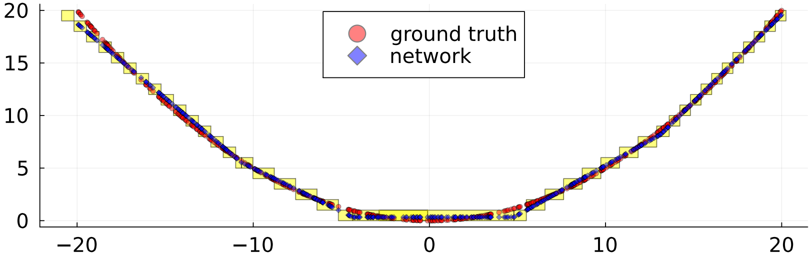
Computation

Applications

Conclusion

Interpretability

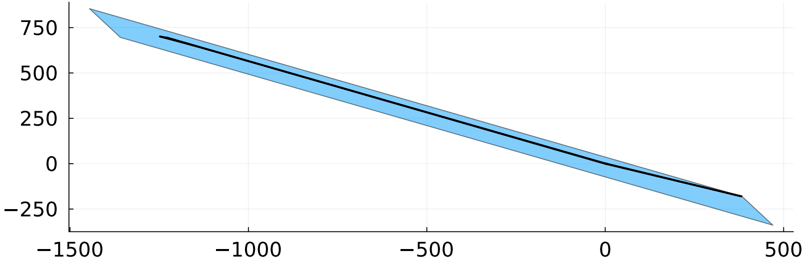
- Train N with structure $\langle 3, 3, 1 \rangle$ to approximate $f(x) = x^2/20$ from 100 samples over $[-20, 20]$
- 500 samples of f (red) and N (blue)
- Preimages (yellow) for 20 intervals over codomain $[0, 20]$



- Can prove that $N^{-1}(\{y : y \leq 0\}) = \emptyset$

Approximation schemes

- **Underapproximation**
  - Issue: subsets may become empty
  - Search problem, need heuristics
- **Overapproximation**
  - Example with interval approximation
  - Structure $\langle 2, 2, 2 \rangle$, leaky-ReLU activations
  - Computation ca. 100x faster



Forward-backward computation

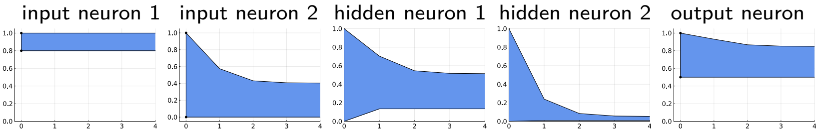
- Intervals useful for monotonic activations (e.g., sigmoids)
- Structure $\langle 2, 1 \rangle$, approximates "XOR over $[0, 1] \subseteq \mathbb{R}$"[1]
- x-axis: iterations of forward-backward computation

- Proves: $x_1 \in [0, 0.2] \wedge x_2 \in [0.8, 1] \implies N(x) \in [0.51, 0.79]$

[1] S. Thrun. Tech. rep. University of Bonn, 1994.

Forward-backward computation

- Intervals useful for monotonic activations (e.g., sigmoids)
- Structure $\langle 2, 1 \rangle$, approximates "XOR over $[0, 1] \subseteq \mathbb{R}$"[1]
- x-axis: iterations of forward-backward computation

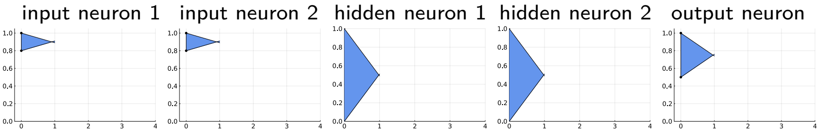


- Proves: $x_1 \in [0, 0.2] \wedge N(x) \in [0.5, 1]$
  $\implies \quad x_2 \in [0, 0.41] \wedge N(x) \in [0.5, 86]$

[1] S. Thrun. Tech. rep. University of Bonn, 1994.

Forward-backward computation

- Intervals useful for monotonic activations (e.g., sigmoids)
- Structure $\langle 2, 1\rangle$, approximates "XOR over $[0,1] \subseteq \mathbb{R}$"[1]
- x-axis: iterations of forward-backward computation



- Proves: $x_1, x_2 \notin [0, 0.2] \vee N(x) \notin [0.5, 1]$

[1] S. Thrun. Tech. rep. University of Bonn, 1994.

Overview

Introduction

Computation

Applications

Conclusion

Conclusion

- Preimage can be obtained fairly easily (at least conceptually)
- Non-injective operations (ReLU, max-pooling) conceal preimage
- Combination with forward image
- Partitioning of input space for classifiers

- Future work:
  - Sound piecewise-affine approximations of activations
  - Use preimage to optimize an objective
    (e.g., find the least robust inputs)