


Machine Learning for Graphs and Sequential Data

Robustness of Machine Learning – Exact Certification

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning 

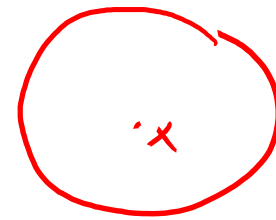
Roadmap

1. Introduction
2. Construction of adversarial examples
3. Improving robustness
4. **Certifiable robustness**
 - **Exact certification**
 - Convex relaxations
 - Lipschitz-continuity
 - Randomized smoothing

Motivation

- Adversarial Training improves robustness, but how can we make sure that the user can really rely on the results?
 - There could still exist some cases where the model behaves in an undesired way
- As discussed, detection of adversarial examples does not seem to work
- **Better approach: Robustness certification.**
 - Idea: try to prove that the classifier's prediction does not change within a radius (measured by some norm)
 - If the proof is successful, we know there cannot be an adversarial example within that radius. We get a guarantee!
 - If the proof is not successful, the prediction could change. Therefore the sample might be an adversarial example (or it might be possible to adversarially change it).
 - In a very conservative approach, we could now refuse our model's prediction and/or consult an expert for manual inspection.

Exact Verification



Goal: Develop an algorithm that answers the question:

“Is the classifier f_θ around the sample \mathbf{x} adversarial-free
(within an ϵ -ball measured by some norm)?”

The algorithm should return **YES** if and only if there are no adversarial example within an ϵ ball around the input sample (i.e. **NO** iff there is an adv. example).

Exact verification methods are typically designed for neural networks with **ReLU activation function**. ReLU networks are very prevalent in deep learning and are well-suited for **combinatorial** exact verification methods.

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

Exact Verification

$$F(x) = \begin{bmatrix} 100 \\ -10 \\ 70 \end{bmatrix} \leadsto \text{softmax}(F(x))$$

- We view the neural network as a **sequence of functions** (i.e. the layers).
- Each **layer** is defined as $f_i(x) = \sigma(W_i x + b_i)$, where W_i and b_i are the weight matrix and the bias of layer i , respectively.
- The **ReLU activation** function is defined as $\sigma(x) = \max(0, x)$ and is applied entry-wise to the input.
- The **overall network** is a function $F: \mathbb{R}^d \rightarrow \mathbb{R}^{|y|}$ given by:

$$F(x) = W_L f_{L-1} \circ f_{L-2} \circ \dots \circ f_1(x) + b_L$$
- The output of F are the **logits** which are subsequently fed into the softmax function to obtain a categorical distribution.
 - We can omit the softmax for certification since the operation is order-preserving (i.e., the ‘winning’ class does not change).

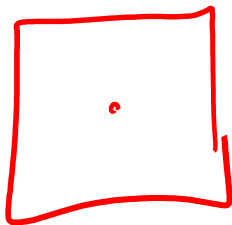
Exact Certification: Complexity

Exact certification is a very powerful method for a defending system: we know exactly when a sample could be an adversarial example and can potentially even use this knowledge to get the worst-case perturbation for adversarial training.

Unfortunately, [Katz et al. 2017] report the following result:

Theorem: Exact certification of neural networks with ReLU activation function and L_∞ -bounded perturbations is **NP-complete**.

Nevertheless, solvers for NP-complete problems have made significant progress, so certifying small to medium-size neural networks is sometimes possible.

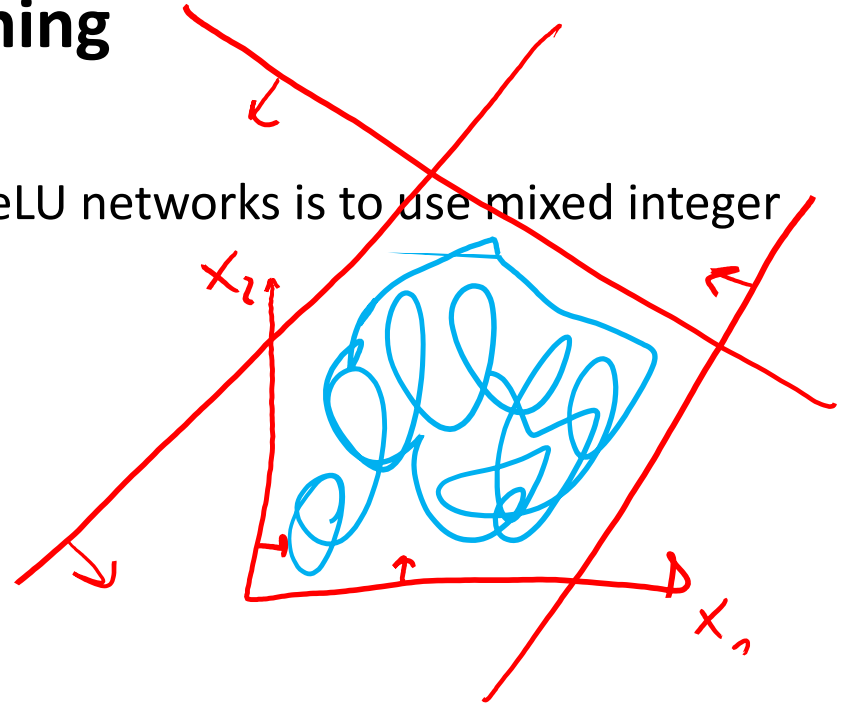


Mixed Integer Linear Programming

- One approach for exact certification of ReLU networks is to use mixed integer linear programming (**MILP**).
- Recall **linear programs** (LPs):

$$\begin{array}{ll}
 \text{minimize} & \mathbf{c}^T \mathbf{x} \\
 \text{subject to} & \mathbf{Ax} \leq \mathbf{b} \\
 & \mathbf{x} \geq 0
 \end{array}$$

$\mathbf{x} \in \mathbb{R}^D$



- Integer linear programs: we have the additional constraints $\mathbf{x}_i \in \mathbb{Z}$, i.e. the variables are integer-valued
- Mixed integer linear programs (**MILP**): *some* variables constrained to be integers, others not.

Mixed Integer Linear Programming: Complexity

- One approach for exact certification of ReLU networks is to use mixed integer linear programming (**MILP**).

- Recall **linear programs** (LPs):

$$\begin{array}{ll}\text{minimize} & \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq 0\end{array}$$

Can be solved **efficiently**
(in polynomial time)

- Integer linear programs: we have the additional constraints $\mathbf{x}_i \in \mathbb{Z}$, i.e. the variables are integer-valued
- Mixed integer linear programs (**MILP**): *some* variables constrained to be integers, others not.

NP-complete

Expressing Exact Certification as MILP

$$F(\mathbf{x}) = \begin{bmatrix} 100 \\ -10 \\ 70 \end{bmatrix}$$

- Suppose our classifier predicts class c^* for \mathbf{x} , i.e. $c^* = \arg \max_c F(\mathbf{x})_c$
- We call $m_t = F(\mathbf{x})_{c^*} - F(\mathbf{x})_t$ the classification margin of classes c^* and t .
- Worst-case margin:**

Handwritten diagram for $c^* = 1, t = 2$:

$$F(\tilde{\mathbf{x}}) = \begin{bmatrix} 80 \\ 90 \\ 10 \end{bmatrix}$$

$$m_t^* = \min_{\tilde{\mathbf{x}}} F(\tilde{\mathbf{x}})_{c^*} - F(\tilde{\mathbf{x}})_t$$

subject to $\|\tilde{\mathbf{x}} - \mathbf{x}\|_p \leq \epsilon$

Handwritten diagram for $c^* = 1, t = 3$:

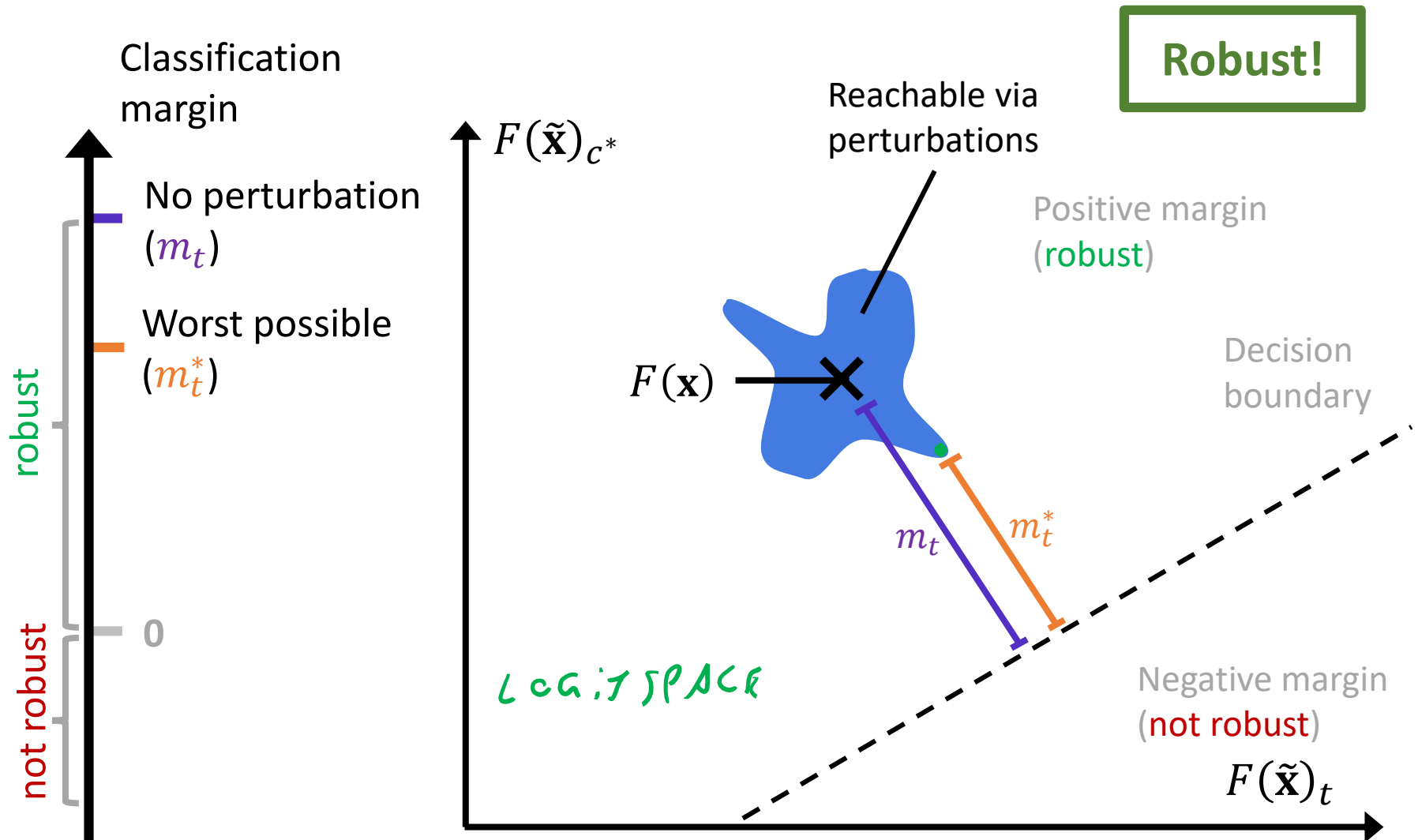
$$F(\tilde{\mathbf{x}}') = \begin{bmatrix} 90 \\ -10 \\ 80 \end{bmatrix}$$

- $m_t^* > 0$: the classifier's prediction cannot be changed from class c^* to t
- If for **all classes** $t \neq c^*$ we have $m_t^* > 0 \rightarrow$ we can certify robustness
- If for **any class** $t \neq c^*$ we have $m_t^* < 0 \rightarrow$ there exists an adversarial example $\tilde{\mathbf{x}} \in \mathcal{P}_{\epsilon,p}(\mathbf{x})$

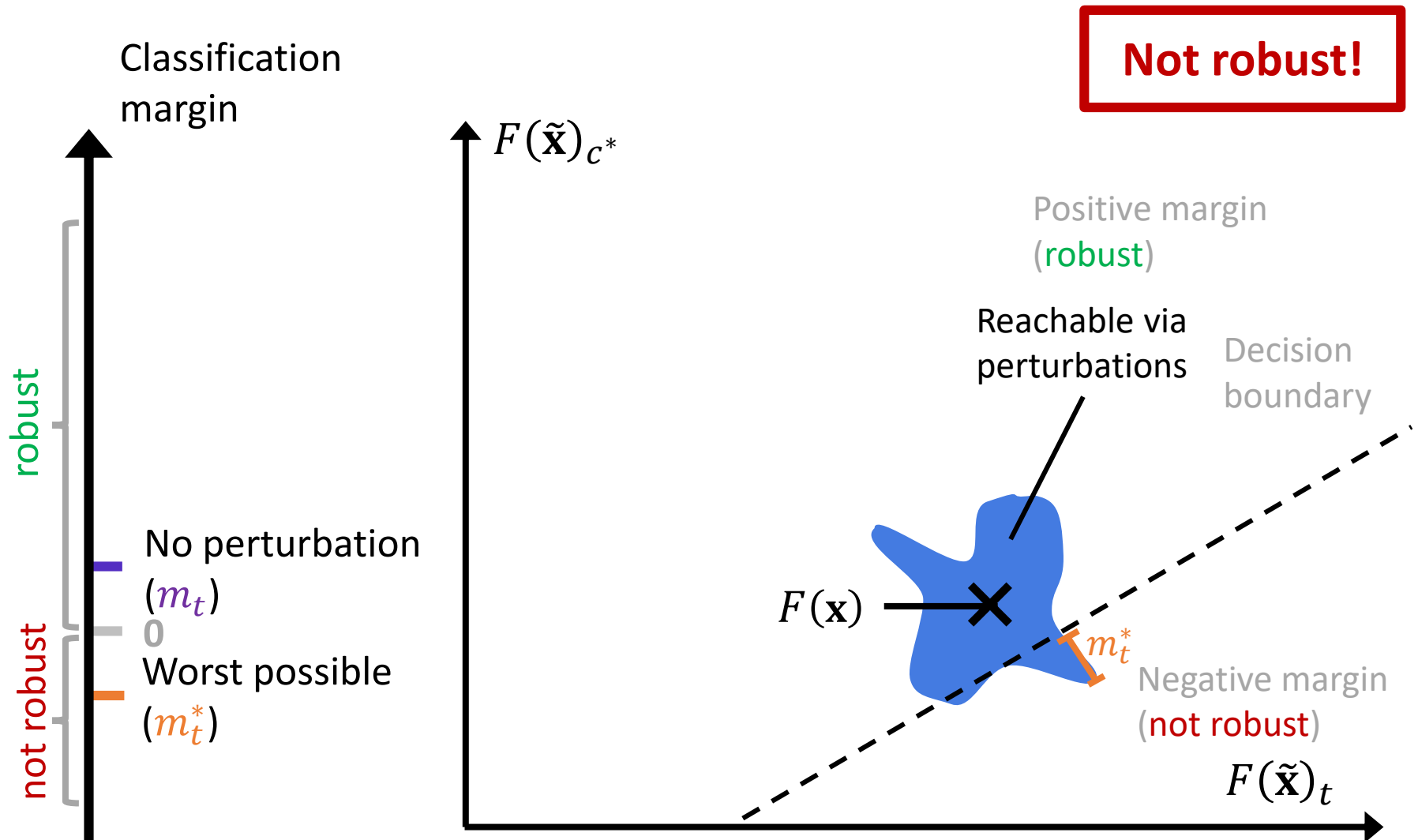
input
SPACE



Exact Robustness Certification: Illustration



Exact Robustness Certification: Illustration

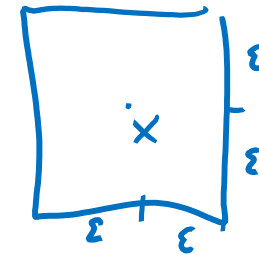


Expressing Exact Certification as MILP: L_p Constraints

- L_∞ constraints are straightforward to include in linear programs
- We add the following constraints to the optimization:

$$\begin{aligned} \mathbf{x}_i - \tilde{\mathbf{x}}_i &\leq \epsilon \quad \forall i \\ \tilde{\mathbf{x}}_i - \mathbf{x}_i &\leq \epsilon \quad \forall i \end{aligned}$$

↑ in all dimension

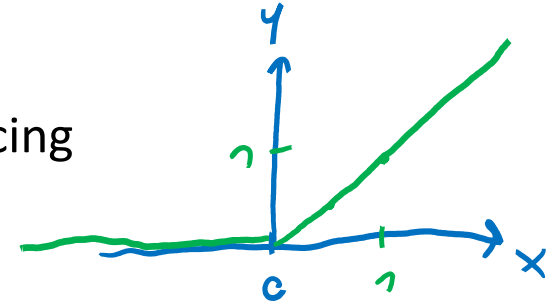


- L_1 constraints are also straightforward to encode
- L_2 constraints can be captured using mixed integer quadratic programming

Expressing Exact Certification as MILP: ReLU (1)

$$\rightarrow * \rightarrow a = 1 \rightarrow y \leq x \wedge y \geq 0 \wedge y \geq x \Rightarrow y = x$$

- The ReLU activation function is where most of the difficulty comes from.
- We want to encode $y = \text{ReLU}(x)$
- Naively, we can encode the ReLU activation by introducing a binary variable (vector of variables) a :



$$\boxed{y_i \leq a_i \cdot x_i} \text{ and } y_i \geq 0 \text{ and } y_i \geq x_i$$

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

- However, now we have a constraint with a product of two variables, hence, is not linear and not convex.

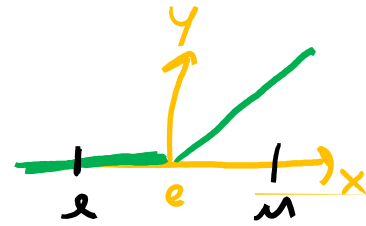
$$x < 0 \rightarrow a = 0 \rightarrow y \leq 0 \wedge y \geq 0 \wedge y \geq x \Rightarrow y = 0$$

$$[y \wedge a = 1 \rightarrow y \leq x \wedge y \geq 0] \downarrow$$

$$x > 0 \rightarrow [a = 0 \rightarrow y \leq 0 \wedge y \geq 0 \wedge y \geq x > 0] \downarrow$$

$\rightarrow *$

Expressing Exact Certification as MILP: ReLU (2)



- Suppose we have lower and upper bounds $[l, u]$ on the input x to the ReLU activation.
- Then, we can encode the ReLU activation using linear and integer constraints [Tjeng et al. 2019]):

$$y_i = \text{ReLU}(x_i) \Leftrightarrow \begin{aligned} & (y_i \leq x_i - l_i(1 - a_i)) \\ & \wedge (y_i \leq a_i \cdot u_i) \\ & \wedge (y_i \geq x_i) \\ & \wedge (y_i \geq 0) \\ & \wedge (a_i \in \{0, 1\}) \end{aligned} \quad \forall i$$

$$l \leq x \leq u$$

$$\begin{aligned} x > 0 & \rightarrow \left[a = 1 \rightarrow y \leq x - l \wedge y \leq 0 \wedge y \geq x \wedge y \geq 0 \right] \downarrow \\ & \rightarrow a = 1 \rightarrow y \leq x \wedge y \leq u \wedge y \geq x \wedge y \geq 0 \Rightarrow y = x \\ x < 0 & \rightarrow a = 0 \rightarrow y \leq \underbrace{x - l}_{\geq 0} \wedge y \leq 0 \wedge y \geq x \wedge y \geq 0 \Rightarrow y = 0 \\ & \quad y[a = 1] \downarrow \end{aligned}$$

Overall MILP

- Note that the overall MILP optimizes over multiple variables
 - Efficient solvers will remove some of them due to the equality constraints

$$m_t^* = \min_{\tilde{\mathbf{x}}, \mathbf{y}^{(l)}, \mathbf{x}^{(l)}, \mathbf{a}^{(l)}} [\hat{\mathbf{x}}^{(L)}]_{c^*} - [\hat{\mathbf{x}}^{(L)}]_t$$

subject to

$$\begin{aligned} \mathbf{x}_i - \tilde{\mathbf{x}}_i &\leq \epsilon \quad \forall i \\ \tilde{\mathbf{x}}_i - \mathbf{x}_i &\leq \epsilon \quad \forall i \end{aligned}$$



$$\mathbf{y}^{(0)} = \tilde{\mathbf{x}}$$

$$\hat{\mathbf{x}}^{(l)} = \mathbf{W}_l \mathbf{y}^{(l-1)} + \mathbf{b}_l \quad \forall l = 1 \dots L$$

Handwritten green notes: $y_i^{(l)} = \text{ReLU}(\hat{x}_i^{(l)})$

$$\left\{ \begin{aligned} y_i^{(l)} &\leq \hat{x}_i^{(l)} - l_i^{(l)} (1 - a_i^{(l)}) \\ y_i^{(l)} &\geq \hat{x}_i^{(l)} \\ y_i^{(l)} &\leq u_i^{(l)} \cdot a_i^{(l)} \\ y_i^{(l)} &\geq 0 \\ a_i^{(l)} &\in \{0, 1\} \end{aligned} \right. \quad \forall l = 1 \dots L - 1 \quad \forall i$$

- Remark: We can also handle all classes $t \neq c^*$ at the same time in a single MILP

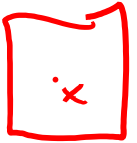
On the Lower and Upper Bounds

$$l^{(x)} \leq \hat{x}^{(x)} \leq u^{(x)}$$

- The MILP formulation relies on being able to compute lower and upper bounds $[l^{(l)}, u^{(l)}]$ on the input $\hat{\mathbf{x}}^{(l)}$ to the ReLU activation (for every layer l)
- One simple way to get (loose) lower and upper bounds is interval arithmetic:

$$u = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad l = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\begin{aligned} u^{(l)} &= [W_l]_+ u^{(l-1)} - [W_l]_- l^{(l-1)} + b_l \\ l^{(l)} &= [W_l]_+ l^{(l-1)} - [W_l]_- u^{(l-1)} + b_l \end{aligned}$$



L_p constraints

$$u_i^{(0)} = x_i + \epsilon$$

$$l_i^{(0)} = x_i - \epsilon$$

- Here, $[W]_+ = \max \{W, 0\}$, $[W]_- = \max \{-W, 0\}$

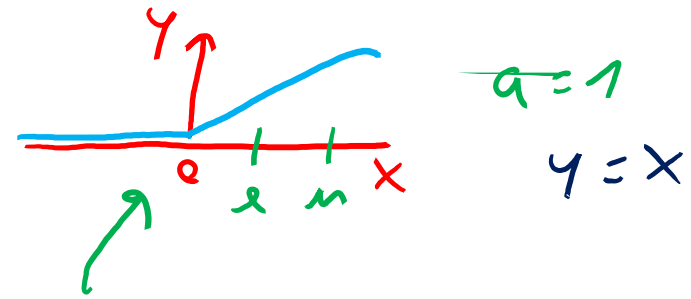
$$Wx + b$$

$$w = \begin{bmatrix} -1 & 5 \\ 2 & -7 \end{bmatrix}$$

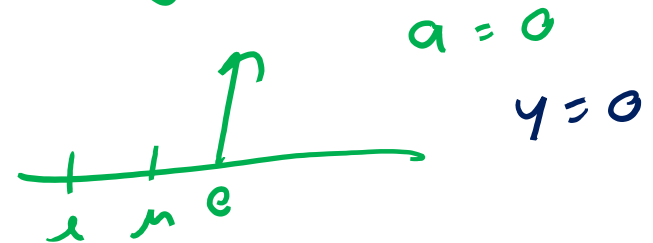
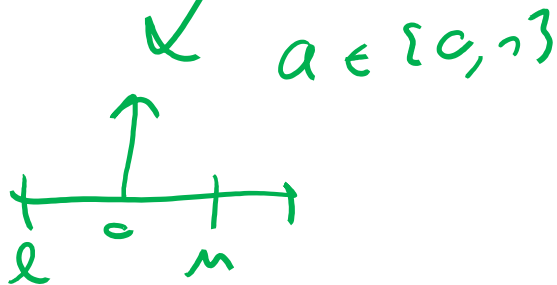
$$[w]_+ = \begin{bmatrix} 0 & 5 \\ 2 & 0 \end{bmatrix}$$

$$[w]_- = \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$$

Stable and Unstable Units



- Units for which $\mathbf{u}_i^{(l)} \geq \mathbf{l}_i^{(l)} \geq 0$ are called *stably active*
 - Units for which $0 \geq \mathbf{u}_i^{(l)} \geq \mathbf{l}_i^{(l)}$ are called *stably inactive*
 - Units for which $\mathbf{u}_i^{(l)} \geq 0 \geq \mathbf{l}_i^{(l)}$ are called *unstable*
- Can be removed from the optimization
- While the tightness of the upper and lower bounds has no influence on the correctness of the result, **tighter bounds** lead to **more stable units** and greatly **speed up** the optimization.



Summary

- Exact verification is possible for ReLU-Networks
 - However, expensive for large neural networks
- Can we find more efficient certificates?
 - Unfortunately not if we focus on exact certificates ("if and only if") due to the NP-hardness (assuming $P \neq NP$)
- However, we can change the allowed answers to our question

“Is the classifier f_θ around the sample \mathbf{x} adversarial-free
(within an ϵ -ball measured by some norm)?”
- If the algorithm returns **YES**, there are no adversarial examples within an ϵ ball around the input sample; if the algorithm returns **POTENTIALLY NOT** there might be adversarial examples or it is adversarial-free
- This is a conservative (careful) answer, i.e. in cases where the algorithms says “yes” we can rely on the prediction → i.e. we still have a guarantee
- We discuss such principles in the next sections!

Recommended Reading

- Lecture 12: Certified defenses I: Exact certification of Jerry Li's course on Robustness in Machine Learning (CSE 599-M), <https://jerryzli.github.io/robust-ml-fall19.html>

References – Exact Verification

- Katz, Guy, et al. "Reluplex: An efficient SMT solver for verifying deep neural networks." *International Conference on Computer Aided Verification*. Springer, Cham, 2017.
- Tjeng, Vincent, et al. "Evaluating Robustness of Neural Networks with Mixed Integer Programming." *International Conference on Learning Representations*, 2019