Building upon MIP and ``classical'' optimization techniques to learn robust deep neural networks

- MIP and OR for AI

Builds on joint work with Ruth Misener, Calvin Tsay, Alexander Thebelt, and Shudian Zhao

CO@Work 2024
ZIB Berlin

Jan Kronqvist

Optimization and Systems Theory, Department of Mathematics
KTH Royal Institute of Technology, Stockholm, Sweden; jankr@kth.se

IN THIS LECTURE

» How to use MIP to **find adversarial examples** of a deep neural network (DNN).
» How to utilize a few adversarial examples to **fine-tune** a DNN.
» My goals for this lecture:

IN THIS LECTURE

- How to use MIP to **find adversarial examples** of a deep neural network (DNN).
- How to utilize a few adversarial examples to **fine-tune** a DNN.
- My goals for this lecture:
  - Show that classical optimization (MIP and nonlinear programming) is also highly relevant for AI.
  - Show that by simple techniques, we can **greatly improve robustness**.
  - Give you an introduction to the areas and present you with some **open challenges**!

THE LECTURE IS BASED ON

1. **Embedding ReLU DNNs in MIPs**
   Collaboration with Ruth Misener, Calvin Tsay, and Alexander Thebelt at Imperial College London.
   - » Based on the papers
     - » Partition-based formulations for mixed-integer optimization of trained ReLU neural networks, NeurIPS 2021, Tsay, Kronqvist, Thebelt, and Misener
       [click to read paper]
     - » P-split formulations: A class of intermediate formulations between big-M and convex hull for disjunctive constraints, ArXiv 2024, Kronqvist, Misener, and Tsay
       [click to read paper]
2. **Robust training**
   Collaboration with Shudian Zhao (Postdoc at KTH)
   - » Based on the papers
     - » A constrained optimization approach to improve robustness of neural networks, Pre-print 2024, Zhao, Kronqvist
       [click to read paper]

ADVERTISEMENTS

- There will be a **MIP computational competition** 2025! An official announcement and details are coming soon.
  - A chance to make it into the MIP hall of fame. ☺

ADVERTISEMENTS

» There will be a **MIP computational competition** 2025! An official announcement and details are coming soon.
  » A chance to make it into the MIP hall of fame. ☺

» I'm often searching for new postdocs and PhD students. Email me *jankr@kth.se*

ADVERTISEMENTS

- There will be a **MIP computational competition** 2025! An official announcement and details are coming soon.
  - A chance to make it into the MIP hall of fame. ☺

- I'm often searching for new postdocs and PhD students. Email me *jankr@kth.se*

- Check out the MINLP solver SHOT **https://shotsolver.dev/shot**

BACKGROUND AND MOTIVATION

WHAT IS ROBUSTNESS ABOUT AND WHY SHOULD YOU CARE

Deep neural nets (DNNs) are known to be sensitive to adversarial perturbations.
– This means that **DNNs are often easily fooled!**

WHAT IS ROBUSTNESS ABOUT AND WHY SHOULD YOU CARE

Deep neural nets (DNNs) are known to be sensitive to adversarial perturbations.
– This means that **DNNs are often easily fooled!**

Our awesome DNN that we trained so well

Classification: Cat

Small change to the picture
(not visible to the human eye)

Our awesome DNN that we trained so well

Classification: Car

This sensitivity is a huge risk in real-world applications!

Example: Eykholt et al. showed that you can successfully change the classification of traffic sign by adding “graffiti” to the sign

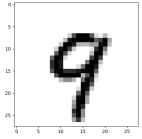
Figure: Misclassified traffic signs by adversarial perturbations. Images from Eykholt et al. [2].

Recommended reading

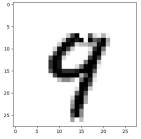
1. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In International Conference on Learning Representations (ICLR).
2. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., ... & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE conference on computer vision and pattern recognition.

» We don't want the DNN's classification of an image x_{image} to change if we make a small perturbation δ to the image.

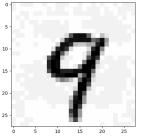
» For a given metric $\|\cdot\|$ and perturbation radius ε, we want $x_{\text{image}} + \delta$ to have the same classification for all $\|\delta\| \leq \varepsilon$.



(a) Original image, classified as 9



(b) Classified as 4, $\|\delta\|_1 \leq 4$



(c) Classified as 4, $\|\delta\|_\infty \leq 0.05$

Let's formally define what we mean by robust.

» Assume we are given a set of training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$, with inputs x_i and correct labels y_i.

Let's formally define what we mean by robust.

» Assume we are given a set of training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$, with inputs x_i and correct labels y_i.

» We have trained a DNN that correctly classifies a subset of the training data $\mathcal{X}_{\text{corr}} \subset \mathcal{X}_{\text{train}}$.

Let's formally define what we mean by robust.

» Assume we are given a set of training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$, with inputs x_i and correct labels y_i.

» We have trained a DNN that correctly classifies a subset of the training data $\mathcal{X}_{\text{corr}} \subset \mathcal{X}_{\text{train}}$.

Definition

We say that the DNN is robust, with a given metric $\|\cdot\|$ and perturbation radius ε, if $x_i + \delta$ is classified as y_i for all $(x_i, y_i) \in \mathcal{X}_{\text{corr}}$ and for all $\|\delta\| \leq \epsilon$.

Let's formally define what we mean by robust.

» Assume we are given a set of training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$, with inputs x_i and correct labels y_i.

» We have trained a DNN that correctly classifies a subset of the training data $\mathcal{X}_{\text{corr}} \subset \mathcal{X}_{\text{train}}$.

Definition

We say that the DNN is robust, with a given metric $\|\cdot\|$ and perturbation radius ε, if $x_i + \delta$ is classified as y_i for all $(x_i, y_i) \in \mathcal{X}_{\text{corr}}$ and for all $\|\delta\| \leq \epsilon$.

How to check if a DNN is robust?

Let's formally define what we mean by robust.

» Assume we are given a set of training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$, with inputs x_i and correct labels y_i.

» We have trained a DNN that correctly classifies a subset of the training data $\mathcal{X}_{\text{corr}} \subset \mathcal{X}_{\text{train}}$.

Definition

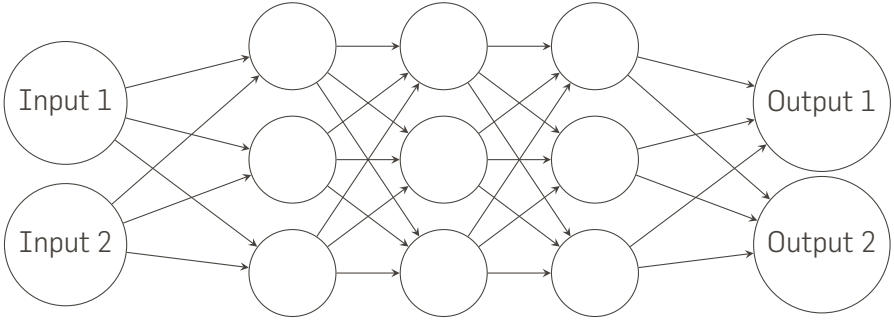
We say that the DNN is robust, with a given metric $\|\cdot\|$ and perturbation radius ε, if $x_i + \delta$ is classified as y_i for all $(x_i, y_i) \in \mathcal{X}_{\text{corr}}$ and for all $\|\delta\| \leq \epsilon$.

How to check if a DNN is robust?

We can use MIP to verify if the DNN is robust and to find perturbations δ that result in misclassifications.

» MIP to the rescue!

ENCODING A DEEP NEURAL NETWORK AS S MIP



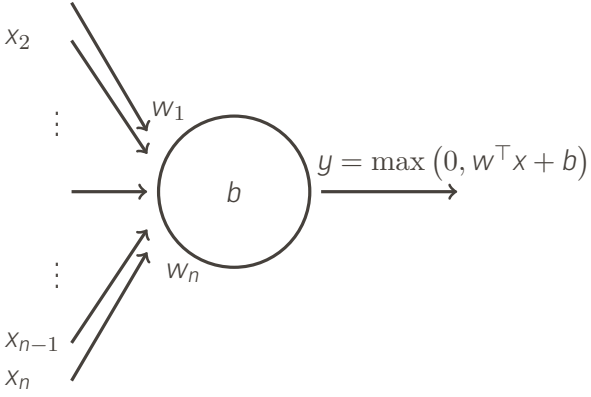
We are focusing on the **ReLU activation** function as it is MIP representable.

» To verify robustness or find adversarial perturbations, we want to optimize over the inputs to produce a certain output.

» For example, maximize (Output 1 - Output 2).

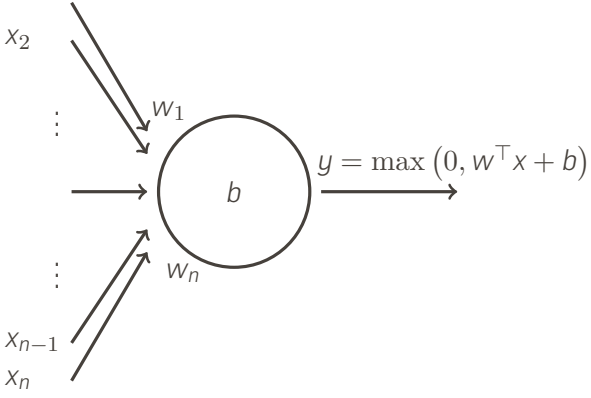
LET'S FOCUS ON A SINGLE NODE IN THE DNN

Consider a single node in the DNN.



LET'S FOCUS ON A SINGLE NODE IN THE DNN

Consider a single node in the DNN.



We can represent the node by the disjunctive constraint

$$\begin{bmatrix} y = 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y = w^\top x + b \end{bmatrix}$$

- Let's see how we can write this in MIP form.

MIP OF A RELU

We have the disjunctive constraint

$$\begin{bmatrix} y = 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y = w^\top x + b \end{bmatrix}.$$

MIP OF A RELU

We have the disjunctive constraint

$$\begin{bmatrix} y = 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y = w^\top x + b \end{bmatrix}.$$

Let's rewrite the disjunctive constraint slightly

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix}.$$

We have the disjunctive constraint

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix}.$$

Let's **introduce two binary variables** σ_0 **and** σ_1. We use these as "indicators" if $\sigma_0 = 1$, then the first set of constraints must be satisfied, and with $\sigma_1 = 1$, the other set must be satisfied.

- We must **enforce one set of constraints and relax the other** set. We use the big-M approach.
- We need the largest value y can take for any input x that we care about. We denote this by y_{max}, and y_{min} as the smallest value $w^\top x + b$ can take.

MIP OF A RELU

We can now represent the disjunctive constraint

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix},$$

by

We can now represent the disjunctive constraint

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix},$$

by

$$y \leq y_{\mathsf{max}}(1 - \sigma_0), \tag{1}$$
$$y \geq 0, \tag{2}$$
$$w^\top x + b \leq y_{\mathsf{max}}(1 - \sigma_0), \tag{3}$$
$$y \geq 0, \tag{4}$$
$$y \leq w^\top x + b - y_{\mathsf{min}}(1 - \sigma_1), \tag{5}$$
$$y \geq w^\top x + b, \tag{6}$$
$$\sigma_0 + \sigma_1 = 1, \tag{7}$$
$$\sigma_0, \sigma_1 \in \{0, 1\}. \tag{8}$$

We can now represent the disjunctive constraint

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix},$$

by

$$y \leq y_{\text{max}}(1 - \sigma_0), \tag{1}$$
$$y \geq 0, \tag{2}$$
$$w^\top x + b \leq y_{\text{max}}(1 - \sigma_0), \tag{3}$$
$$y \geq 0, \tag{4}$$
$$y \leq w^\top x + b - y_{\text{min}}(1 - \sigma_1), \tag{5}$$
$$y \geq w^\top x + b, \tag{6}$$
$$\sigma_0 + \sigma_1 = 1, \tag{7}$$
$$\sigma_0, \sigma_1 \in \{0, 1\}. \tag{8}$$

If we substitute in $\sigma_0 = 1 - \sigma_1$ and remove some redundant constraints, we get

We can now represent the disjunctive constraint

$$\begin{bmatrix} y \leq 0 \\ y \geq 0 \\ w^\top x + b \leq 0 \end{bmatrix} \vee \begin{bmatrix} y \geq 0 \\ y \leq w^\top x + b \\ y \geq w^\top x + b \end{bmatrix},$$

by

$$y \geq 0, \tag{9}$$
$$y \leq y_{\max}\sigma_1, \tag{10}$$
$$y \leq w^\top x + b - y_{\min}(1 - \sigma_1), \tag{11}$$
$$y \geq w^\top x + b, \tag{12}$$
$$\sigma_1 \in \{0, 1\}. \tag{13}$$

This is the so-called Big-M formulation that was presented in

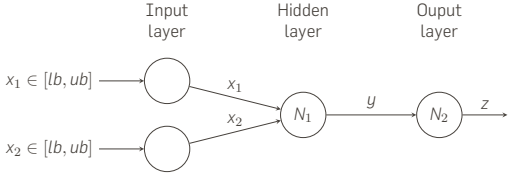
» Lomuscio, A. and Maganti, L. An approach to reachability analysis for feed-forward ReLU neural networks. arXiv preprint arXiv:1706.07351, 2017

» Fischetti, M. and Jo J.. Deep neural networks and mixed integer linear optimization. Constraints, 23(3):296–309, 2018.

Takeaway:
We can represent any node with a ReLU activation function in a DNN by

$$y \geq 0, \tag{9}$$
$$y \leq y_{\max}\sigma_1, \tag{10}$$
$$y \leq w^\top x + b - y_{\min}(1 - \sigma_1), \tag{11}$$
$$y \geq w^\top x + b, \tag{12}$$
$$\sigma_1 \in \{0, 1\}. \tag{13}$$

Consider the small neural network



The input-output relation of the network is given by

$$y = \max\{f_1(x),\ 0\},$$
$$z = \max\{f_2(y_1),\ 0\},$$

where $f_1(x) = w_1x_1 + w_2x_2 + b_1,\ f_2(x) = w_3y_1 + b_2$.

We have the parameters: $w_1 = 5$, $w_2 = -4$, $b_1 = 3$, $w_3 = -2$, $b_2 = 50$, and $x_1 \in [-10, 10]$, $x_2 \in [-5, 5]$.

EXAMPLE CONT.

For node N_1 the output is given by

$$y = \max\{5x_1 - 4x_2 + 3,\ 0\},$$

and we know $x_1 \in [-10, 10]$, $x_2 \in [-5, 5]$.

EXAMPLE CONT.

For node N_1 the output is given by

$$y = \max\{5x_1 - 4x_2 + 3,\ 0\},$$

and we know $x_1 \in [-10, 10]$, $x_2 \in [-5, 5]$.
We get y_{max} and y_{min} by simple bound propagation

$$y_{\text{min}} = 5(-10) - 4(5) + 3 = -67,$$
$$y_{\text{max}} = 5(10) - 4(-5) + 3 = 73.$$

EXAMPLE CONT.

For node N_1 the output is given by

$$y = \max\{5x_1 - 4x_2 + 3,\ 0\},$$

and we know $x_1 \in [-10, 10]$, $x_2 \in [-5, 5]$.
We get y_{max} and y_{min} by simple bound propagation

$$y_{\text{min}} = 5(-10) - 4(5) + 3 = -67,$$
$$y_{\text{max}} = 5(10) - 4(-5) + 3 = 73.$$

We plug this into the big-M constraints and get

$$\begin{aligned}
&y \geq 0,\\
&y \leq 73\sigma_1,\\
&y \leq 5x_1 - 4x_2 + 3 + 67(1 - \sigma_1),\\
&y \geq 5x_1 - 4x_2 + 3,\\
&\sigma_1 \in \{0, 1\}.
\end{aligned}$$

EXAMPLE CONT.

For node N_2 the output is given by

$$z = \max\left\{-2y + 50,\ 0\right\},$$

and we know $y \in [0, 73]$. The bounds on z are given simply by

$$z_{\text{min}} = -2(-73) + 50 = -96,$$
$$z_{\text{max}} = -2(0) + 50 = 50.$$

EXAMPLE CONT.

For node N_2 the output is given by

$$z = \max\{-2y + 50,\ 0\},$$

and we know $y \in [0, 73]$. The bounds on z are given simply by

$$z_{\text{min}} = -2(-73) + 50 = -96,$$
$$z_{\text{max}} = -2(0) + 50 = 50.$$

We plug this into the big-M constraints and get

$$\begin{aligned}
&z \geq 0,\\
&z \leq 50\sigma_2,\\
&z \leq -2y + 50 + 96(1 - \sigma_2),\\
&z \geq -2y + 50,\\
&\sigma_2 \in \{0, 1\}.
\end{aligned}$$

EXAMPLE CONT.

We can now, for example, find the inputs x_1 and x_2 that maximize the output by solving the MILP

$$
\begin{aligned}
\max \; & z \\
\text{s.t.} \; & y \geq 0, \\
& y \leq 73\sigma_1, \\
& y \leq 5x_1 - 4x_2 + 3 + 67(1 - \sigma_1), \\
& y \geq 5x_1 - 4x_2 + 3, \\
& z \geq 0, \\
& z \leq 50\sigma_2, \\
& z \leq -2y + 50 + 96(1 - \sigma_2), \\
& z \geq -2y + 50, \\
& \sigma_1 \in \{0, 1\} \\
& \sigma_2 \in \{0, 1\} \\
& x_1 \in [-10, 10], \; x_2 \in [-5, 5].
\end{aligned}
$$

» The big-M formulation of ReLU neural networks is simple and works well for relatively small networks (a few hundred nodes).

» The big-M formulation of ReLU neural networks is simple and works well for relatively small networks (a few hundred nodes).

» The problem of finding an adversarial perturbation to image x_{img} can be formulated as

$$\begin{aligned} \text{maximize} \quad & \text{misclassification} && (14)\\ \text{s.t.} \quad & \text{big-M constraints for each RelU node,} && (15)\\ & x_{\text{input}} = x_{\text{img}} + \delta, && (16)\\ & \|\delta\| \leq \varepsilon. && (17) \end{aligned}$$

» The big-M formulation of ReLU neural networks is simple and works well for relatively small networks (a few hundred nodes).

» The problem of finding an adversarial perturbation to image x_{img} can be formulated as

$$\begin{aligned} \text{maximize} \quad & \text{misclassification} && (14)\\ \text{s.t.} \quad & \text{big-M constraints for each RelU node,} && (15)\\ & x_{\text{input}} = x_{\text{img}} + \delta, && (16)\\ & \|\delta\| \leq \varepsilon. && (17) \end{aligned}$$

» The resulting MIP problems are difficult to solve and often have a very weak continuous relaxation.

CONTINUOUS RELAXATION OF THE TWO NODE EXAMPLE

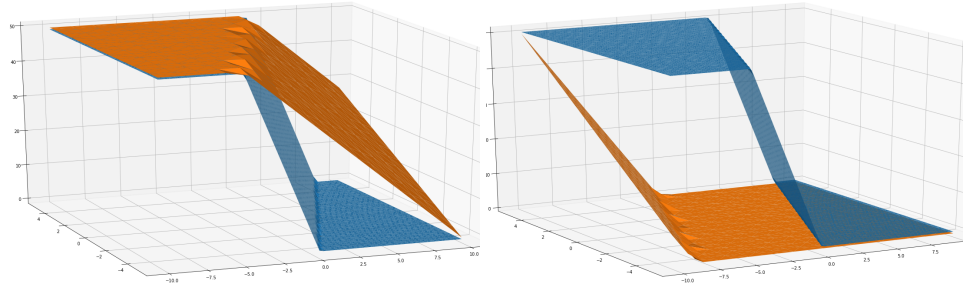


Figure: Upper and lower bounds of the network's output for the continuous relaxation of our two node example. The blue surface represents the output of the neural network.

The relaxations get increasingly worse with the network's number of nodes and layers.

CONTINUOUS RELAXATION OF THE TWO NODE EXAMPLE

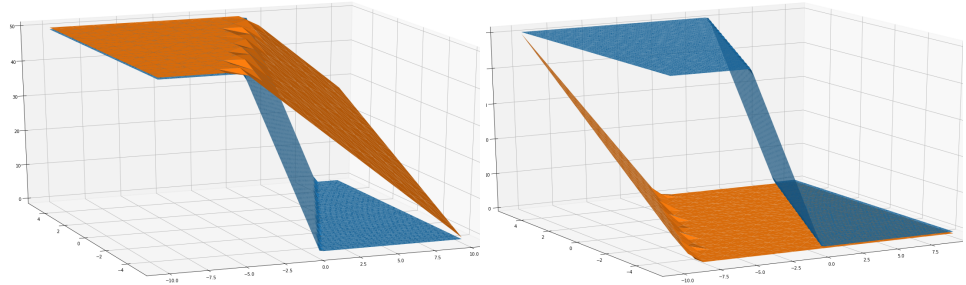


Figure: Upper and lower bounds of the network's output for the continuous relaxation of our two node example. The blue surface represents the output of the neural network.

The relaxations get increasingly worse with the network's number of nodes and layers.

The big-M formulation of DNNs works because of the **magic of MIP solvers**.

IMPROVED FORMULATIONS

» Convex hull formulations (or cuts) for individual nodes, presented in Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., & Vielma, J. P. (2020). Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming.

» Partitioned-based formulations presented in Tsay, C., Kronqvist, J., Thebelt, A., & Misener, R. (2021). Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. NeurIPS 34.

MIP FOR DNN SUMMARY

- We can embed ReLU DNNs into a MIP.
- We can use MIP to find adversarial perturbations (or so-called adversarial examples).
  - Computationally demanding (we are solving difficult MIPs)
  - We need to solve one MIP to find one adversarial example.
  - Finding lots of adversarial examples by MIP is not feasible.

IMPROVING ROBUSTNESS OF DEEP NEURAL NETS

FINE-TUNING TO IMPROVING ROBUSTNESS

Assume we have trained a DNN with weights $\hat{w}$, using the training data $\mathcal{X}_{\text{train}} = \{x_i, y_i\}_{i=1}^{k}$.

Suppose we are given a small set of adversarial examples $\mathcal{X}_{\text{adv}} = \{x_i, y_i\}_{i=1}^{l}$, (data points that will be misclassified due to a small perturbation).

How can we use $\mathcal{X}_{\text{adv}}$ to improve robustness?

- Directly adding $\mathcal{X}_{\text{adv}}$ to the training data will have very little impact if $\mathcal{X}_{\text{adv}}$ is small in comparison to $\mathcal{X}_{\text{train}}$. For example, for MNIST $|\mathcal{X}_{\text{train}}| = 60,000$ and we use $|\mathcal{X}_{\text{train}}| = 10 - 50$.
- Instead, we propose a nonlinear programming approach.

Some notation:

- W all parameters for the given DNN architecture (weights and biases). $\hat{W}$ initial parameters.
- $C(x; W)$: Predicted label of x with DNN parameters W.
- $\mathcal{X}_{\text{corr}} = \{x_i, y_i\}_{i=1}^{t} \subset \mathcal{X}_{\text{train}}$, for which x_i is correctly classified as y_i by the DNN using weights $\hat{W}$.
- Perturbation neighborhood $B_\epsilon(\bar{x}) := \{\tilde{x} \mid \|x - \tilde{x}\|_\infty \leq \epsilon\}$ around $\bar{x}$.
- $\ell(W, \mathcal{X}_{\text{train}})$ loss function evaluated over training data $\mathcal{X}_{\text{train}}$ with DNN parameters W.

Ideally, we would like to solve

$$\begin{aligned} &\min \|W - \hat{W}\|_2^2 \\ &\text{s.t. } C(\tilde{x}; W) = y_i, \quad \forall \tilde{x} \in B_\epsilon(x_i),\ \forall (x_i, y_i) \in \mathcal{X}_{\text{corr}} \\ &\qquad \ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}}). \end{aligned}$$

An optimal solution to the problem above would be robust! But, in practice, we can't solve it (highly nonlinear, nonconvex, and infinite number of constraints).

We consider a relaxed version with a finite number of constraints given by the adversarial examples $\mathcal{X}_{\text{adv}}$

$$\begin{aligned} &\min \|W - \hat{W}\|_2^2 \\ &\text{s. t. } C(x_i; W) = y_i, \quad \forall (x_i, y_i) \in \mathcal{X}_{\text{adv}} \\ &\qquad \ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}}). \end{aligned} \tag{NLP-fineT}$$

- Still a huge nonconvex problem ($\sim 200,000$ variables in our larger examples, with a relatively dense Hessian). There is no hope of finding/verifying a global optimal solution.
- We don't need an optimal solution. We are even happy with an almost feasible solution.

We consider a relaxed version with a finite number of constraints given by the adversarial examples $\mathcal{X}_{\text{adv}}$

$$\begin{aligned} &\min \|W - \hat{W}\|_2^2 \\ &\text{s.t. } C(x_i; W) = y_i, \quad \forall (x_i, y_i) \in \mathcal{X}_{\text{adv}} \\ &\qquad \ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}}). \end{aligned} \tag{NLP-fineT}$$

- » Still a huge nonconvex problem ($\sim 200,000$ variables in our larger examples, with a relatively dense Hessian). There is no hope of finding/verifying a global optimal solution.
- » We don't need an optimal solution. We are even happy with an almost feasible solution.
- » We developed an **iterative linearization approach**. There is room for improvement, but even this simple approach already improves robustness significantly.

We don't want constraints directly based on the classification function $C(\cdot; W) := \arg\max_{i \in \mathcal{I}} f_i(\cdot; w)$, where $\mathcal{I}$ contain the set of all labels.

We don't want constraints directly based on the classification function $C(\cdot; W) := \arg\max_{i \in \mathcal{I}} f_i(\cdot; w)$, where $\mathcal{I}$ contain the set of all labels.

For a pair of labels (i, j), input x is as likely, or more likely, to be labeled i than j if

$$f_i(x; W) \geq f_j(x; W), \tag{18}$$

where $f_i(\cdot; W)$ is the classification confidence of label i, typically the i-th output.

We don't want constraints directly based on the classification function $C(\cdot; W) := \arg\max_{i \in \mathcal{I}} f_i(\cdot; w)$, where $\mathcal{I}$ contain the set of all labels.

For a pair of labels (i, j), input x is as likely, or more likely, to be labeled i than j if

$$f_i(x; W) \geq f_j(x; W), \tag{18}$$

where $f_i(\cdot; W)$ is the classification confidence of label i, typically the i-th output.

Forcing the correct classification of $(\bar{x}_i, y_i) \in \mathcal{X}_{\text{adv}}$ is, thus, be achieved by the constraints

$$f_{y_i}(\bar{x}_i; W) \geq f_j(\bar{x}_i; W), \quad \forall j \in \mathcal{I} \setminus y_i \tag{19}$$

and we can add these constraints for all adversarial data points $(\bar{x}_i, y_i)$.

We want easier constraints to work with, so we linearize

$$f_{y_i}(\bar{x}_i; W) \geq f_j(\bar{x}_i; W), \quad \forall j \in \mathcal{I} \setminus y_i$$

with a first-order Taylor series expansion around the initial parameters W

$$f_{y_i}(\bar{x}_i; \hat{W}) - f_j(\bar{x}_i; \hat{W}) + (W - \hat{W})^\top \left(\nabla_w f_{y_i}(\bar{x}_i; \hat{W}) - \nabla_w f_j(\bar{x}_i; \hat{W}) \right) \geq 0, \ \forall j \in \mathcal{I} \setminus y_i. \quad (20)$$

We want easier constraints to work with, so we linearize

$$f_{y_i}(\bar{x}_i; W) \geq f_j(\bar{x}_i; W), \quad \forall j \in \mathcal{I} \setminus y_i$$

with a first-order Taylor series expansion around the initial parameters W

$$f_{y_i}(\bar{x}_i; \hat{W}) - f_j(\bar{x}_i; \hat{W}) + (W - \hat{W})^\top \left(\nabla_w f_{y_i}(\bar{x}_i; \hat{W}) - \nabla_w f_j(\bar{x}_i; \hat{W}) \right) \geq 0, \ \forall j \in \mathcal{I} \setminus y_i. \quad (20)$$

- We can generate cuts to promote the correct classification of adversarial data.
- Keep in mind, the constraints $f_{y_i}(\bar{x}_i; W) \geq f_j(\bar{x}_i; W)$ are nonconvex.

We can linearize the constraints $f_{y_i}(\bar{x}_I; W) \geq f_j(\bar{x}_i; W)$ with respect to both the parameters W and input x_i, and state that the inequality must hold for all x satisfying $\|x - \bar{x}_i\|_1 \leq \varepsilon_x$.

This results in the linearization

$$
\begin{aligned}
& f_{y_i}(\bar{x}, \hat{W}) - f_j(\bar{x}, \hat{W}) + (W - \hat{W})^\top (\nabla_w f_{y_i}(\bar{x}, \hat{W}) - \nabla_w f_j(\bar{x}, \hat{W})) \\
& \geq \max_{\|x - \bar{x}\|_1 \leq \epsilon_x} \{(x - \bar{x})^\top (\nabla_x f_j(\bar{x}, \hat{W}) - \nabla_x f_{y_i}(\bar{x}, \hat{W}))\},
\end{aligned}
$$

We can linearize the constraints $f_{y_i}(\bar{x}_I; W) \geq f_j(\bar{x}_i; W)$ with respect to both the parameters W and input x_i, and state that the inequality must hold for all x satisfying $\|x - \bar{x}_i\|_1 \leq \varepsilon_x$.

This results in the linearization

$$
\begin{aligned}
& f_{y_i}(\bar{x}, \hat{W}) - f_j(\bar{x}, \hat{W}) + (W - \hat{W})^\top (\nabla_w f_{y_i}(\bar{x}, \hat{W}) - \nabla_w f_j(\bar{x}, \hat{W})) \\
& \geq \max_{\|x - \bar{x}\|_1 \leq \epsilon_x} \{(x - \bar{x})^\top (\nabla_x f_j(\bar{x}, \hat{W}) - \nabla_x f_{y_i}(\bar{x}, \hat{W}))\},
\end{aligned}
$$

$$
\begin{aligned}
\implies & f_{y_i}(\bar{x}, \hat{W}) - f_j(\bar{x}, \hat{W}) + (W - \hat{W})^\top (\nabla_w f_{y_i}(\bar{x}, \hat{W}) - \nabla_w f_j(\bar{x}, \hat{W})) \geq \\
& \geq \epsilon_x \cdot \max_m |(\nabla_x f_j^m(\bar{x}, \hat{W}) - \nabla_x f_{y_i}^m(\bar{x}, \hat{W}))|, \quad (= \text{small constant})
\end{aligned}
$$

where $\nabla_x f_j^m(\bar{x}, \hat{W})$ refers to the m-th component of the vector $\nabla_x f_j(\bar{x}, \hat{W})$.

Similarly we linearize the constraint the no-worse loss constraint $\ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}})$ as

$$(W - \hat{W})^\top \nabla_w \ell(W, \mathcal{X}_{\text{train}}) \leq 0. \tag{21}$$

LINEARIZED FINE-TUNING PROBLEM

We can then form the linearized fine-tuning problem

$$
\begin{aligned}
&\min \|W - \hat{W}\|_2^2 \\
&\text{s. t. linearizations of } f_{y_i}(\bar{x}_{\mathcal{I}}; W) \geq f_j(\bar{x}_i; W) \quad \forall j \in \mathcal{I} \setminus y_i,\ \forall (\bar{x}_i, y_i) \in \mathcal{X}_{\text{adv}}, \quad \text{(QP-fineT)} \\
&\qquad \text{linearizations of } \ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}}).
\end{aligned}
$$

LINEARIZED FINE-TUNING PROBLEM

We can then form the linearized fine-tuning problem

$$
\begin{aligned}
&\min \|W - \hat{W}\|_2^2 \\
&\text{s. t. linearizations of } f_{y_i}(\bar{x}_{\mathcal{I}}; W) \geq f_j(\bar{x}_i; W) \quad \forall j \in \mathcal{I} \setminus y_i,\ \forall (\bar{x}_i, y_i) \in \mathcal{X}_{\text{adv}}, \quad \text{(QP-fineT)} \\
&\qquad \text{linearizations of } \ell(W, \mathcal{X}_{\text{train}}) \leq \ell(\hat{W}, \mathcal{X}_{\text{train}}).
\end{aligned}
$$

- » The minimizer of (QP-fineT) typically don't satisfy all nonlinear constraints.
- » We can refine the linearized problem by generating linearizations (cuts) at the minimizer of (QP-fineT).
- » We can't guarantee convergence (original problem nonconvex).

MAIN ALGORITHM

The main steps are

1. At iteration k, solve problem (QP-fineT) and store the minimizer as W^k.
2. Generate new linearizations at W^k and add them to problem (QP-fineT)
3. Increase iteration counter k and repeat.
4. Terminate on a maximal number of iterations.

SOLUTION REFINEMENT (LINE SEARCH)

The main algorithm gives candidate solutions $W^0, W^1, \ldots W^m$.

We further expand the set of candidate solutions by the rough line search

$$W(\alpha, k) := \alpha W^k + (1 - \alpha)\hat{W}, \forall \alpha = 0.1, \ldots, 1, \tag{22}$$

where $\hat{w}$ denote the initial parameter.

We use two criteria for evaluating the quality of the candidate solutions: 1) maximal violation of the constraints $f_{y_i}(\bar{x}_I; W) \geq f_j(\bar{x}_i; W)$, and 2) loss function on the training data $\ell(W, \mathcal{X}_{\text{train}})$.

- » Only Pareto optimal (efficient) solutions are interesting.
- » We choose the final solution by a weighted sum.

SOME NUMERICAL RESULTS

IMPLEMENTATION DETAILS

- Test instances:
  - MNIST: gray-scaled images of numbers 0 to 9, scaled pixels between 0 and 1.
  - CNN: 2 convolutional layers, 16×32 (stride of 2, max pooling layers to reduce half resolutions);
    2 fully connected layers stepping down to 1568×100, 100×10.
  - 162,710 parameters.
- Adversarial attackers (to evaluate robustness)
  - The fast gradient sign method (FGSM) by Goodfellow et al.: ℓ_∞-norm with radius 0.1.
  - The projected gradient descent method (PGD) by Madry et al.: run 50 iterations with a step size of 0.01, ℓ_∞-norm with radius 0.1

- Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). "Explaining and harnessing adversarial examples". In: arXiv preprint arXiv:1412.6572.
- Madry, Aleksander et al. (2017). "Towards deep learning models resistant to adversarial attacks". In: arXiv preprint arXiv:1706.06083.

NUMERICAL RESULTS FOR MNIST WITH CNN MODEL

$\lvert\mathcal{X}_{adv}\rvert$	ω	MNIST		Adversary acc.(%)	
		loss	acc.(%)	FGSM	PGD
0		0.0197	98.55	15.67	4.02
10		0.0273	98.39	59.48	43.14
	0	0.0354	98.19	59.49	43.76
	0.2	0.0269	98.46	61.54	45.73
	0.4	0.0269	**98.46**	**61.54**	**45.87**
50		0.0305	98.30	72.12	62.96
	0	0.0310	98.44	73.79	64.33
	0.2	0.0290	**98.46**	**74.77**	**65.94**
	0.4	0.0262	98.52	71.24	57.70

Table 1: CNN models found with various $\lvert\mathcal{X}_{adv}\rvert$, $M = 20$.

For the initial model, the accuracy after an adversarial attack 4 - 16%.
By only using 10 adversarial data points, we could increase the accuracy to 46 - 62%.

TAKE AWAY

» MIP can be used to check if a DNN is robust and to find adversarial examples.
» Classical optimization techniques can be used to greatly improve robustness.
» But, still plenty of room for improvement!

THANK YOU