Reliable and Interpretable Artificial Intelligence

Lecture 5: Certification with complete methods

Martin Vechev

ETH Zurich

Fall 2020

SRILAB

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

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Complete method: MILP

Let us incorporate MILP (Mixed Integer Linear Program) solvers in order to verify robustness. For ReLU networks, MILP is complete. However, this comes at the cost that MILP is at least NP-complete (cannot be solved in polynomial time in any known manner) due to underlying linear programs. Further, there is an additional exponential factor due to splitting.

We will first define the standard MILP problem. Then, we will use the Box bounds to constrain MILP. So, Box will help MILP be faster by doing less work (MILP still remains complete).

MILP Generic Problem Definition

$$\boldsymbol{min}\ \boldsymbol{c_1 x_1 + c_2 x_2 + \cdots + c_n x_n}$$ objective

$$\boldsymbol{a_{11} x_1 + \cdots + a_{1n} x_n \leq b_1}$$
$$\ldots.$$
$$\boldsymbol{a_{m1} x_1 + \cdots + a_{mn} x_n \leq b_m}$$ constraints

$$\boldsymbol{l_i \leq x_i \leq u_i \quad 1 \leq i \leq n}$$ bounds on continuous $\boldsymbol{x_i}$

$$\boldsymbol{x_j \in Z}$$ some $\boldsymbol{x_j}$ are integer

- $\boldsymbol{c_i, a_{ij}, b_i \in R}$ **and** $\boldsymbol{l, u \in R^n}$
- some $\boldsymbol{x_j}$'s can be integers (or even binary), hence Mixed-Integer problem
- state-of-the-art solvers (e.g., Gurobi) require bounds on $\boldsymbol{x_i}$'s

Encoding introduced in: Evaluating Robustness of Neural Networks with Mixed Integer Programming, ICLR 2019 (see Appendix)
https://openreview.net/pdf?id=HyGIdiRqtm

MILP encoding of Neural Network

To encode the network as a MILP instance, we need to:

1. Encode the affine layer
2. Encode the ReLU layer
3. Encode the pre-condition $\boldsymbol{\phi}$
4. Encode the post-condition $\boldsymbol{\psi}$

Encode Affine layer as MILP

This direct as it is just a linear constraint

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

where $\boldsymbol{W}$ are the weights and $\boldsymbol{b}$ is the bias

(note: convolution is also an affine transformation, can be encoded directly)

Encode ReLU layer as MILP

ReLU definition is:

$$\boldsymbol{y = \ ReLU(x) = max(0, x)}$$

MILP ReLU encoding is:

$$\boldsymbol{y \leq x \ - l \ * (1 - a)}$$

$$\boldsymbol{y \geq x}$$

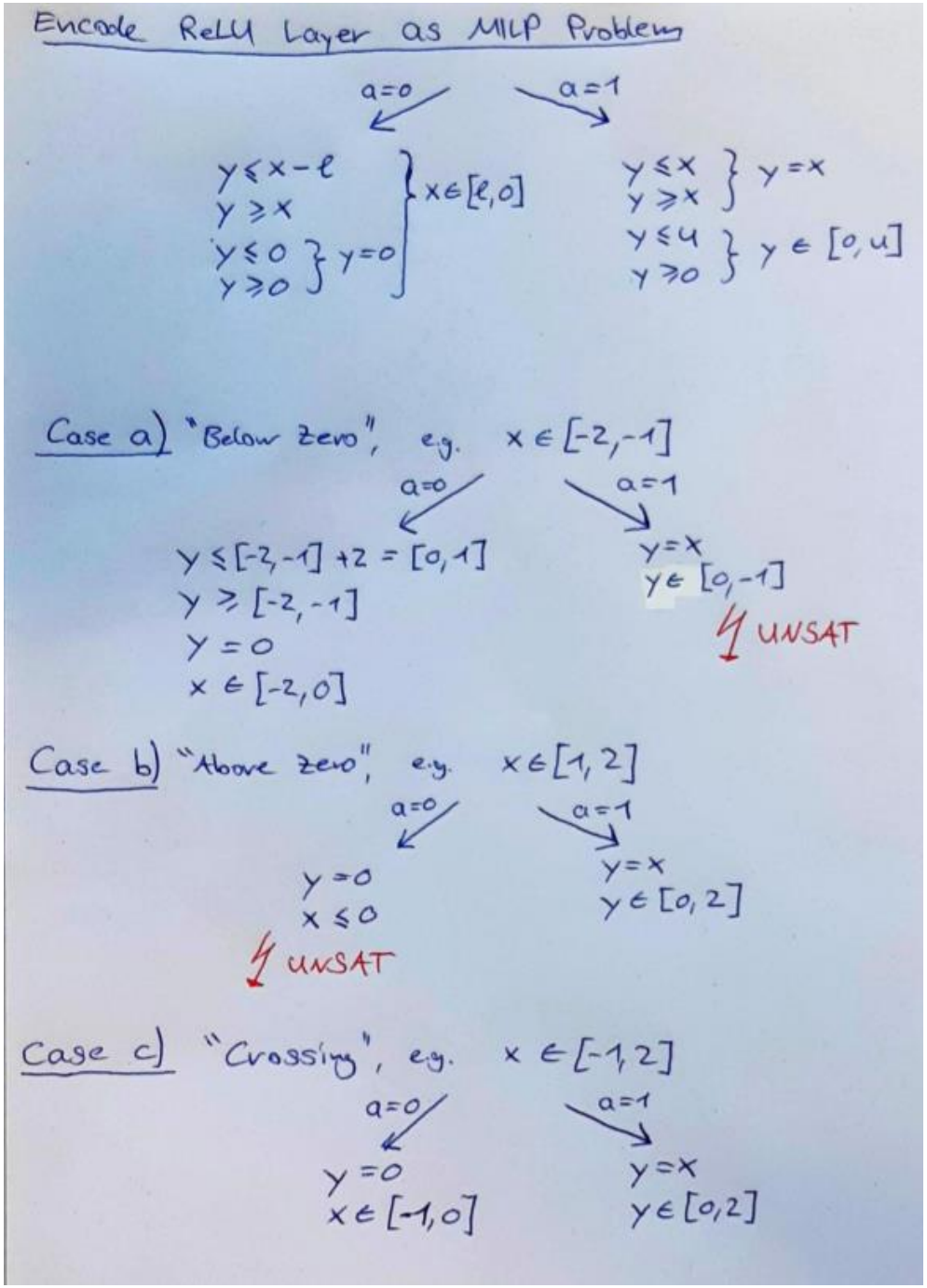
$$\boldsymbol{y \leq u * a}$$

$$\boldsymbol{y \geq 0}$$

$$\boldsymbol{a \in \{0, 1\}}$$ → $\boldsymbol{a}$ is a binary integer variable

This assumes we have computed lower $\boldsymbol{l}$ and upper $\boldsymbol{u}$ bounds for the input neuron $\boldsymbol{x}$ (e.g., by using Box beforehand).

(*Intuition of the encoding on board)



Reminder: ReLU encoding

$$y \leq x - l * (1 - a)$$

$$y \geq x$$

$$y \leq u * a$$

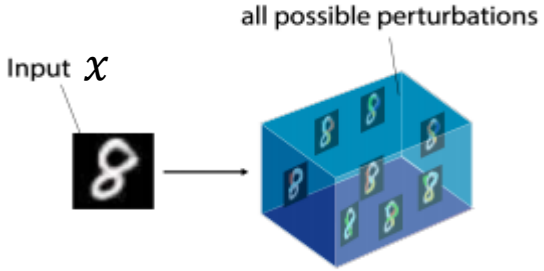
$$y \geq 0$$

$$a \in \{0, 1\}$$

Encode Pre-condition ϕ as MILP

Lets take ϕ = L_∞ ball around x as an example:

L_∞ ball: $\text{Ball}(x)_\epsilon = \{x' \mid ||x - x'||_\infty < \epsilon\}$



MILP encoding:

$x_i - \epsilon \leq x' \leq x_i + \epsilon$ → That is, we will introduce lower and upper bound constraints for each input neuron x_i'

Encode Post-condition ψ as MILP

Lets take ψ = label 0 is more likely than label 1 (our example network)

$$\psi = o_0 > o_1$$

We want to prove this. Hence, this must be forming our MILP objective

MILP encoding:

$$\boldsymbol{min}\ o_0 - o_1$$

After MILP finishes, final verification takes the computed values for o_0 , o_1 and checks if o_0 is indeed greater than o_1 . If yes, verification succeeds, if not, verification fails.

Generic vs. Instantiated MILP

Generic MILP problem

$$\boldsymbol{min}\ \boldsymbol{c_1x_1 + c_2x_2 + \cdots + c_nx_n}$$

$$\boldsymbol{a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1}$$
$$\boldsymbol{\ldots.}$$
$$\boldsymbol{a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m}$$

$$\boldsymbol{l_i \leq x_i \leq u_i \quad 1 \leq i \leq n}$$

$$\boldsymbol{x_j \in Z}$$

Our MILP instance

$$\boldsymbol{min}\ o_0 - o_1$$

Plug in the affine and ReLU MILP encodings as defined

$$l_i \leq x_i^p \leq u_i$$

pre-computed Box bounds on all neurons in each layer p

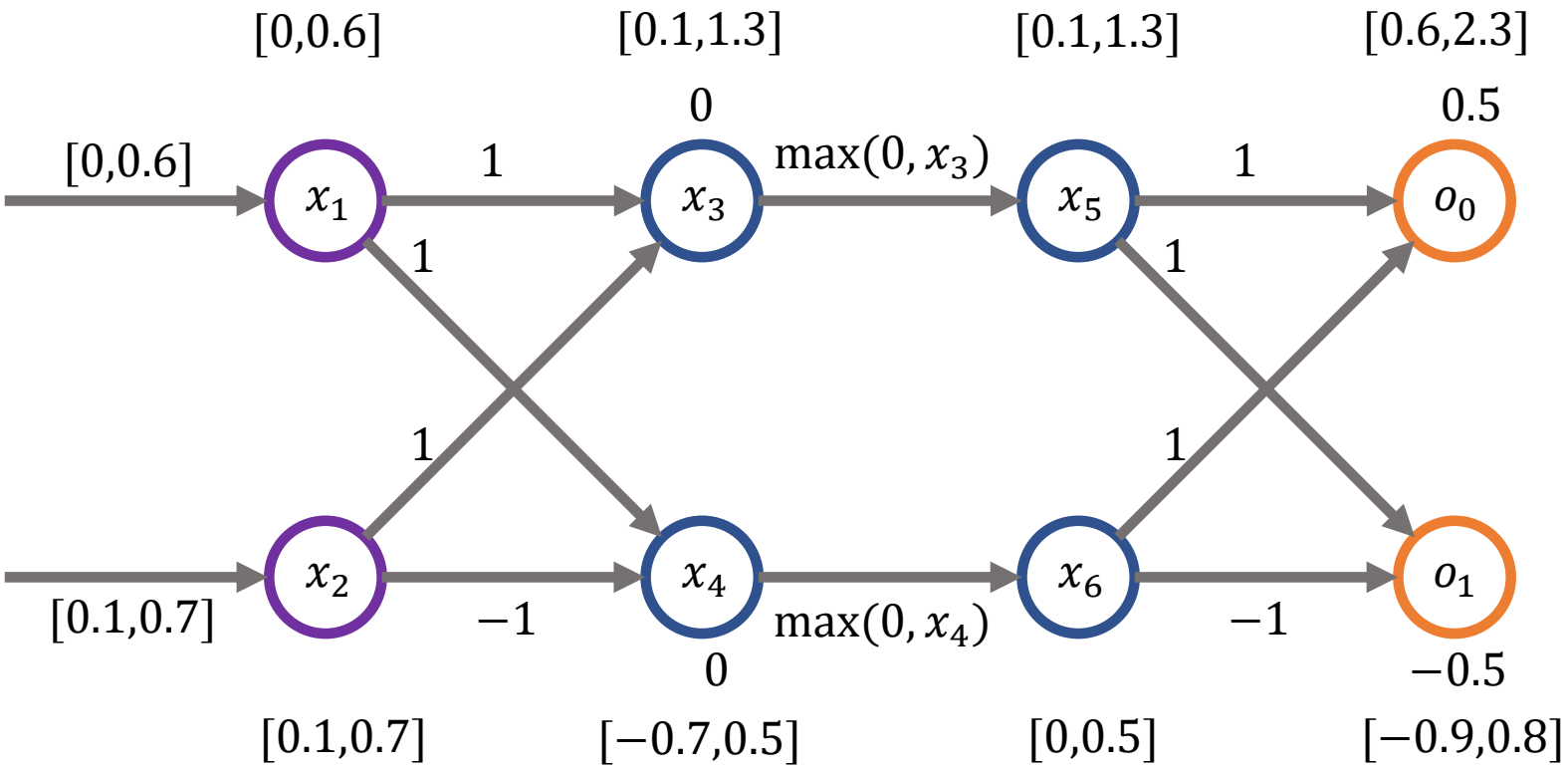
$$x_i - \epsilon \leq x_i' \leq x_i + \epsilon$$

ϕ: bounds on input neurons

$$a_j \in \{0, 1\}$$

These are the $\boldsymbol{a} \in \{\boldsymbol{0}, \boldsymbol{1}\}$ vars from the ReLU encoding

Reminder: these are the bounds with Box



MILP Instance for this network

$$\boldsymbol{min}\ o_0 - o_1$$

Affine

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$o_0 = x_5 + x_6 + 0.5$$
$$o_1 = x_5 - x_6 - 0.5$$

ReLU: $x_5 = \max(0, x_3)$

$$x_5 \leq x_3 - 0.1 * (1 - a_5)$$
$$x_5 \geq x_3$$
$$x_5 \leq 1.3 * a_5$$
$$x_5 \geq 0$$

ReLU: $x_6 = \max(0, x_4)$

$$x_6 \leq x_4 + 0.7 * (1 - a_6)$$
$$x_6 \geq x_4$$
$$x_6 \leq 0.5 * a_6$$
$$x_6 \geq 0$$

Input bounds

$$0 \leq x_1 \leq 0.6$$
$$0.1 \leq x_2 \leq 0.7$$

Pre-computed Box bounds

$$0.1 \leq x_3 \leq 1.3$$
$$-0.7 \leq x_4 \leq 0.5$$
$$0.1 \leq x_5 \leq 1.3$$
$$0 \leq x_6 \leq 0.5$$

Binary integer variables

$$\boldsymbol{a_5, a_6} \in \{\boldsymbol{0, 1}\}$$

Solving this MILP instance will lead to proving the property

Lets solve MILP to see why Box helps

(Note: MILP solvers employ elaborate algorithms, the idea here is to show how Box can help even state-of-the-art MILP solvers)

Case $\mathbf{a_5 = 0}$

$$\boldsymbol{min}\ o_0 - o_1$$

Affine

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$o_0 = x_5 + x_6 + 0.5$$
$$o_1 = x_5 - x_6 - 0.5$$

ReLU: $x_5 = \max(0, x_3)$

$$x_5 \leq x_3 - 0.1$$
$$x_5 \geq x_3$$
...

ReLU: $x_6 = \max(0, x_4)$

$$x_6 \leq x_4 + 0.7 * (1 - a_6)$$
$$x_6 \geq x_4$$
$$x_6 \leq 0.5 * a_6$$
$$x_6 \geq 0$$

Pre-computed Box bounds

$$0.1 \leq x_3 \leq 1.3$$
$$-0.7 \leq x_4 \leq 0.5$$
$$0.1 \leq x_5 \leq 1.3$$
$$0 \leq x_6 \leq 0.5$$

This is an infeasible LP instance so MILP does not have to consider it .

Here, Box bounds helped MILP in not exploring further values for $\boldsymbol{a_6}$. So it saved generating two cases for $\boldsymbol{a_6}$.

In practice, we directly generate this MILP

$$\boldsymbol{min}\ o_0 - o_1$$

Affine

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$o_0 = x_5 + x_6 + 0.5$$
$$o_1 = x_5 - x_6 - 0.5$$

ReLU: $x_5 = \max(0, x_3)$

$$x_5 = x_3$$
$$0.1 \le x_5 \le 1.3$$

ReLU: $x_6 = \max(0, x_4)$

$$x_6 \le x_4 + 0.7 * (1 - a_6)$$
$$x_6 \ge x_4$$
$$x_6 \le 0.5 * a_6$$
$$x_6 \ge 0$$

can prove x_3 is positive using Box bounds

Input bounds

$$0 \le x_1 \le 0.6$$
$$0.1 \le x_2 \le 0.7$$

Pre-computed Box bounds

$$0.1 \le x_3 \le 1.3$$
$$-0.7 \le x_4 \le 0.5$$
$$0.1 \le x_5 \le 1.3$$
$$0 \le x_6 \le 0.5$$

Binary integer variable

$$\boldsymbol{a_6} \in \{\boldsymbol{0}, \boldsymbol{1}\}$$

Case $a_6 = 0$

$$\boldsymbol{min}\ o_0 - o_1$$

Affine

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$o_0 = x_5 + x_6 + 0.5$$
$$o_1 = x_5 - x_6 - 0.5$$

ReLU: $x_5 = \max(0, x_3)$

$$x_5 = x_3$$
$$0.1 \le x_5 \le 1.3$$

ReLU: $x_6 = \max(0, x_4)$

$$x_4 \le x_6 \le x_4 + 0.7$$
$$x_6 = 0$$

simplifying

$$o_0 - o_1 = 2 * x_6 + 1$$
$$o_0 - o_1 = 1$$

Input bounds

$$0 \le x_1 \le 0.6$$
$$0.1 \le x_2 \le 0.7$$

Pre-computed Box bounds

$$0.1 \le x_3 \le 1.3$$
$$-0.7 \le x_4 \le 0.5$$
$$0.1 \le x_5 \le 1.3$$
$$0 \le x_6 \le 0.5$$

Here, we proved, subject to the constraints, that the minimum is always 1, hence property holds.

Note: cannot just use the bounds for x_5 and x_6 for computing $o_0 - o_1$: this will fail! Good to try.

Case $a_6 = 1$

$$\boldsymbol{min}\ o_0 - o_1$$

Affine

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$o_0 = x_5 + x_6 + 0.5$$
$$o_1 = x_5 - x_6 - 0.5$$

ReLU: $x_5 = \max(0, x_3)$

$$x_5 = x_3$$
$$0.1 \leq x_5 \leq 1.3$$

ReLU: $x_6 = \max(0, x_4)$

$$x_6 = x_4$$
$$0 \leq x_6 \leq 0.5$$

simplifying

Input bounds

$$0 \leq x_1 \leq 0.6$$
$$0.1 \leq x_2 \leq 0.7$$

Pre-computed Box bounds

$$0.1 \leq x_3 \leq 1.3$$
$$-0.7 \leq x_4 \leq 0.5$$
$$0.1 \leq x_5 \leq 1.3$$
$$0 \leq x_6 \leq 0.5$$

$$o_0 - o_1 = 2 * x_6 + 1$$

$$1 \leq o_0 - o_1 \leq 2$$

$$min(o_0 - o_1) = 1$$

Here, we proved the property again

Summary of Lecture

- We showed how to create a MILP instance for solving the problem of (complete) neural network certification.

- This is useful as MILP alone is prohibitively expensive. This was an example of using an incomplete method to speed up a complete method.

- We discussed one combination of combining complete and incomplete methods. There are other ways for both methods to benefit from each other, we mention some of these in later lectures.