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

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**

Introduction

- The robustness certificates studied so far have in common that they try to prove that the predicted class for a given input does not change for some type of perturbations.

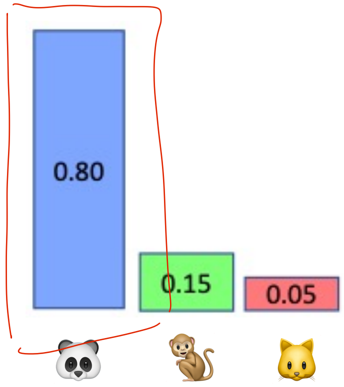
- The idea behind randomized smoothing is simple: we transform any given base classifier into a smoothed classifier by **randomly** adding noise (e.g. Gaussian) to the input and predicting the majority class given many samples.

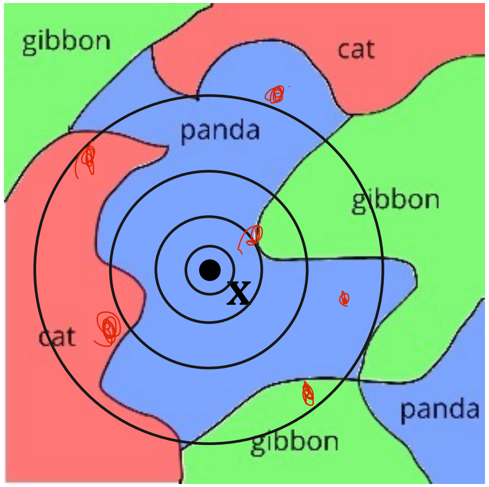
- We certify that the predictions of the resulting **smoothed** classifier do not change when the input is (adversarially) perturbed.

Graphical Overview

- The image shows the predictions of some base classifier f (e.g. a Neural Network) in different colors for any given input.
- Given a test input $\mathbf{x}$ we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$ and observe the output of the base classifier $f(\mathbf{x} + \boldsymbol{\epsilon})$.
- We notice that the **majority** of instances are still classified correctly, i.e. $f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐼 even though sometimes $f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐒 or $f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐱.
- By slightly moving the center point $\mathbf{x}$ (i.e. perturbing the original image), the probabilities of observing the different classes will only **change slowly**.

vote maximum





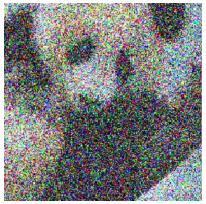
input space

Example and figures from:
Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

General Idea

- We smooth any Base classifier $f(\mathbf{x}) = \arg\max_{c \in \mathcal{Y}} F(\mathbf{x})_c$ into a smooth classifier g
  - e.g. $F(\mathbf{x})$ is the vector of logits returned by a Neural Network, $f(\mathbf{x})$ returns the class.

- $g(\mathbf{x})$ = the **most probable** prediction of f under random Gaussian noise.

- Example: consider input $\mathbf{x} =$ and noisy input $\mathbf{x} + \boldsymbol{\epsilon}$ = 

- Suppose when we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$ and evaluate $f\ (\mathbf{x} + \boldsymbol{\epsilon})$ it holds:
  - $\mathbb{P}_\epsilon\big(f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐼$\big) = 0.80$
  - $\mathbb{P}_\epsilon\big(f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐒$\big) = 0.15$
  - $\mathbb{P}_\epsilon\big(f(\mathbf{x} + \boldsymbol{\epsilon}) =$ 🐱$\big) = 0.05$

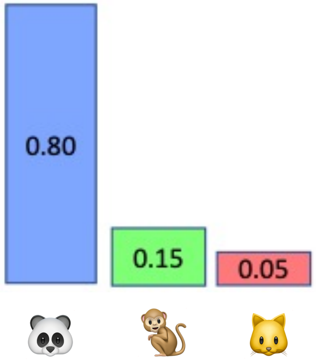
- Then $g(\mathbf{x}) =$ 🐼 i.e. g predicts the majority class when randomly sampling.

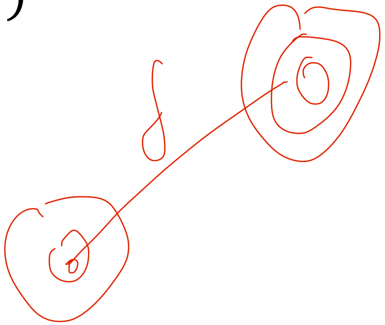
Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

Formal Definition of the Smoothed Classifier

- The output of the smoothed classifier $g(\mathbf{x})$ is a vector with entries

$$g(\mathbf{x})_c = \mathbb{P}_{\boldsymbol{\epsilon}}(f(\mathbf{x}+\boldsymbol{\epsilon}) = c) \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$





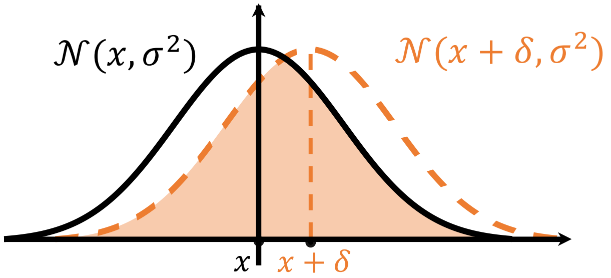
- Now denote with $c^* = \arg\max_c g(\mathbf{x})_c$ the most likely class and with $\boxed{p^*_{\mathbf{x}} = g(\mathbf{x})_{c^*}}$ the probability of observing c^* (In the example c^*= 🐼 and $\boxed{p^*_{\mathbf{x}} = 0.8}$)

- Goal: We want to certify that for any admissible perturbation $\boldsymbol{\delta}$ it holds

$$\boxed{\arg\max_c g(\mathbf{x}+\boldsymbol{\delta})_c = \boxed{c^*} \quad \text{for all } \|\boldsymbol{\delta}\|_2 \leq r}$$

Adervsian Perturbation

Towards Randomized Smoothing Certificates



Observation: When we **slightly perturb** x the samples from $\mathcal{N}(\boldsymbol{x},\sigma^2)$ and $\mathcal{N}(\boldsymbol{x}+\boldsymbol{\delta},\sigma^2)$ have a large overlap → $g(\boldsymbol{x}+\boldsymbol{\delta})$ and $g(\boldsymbol{x})$ produce similar output

Challenges:

1. We wish to certify arbitrary classifiers f without any further assumptions
2. We know $\|\boldsymbol{\delta}\|_2$ but can only evaluate $f(\boldsymbol{x}+\boldsymbol{\epsilon})$ and not $f(\boldsymbol{x}+\boldsymbol{\delta}+\boldsymbol{\epsilon})$

dont know δ

Solution: Assume the **worst-case** → consider the **worst-possible** classifier h^* that

- classifies $h^*(\boldsymbol{x}+\boldsymbol{\epsilon})$ with probability $p_{\mathbf{x}}^*$ (i.e. behaves like our actual classifier f around $\boldsymbol{x}$)
- but with the smallest possible probability to classify perturbed data $\boldsymbol{x}+\boldsymbol{\delta}$ as class c^*

Formally:

worst → not panda

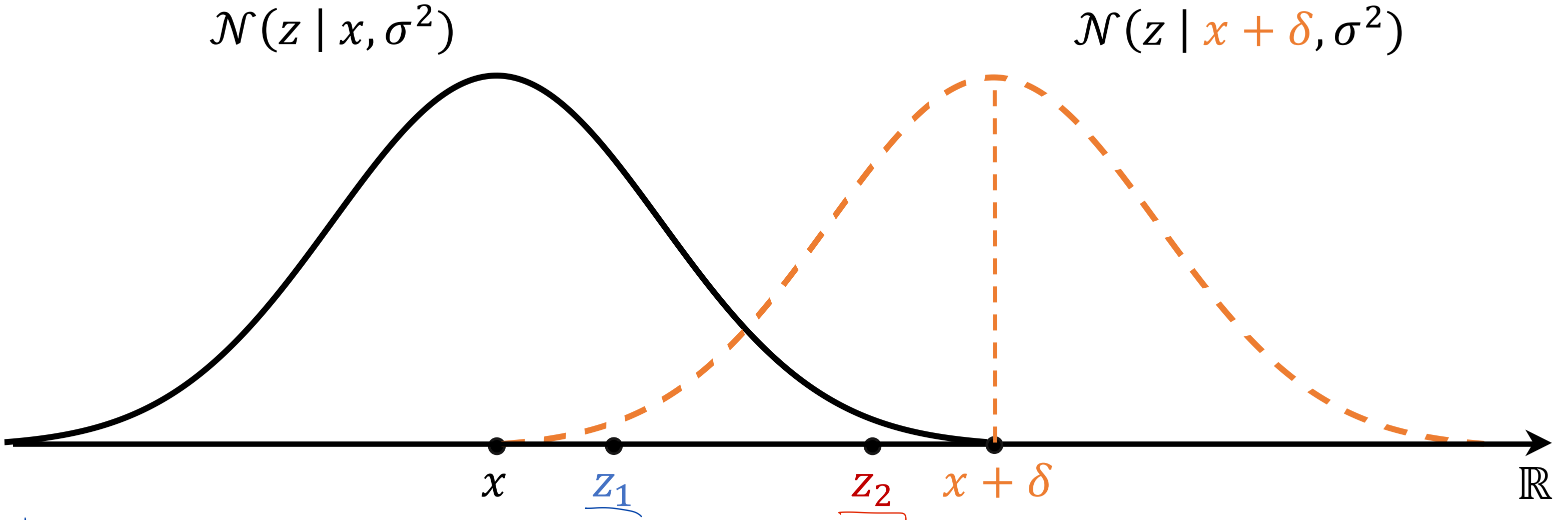
clean

$$\Rightarrow \boxed{\min_{h\in\mathbb{H}} \mathbb{P}_{\epsilon}(h(\boldsymbol{x}+\boldsymbol{\delta}+\boldsymbol{\epsilon})=c^*)} \quad \text{such that} \quad \mathbb{P}_{\epsilon}(h(\boldsymbol{x}+\boldsymbol{\epsilon})=c^*)=p_{\boldsymbol{x}}^*$$

where $\mathbb{H}$ is the set of all possible classifiers (and $f\in\mathbb{H}$)

Worst-Case View on Randomized Smoothing

How does the **worst-case** look like? We first consider the following 1-D example, where we can draw both z_1 and z_2 from $\mathcal{N}(z \mid x, \sigma^2)$ and $\mathcal{N}(z \mid x + \delta, \sigma^2)$



label blue

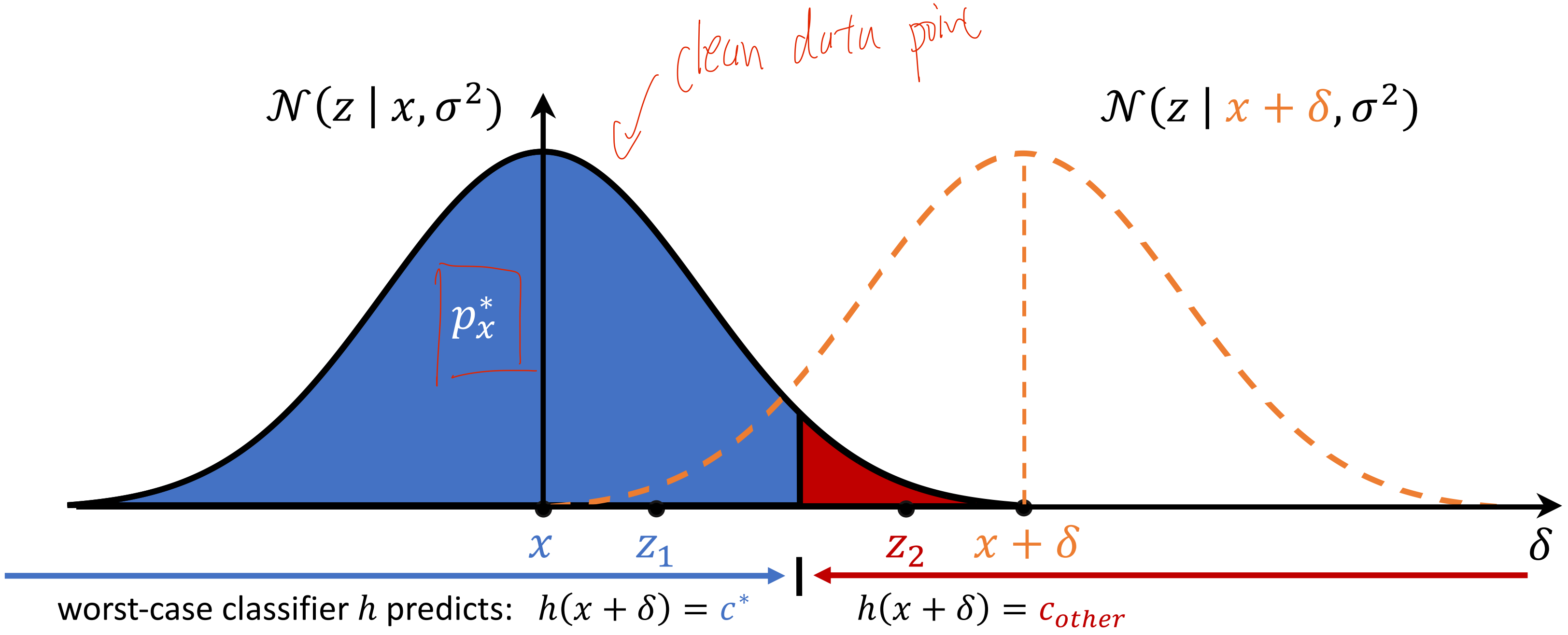
label red

In the **worst-case**:

- Sample z_1 with $\mathcal{N}(z_1 \mid x, \sigma^2) > \mathcal{N}(z_1 \mid x + \delta, \sigma^2)$ is classified as $f(z_1) = c^*$
- Sample z_2 with $\mathcal{N}(z_2 \mid x, \sigma^2) < \mathcal{N}(z_2 \mid x + \delta, \sigma^2)$ is classified as $f(z_2) = c_{other} \neq c^*$

Worst-Case View on Randomized Smoothing

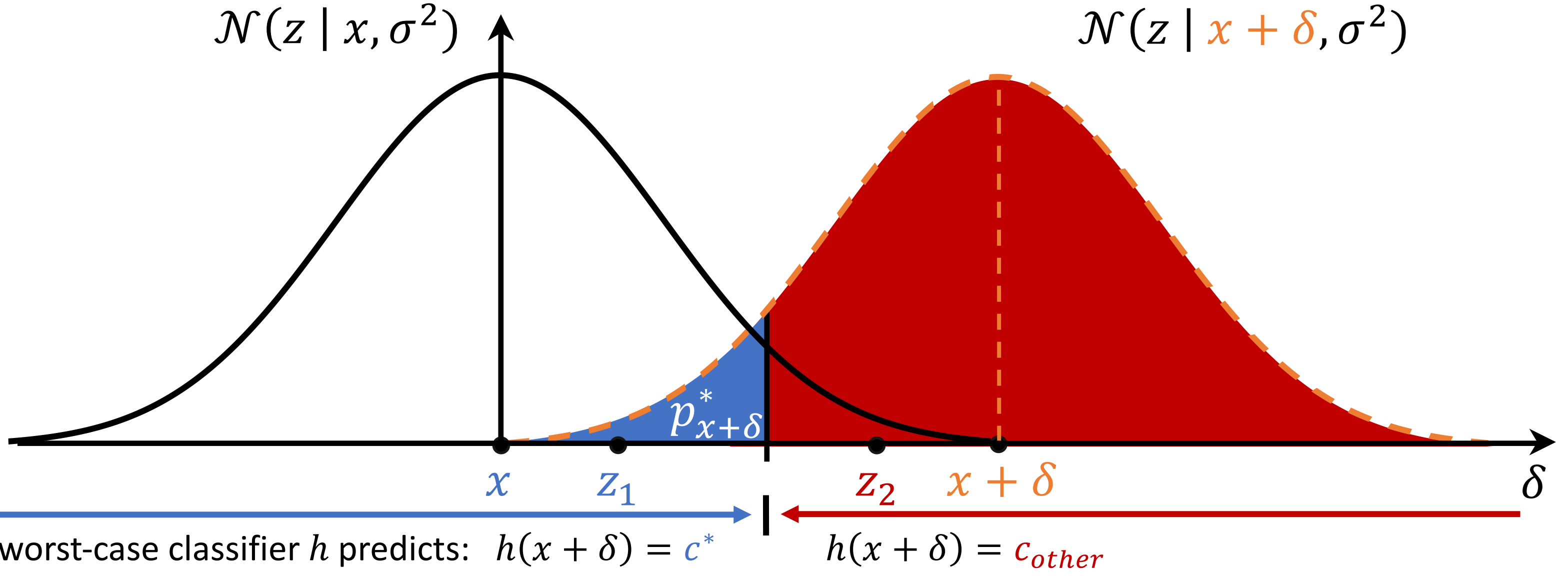
In the **worst-case**, all the samples from c^* are **concentrated on one side**, and all samples from $c_{other} \neq c^*$ are **concentrated on the other side**


clean data point
$\mathcal{N}(z \mid x, \sigma^2)$
$\mathcal{N}(z \mid x+\delta, \sigma^2)$
p_x^*
x
z_1
z_2
$x+\delta$
δ
worst-case classifier h predicts: $h(x+\delta) = c^*$
$h(x+\delta) = c_{other}$

1. This is a correct solution because the probability to classify x as c^* is p_x^*

Worst-Case View on Randomized Smoothing

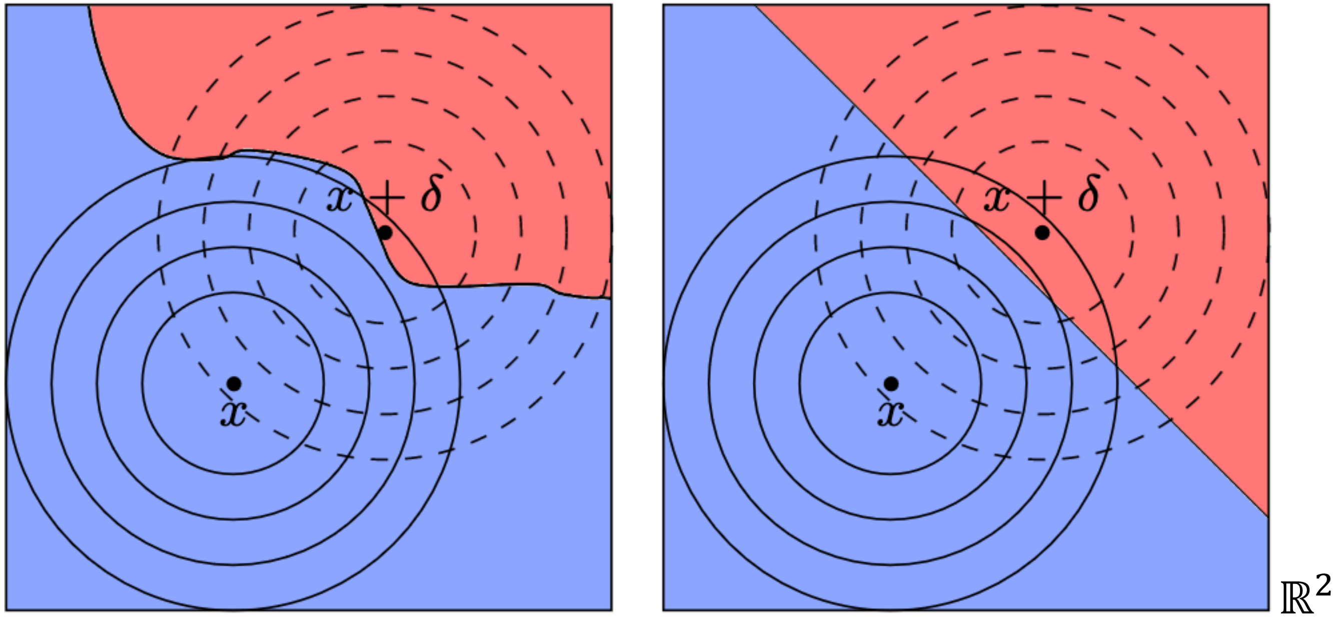
In the **worst-case**, all the samples from c^* are **concentrated on one side**, and all samples from $c_{other} \neq c^*$ are **concentrated on the other side**


$\mathcal{N}(z \mid x, \sigma^2)$
$\mathcal{N}(z \mid x+\delta, \sigma^2)$
$p^*_{x+\delta}$
x
z_1
z_2
$x+\delta$
δ
worst-case classifier h predicts: $h(x+\delta) = c^*$
$h(x+\delta) = c_{other}$

1. This is a correct solution because the probability to classify x as c^* is p^*_x
2. This is the worst-case since we minimize the probability $p^*_{x+\delta} = g(x+\delta)_{c^*}$ of classifying the perturbed sample $x+\delta$ as c^*

Worst-Case in Higher Dimensions

In general, the worst-case classifier h is a **linear classifier** whose decision boundary is orthogonal to the perturbation $\boldsymbol{\delta}$



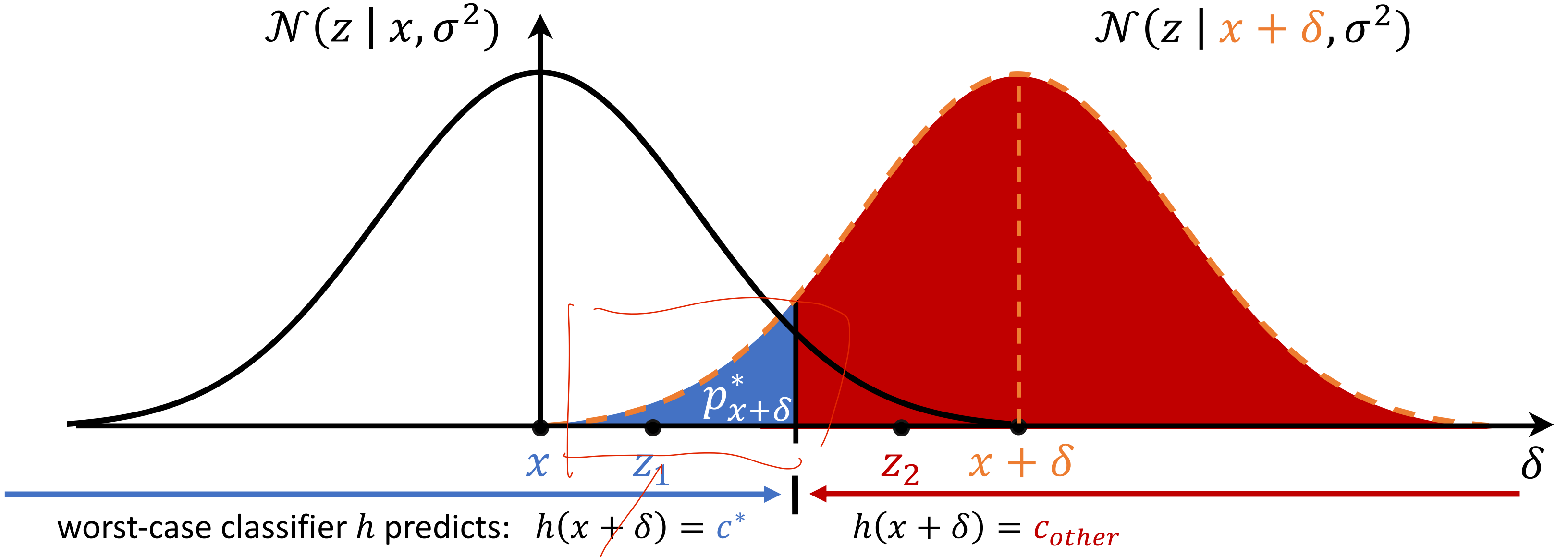
not the worst-case h

worst-case h

Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

Worst-Case View on Randomized Smoothing

In the **worst-case**, all the samples from c^* are **concentrated on one side**, and all samples from $c_{other} \neq c^*$ are **concentrated on the other side**



(Cohen et al., 2019) derive a close-form solution of the minimization problem and the optimal value is given by:

$$p^*_{x+\delta} = \overset{\text{CDF}}{\Phi}\left(\overset{\text{CDF}^{-1}}{\boxed{\Phi^{-1}(p^*_x)}} - \overset{\text{shifting}}{\boxed{\frac{\|\boldsymbol{\delta}\|_2}{\sigma}}}\right)$$

$\Phi^{-1}(p)$: inverse CDF of a standard Gaussian

Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

Largest Certifiable Radius

The probability to classify perturbed data as class c^* is

tiny blue ⇒ $$p^*_{x+\delta} = \Phi\left(\Phi^{-1}(p^*_x) - \frac{\|\delta\|_2}{\sigma}\right)$$

How large can we make $\|\delta\|_2$ (how much can we perturb x) and still **guarantee** that

$$\underset{c}{\operatorname{argmax}}\, g(x + \delta)_c = \arg\max_c g(x)_c?$$

The smoothed classifier g is robust if $p^*_{x+\delta} > 0.5$. We have

$$\boxed{p^*_{x+\delta} > 0.5}$$
$$\Leftrightarrow \Phi\left(\Phi^{-1}(p^*_x) - \frac{\|\delta\|_2}{\sigma}\right) > \frac{1}{2}$$
$$\Leftrightarrow \Phi^{-1}(p^*_x) > \frac{\|\delta\|_2}{\sigma}$$
$$\Leftrightarrow \|\delta\|_2 < \sigma \cdot \Phi^{-1}(p^*_x)$$

Recall: $\Phi^{-1}(0.5) = 0$

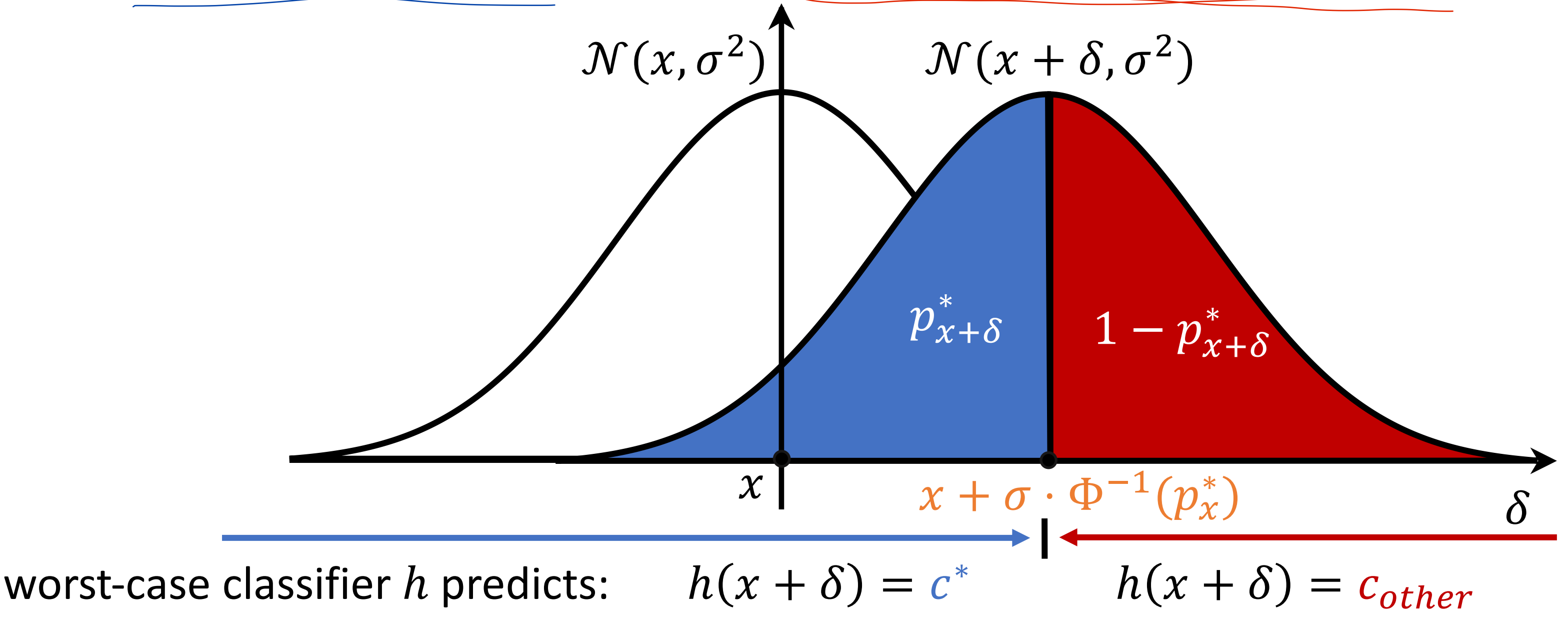
If $\|\delta\|_2 = \sigma \cdot \Phi^{-1}(p^*_x)$, then (in the worst case) $\boxed{p^*_{x+\delta} = 1 - p^*_{x+\delta} = 0.5.}$

Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

Largest Certifiable Radius

$p^*_{x+\delta} > 0.5$

- If $\|\boldsymbol{\delta}\|_2 = \sigma \cdot \Phi^{-1}(p^*_{\boldsymbol{x}})$, then (in the worst case), $p^*_{\boldsymbol{x}+\boldsymbol{\delta}} = 1 - p^*_{\boldsymbol{x}+\boldsymbol{\delta}} = 0.5$.
- Thus, for any perturbation $\|\boldsymbol{\delta}\|_2 < \sigma \cdot \Phi^{-1}(p^*_{\boldsymbol{x}})$, the smoothed classifier will not change its prediction, i.e. $\arg\max g(\boldsymbol{x} + \boldsymbol{\delta}) = \arg\max g(\boldsymbol{x})$.


$\mathcal{N}(x, \sigma^2)$
$\mathcal{N}(x + \delta, \sigma^2)$
$p^*_{x+\delta}$
$1 - p^*_{x+\delta}$
x
$x + \sigma \cdot \Phi^{-1}(p^*_x)$
δ
worst-case classifier h predicts:
$h(x + \delta) = c^*$
$h(x + \delta) = c_{other}$

- Larger variance σ^2 could lead to larger radii $\|\boldsymbol{\delta}\|_2$, however it could also lead to reduced $p^*_{\boldsymbol{x}}$ by introducing too much noise

Lipschitzness of Smoothed Classifiers

- The smoothed classifier g is naturally Lipschitz-continuous
- Gaussian Randomized Smoothing amounts to the Weierstrass transform, i.e. a convolution with a Gaussian $g(\boldsymbol{x}) = \big(\overset{\text{Convolution filter}}{f \ *}\ \overset{\text{Gaussian}}{\mathcal{N}(0, I)}\big)(\boldsymbol{x})$, which is $\sqrt{\frac{2}{\pi}}$ - Lipschitz
- This convolution with a Gaussian corresponds to a multiplication with a Gaussian in frequency domain
- In this context we can interpret Gaussian smoothing as a low-pass filter that filters out high frequencies (which typically correspond to noise)

Hadi Salman, Jerry Li, Ilya P. Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, Greg Yang. Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers. NeurIPS 2019.

How to determine $p_{\mathbf{x}}^*$?

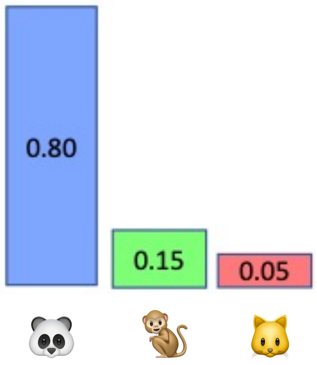
can't compute

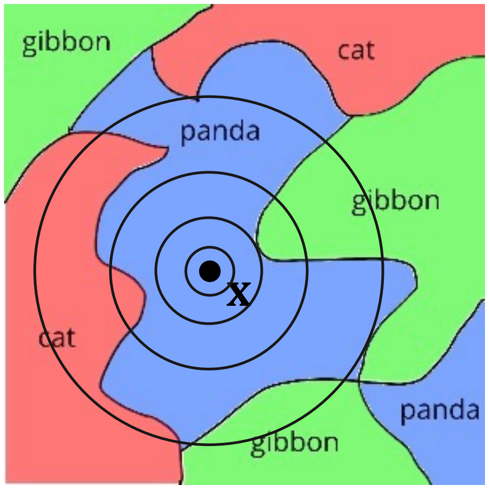
So far, we assumed we know the true $p_{\mathbf{x}}^*$, i.e. the proportion of the samples classified as c^* under the Gaussian noise.

- However, in general we cannot exactly compute these class probabilities since the prediction $f(\mathbf{x}+\boldsymbol{\epsilon})$ may change for any $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$ (infinite possibilities)

Solution: Monte Carlo estimation

- Sample a large number of samples from the Gaussian to estimate $p_{\mathbf{x}}^*$.
- As the number of samples goes to infinity, we are guaranteed to converge to the true proportion $p_{\mathbf{x}}^*$





Jeremy M. Cohen, Elan Rosenfeld and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. ICML 2019.

How to determine $p_{\mathbf{x}}^*$?

We need to be very certain not to overestimate $p_{\mathbf{x}}^*$, since this would lead to invalid certificates!

measure see c^*

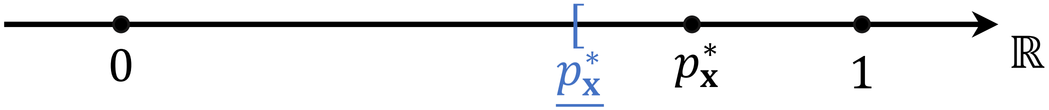
Solution: Compute a one-sided $(1-\alpha)$ lower confidence interval for $p_{\mathbf{x}}^*$ given the outcome for many samples (where α is small, e.g. 5%)

- Let $\mathrm{A} = \mathbb{I}[\boxed{f(\mathbf{x}+\boldsymbol{\epsilon}) = c^*}] \in \{0,1\}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ denote the random variable corresponding to the event of observing the class c^*.
- A is a Bernoulli random variable with probability of observing 1 equal to $p_{\mathbf{x}}^*$.
  Think of a (biased) coin flip with probability $p_{\mathbf{x}}^*$

  A 出现 m次？
- If we sample n times and $\mathrm{A} = 1$ in m/n then $P(A = m) = Binomial(n, p_x^*)$

$$\begin{pmatrix} n \\ p_x^* \end{pmatrix}$$

- We can get a **lower bound** $\underline{p_{\mathbf{x}}^*}$ from the confidence interval for the Binomial



⇒ The final certificates are **probabilistic** and hold with a confidence level of $1-\alpha$

Practical Considerations

- For certification, we have to determine
  - The most likely class $\boxed{c^*}$
  - Lower bound for $\boxed{\underline{p}_{\mathbf{x}}^*}$

- We need to take care from a statistical point of view.

Two steps

1. Using a small set of samples we first take a guess at c^*
2. Then, using a large set of samples we compute $\boxed{\underline{p}_{\mathbf{x}}^*}$ based on the confidence interval

Training for Randomized Smoothing

- If we train the base classifier f normally the predictions for the noisy inputs might not be accurate for large variance σ^2.
- Idea: **data augmentation** during training.
- Instead of training on the original data we randomly perturb it during training:

$$\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

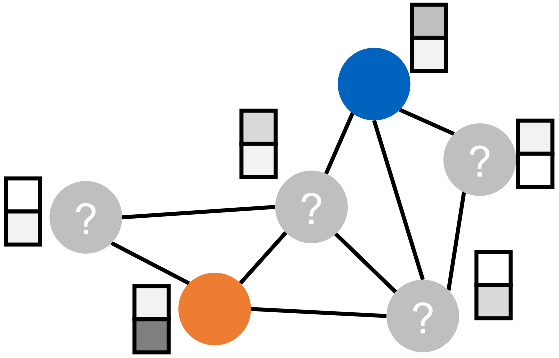
- and train on $\tilde{\mathbf{x}}$ rather then $\mathbf{x}$ to make sure f can predict the noisy inputs well.
- Note: we re-sample a new $\boldsymbol{\epsilon}$ every time.

- We can also (approximately) train the smoothed classifier g or even perform adversarial training (e.g. with FGSM) on g but this is outside of the scope.

Randomized Smoothing for discrete data

How about using Randomized Smoothing for graph data or for text?

In general: How about discrete data?



Challenge: Adding **Gaussian noise** to the **discrete graph structure** is not suitable

Randomized Smoothing for discrete data

In general: Smooth classifier $g(\mathbf{x})_c$ returns the probability that the base classifier f classifies a smoothed sample $\tilde{\mathbf{x}} \sim \phi(\mathbf{x})$ as class c

← noisy as class c

$$g(\mathbf{x})_c := \mathbb{P}\left(\boxed{f}\left(\phi(\mathbf{x})\right) = c\right) = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}} \sim \phi(\mathbf{x})}\left(\mathbb{I}[f(\tilde{\mathbf{x}}) = c]\right)}_{\text{Expectation}}$$

with a **randomization scheme** $\phi(\mathbf{x})$

For simplicity, we assume binary data (e.g. an unweighted graph)

→ **$\mathbf{L_0}$-norm radius certification.**

Goal: We want to certify the smooth classifier g. That is, we aim to show that for a radius r it holds:

$$\text{for all } \mathbf{x}' \text{ with } ||\mathbf{x}' - \mathbf{x}||_0 \le r: \quad c^* = \arg\max_c g(\mathbf{x}')_c$$

How to Smooth Graph Data?

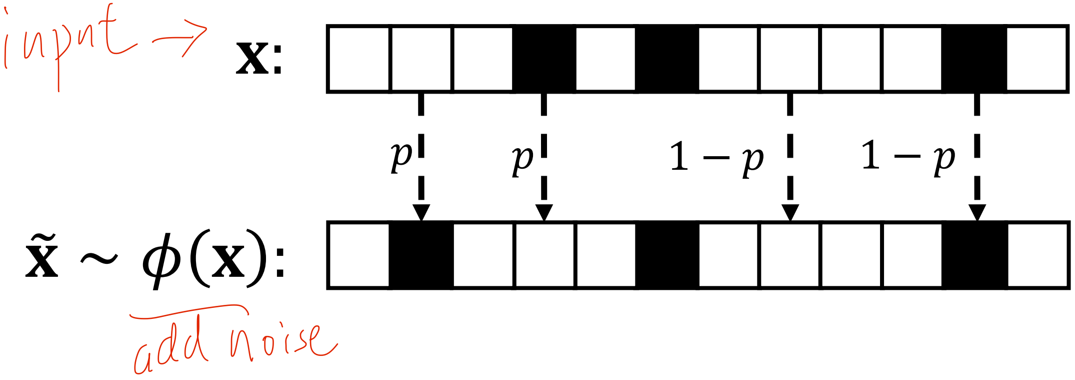
Problem: Adding **Gaussian noise** $\phi(\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \boldsymbol{I})$ to the **discrete** graph structure is not suitable

Solution: We model the n^2 possible edges in the adjacency matrix as a **Bernoulli random variable**

First idea: Same ”flip probability” for every element

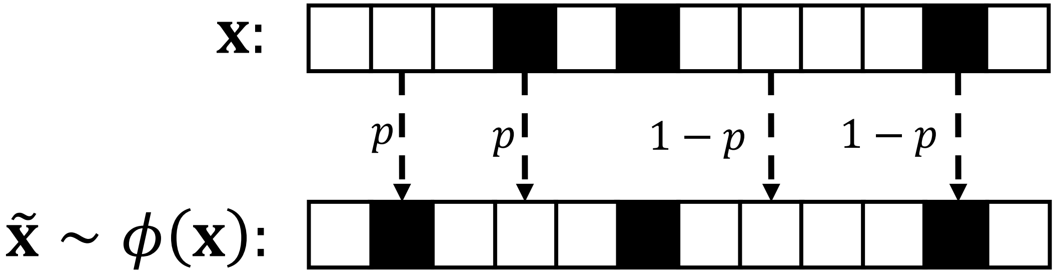
$$\tilde{\mathbf{x}} \sim \phi(\mathbf{x}) \text{ defined via } \mathbb{P}(\tilde{\mathbf{x}}_i \mid \mathbf{x}) = \begin{cases} p & , \tilde{\mathbf{x}}_i = 1 - \mathbf{x}_i \\ 1 - p & , \tilde{\mathbf{x}}_i = \mathbf{x}_i \end{cases}$$

$$p(\tilde{x} \mid x) = \prod_{i=1}^{D} p(\tilde{x}_i \mid x)$$



How to Smooth Graph Data?

First idea: Same ”flip probability” for every element



Problem: Real-world graphs are typically very **sparse** ($m \ll n^2$) and hence picking a meaningful p almost impossible

- Large flip probability p: most certainly we will add more random edges than original edges exist
- Small flip probability p: usually only a very few edges would be deleted

How to Smooth Graph Data? Sparsity Matters!

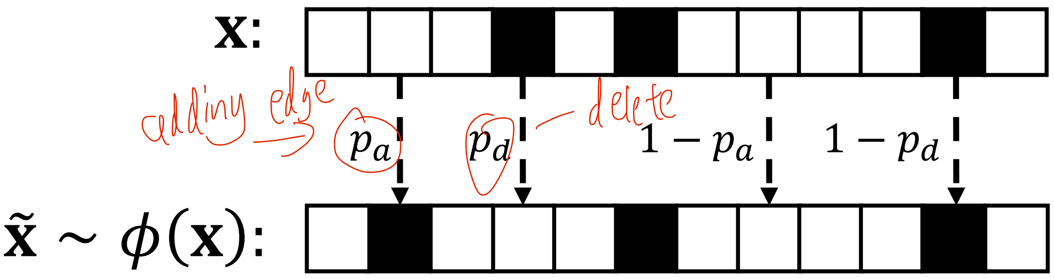
Sparsity-aware random sampling $\tilde{\mathbf{x}} \sim \phi(\mathbf{x})$:

Every element of the adjacency matrix is modelled via a **Bernoulli random variable** with **data dependent probability**:

flipping

$$\mathbb{P}(\tilde{\mathbf{x}}_i \mid \mathbf{x}) = \begin{cases} p_d^{\mathbf{x}_i} p_a^{1-\mathbf{x}_i} & , \tilde{\mathbf{x}}_i = 1 - \mathbf{x}_i \\ (1-p_d)^{\mathbf{x}_i}(1-p_a)^{1-\mathbf{x}_i} & , \tilde{\mathbf{x}}_i = \mathbf{x}_i \end{cases}$$

That is, each of the n^2 elements in the adjacency matrix is flipped with probability p_a if the value was previously 0 (no edge) or with p_d if previously an edge existed:



Aleksandar Bojchevski, Johannes Klicpera and Stephan Günnemann. Efficient Robustness Certificates for Discrete Data: Sparsity-Aware Randomized Smoothing for Graphs, Images and More, ICML 2020.

Smoothed Classifier for Discrete Data

noisy data classified by C

With this randomization scheme we can write:

$$g(\mathbf{x})_c \coloneqq \underline{\mathbb{P}\big(f\big(\phi(\mathbf{x})\big) = c\big)} = \mathbb{E}_{\tilde{\mathbf{x}}\sim\phi(\mathbf{x})}(\mathbb{I}[f(\tilde{\mathbf{x}}) = c])$$

$$= \sum_{\tilde{\mathbf{x}}} \mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x})\mathbb{I}[f(\tilde{\mathbf{x}}) = c] = \sum_{\tilde{\mathbf{x}}\text{ s.t. }\boxed{f(\tilde{\mathbf{x}})=c}} \underline{\mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x})} = \sum_{\tilde{\mathbf{x}}\text{ s.t. }f(\tilde{\mathbf{x}})=c}\ \prod_{i=1}^{n^2} \mathbb{P}(\tilde{\mathbf{x}}_i\,|\mathbf{x})$$

We illustrate $\mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x})$ with a hypothetical subgraph: $\mathbf{x} =$ □□🟦

↓ readed by X ←

$\mathbb{P}($□□🟦$) = (1-p_a)^2(1-p_d)$	$\mathbb{P}($□□□$) = \underline{(1-p_a)^2}p_d$
$\mathbb{P}($□🟧🟧$) = p_a(1-p_a)(1-p_d)$	$\mathbb{P}($□🟦□$) = p_a(1-p_a)p_d$
$\mathbb{P}($🟦□🟦$) = p_a(1-p_a)(1-p_d)$	$\mathbb{P}($🟧□□$) = p_a(1-p_a)p_d$
$\mathbb{P}($🟧🟧🟧$) = p_a^2(1-p_d)$	$\mathbb{P}($🟦🟦□$) = p_a^2p_d$

Worst-Case Base Classifier

Let's assume we know the value of $g(\mathbf{x})_{c^*}$ for the original sample $\mathbf{x}$

- Note: Of course, like in the Gaussian/continuous case, we do **not** compute this term exactly (far too expensive) but rather derive a bound based on MC samples

How far can we deviate from $\mathbf{x}$, e.g. obtaining $\mathbf{x}'$, and still **guarantee** that we do not change the prediction, i.e. still have $\arg\max_c g(\mathbf{x})_c = c^* = \arg\max_c g(\mathbf{x}')_c$?

→ Similarly to the Gaussian/continuous randomized smoothing, to answer this question, we can inspect the **worst-case base classifier.**

Since the worst-case base classifier has a simple form (e.g. linear in the Gaussian case), once we know it, it is "rather simple" to obtain the certification radius

How to Find the Worst-Case Base Classifier?

For any **chosen location $\mathbf{x}'$**, we can express the **worst-case base classifier** as a minimization problem:

$$\min_{h \in \mathbb{H}} \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}') \qquad \boxed{\left(\leq \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{f}(\tilde{\mathbf{x}})=c} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}') = g(\mathbf{x}')_{c^*} \right),}$$

can't compute

$$s.t. \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}) = g(\mathbf{x})_{c^*}$$

clean

Intuition:

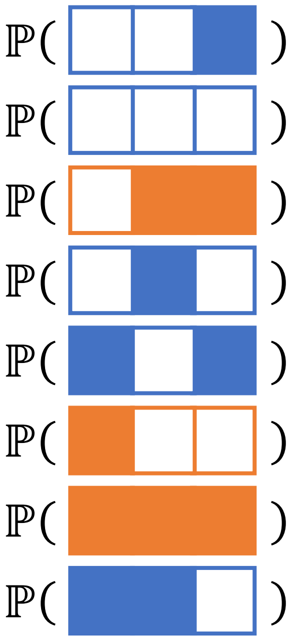
We search for a base classifier h such that the resulting smooth classifier

- maintains the probability mass for the clean sample $\mathbf{x}$, i.e. $\mathbb{P}\left(h\left(\phi(\mathbf{x})\right) = c^*\right) = \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}) = g(\mathbf{x})_{c^*}$
- and simultaneously minimizes the probability mass at the perturbed sample $\mathbf{x}'$, i.e. $\mathbb{P}\left(h\left(\phi(\mathbf{x}')\right) = c^*\right) = \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}')$

The Space of Base Classifiers $\mathbb{H}$

- Recall: we only assumed knowledge about the value of $g(\mathbf{x})_{c^*}$
  - We do **not** know the output of the actual base classifier f at every possible input
- Various base classifiers $h \in \mathbb{H}$ fulfill the property $\underline{\sum_{\tilde{\mathbf{x}} \text{ s.t. } h(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \mid \mathbf{x}) = g(\mathbf{x})_{c^*}}$

ha $\cancel{h_1} \in \mathbb{H}$.



$$= (1 - p_a)^2(1 - p_d) =$$

$$= (1 - p_a)^2 p_d =$$

$$= p_a(1 - p_a)(1 - p_d) =$$

$$= p_a(1 - p_a)p_d =$$

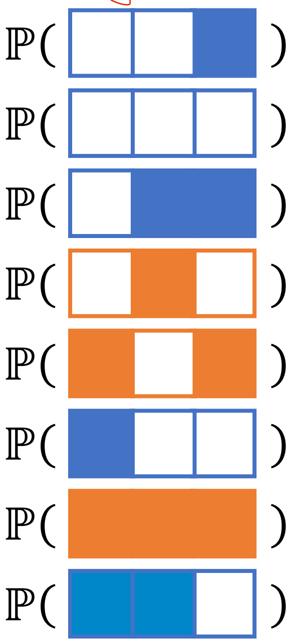
$$= p_a(1 - p_a)(1 - p_d) =$$

$$= p_a(1 - p_a)p_d =$$

$$= p_a^2(1 - p_d) =$$

$$= p_a^2 p_d =$$

hb $\cancel{h_2} \in \mathbb{H}$.



Which classifier h ∈ $\mathbb{H}$ represents the worst-case?

Solution for the Worst-Case Base Classifier

The previous minimization problem can be formulated as a linear program!

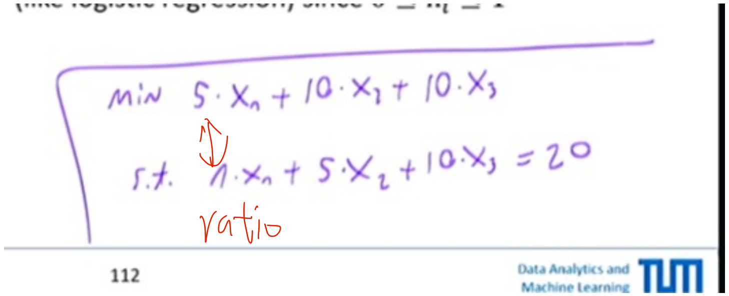
Denote with $\tilde{\mathbf{x}}^{(i)}$ the enumeration of all possible $\tilde{\mathbf{x}}$.

$$\min_{\mathbf{h}} \sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}'\right)$$

$$\text{subject to} \sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}\right) = g(\mathbf{x})_{c^*} \text{ and } 0 \leq \mathbf{h}_i \leq 1$$

The vector $\mathbf{h}$ represents the worst-case base classifier: $\mathbf{h}_i$ indicates whether $h(\tilde{\mathbf{x}}^{(i)})$ outputs c^* ($\mathbf{h}_i = 1$) or some other class

- Technically it is a soft classifier (like logistic regression) since $0 \leq \mathrm{h}_i \leq 1$



(like logistic regression) since $0 \leq \mathrm{h}_i \leq 1$

$$\min \; 5 \cdot x_1 + 10 \cdot x_2 + 10 \cdot x_3$$

$$\text{s.t.} \; 1 \cdot x_1 + 5 \cdot x_2 + 10 \cdot x_3 = 20$$

ratio

Solution for the Worst-Case Base Classifier

Interesting fact: This is a very special LP, which can efficiently and exactly be solved with a greedy approach

- Initialize all $\mathbf{h}_i$ with zero
- Compute likelihood ratios $\eta_i = {}^{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x})}\big/_{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x}')}$ and sort them
  - i.e. get indices $j_1, j_2, j_3, \ldots$ such that $\eta_{j_1} \geq \eta_{j_2} \geq \eta_{j_3} \geq \cdots$
- For $k = 1, \ldots$ set $\mathbf{h}_{j_k} = 1$ while budget $\sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(\mathrm{i})}|\mathbf{x}\right) = g(\mathbf{x})_{c^*}$ is not used up
  - i.e. process the sorted indices from left to right and assign a 1 (again: at the “switch point” we might have $0 < \mathbf{h}_{j.} < 1$ to consume the budget fully).

➢ Result: We do not even have to solve an optimization problem! We just sort based on the **likelihood ratio** and assign class c^* to the “left part”

– Note the similarity to the linear classifier in the Gaussian/continuous case

➢The worst-case base classifier has a very simple form

Some Details We Skip

- Knowing the worst-case base classifier, enables us to find the certification radius r (technically we even have two radii: addition/deletion)
  - Core insight: The general form of the worst-case classifier is always the same, independent of which $\mathbf{x}'$ we consider; similar to the Gaussian/continuous case
- In the linear program the dimensionality of $\mathbf{h}$ would be enormous (all possible graphs). We can use the fact that only the likelihood ratio $\eta_i = {}^{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x})}/_{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x}')}$ matters for the solution.
  - Intuition: Group together all graphs that have the same value for η_i into one large region → dimensionality of $\mathbf{h}$ equals to the number of regions
  - Indeed we have only a small number of regions: linear in the radius/~~dimensionality of the input;~~ very fast certification possible
- For further details we refer to (Bojchevski et al., 2020).

Most importantly, this randomized smoothing technique works for all models with binary input data: GNNs, CNNs, SVMs, Decision Trees, …

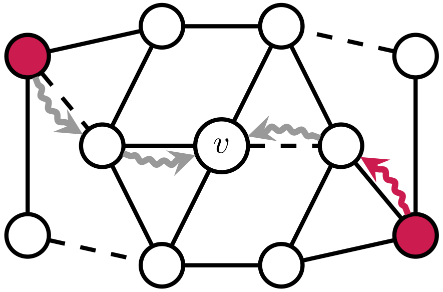
Gray-box Certificates

So far there were only two types of certificates:

- White-box certificates apply only to very specific models
- Black-box certificates are strong, model-agnostic but also ignore important properties of the underlying classifier

Can we improve black-box certificates by exploiting model knowledge?

- Gray-box certificates: Stronger certificates by restricting the set of possible classifiers $\mathbb{H}$
  - Exploit invariances of a model under e.g. rotation/translation [1]
  - Exploit graph-structure in message-passing GNNs [2]



[1] Jan Schuchardt and Stephan Günnemann. Invariance-Aware Randomized Smoothing Certificates. NeurIPS, 2022.

[2] Yan Scholten, Jan Schuchardt, Simon Geisler, Aleksandar Bojchevski, Stephan Günnemann. Randomized Message-Interception Smoothing: Gray-box Certificates for Graph Neural Networks. NeurIPS 2022.

Randomized Smoothing – Summary

- Randomized smoothing is an easy to implement certification method for perturbations which are L_2 bounded.
- It does not make any assumptions about the model, i.e. we can certify any deep neural network with this strategy.
- Since we need many samples for high-quality certificates, evaluating the model should be relatively cheap.
- Randomized smoothing also lends itself to robust training.

Robustness certification of GNNs is challenging but possible

- specialized approaches enable to exploit structure of the GNN models

Randomized smoothing can be adapted to **discrete input** data via **Bernoulli random variables**

→ We draw Monte Carlo samples for $g(\mathbf{x})$ and obtain the certified radii analytically.

→ Most importantly, **randomized smoothing with the proposed noise model works for all models with binary input data**: GNNs, CNNs, SVMs, Decision Trees, …

Robustness of Machine Learning Models: Summary

- Robustness of machine learning models is a crucial requirement to enable their application in the real world.
- As we have seen, there are **multiple strategies** for **certifying** the robustness of machine learning models and for **robust training**.
- The certification strategies we covered were for L_p-bounded perturbations.
- Of course there are many **other relevant perturbations**, e.g. rotations, translations, illumination changes, etc.
- While adversarial training is often easy to implement even for those perturbations, robustness certification for these scenarios is an active research field.
- **Our Chair** is active in robustness certification for images, vector data, and non-i.i.d. data such as graphs → reach out to us for thesis opportunities/Hiwi

Questions – Robustness (IV)

1. For which class of classifiers does the certificate for the smoothed classifier g equal the certificate for the underlying base classifier f and why?
2. Given two classifiers f_1 and f_2 with Lipschitz constants k_1 and k_2 with $k_1 < k_2$ which one would provide better guarantees? What if we form smoothed classifiers g_1 and g_2?
3. Suppose you want to determine the worst-case structure perturbation Δ, which is limited to (i) insert or (ii) remove at most k edges. How many possible perturbations are there (in big-O notation w.r.t. the number of nodes N and number of edges E)?
4. Given a graph with 2810 nodes and 7336 edges. What value of p_a do we need to choose if in expectation we want to sample 7336 further edges?