

Machine Learning for Graphs and Sequential Data

Robustness of Machine Learning Models

Lecturer: Prof. Dr. Stephan Günnemann

cs.cit.tum.de/daml

Summer Term 2023

Roadmap

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

Lipschitz Continuity: Idea

- Roughly speaking, adversarial examples result from a **small change** in the input space (e.g. image) leading to a **large change** in the output space (logits).
- The idea of robustness via Lipschitz continuity is to bound how much the output of the network can change for a given perturbation of the input.
- More formally, a function $f: \mathcal{X} \rightarrow \mathcal{Y}$ is **Lipschitz continuous** if

$$\mathcal{D}_y(f(\mathbf{x}_1), f(\mathbf{x}_2)) \leq k \cdot \mathcal{D}_x(\mathbf{x}_1, \mathbf{x}_2) \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$$

- We refer to k as the **Lipschitz constant** of f , and to f as being **k-Lipschitz**.
- \mathcal{D}_x and \mathcal{D}_y are metrics on the input and output space, respectively (e.g., L_p norms).

Lipschitz Continuity for Robustness Certification

- Idea: determine (or bound) the Lipschitz constant of a neural network F .
- Then we can compute the **worst-case change** of the logits given a **bounded perturbation** of the input.
- How can we compute the Lipschitz constant $L(F)$?
- Given a function $f(\mathbf{x}) = (\phi_L \circ \phi_{L-1} \circ \dots \circ \phi_1)(\mathbf{x})$, the Lipschitz constant of f is bounded from above by:

$$L(f) \leq \prod_{l=1}^L L(\phi_l)$$

- Thus, we can compute the Lipschitz constants of the individual functions to upper bound the Lipschitz constant of f .

Lipschitz Continuity for Robustness Certification

- Given a function $f(\mathbf{x}) = (\phi_L \circ \phi_{L-1} \circ \dots \circ \phi_1)(\mathbf{x})$, the Lipschitz constant of f is bounded from above by:

$$L(f) \leq \prod_{l=1}^L L(\phi_l)$$

- Recall:** $F(\mathbf{x}) = \mathbf{W}_L f_{L-1} \circ f_{L-2} \circ \dots \circ f_1(\mathbf{x}) + \mathbf{b}_L$
- We can compute the Lipschitz constant of the **individual layers** to upper bound the **Lipschitz constant of the neural network F** .
- In the following, we thus need to learn how to compute the Lipschitz constant of different layers and activation functions.

Lipschitz Constant of Fully Connected Layers

- Fully connected layers consist of an affine transformation and an activation function. We first consider the affine transformation:

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b},$$

- Plugging $f(\mathbf{x})$ into the definition of the Lipschitz constant we get:

$$\|(\mathbf{W}\mathbf{x}_1 + \mathbf{b}) - (\mathbf{W}\mathbf{x}_2 + \mathbf{b})\|_p \leq k\|\mathbf{x}_1 - \mathbf{x}_2\|_p$$

- Setting $\mathbf{a} = \mathbf{x}_1 - \mathbf{x}_2$ and rearranging we obtain

$$L(f) = \sup_{\mathbf{a} \neq \mathbf{0}} \frac{\|\mathbf{W}\mathbf{a}\|_p}{\|\mathbf{a}\|_p},$$

which is the definition of the operator norm.

Lipschitz Constant of Fully Connected Layers

$$L(f) = \sup_{\mathbf{a} \neq \mathbf{0}} \frac{\|\mathbf{W}\mathbf{a}\|_p}{\|\mathbf{a}\|_p},$$

which is the definition of the operator norm.

- For $p = 1$, $L(f)$ is the maximum L_1 norm of the columns of \mathbf{W} .
- For $p = \infty$, $L(f)$ is the maximum L_1 norm of the rows of \mathbf{W} .
- For $p = 2$, $L(f)$ is the maximum singular value of \mathbf{W} .

Convolutional layers:

- Computing the Lipschitz constant of convolutional layers is a bit trickier in the case of $p = 2$ but can be approximated relatively efficiently [Gouk+ 2018].

Lipschitz Constant of Neural Networks

Activation functions:

- Typical activation functions (ReLU, sigmoid, tanh, softmax, ...) have a Lipschitz constant of at worst 1.
- Specifically, ReLU has a Lipschitz constant of exactly 1.
- For a feed-forward neural network F with ReLU activation functions we therefore upper bound its Lipschitz constant as:

$$L(F) \leq \prod_{l=1}^L \|W_l\|_p$$

// note again: this is the operator norm

- By regularizing the norm of the weight matrices we can therefore enforce any desired Lipschitz constant for the network.

Certifying Robustness via the Lipschitz Constant

- Suppose we have a neural network F with Lipschitz constant k .
- Given an input sample \mathbf{x} and corresponding logits $\hat{\mathbf{y}} = F(\mathbf{x})$ we can compute the maximum norm ϵ of a perturbation $\boldsymbol{\delta}$: $\|\boldsymbol{\delta}\|_p \leq \epsilon$ for which we can guarantee robustness of the classifier.

Approach:

1. Compute the minimum perturbation norm $\hat{\epsilon} = \|\hat{\boldsymbol{\delta}}\|_p$ in **logit space** to change the classification.
2. Compute the corresponding perturbation norm $\epsilon = \|\boldsymbol{\delta}\|_p$ in **input space** based on the **Lipschitz constant**.

Certifying Robustness via the Lipschitz Constant

Approach:

1. Compute the minimum perturbation norm $\hat{\epsilon} = \|\hat{\delta}\|_p$ in **logit space** to change the classification.

Illustration: see annotation

Determining the maximum certifiable Input Perturbation

2. Compute the corresponding perturbation norm $\epsilon = \|\delta\|_p$ in **input space** based on the **Lipschitz constant**.

Recall

- a) $\hat{\epsilon}$ is the minimum perturbation norm in **logit space**, that is we require $\|F(\mathbf{x}) - F(\tilde{\mathbf{x}})\|_p \leq \hat{\epsilon}$
- b) We know F is Lipschitz, that is $\|F(\mathbf{x}) - F(\tilde{\mathbf{x}})\|_p \leq k \cdot \|\mathbf{x} - \tilde{\mathbf{x}}\|_p$

Observation: As long as $\|\mathbf{x} - \tilde{\mathbf{x}}\|_p \leq \frac{\hat{\epsilon}}{k}$ we have $\stackrel{b)}{\Rightarrow} \|F(\mathbf{x}) - F(\tilde{\mathbf{x}})\|_p \leq k \cdot \frac{\hat{\epsilon}}{k} = \hat{\epsilon}$

\Rightarrow The maximum certifiable radius is $\epsilon = \frac{\hat{\epsilon}}{k}$

- ➔ We can certify that for any $\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x}) = \{\tilde{\mathbf{x}}: \|\mathbf{x} - \tilde{\mathbf{x}}\|_p \leq \frac{\hat{\epsilon}}{k}\}$ the predicted class for sample \mathbf{x} does not change.
- ➔ If the Lipschitz constant k of the network is small, we expect to get more robustness certificates.

Practical Considerations

- We have seen that, from a robustness point of view, we want our classifiers to have **small Lipschitz constant**.
- Taking this to the extreme, a Lipschitz constant of 0 would mean that it is impossible to change the prediction with perturbations of the input.
- However, small Lipschitz constants **limit the expressiveness** of the function F .
- Therefore, we need to **trade off expressiveness and robustness**.
- Moreover, [Huster+ 2018] argue that there are **theoretical limitations** in using only atomic (i.e. layer-wise) Lipschitz constants to upper bound k .

Questions – Robustness (III)

1. Suppose we define a **local** variant of the Lipschitz constant around a given point \mathbf{x}_0 as follows

$$\mathcal{D}_y(f(\mathbf{x}_0), f(\mathbf{x})) \leq k_{\mathbf{x}_0} \cdot \mathcal{D}_x(\mathbf{x}_0, \mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}$$

How does the local constant $k_{\mathbf{x}_0}$ relate to the global constant k of f ? Which one would provide better guarantees and why?

Recommended Reading

- Cisse, Moustapha, et al. "Parseval networks: Improving robustness to adversarial examples." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.

References

- Gouk, Henry, et al. "Regularisation of neural networks by enforcing lipschitz continuity." *arXiv preprint arXiv:1804.04368*(2018).
- Huster, Todd, Cho-Yu Jason Chiang, and Ritu Chadha. "Limitations of the Lipschitz constant as a defense against adversarial examples." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, Cham, 2018.