

ECE/CS 584

Verification of Embedded & Cyber-physical Systems

Lecture 8: Neural Network Verification with Bound Propagation Algorithms (Part II)

Prof. Huan Zhang

huan@huan-zhang.com

Reminders

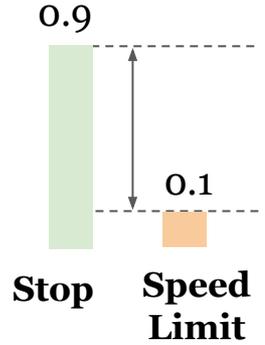
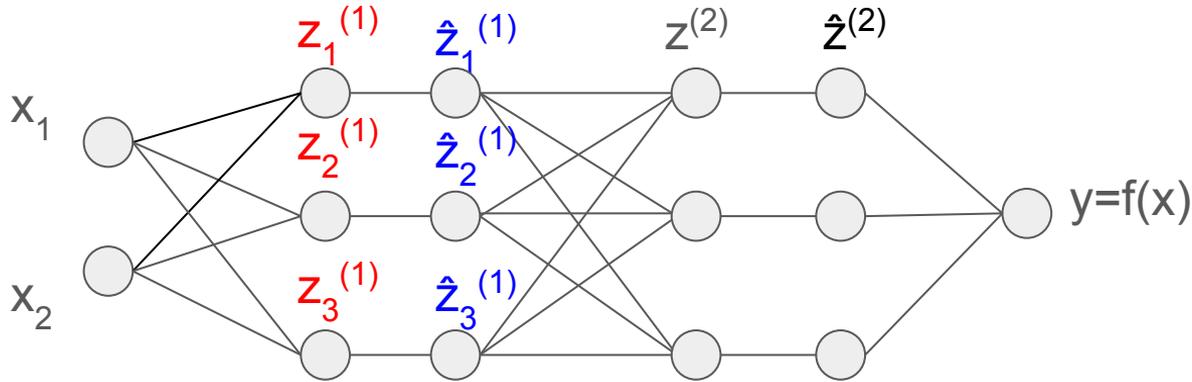
Project proposal due 3/5 (extended by 2 days)

HW2 will be released in the next a few days (deadline after ~2 weeks)

This Thursday (2/27), we have a guest lecture (from Prof. Haoze Wu):

<https://illinois.zoom.us/j/89423827505?pwd=yszKhjHU4985RTNUze0N1DkJOKNjL.1>

Review: NN verification as an **optimization** problem



$$\exists x \in S \wedge y \leq 0 \wedge y = f(x)$$

Input domain under consideration

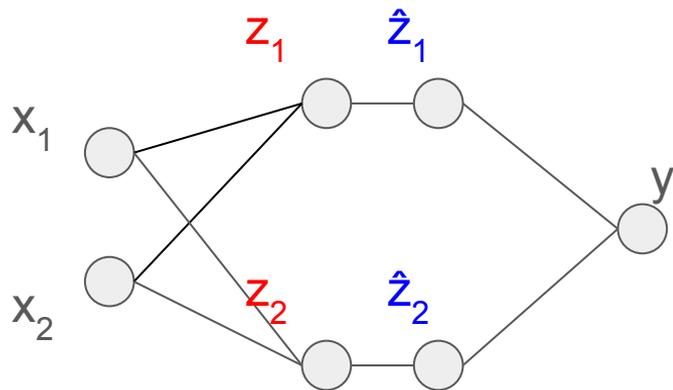
Negation of the desired property

$$y^* = \min_{x \in S} f(x)$$

MILP and LP

Review: bound propagation with linear bounds (CROWN)

Simple example: linear \rightarrow ReLU \rightarrow linear

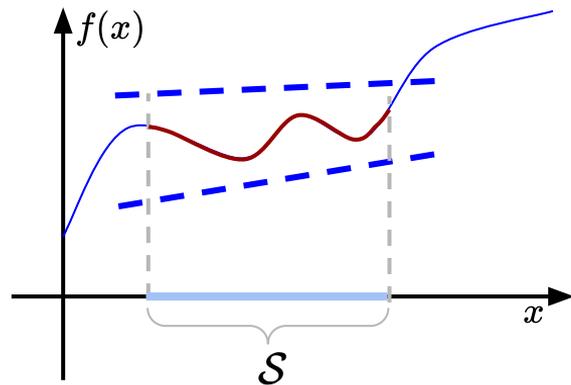


$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$

$$z_1 = x_1 - x_2 \quad z_2 = 2x_1 - x_2$$

$$y = \text{ReLU}(z_1) - \text{ReLU}(z_2)$$

Goal: bound y using symbolic linear functions of x
(linear inequalities)



Review: bound propagation with linear bounds (CROWN)

Prerequisite: all pre-activation bounds (can be computed using CROWN by treating z_1 and z_2 as the output neuron)

$$x_1 \in [-1, 2], x_2 \in [-2, 1] \quad z_1 = x_1 - x_2 \quad z_2 = 2x_1 - x_2$$

$$z_1 \in [-2, 4] \quad z_2 \in [-3, 6]$$

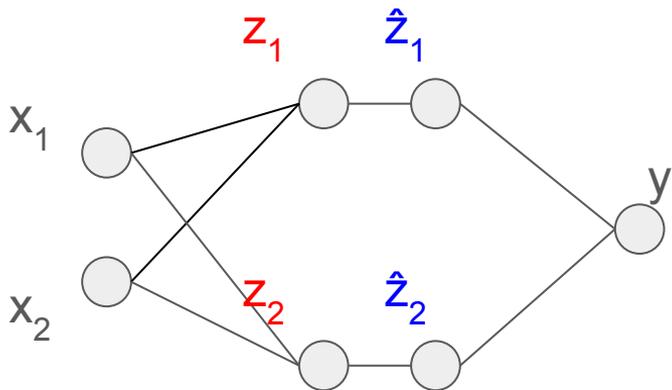
Pre-activation bounds needed for linear bounds
of ReLU or other non-linear functions

Review: bound propagation with linear bounds (CROWN)

Propagation starts from the output y .

Step 1: bound y using linear functions of y (base case): $y \leq y, y \geq y$

Step 2: bound y using linear functions of $\hat{\mathbf{z}}$: simply plugin the definition of the second linear layer: $y = \hat{z}_1 - \hat{z}_2$



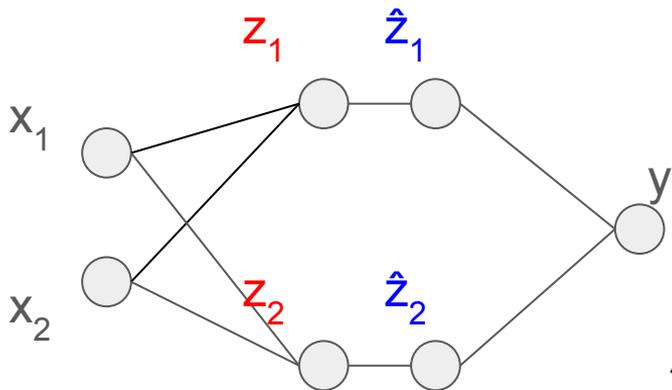
$$y \leq \hat{z}_1 - \hat{z}_2, y \geq \hat{z}_1 - \hat{z}_2$$

Linear layer: simple substitution

Review: bound propagation with linear bounds (CROWN)

Step 3: bound y using linear functions of z : need linear bounds for ReLU functions, which allows us to replace \hat{z} with z

ReLU layer: use linear bound
Check **sign** of coefficients and take the lower or upper bound



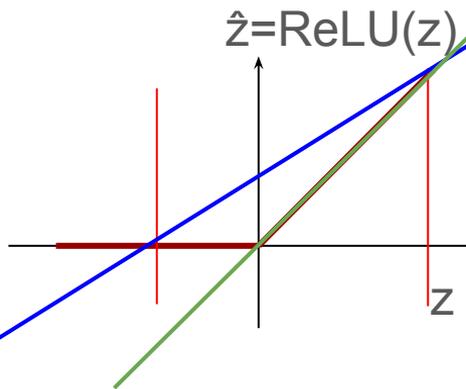
$$y \leq 1 \cdot \hat{z}_1 + (-1) \cdot \hat{z}_2$$
$$y \geq 1 \cdot \hat{z}_1 + (-1) \cdot \hat{z}_2$$

$$z_1 \leq \text{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

$$z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$



$$z_1 - \left(\frac{2}{3}z_2 + 2\right) \leq y \leq \left(\frac{2}{3}z_1 + \frac{4}{3}\right) - z_2$$



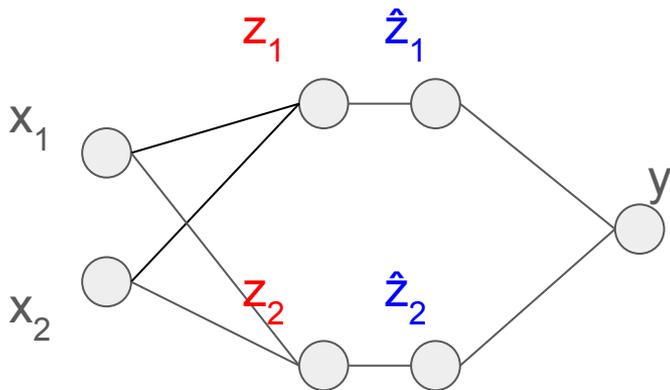
Review: bound propagation with linear bounds (CROWN)

Step 4: bound y using linear functions of \mathbf{x}

$$z_1 - \left(\frac{2}{3}z_2 + 2\right) \leq y \leq \left(\frac{2}{3}z_1 + \frac{4}{3}\right) - z_2$$

$$z_1 = x_1 - x_2$$

$$z_2 = 2x_1 - x_2$$



$$-\frac{1}{3}x_1 - \frac{1}{3}x_2 - 2 \leq y \leq -\frac{4}{3}x_1 + \frac{1}{3}x_2 + \frac{4}{3}$$

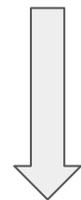
Linear layer: simple substitution

Review: bound propagation with linear bounds (CROWN)

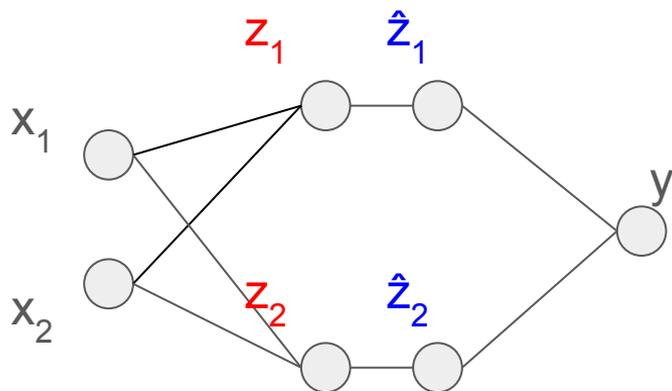
Step 5: concretize linear bounds

$$-\frac{1}{3}x_1 - \frac{1}{3}x_2 - 2 \leq y \leq -\frac{4}{3}x_1 + \frac{1}{3}x_2 + \frac{4}{3}$$

$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$



$$y \in [-3, 3]$$



How to improve bound propagation

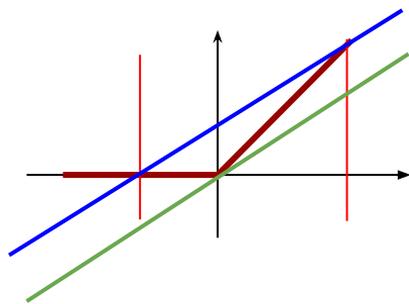
Bound propagation is fast, but what if the bounds are not tight enough?

Goal: use more time to “refine” the bounds. Two techniques:

- Bound optimization (previous lecture)
- **Branch and bound (this lecture)**

Bound optimization (α -CROWN)

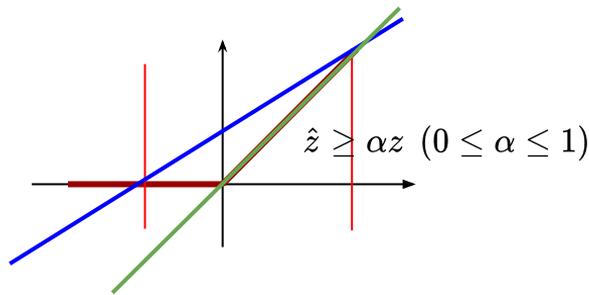
In the previous lecture, we discussed the possibility of making the lower bound of a ReLU function optimizable. α can be optimization used gradient descent.



$$\frac{2}{3}z_1 \leq \text{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

$$\frac{2}{3}z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

$$y \in \left[-\frac{10}{3}, 2\right]$$



$$z_1 \leq \text{ReLU}(z_1) \leq \frac{2}{3}z_1 + \frac{4}{3}$$

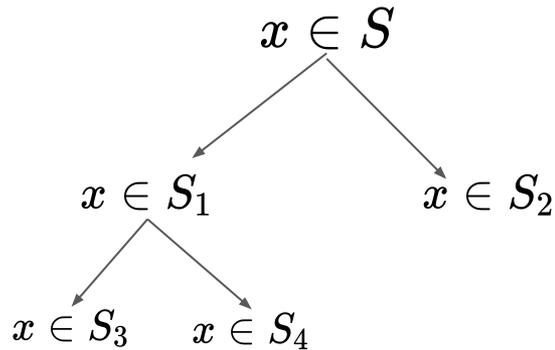
$$z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

$$y \in [-3, 3]$$

Branch and bound

General idea: split (branch) the original problem into easier subproblems; obtain bounds on each subproblem

Define **LB(S)** as the lower bound obtained using bound propagation for $\min_{x \in S} f(x)$



$$S_1 \cup S_2 = S$$

$$S_3 \cup S_4 \cup S_2 = S$$

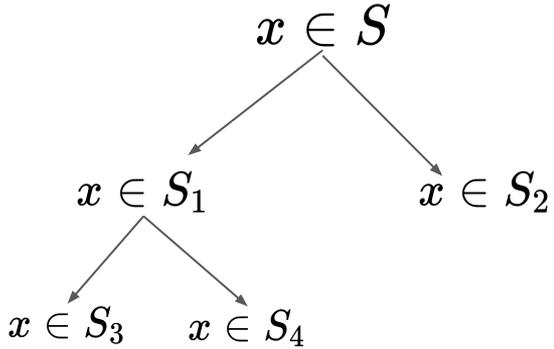
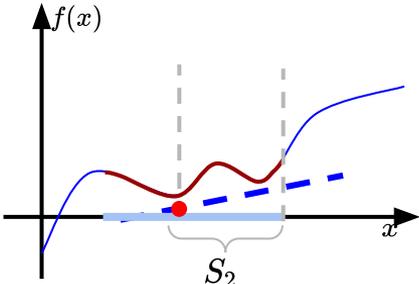
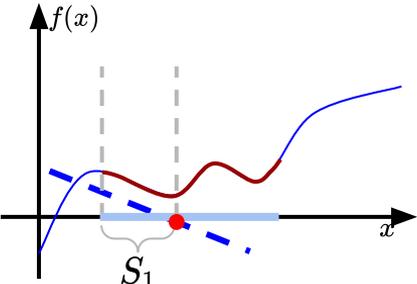
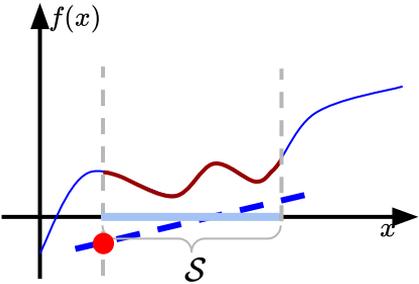
All leaf nodes

$$LB(S)$$

$$\min(LB(S_1), LB(S_2))$$

$$\min(LB(S_3), LB(S_4), LB(S_2))$$

Branch and bound: why the lower bounds become tighter?



$$S_1 \cup S_2 = S$$

$$S_3 \cup S_4 \cup S_2 = S$$

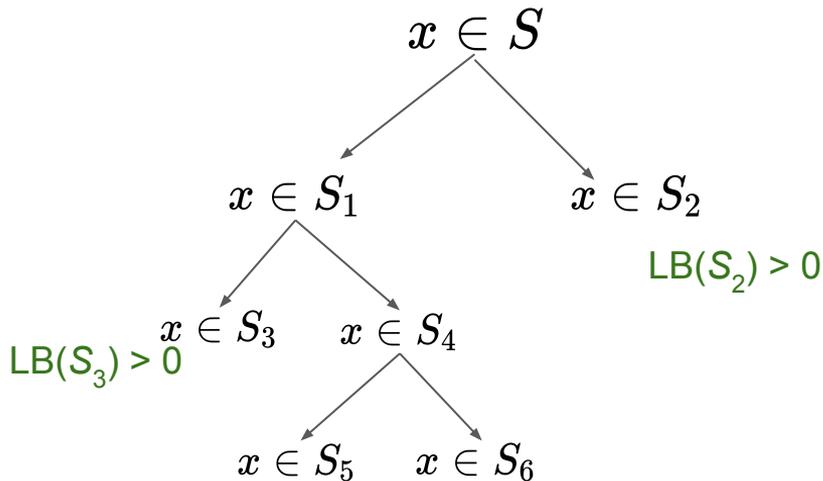
$$LB(S)$$

$$\min(LB(S_1), LB(S_2))$$

$$\min(LB(S_3), LB(S_4), LB(S_2))$$

Branch and bound

If $LB(S_i) > 0$, it can be removed from our problem since the property is verified on this subdomain S_i ; branch and bound is needed for unverified subdomains only.



List of unverified subproblems

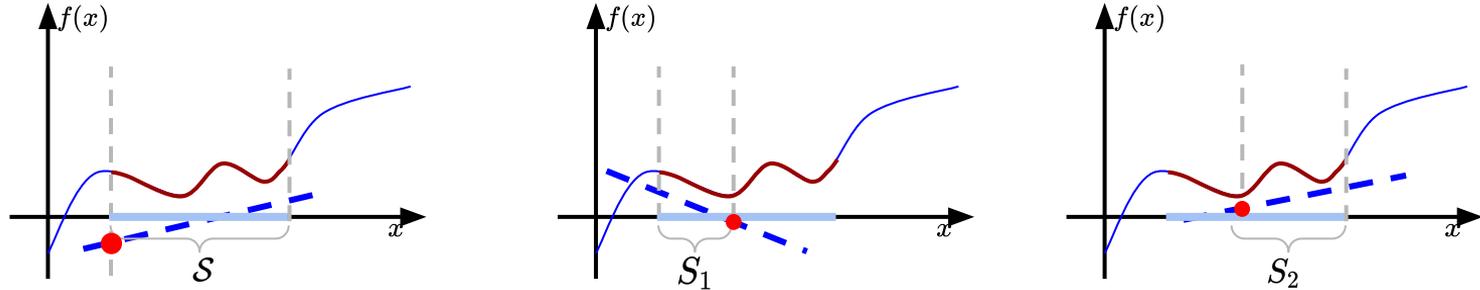
$\{S\}$

~~$\{S_1, S_2\}$~~

~~$\{S_3, S_4\}$~~

$\{S_5, S_6\}$

Branch and bound on input



Split each into domain S , typically by

$$S = \{x_1 \in [-1, 1], x_2 \in [-1, 1]\} \Rightarrow$$

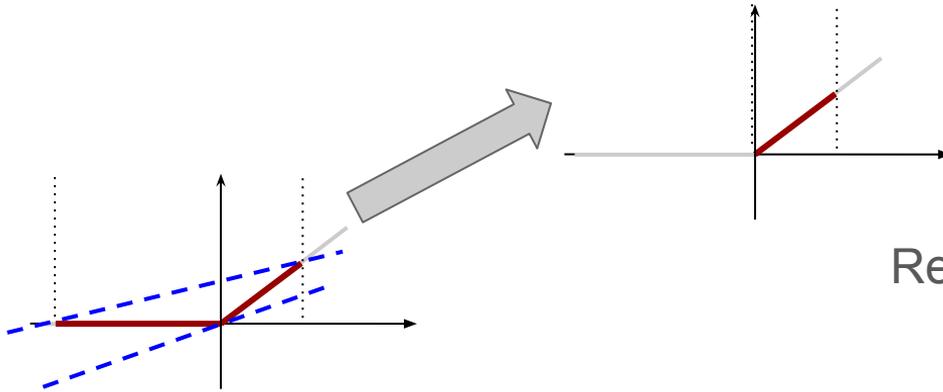
$$S_1 = \{x_1 \in [-1, 0], x_2 \in [-1, 1]\}, S_2 = \{x_2 \in [0, 1], x_2 \in [-1, 1]\}$$

Implementation is easy

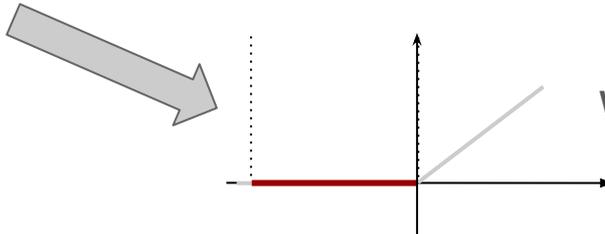
Does not work well when input dimension is very high (e.g., image inputs)

Branch and bound on ReLU

Implicitly split input domain S by considering a ReLU neuron in two cases: active and inactive.



ReLU becomes **linear** in both subproblems



Works best when the number of unstable neurons is not very large

Branch and bound on ReLU

Implicitly split input domain S by considering a ReLU neuron in two cases: active and inactive.

$$S = \{x_1 \in [-1, 1], x_2 \in [-1, 1]\} \Rightarrow$$

$$S_1 = \{x_1 \in [-1, 1], x_2 \in [-1, 1], z_j^{(i)}(x_1, x_2) \geq 0\},$$

z is a function of input x

$$S_2 = \{x_1 \in [-1, 1], x_2 \in [-1, 1], z_j^{(i)}(x_1, x_2) \leq 0\}$$

E.g., for our example $x_1 \in [-1, 2], x_2 \in [-2, 1] \quad z_1 = x_1 - x_2$

Splitting z_1 essentially consider two cases $x_1 - x_2 \geq 0$ and $x_1 - x_2 \leq 0$

Let's go over our example again with split

Prerequisite: all pre-activation bounds

$$x_1 \in [-1, 2], x_2 \in [-2, 1] \quad z_1 = x_1 - x_2 \quad z_2 = 2x_1 - x_2$$

Split constraint: $z_1 \leq 0$

$$z_1 \in [-2, 0] \quad z_2 \in [-3, 6]$$

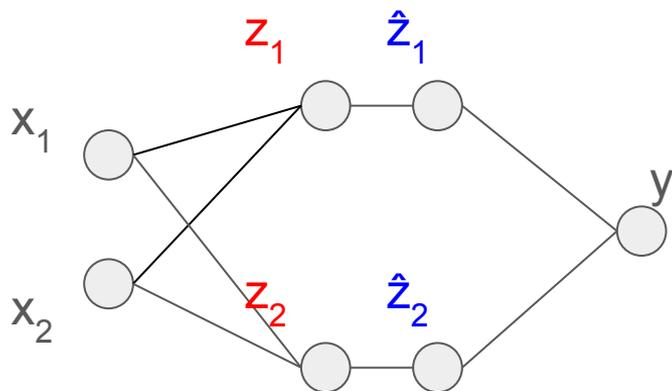
Pre-activation bounds for z_1 updated!

CROWN with neuron split

Let's look at the **lower bound** only (since only lower bound is needed)

Step 1: bound y using linear functions of y (base case): $y \leq y$, $y \geq y$

Step 2: bound y using linear functions of $\hat{\mathbf{z}}$: simply plugin the definition of the second linear layer: $y = \hat{z}_1 - \hat{z}_2$



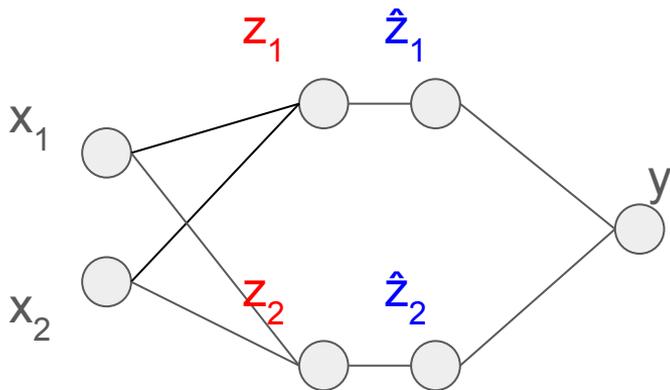
$$y \geq \hat{z}_1 - \hat{z}_2$$

Linear layer: simple substitution

CROWN with neuron split (changed with the split)

Step 3: bound y using linear functions of z : need linear bounds for ReLU functions, which allows us to replace \hat{z} with z

ReLU layer: use linear bound
Check **sign** of coefficients and take the lower or upper bound



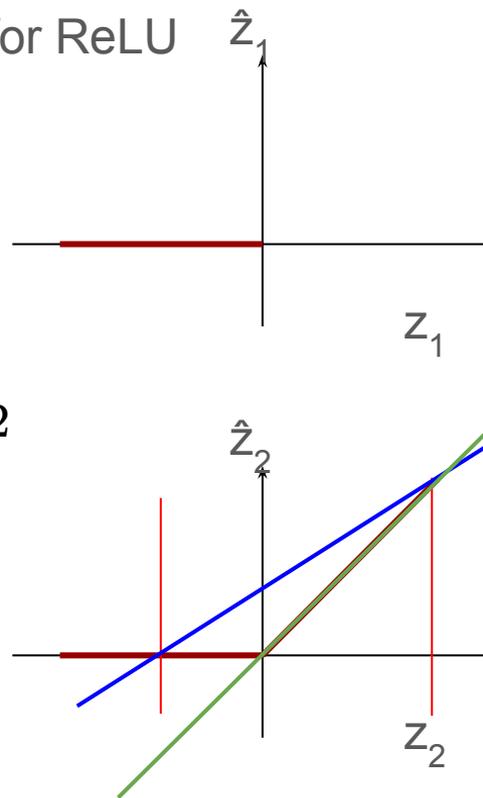
$$y \geq 1 \cdot \hat{z}_1 + (-1) \cdot \hat{z}_2$$

$$0 \leq \text{ReLU}(z_1) \leq 0$$

$$z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$

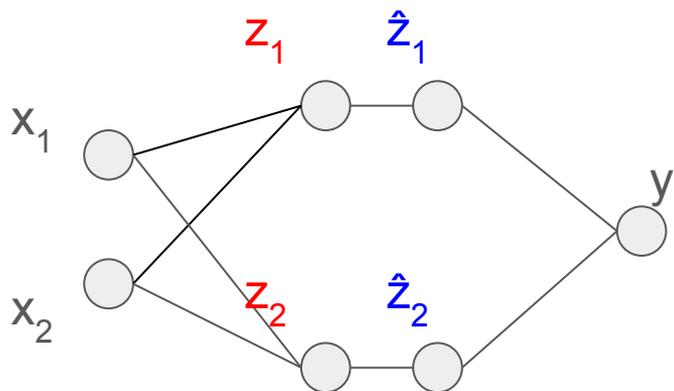


$$y \geq -\frac{2}{3}z_2 - 2$$



CROWN with neuron split (changed with the split)

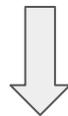
Step 4: bound y using linear functions of \mathbf{x}



$$y \geq -\frac{2}{3}z_2 - 2$$

$$z_1 = x_1 - x_2$$

$$z_2 = 2x_1 - x_2$$

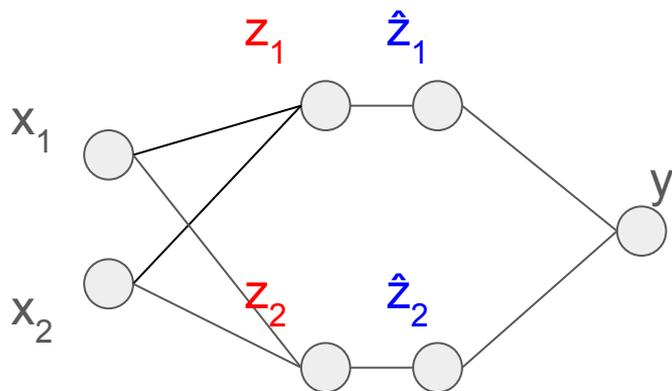


$$y \geq -\frac{4}{3}x_1 + \frac{2}{3}x_2 - 2$$

Linear layer: simple substitution

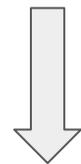
CROWN with neuron split (changed with the split)

Step 5: concretize linear bounds



$$y \geq -\frac{4}{3}x_1 + \frac{2}{3}x_2 - 2$$

$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$



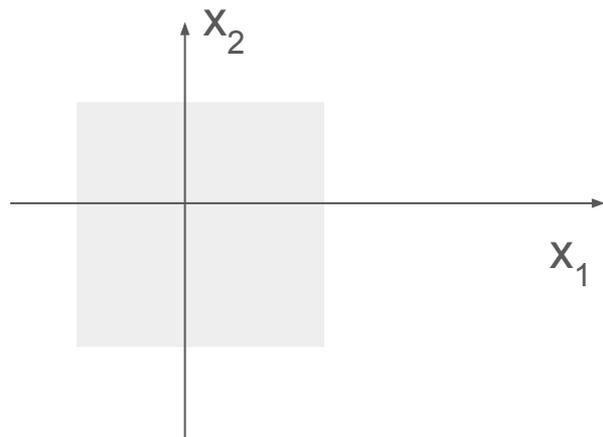
$$y \geq -6$$

Recall that without the split, we have $y \geq -3$
With the split we expect the lower bound to improve??

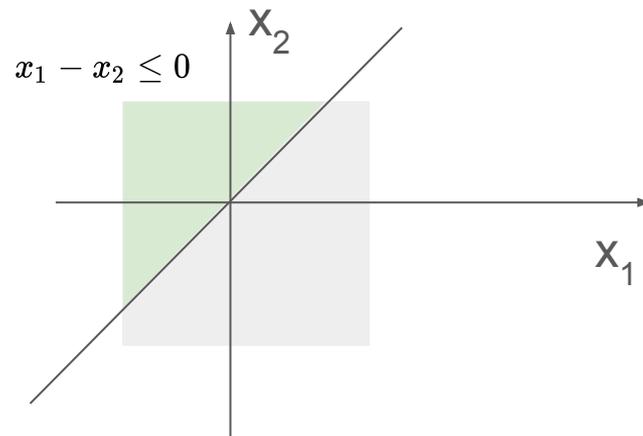
What is going wrong with CROWN?

The split constraint is not fully used during the process.

$$z_1 \leq 0 \implies x_1 - x_2 \leq 0$$



$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$

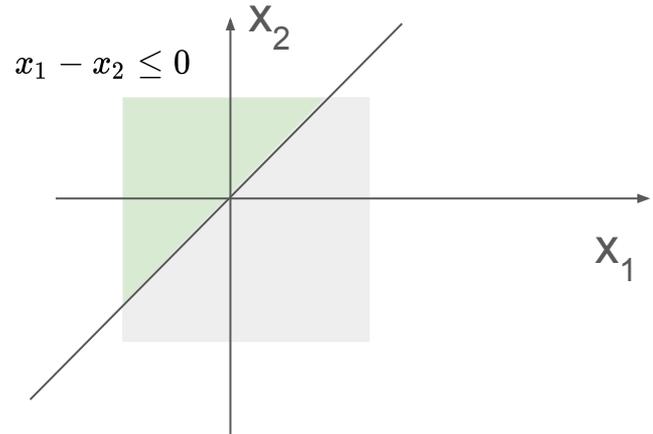


What is going wrong with CROWN?

In the concretization step, we still consider the worst case scenario in the larger box, rather than the green triangle.

$$y \geq -\frac{4}{3}x_1 + \frac{2}{3}x_2 - 2$$

$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$



How to address the problem?

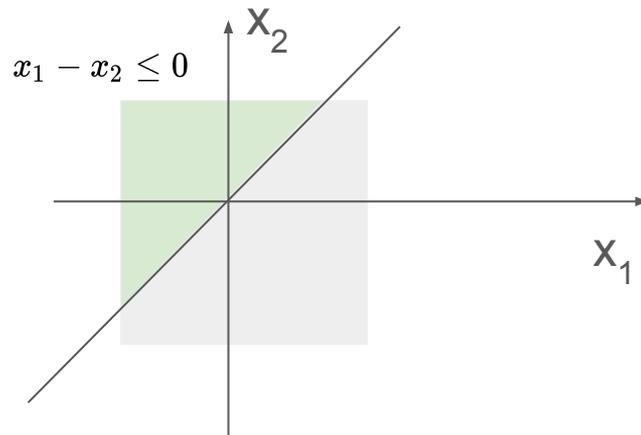
Instead we should solve this optimization problem during concretization:

$$\min_{x_1, x_2} -\frac{4}{3}x_1 + \frac{2}{3}x_2 - 2$$

$$\text{s.t. } z_1 \leq 0$$

$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$

CROWN cannot handle this **constraint!**



β -CROWN: bound propagation with split constraint

We use *Lagrangian multipliers* to handle this constraint.

To solve a constrained optimization problem:

$$\begin{aligned} & \min f_0(x) \\ & \text{such that } f_i(x) \leq 0 \qquad \forall i \in 1, \dots, m \end{aligned}$$

We can define Lagrangian with $\lambda_i \geq 0$:

$$L(x, \lambda) = f_0(x) + \sum_i \lambda_i f_i(x)$$

So the optimization problem can be written as

$$\min_x \max_{\lambda} L(x, \lambda)$$

β -CROWN: bound propagation with split constraint

$$\min_x \max_{\lambda} L(x, \lambda)$$

It is hard to solve directly. But we can then apply *weak duality*, which gives a lower bound

$$\max_{\lambda} \min_x L(x, \lambda) \leq \min_x \max_{\lambda} L(x, \lambda)$$

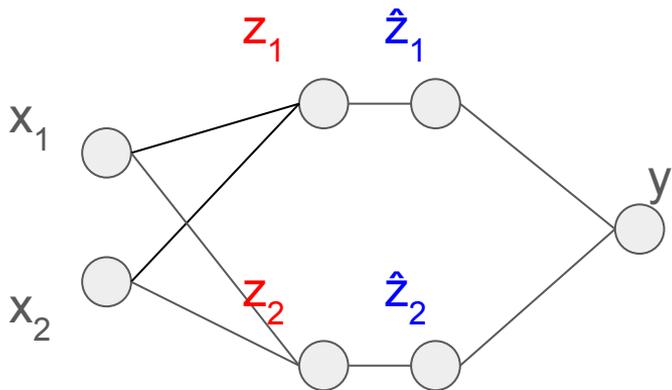
It has an intuitive game-theoretic explanation: whoever plays second may have an advantage, because they know the move of the first player.

Closed form solution exist for the inner minimization
(basically the concretization process without constraints)

β -CROWN with neuron split

Step 3: bound y using linear functions of z : need linear bounds for ReLU functions, which allows us to replace \hat{z} with z

ReLU layer: use linear bound
Check **sign** of coefficients and take the lower or upper bound

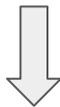


Change in bound propagation: add β for each split constraint

$$y \geq 1 \cdot \hat{z}_1 + (-1) \cdot \hat{z}_2$$

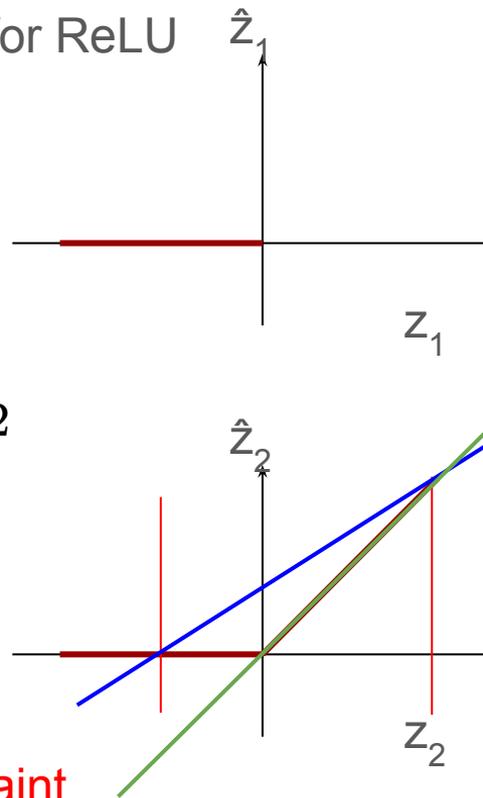
$$0 \leq \text{ReLU}(z_1) \leq 0$$

$$z_2 \leq \text{ReLU}(z_2) \leq \frac{2}{3}z_2 + 2$$



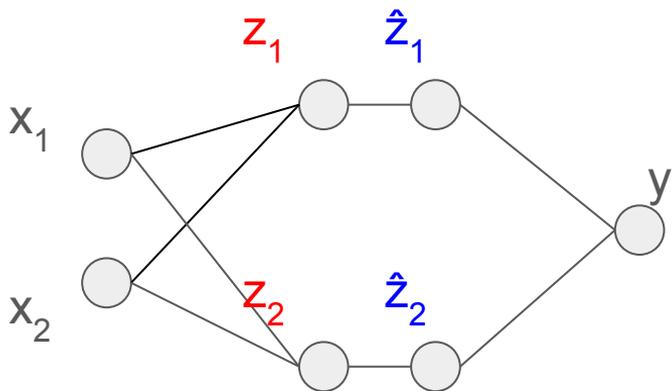
$$y \geq -\frac{2}{3}z_2 - 2 + \beta z_1$$

$\beta \geq 0$



β -CROWN with neuron split

Step 4: bound y using linear functions of \mathbf{x} , Now our bound has a parameter β



$$y \geq -\frac{2}{3}z_2 - 2 + \beta z_1$$

$$z_1 = x_1 - x_2$$

$$z_2 = 2x_1 - x_2$$



$$y \geq (\beta - \frac{4}{3})x_1 + (\frac{2}{3} - \beta)x_2 - 2$$

Linear layer: simple substitution

β -CROWN with neuron split

Step 5: concretize linear bounds

$$y \geq (\beta - \frac{4}{3})x_1 + (\frac{2}{3} - \beta)x_2 - 2$$

$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$

Concretization depends on the sign of the coefficients, so we must discuss three cases:

$$0 \leq \beta \leq \frac{2}{3}$$

$$\frac{2}{3} \leq \beta \leq \frac{4}{3}$$

$$\beta \geq \frac{4}{3}$$

β -CROWN with neuron split

Step 5: concretize linear bounds

$$y \geq (\beta - \frac{4}{3})x_1 + (\frac{2}{3} - \beta)x_2 - 2$$
$$x_1 \in [-1, 2], x_2 \in [-2, 1]$$

$$0 \leq \beta \leq \frac{2}{3} \quad y \geq (\beta - \frac{4}{3}) \cdot 2 + (\frac{2}{3} - \beta) \cdot (-2) - 2$$

The optimal β to maximize y is $2/3$, with objective = $-10/3$

$$\frac{2}{3} \leq \beta \leq \frac{4}{3} \quad y \geq (\beta - \frac{4}{3}) \cdot 2 + (\frac{2}{3} - \beta) \cdot 1 - 2$$

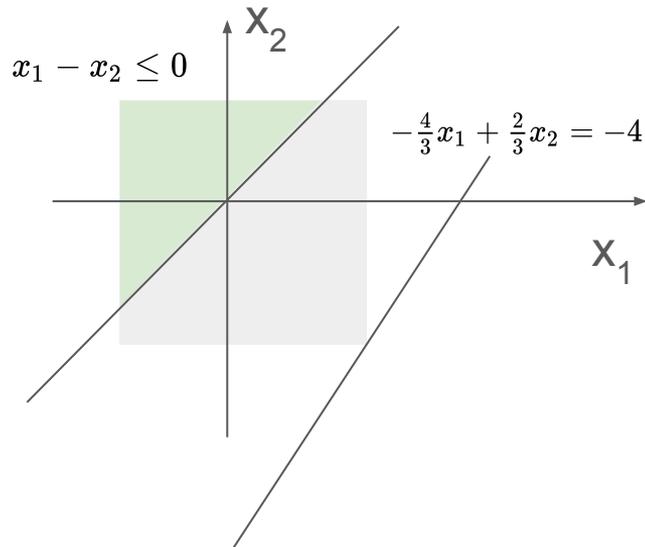
The optimal β is $4/3$, with objective = **$-8/3$**

$$\beta \geq \frac{4}{3} \quad y \geq (\beta - \frac{4}{3}) \cdot (-1) + (\frac{2}{3} - \beta) \cdot 1 - 2$$

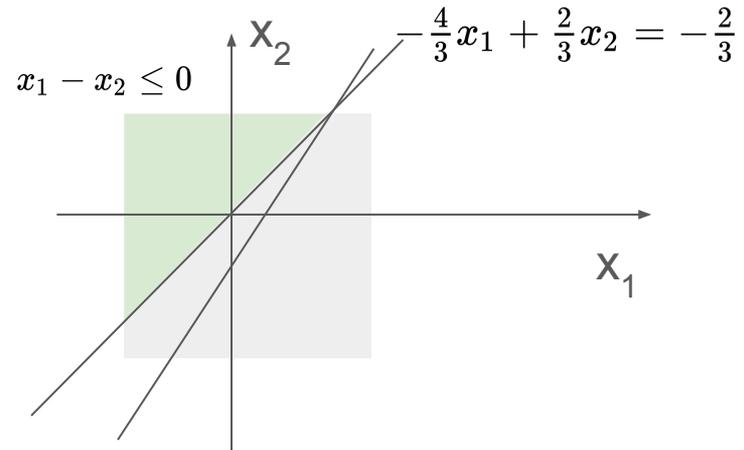
The optimal β is $4/3$, with objective = **$-8/3$**

Geometric interpretation

$$\min_{x_1, x_2} -\frac{4}{3}x_1 + \frac{2}{3}x_2 - 2$$



No constraint, obj = -6



With constraint, obj = $-8/3$, improved!

Which dimension/which neuron to branch?

Similar to the backtracking process in DPLL, the selection of which dimension (for input split) or which neuron (ReLU split) is very important.

Strong branching: try every possible branch and choose the one with actual largest improvements in lower bound

Heuristic branching: estimate how good a branch is, and choose the neuron/dimension with highest score.

Example branching heuristic

$$S = \{x_1 \in [-1, 1], x_2 \in [-1, 1]\} \Rightarrow$$

$$S_1 = \{x_1 \in [-1, 0], x_2 \in [-1, 1]\}, S_2 = \{x_2 \in [0, 1], x_2 \in [-1, 1]\}$$

OR

$$S_1 = \{x_1 \in [-1, 1], x_2 \in [-1, 0]\}, S_2 = \{x_2 \in [-1, 1], x_2 \in [0, 1]\}$$

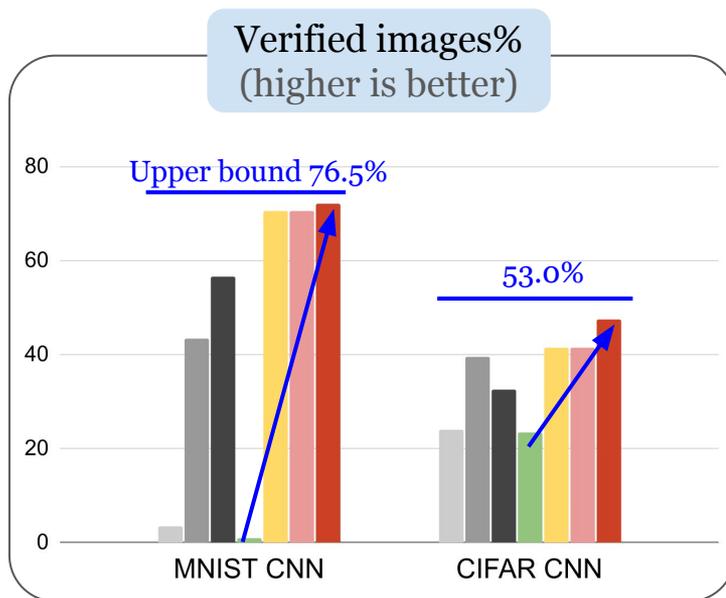
We can estimate the impact on lower bound given changes on x_1 and x_2

Given the CROWN linear bound $y \geq a_1 x_1 + a_2 x_2 + c$, we branch on dimension i where $|a_i|$ is largest.

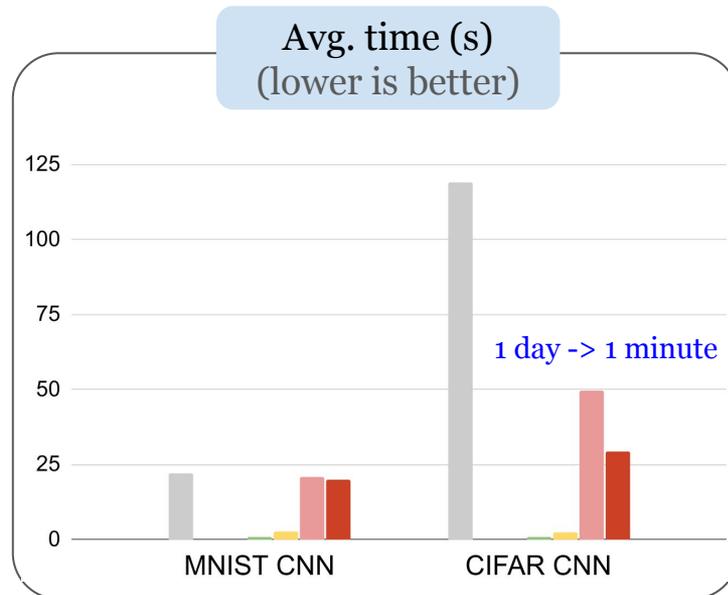
Benchmarks: CROWN-family bound propagation algorithms

- Linear Programming (Salman et al. 2019)
- Semidefinite Programming (Dathathri et al. 2020)
- Integer Programming (Tjeng et al. 2017)
- CROWN
- α -CROWN
- β -CROWN
- GCP-CROWN

Pixel perturbation magnitude:
0.3 for MNIST, 2/255 for CIFAR



Model size: ~5k neurons



Integer programming and semidefinite programming **not plotted** (~1 day)

Key enablers: specialized bound propagation solver + GPU acceleration + BaB

Theoretical Connections: CROWN vs MIP/LP

Prove: $\forall x \in \mathcal{S}, f(x) > 0$

