Reliable and Interpretable Artificial Intelligence

Lecture 6: The Zonotope convex relaxation

Martin Vechev

ETH Zurich

Fall 2020

SRILAB

http://www.sri.inf.ethz.ch

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The problem with Box

Box is simple and efficient, but the problem is that it looses too much precision, in both, the ReLU abstract transformer but also in the affine abstract transfomer.

Zonotope will be a convex relaxation whose affine transformer will not lose precision (that is, it will be exact); however, its ReLU transformer will again lose some precision. Zonotope has been shown effective in both verification and provable training (discussed later).

Incomplete method II: Zonotope

With the Zonotope relaxation, each variable (neuron) is captured in an **affine form**. We can think of Zonotope as a more extensive version of Box. Unlike Box, the variables can be related (in limited ways) through parameters.

The Zonotope Convex Relaxation

For d (concrete) neurons x_1 to x_d, the abstract neurons will look like:

$$\hat{x}_1 = a_0^1 + \sum_{i=1}^{k} a_i^1 \epsilon_i$$

.....

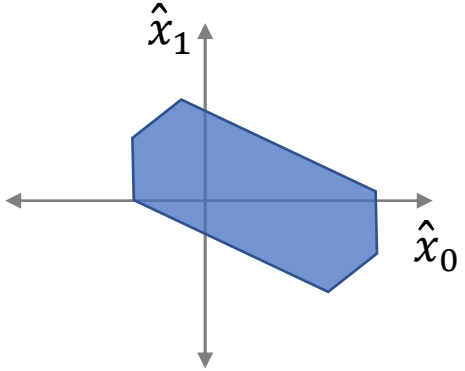
$$\hat{x}_d = a_0^d + \sum_{i=1}^{k} a_i^d \epsilon_i$$

ϵ_i : noise terms ranging [-1,1] shared between abstract neurons

a_i^j : real number that controls magnitude of noise

This represents a polytope centered around the vector $a_0 = (a_0^1, \ldots, a_0^d)$

Example of a concretization in 2D (d = 2) with k = 3:



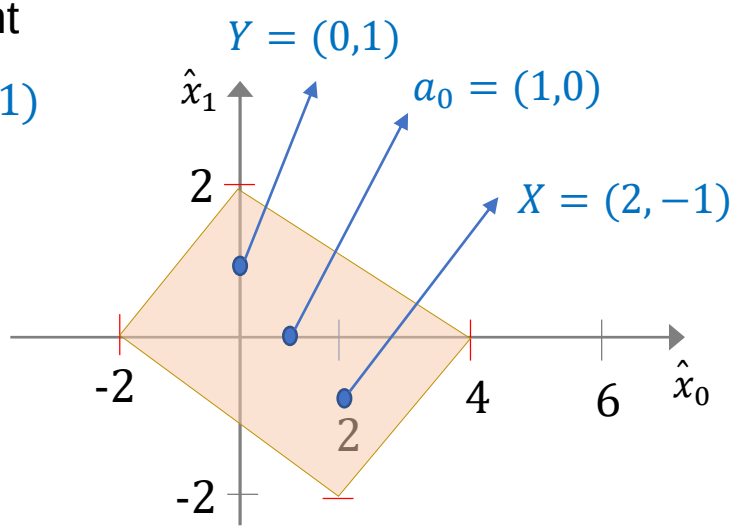
Centering

We can flip any point X in the zonotope around the center a_0 so to obtain a flipped point Y of X, where $Y = 2 * a_0 - X$ and Y is in the zonotope and X and Y are equal distance from a_0. That is, the zonotope is point-symmetric around a_0.

Example:

The zonotope **Z** is centered around $a_0 = (1,0)$. The point $X = (2,-1)$ can be flipped to obtain $Y = 2 * a_0 - X = (0,1)$

Z = $$\begin{aligned} \hat{x}_0 &= 1 - 2\epsilon_1 + \epsilon_2 \\ \hat{x}_1 &= 0 + \epsilon_1 + \epsilon_2 \end{aligned}$$ is visualized:

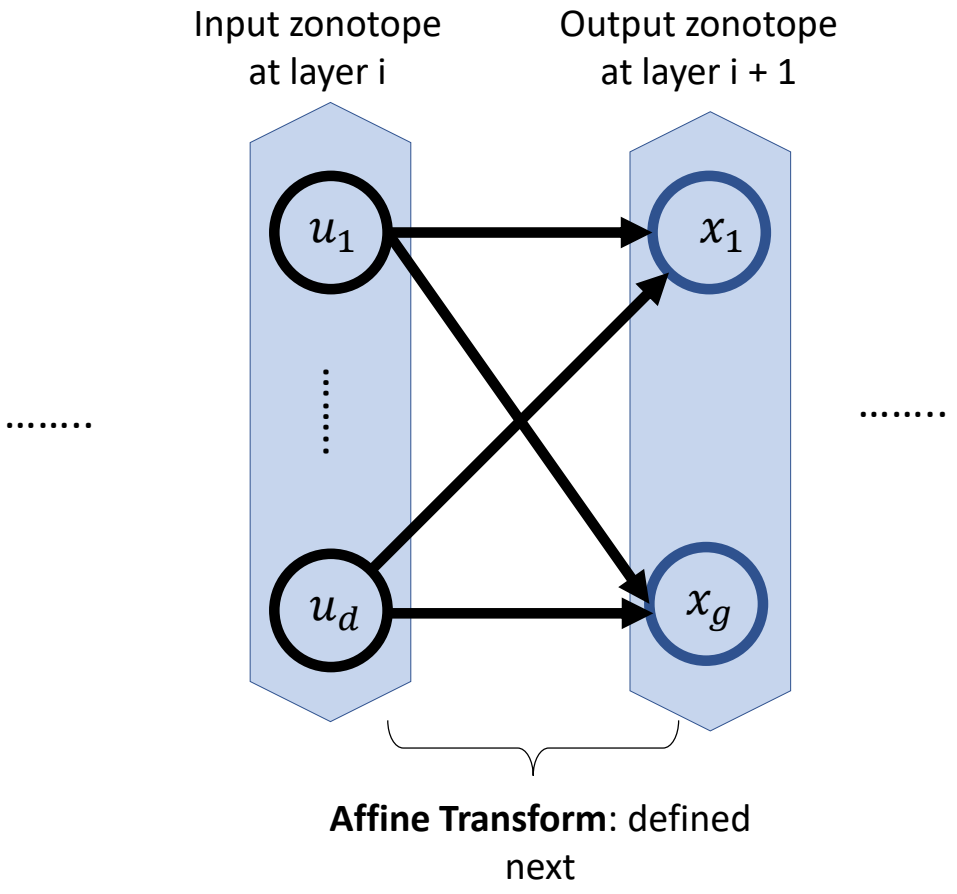


The 4 extreme values $\epsilon_1 \in \{-1,1\}$ and $\epsilon_2 \in \{-1,1\}$ determine the 4 corners.

We have formally defined what a Zonotope is. However, we need to also show how to transform a zonotope, specifically with the operations found in neural networks: affine layers and ReLU.

(side note: for provable training, we need to also know how to project to a zonotope).

Zonotope Affine Transformer



Note: the affine transformer can be computed in parallel for all output neurons $x_1 \ldots x_g$

Zonotope Affine Transformer

Multiplication of the p'th neuron by a real-valued constant C :

$$(a_0^p + \sum_{i=1}^{k} a_i^p \, \epsilon_i) * \, C = (C * a_0^p + \sum_{i=1}^{k} C * \, a_i^p \, \epsilon_i)$$

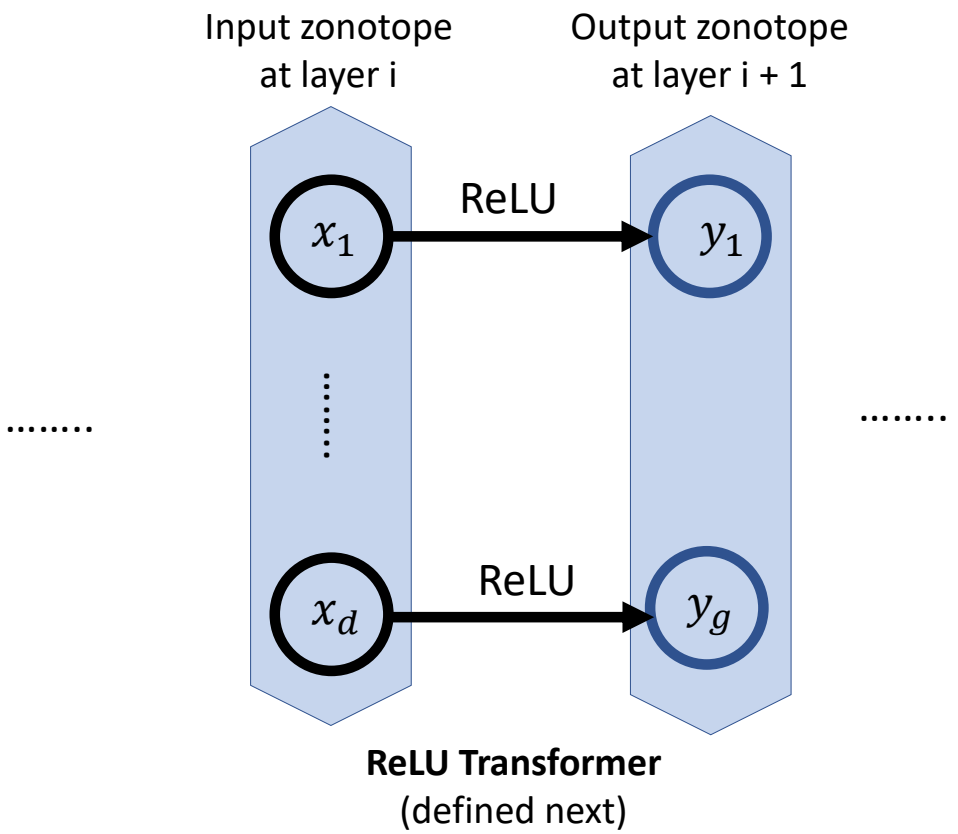
Adding two neurons p and q is done component-wise:

$$(a_0^p + \sum_{i=1}^{k} a_i^p \, \epsilon_i) + (a_0^q + \sum_{i=1}^{k} a_i^q \, \epsilon_i) = (a_0^q + a_0^p) + \sum_{i=1}^{k} (a_i^p + a_i^q) \, \epsilon_i$$

Note: abstract transformer for affine is exact

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$



ReLU Transformer
(defined next)

Note: the ReLU transformer is computed in parallel for all output neurons $y_1 \ldots y_g$

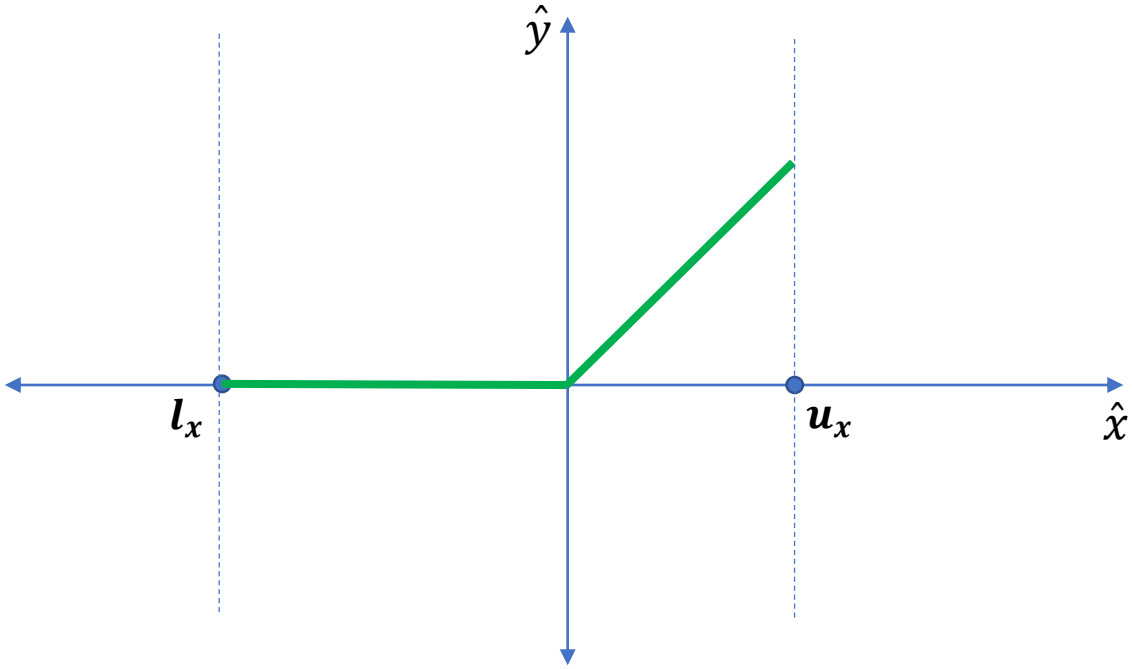
Fast and Effective Robustness Certification, NeurIPS'18
Singh et.al.

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 1: Obtain box bounds for $\hat{x}$, by exploring $\epsilon_i = -1$ and $\epsilon_i = 1$ values to compute end points. To compute the lower bound $\boldsymbol{l_x}$ use $\epsilon_i = -1$ if a_i is positive and use $\epsilon_i = 1$ if a_i is negative.

To compute the upper bound $\boldsymbol{u_x}$ use $\epsilon_i = 1$ if a_i is positive and use $\epsilon_i = -1$ if a_i is negative.



Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 2: Check if the bounds are on one side of the plane (easy cases):

→ If $\boldsymbol{u_x \leq 0}$ then $\hat{y} = 0$ (strictly negative)

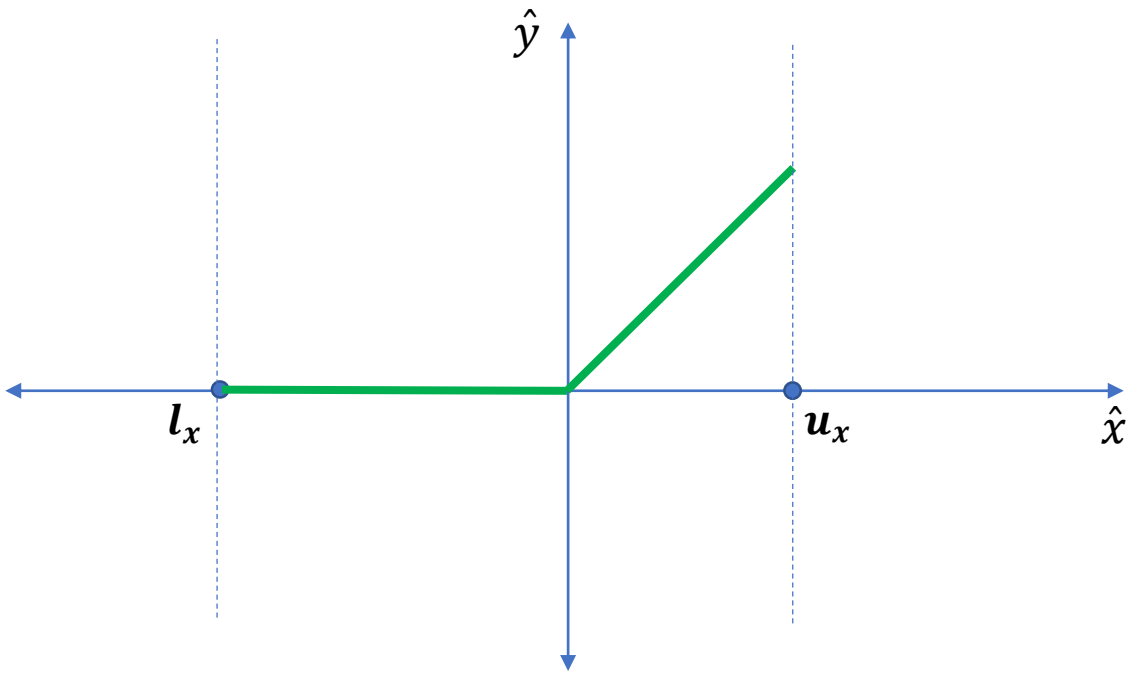
→ If $\boldsymbol{l_x > 0}$ then $\hat{y} = \hat{x}$ (strictly positive)

→ If $\boldsymbol{l_x < 0}$ and $\boldsymbol{u_x > 0}$ then we get into the ``crossing boundary'' case [discussed next].

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

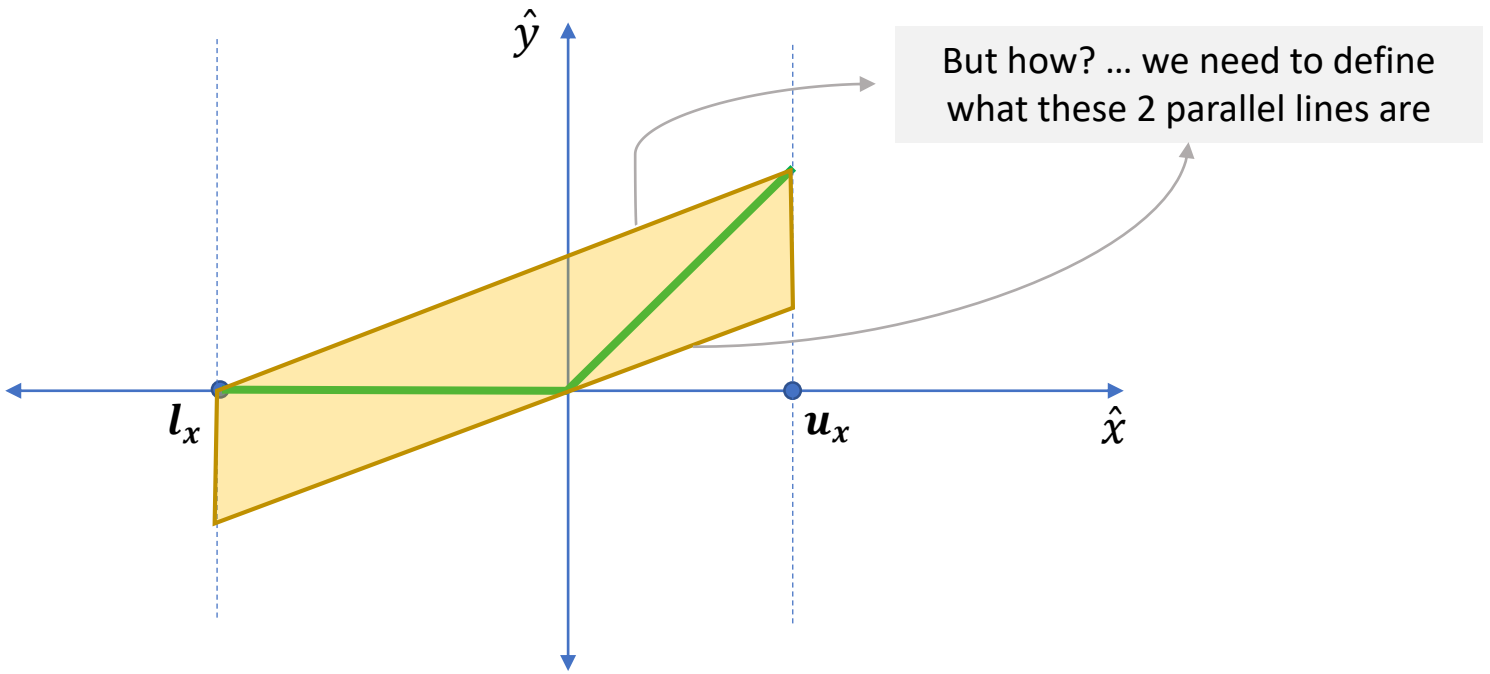
Step 2: The ``crossing-boundary'' case looks as follows (where in green is the ReLU function):



Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 2: We now compute a Zonotope (yellow) which encloses the green ReLU function.

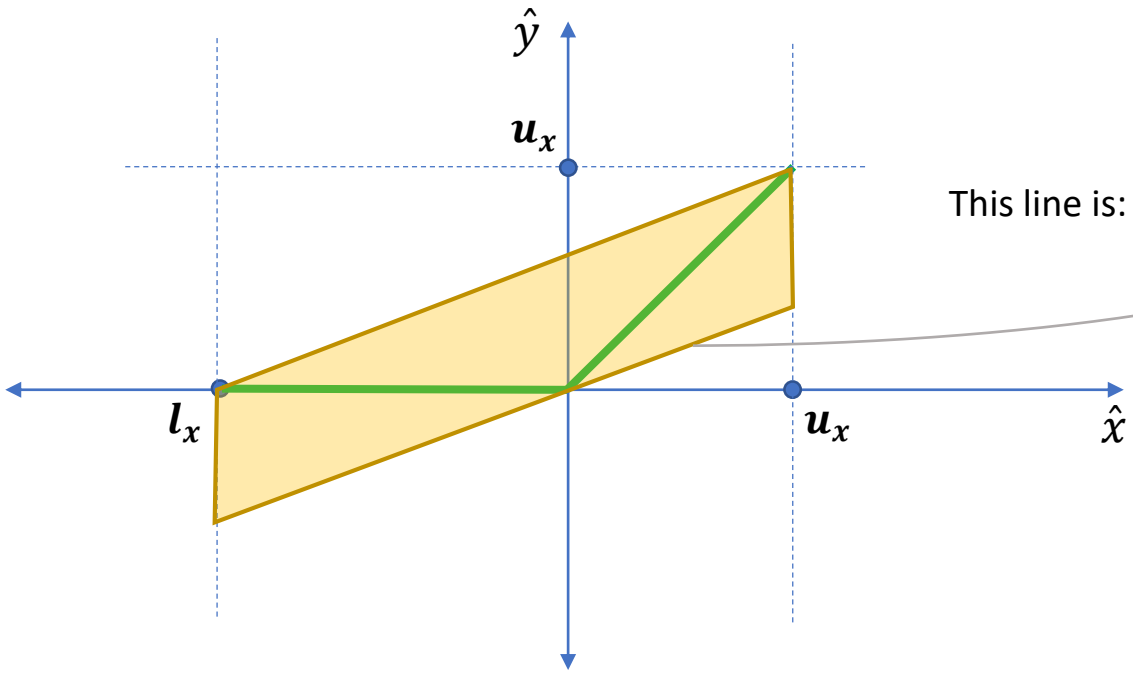


Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 2a: Define the first (the bottom) line y_1.

The slope is: $\lambda = \frac{u_x}{u_x - l_x}$



This line is: $y_1(\hat{x}) = \lambda * \hat{x}$

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

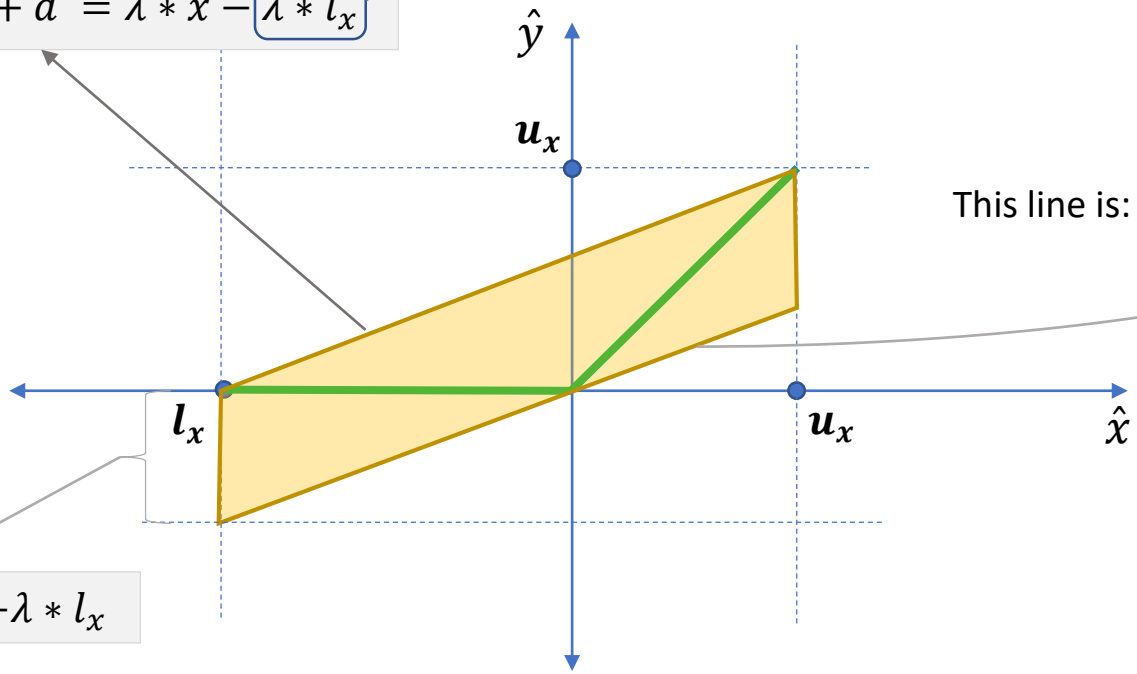
Step 2b: Define the second (upper) line y_2.

$$y_2(\hat{x}) = y_1(\hat{x}) + d = \lambda * \hat{x} - \lambda * l_x$$

Constant term

The slope is: $\lambda = \frac{u_x}{u_x - l_x}$

This line is: $y_1(\hat{x}) = \lambda * \hat{x}$



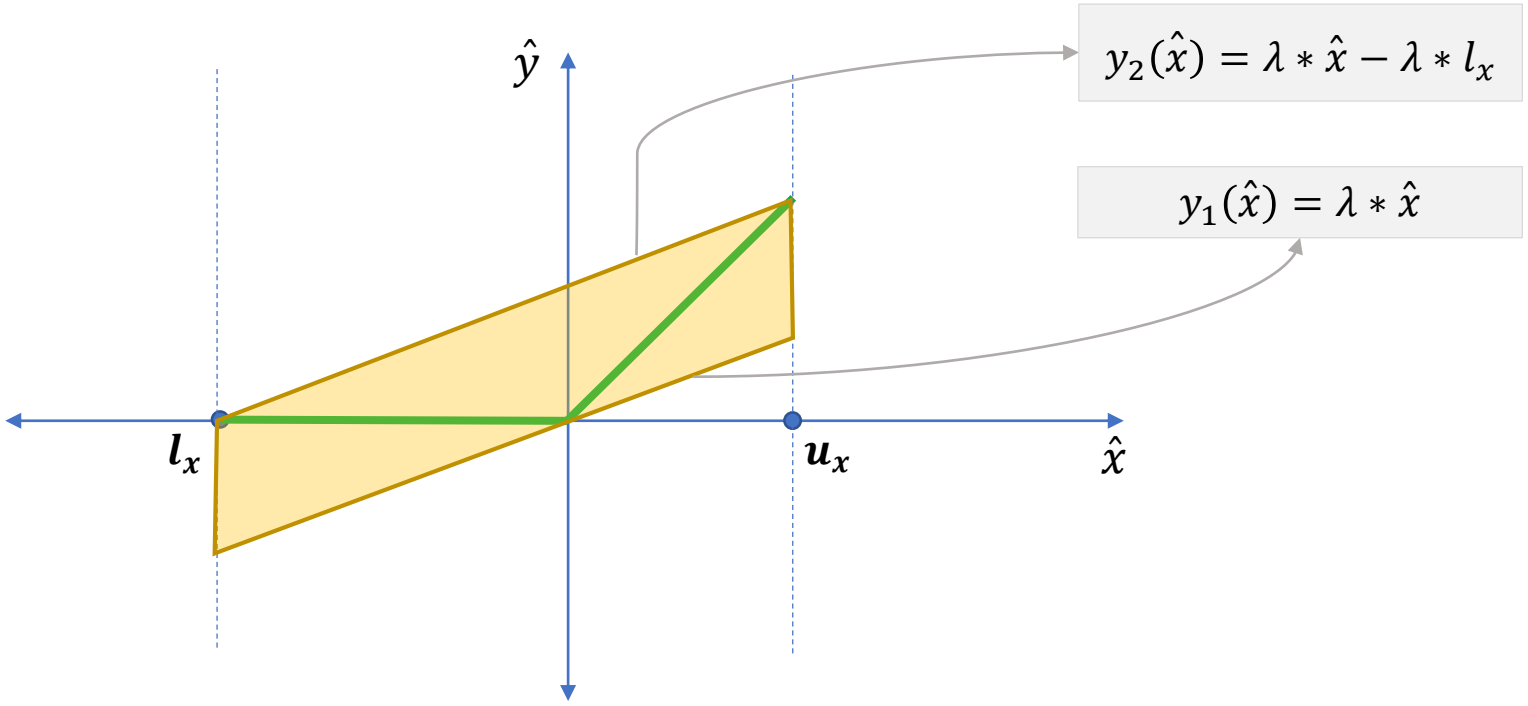
distance is:

$$d = -y_1(l_x) = -\lambda * l_x$$

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 3: We now have the two lines, but we need to capture the yellow zonotope with them…



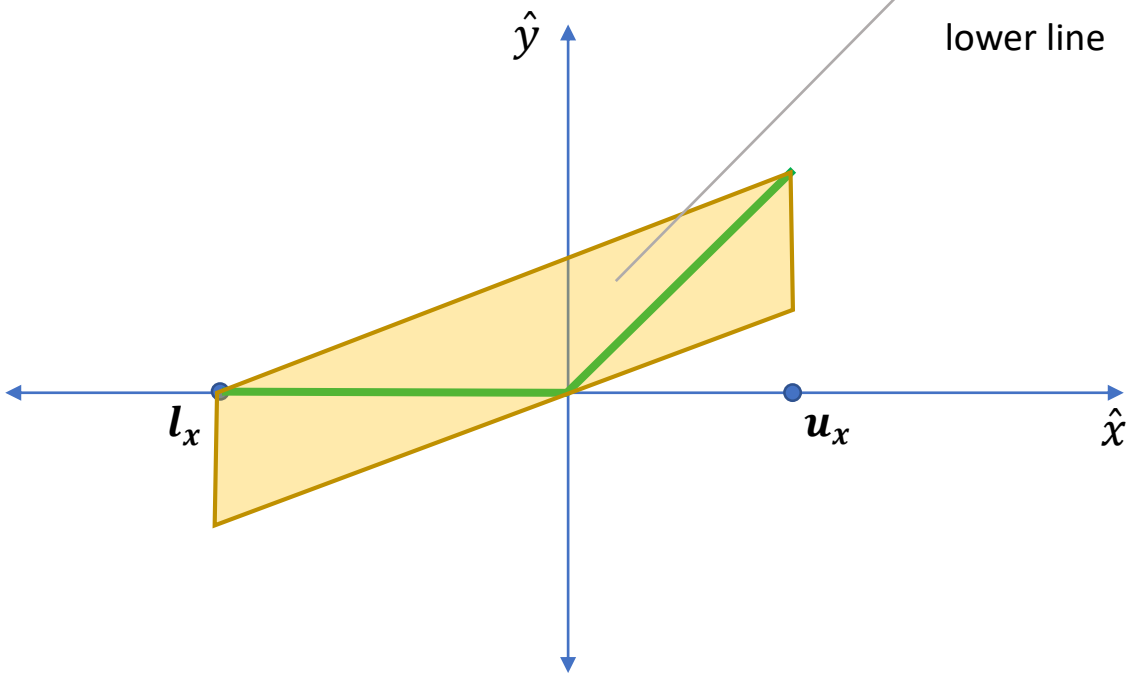
Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 3: The yellow zonotope is captured by the following inequality:

$$\lambda * \hat{x} \leq y(\hat{x}) \leq \lambda * \hat{x} - \lambda * l_x$$

lower line upper line



Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 3: The yellow zonotope is captured by the following inequality: $\lambda * \hat{x} \leq y(\hat{x}) \leq \lambda * \hat{x} - \lambda * l_x$

But this inequality is **not in the Zonotope format** so we need to make the transformer produce an equivalent zonotope in the format

Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$ we need to somehow apply $y = \max(0, x)$ to get $\hat{y} = ???$

Step 3: The yellow zonotope is captured by the following inequality: $\lambda * \hat{x} \leq y(\hat{x}) \leq \lambda * \hat{x} - \lambda * l_x$

This is the same as: $y(\hat{x}) = \lambda * \hat{x} - c * \lambda * l_x$ where $c \in [0,1]$

But our error terms $\epsilon \in [-1,1]$ and not $[0,1]$

So if we want to use $[-1,1]$ we need to have $c = \frac{\epsilon_{new} + 1}{2}$ where ϵ_{new} is a new error term

Thus we obtain $y(\hat{x}) = \lambda * \hat{x} - \frac{\epsilon_{new} + 1}{2} * \lambda * l_x$

The final ReLU transformer is: $y(\hat{x}) = \lambda * \hat{x} - \epsilon_{new} * \frac{\lambda * l_x}{2} - \frac{\lambda * l_x}{2}$ → This is in zonotope format

Applying the Zonotope ReLU Transformer

Given this: $\hat{x} = a_0 + \sum_{i=1}^{k} a_i \, \epsilon_i$, **to apply the transformer**, we first compute λ , l_x, u_x

Plug $\hat{x}$ into transformer definition: $y(\hat{x}) = \lambda * \hat{x} - \epsilon_{new} * \frac{\lambda * l_x}{2} - \frac{\lambda * l_x}{2}$

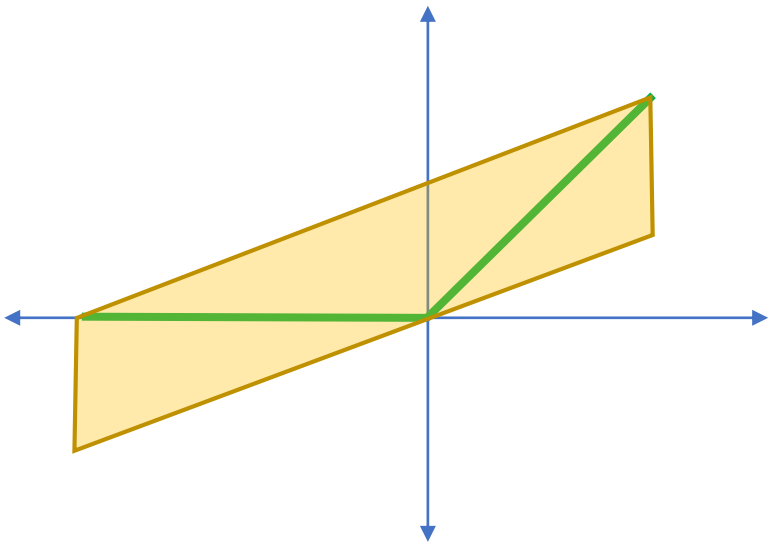
And obtain: $y(\hat{x}) = \lambda a_0 + \sum_{i=1}^{k} \lambda \, a_i \, \epsilon_i - \epsilon_{new} * \frac{\lambda * l_x}{2} - \frac{\lambda * l_x}{2}$

After doing basic calculations: $y(\hat{x}) = b_0 + \sum_{i=1}^{k+1} b_i \, \epsilon_i$

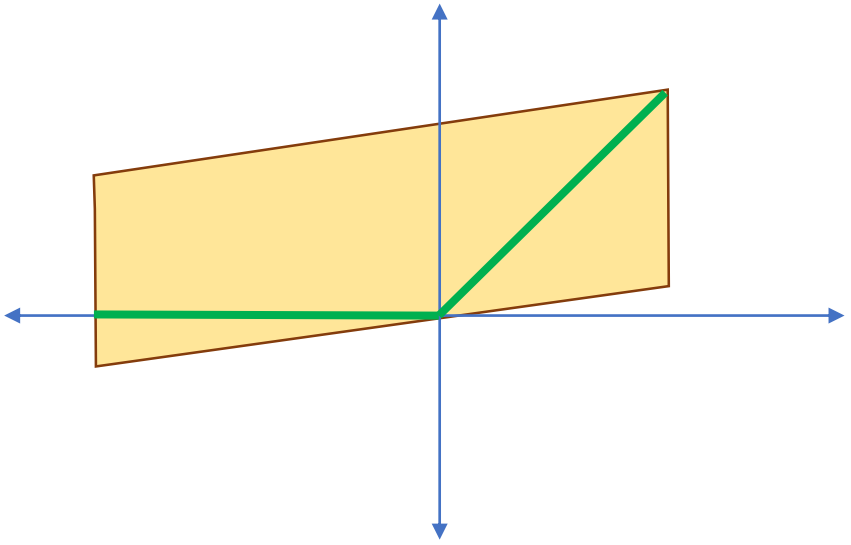
This is our final result

Not great news:
Error terms increase as depth of network increases!

Two Non-Comparable Zonotope Transformers



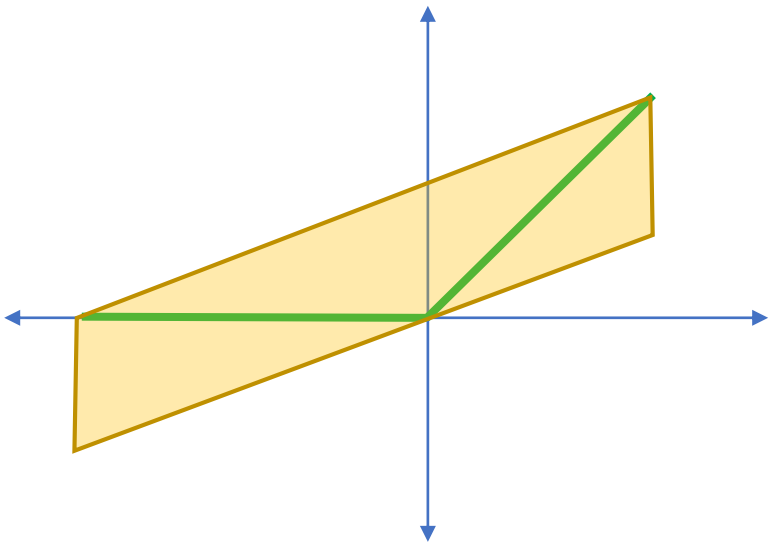
Zonotope transformer I



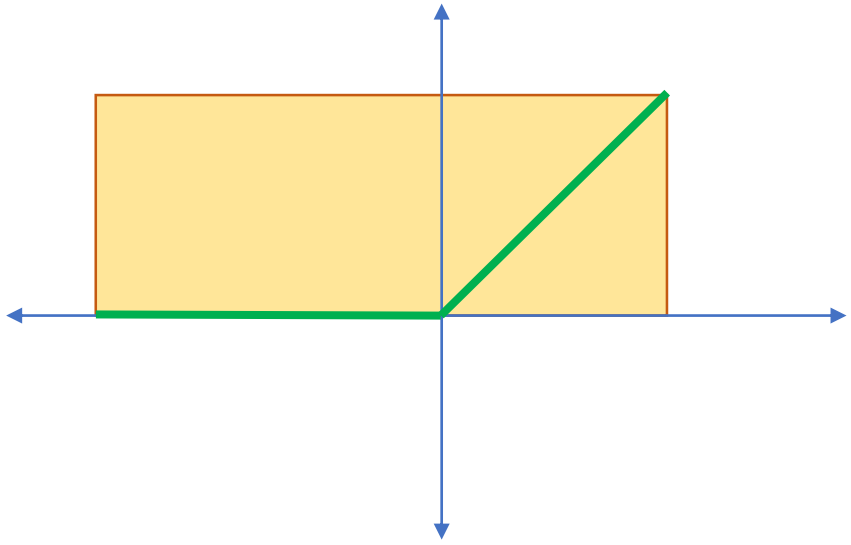
Zonotope transformer II

- First transformer is what we looked at and is optimal area-wise (in the input-output plane).
- Second transformer produces bigger area, but is non-comparable to the first.
- There are many non-comparable ReLU transformers for Zonotope.

Zonotope vs. Box ReLU Transformer



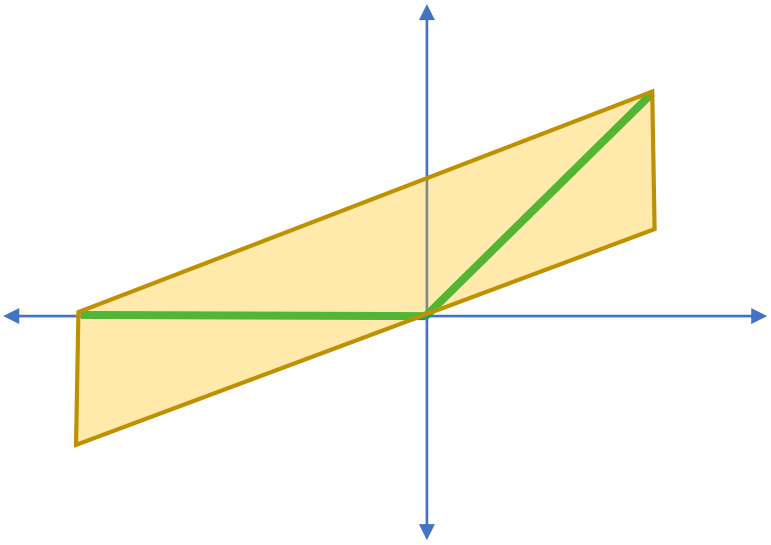
Zonotope transformer



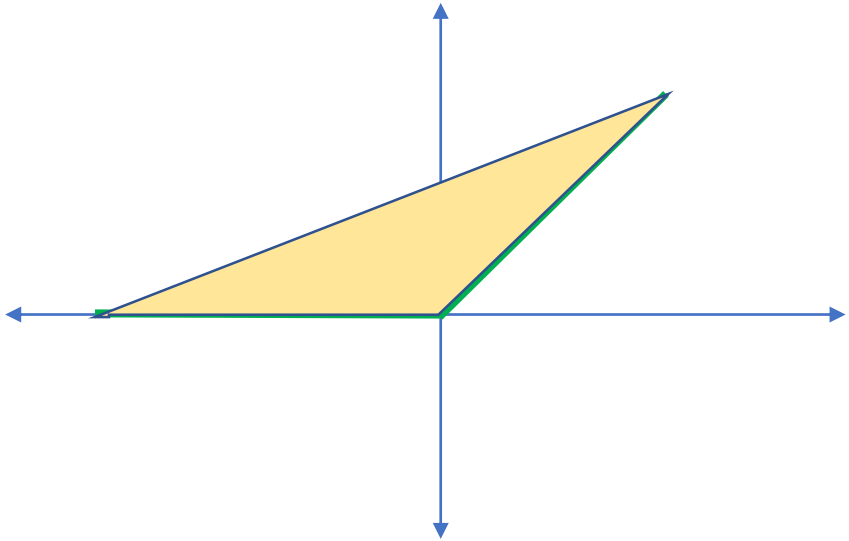
Box transformer

- Mathematically, the two transformers produce **non-comparable** results (no shape is included in the other). However, our Zonotope transformer is **optimal in terms of area**.
- Both transformers can process one layer in parallel, so they are GPU-friendly.
- The Box transformer is memory friendly as it does not introduce new error terms.

Zonotope vs. Optimal ReLU Transformer



Zonotope transformer

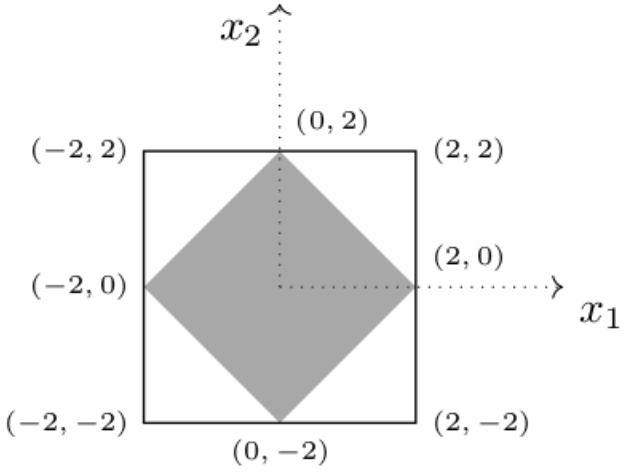


Optimal convex transformer
(known as triangle relaxation)

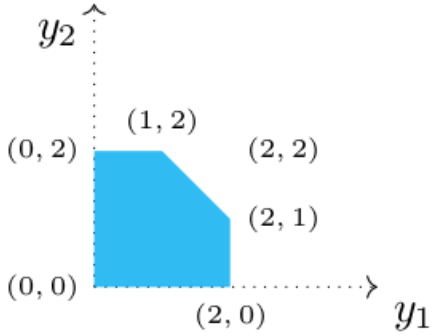
- The area of our transformer is **2x the area** of the optimal triangle transformer.
- Optimal convex transformer is too expensive to work with (so can only be used selectively).
- Optimal transformer produces optimal convex shape if we consider **one neuron at a time**.
- However, see next slides:… ☺

One-neuron at a time is not optimal

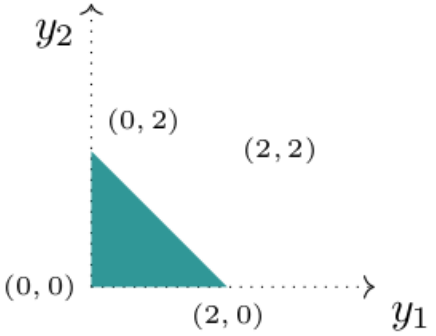
[NeurIPS'19: "Beyond Single Neuron Relaxations for Certification" – Singh, Ganvir, Pueschel, Vechev]



(a) Input shape



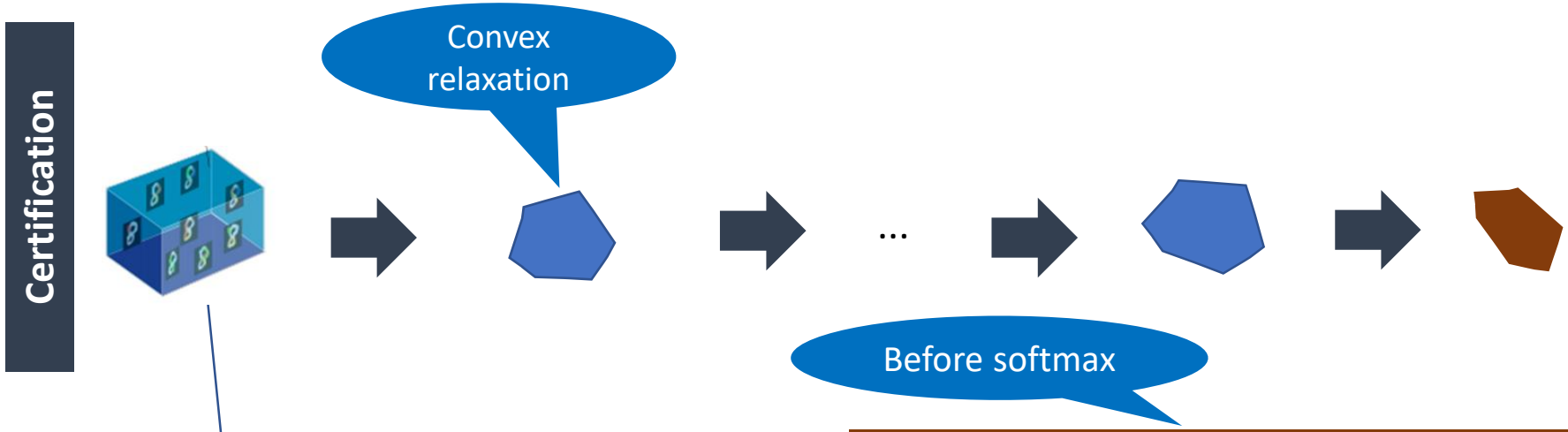
(b) 1-ReLU output



(c) 2-ReLU output

- Contrary to popular belief [see: https://arxiv.org/abs/1902.08722, NeurIPS'19], if we consider 2 neurons at a time, we can produce more precise results.
- Here, we have two neurons x_1 and x_2. The blue figure (b) shows the result for 1-ReLU for triangle, where we compute neuron-wise, that is we compute y_1 from x_1 and y_2 from x_2.
- However, if we compute y_1 and y_2 by considering x_1 and x_2 together, we can get a more precise result, shown in figure (c).
- We will see a lecture on how to compute the result in figure (c) later in the course.

Lets revisit this slide: propagating ϕ



Region ϕ:

$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
...
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i. \epsilon_i \in [0,1]$

Output region ϕ_n

$o_0 = 0$
$o_1 = 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \cdots$
$o_2 = 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \cdots$
...
$o_9 = 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \cdots$
$\forall i. \epsilon_i \in [0,1]$

We saw how to compute the intermediate shapes (in Box or Zonotope format).

For example here ϕ_n follows the Zonotope format.

Summary

- We introduced another incomplete method, the Zonotope relaxation.

- We introduced the abstract transformers for affine and ReLU for Zonotope. Both transformers can be computed in parallel for a layer.

- Unlike Box, Zonotope is exact for affine. Both lose precision on ReLUs.

Next time: some discussion on the general theory of relaxations

(what is optimal, what is sound).