

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

- Often ML models and algorithms are optimized w.r.t. simple metrics
 - e.g. misclassification rate, reconstruction error, etc.
- As ML/AI is becoming more widespread and is used in critical applications (e.g. autonomous driving, algorithmic decision-making involving humans) we must consider further aspects
- As ML models get deployed in the real-world they create feedback loops which can have potentially unintended consequences
- One important aspect: Are the ML models reliable?
 - How do they behave in the wild? When your data might, e.g., be corrupted?

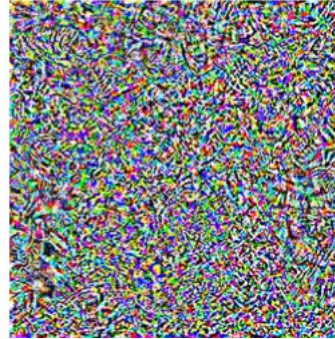
What Are Adversarial Examples?

Predicted
class:

“pig”



+ 0.005 x



=

?



Image from http://gradientscience.org/intro_adversarial/

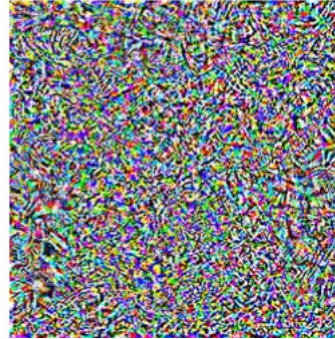
What Are Adversarial Examples?

Predicted
class:

“pig”



+ 0.005 x



=

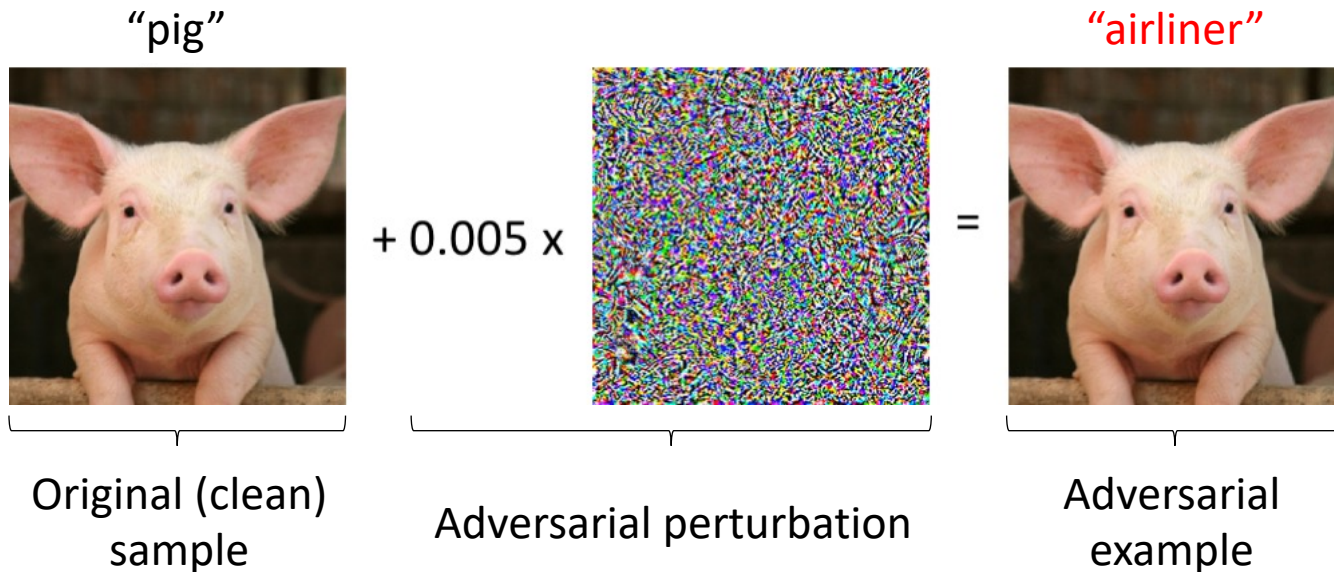
“airliner”



Image from http://gradientscience.org/intro_adversarial/

What Are Adversarial Examples?

Predicted
class:



Small (imperceptible) but **specifically crafted perturbations** lead to **false predictions** in machine learning models.

- Why should we care about adversarial examples?
- What does “small” mean?
- How are adversarial examples created?
- How to protect against adversarial examples?

Image from http://gradientscience.org/intro_adversarial/

Why We (Should) Care About Adversarial Examples

Real-world risks

- Adversarial examples are an obvious security threat for many real-world applications, e.g. self-driving cars.
- Adversarial examples also exist in the real world, e.g. 2D / 3D prints, special glasses to disturb face recognition, etc.

Conceptual gaps

- Neural networks are hypothesized to learn meaningful representations that capture semantic understanding of the domain and task.
 - Adversarial examples are counterexamples to this hypothesis: the semantic content of the samples is unchanged but the network is fooled.
- Nature as an adversary: Even if there is no adversary in our use-case, we should quantify robustness to **worst-case noise**

Adversarial Examples



Adversarial glasses fool facial recognition systems into classifying the wearer as someone else, [Sharif 2016]



ML systems classify the adversarially modified STOP sign as a speed limit sign, [Eykholt 2018]

Mahmood Sharif et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." *SIGSAC* 2016.
Kevin Eykholt, et al. "Robust physical-world attacks on deep learning visual classification." *CVPR* 2018.

Adversarial Examples – Definition

Classification task:

- Dataset: $(\mathbf{x}_i, y_i) \sim \mathbb{P}_{\text{data}}, \quad (\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$
- Classifier: $f: \mathbb{R}^d \rightarrow \mathcal{Y}$
- Specify a perturbation set $\mathcal{P}(\mathbf{x})$, i.e. a set of perturbations which when applied to \mathbf{x} do not change its semantic
 - and, thus, should also not change its classification
- We say that a point $\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})$ is an adversarial example for f at (\mathbf{x}, y)
 - if $f(\mathbf{x}) = y$, i.e., f correctly classifies \mathbf{x}
 - but $f(\tilde{\mathbf{x}}) \neq y$, i.e. fails to correctly classify $\tilde{\mathbf{x}}$

On “small” Perturbations

- Perturbations should not change the **semantic content** of a sample.
- This is often translated into L_p constraints with some small ϵ
- L_p norm: $\mathcal{P}_{\epsilon,p}(\mathbf{x}) = \{\tilde{\mathbf{x}}: \|\tilde{\mathbf{x}} - \mathbf{x}\|_p < \epsilon\}$, typically $p \in \{1, 2, \infty\}$
- While mathematically convenient, L_p norms with small ϵ do not contain **all** semantically meaningless perturbations.
- For example, a **small rotation** does typically not change the meaning of a picture but often corresponds to **large changes** in L_p norm.

Attack Variants

- **Evasion attacks:** given a **fixed**, trained classifier f , the attacker aims to find an adversarial perturbation (at test time)
- **Poisoning attacks:** the adversary aims to modify the **training dataset** such that a classifier trained on the dataset has properties desired by the attacker.
 - i.e. the manipulation/corruption is done **before** training
 - ➔ **Not covered in this course.**
- **Targeted attacks:** the attacker aims to have a certain sample classified as a specific class (e.g. speed limit 100 km/h sign).
- **Untargeted attacks:** the attacker aims to have a sample misclassified as **any** class different than the correct one.

Roadmap

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

Adversarial Attacks: Objective Function

Construction of adversarial examples can be phrased as an optimization problem.
For example:

$$\tilde{\mathbf{x}}_{\mathbf{x}}^* = \arg \max_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell_{0/1}(f(\tilde{\mathbf{x}}), y)$$

- **Recall:** $\ell_{0/1}$ is the zero/one loss (0 if correct, 1 if incorrect).
- **However:** $\ell_{0/1}$ has either zero or undefined gradient.
- Therefore, the **cross-entropy loss** \mathcal{L} is often used as a surrogate:

$$\tilde{\mathbf{x}}_{\mathbf{x}}^* = \arg \max_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \mathcal{L}(f(\tilde{\mathbf{x}}), y)$$

Projected Gradient Descent

$$\tilde{\mathbf{x}}_{\mathbf{x}}^* = \arg \max_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \mathcal{L}(f(\tilde{\mathbf{x}}), y)$$

- One common method is **Projected gradient descent** (PGD): after each gradient step on the objective, project onto the valid domain.
- $\mathbf{x}_{t+1} = \Pi(\mathbf{x}_t + \eta_t \nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}_t), y))$
- Like training the model but **updating the data instead of the weights**.
- Note: since $f(\tilde{\mathbf{x}})$ is **not convex**, in general we cannot find the global optimum.

Fast Gradient-Sign Method (FGSM):

- $\tilde{\mathbf{x}} = \Pi(\mathbf{x} + \eta \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}), y)))$
- When $\mathcal{P}(\mathbf{x})$ is a ball with radius ϵ measured by the L_{∞} norm, setting $\eta = \epsilon$ yields valid perturbations with only a single step and without projection.

Alternative Optimization Problem

- An alternative formulation is to optimize:

$$\min_{\tilde{\mathbf{x}}} \mathcal{D}(\mathbf{x}, \tilde{\mathbf{x}}) \quad \text{subject to } \ell_{0/1}(f(\tilde{\mathbf{x}}), y) > 0$$

- Here, \mathcal{D} is a term that is large when $\tilde{\mathbf{x}}$ is far from \mathbf{x} (e.g., an L_p distance)
- [Carlini and Wagner, 2017] convert this constrained into an **unconstrained** optimization problem:

$$\min_{\tilde{\mathbf{x}}} \mathcal{D}(\mathbf{x}, \tilde{\mathbf{x}}) + \lambda \cdot L(\tilde{\mathbf{x}}, y),$$

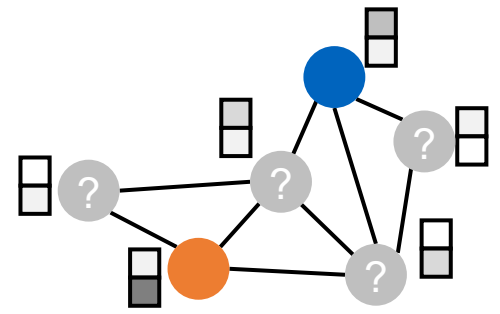
- A very effective **loss function** is

$$L(\tilde{\mathbf{x}}, y) = \left[Z(\tilde{\mathbf{x}})_y - \max_{i \neq y} (Z(\tilde{\mathbf{x}})_i) \right]_+$$

- y is the original class we want $\tilde{\mathbf{x}}$ to deviate from
- $[\mathbf{x}]_+ = \max(\mathbf{x}, 0)$
- $Z(\tilde{\mathbf{x}})_i = \log f(\tilde{\mathbf{x}})_i$ (log probability of class i)
- The loss L is positive if $\tilde{\mathbf{x}}$ is classified as y and 0 otherwise.

Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." *IEEE symposium on security and privacy (sp)*. 2017.

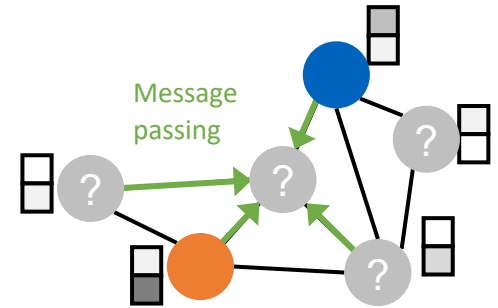
Adversarial Attacks on GNNs



- Graph Neural Networks are also not robust under adversarial perturbations
- In contrast to e.g. image classifiers, GNNs use **both** the **node's attributes** as well as their **connections** to make a prediction.
 - Therefore, adversarial attacks can happen through both the **node attributes** as well as the **graph structure**
 - Structural attacks are indeed quite common in the real world (e.g. adding fake connections in a social network)
- Structure attacks are specifically challenging since they change the flow of messages passed through the GNN

Adversarial Attacks on GNNs

- Example: two-layer GCN in matrix form:



node attributes

$$\mathbf{Z} \in \mathbb{R}^{N \times C} = f_{\theta}(\mathbf{A}, \mathbf{X}) = \text{softmax}(\hat{\mathbf{A}} \text{ReLU}(\hat{\mathbf{A}} \mathbf{X} \mathbf{W}^{(1)} + \mathbf{b}^{(1)}) \mathbf{W}^{(2)} + \mathbf{b}^{(2)})$$

message passing

- $\theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$ are learnable model weights.
- Adversarial attack:** Modify node attributes \mathbf{X} and/or adjacency matrix \mathbf{A} in order to maximize classification loss
 - of an individual target node or
 - on the whole dataset/test set (global attack).

GNN Adversarial Attacks: Challenges

1. Optimization over **discrete variables** (the graph structure). Perturbations are measured via non-convex L_0 norm.
2. **Relational dependencies** between the nodes: cannot view samples in isolation.
3. $(A', X') \approx (A, X)$: What is a sensible measure of perturbations that do not change the semantics for (attributed) graphs?
4. **Transductive setting**: unlabeled data is **used during training**; most realistic scenario is a **poisoning attack**, where the attacker modifies the training data, which corresponds to a challenging **bilevel optimization problem**:

$$\max_{A, X} \mathcal{L}_{test}(f_{\theta^*}(A, X)) \quad s. t. \quad \theta^* = \arg \min_{\theta} \mathcal{L}_{train}(f_{\theta}(A, X))$$

GNN Adversarial Attack: Nettack

- One of the earliest GNN attack algorithms [Zügner, 2018].
- Targets a **single node's prediction**.

$$\mathbf{Z} = f_{\theta}(\mathbf{A}, \mathbf{X}) = \text{softmax}(\hat{\mathbf{A}} \text{ReLU}(\hat{\mathbf{A}} \mathbf{X} \mathbf{W}^{(1)}) \mathbf{W}^{(2)})$$

Linearize classifier

$$\log \mathbf{Z}' = \hat{\mathbf{A}}^2 \mathbf{X} \mathbf{W}'$$

Structure perturbations: $\max_{\hat{\mathbf{A}}} \mathcal{L}'(\log \mathbf{Z}'_v)$ where $\log \mathbf{Z}'_v = [\hat{\mathbf{A}}^2 \mathbf{c}]_v$ ← Constants

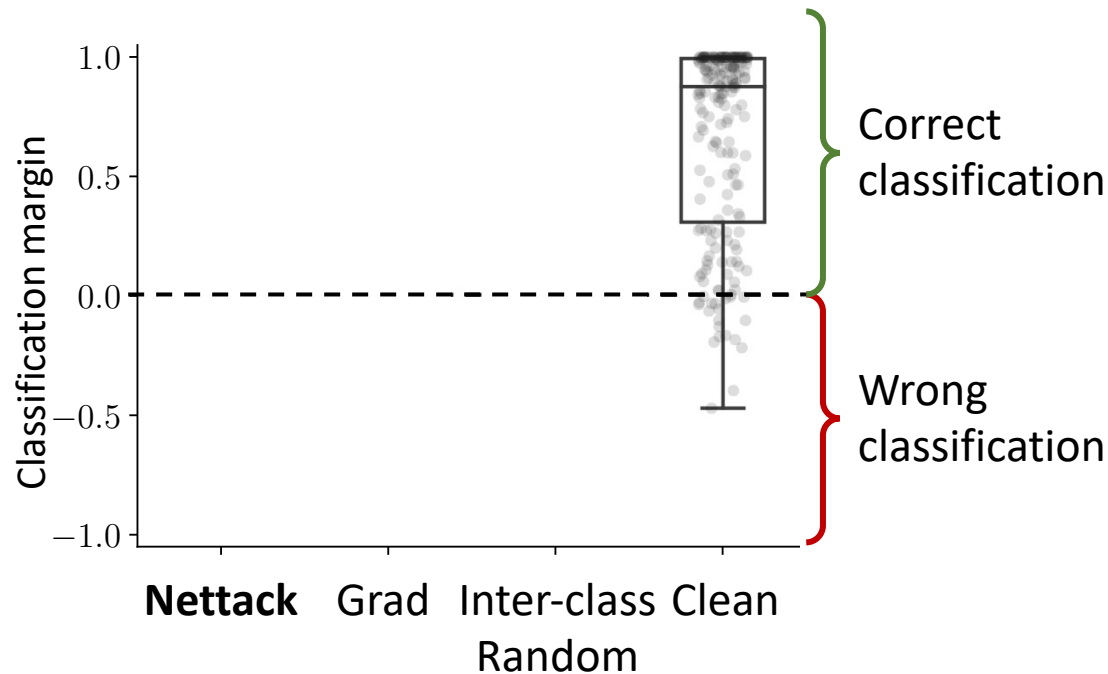
Feature perturbations: $\max_{\mathbf{X}} \mathcal{L}'(\log \mathbf{Z}'_v)$ where $\log \mathbf{Z}'_v = [\mathbf{c}_1^T \mathbf{X} \mathbf{c}_2]_v$ ← Constants

→ **Greedy** pick the **optimal perturbation** at each step.

→ Uses closed-form solutions for the **optimal perturbation** at each step

GNN Adversarial Attack: Nettack results

- Poisoning attack scenario (model is trained on perturbed data)
- Each point represents one attacked node
- Attack budget per node: $\Delta(i) = \deg(i) + 2$



% Correct:

90.3%

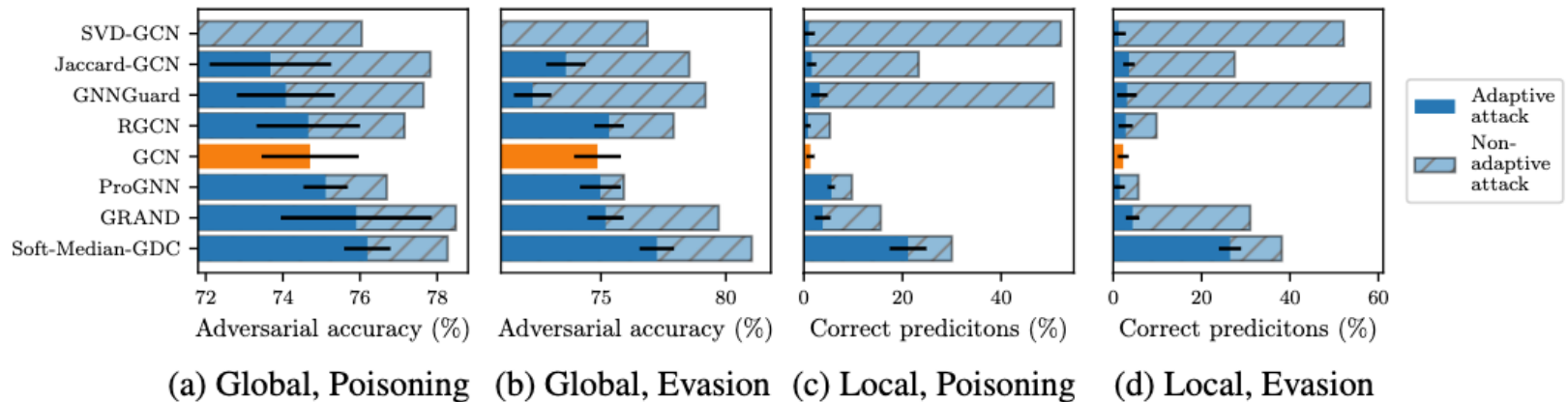
Adversarial Attacks for Model Evaluation

Adversarial attacks can be used to evaluate a model's robustness

- I.e., evaluating the performance of models under worst-case noise (**adversarial accuracy**)

But adversarial attacks should be model-specific

- Just because an attack is successful against one architecture does not mean the attack is successful for other architectures
- Adversarial attacks must be adapted to architectural changes (**adaptive adversarial attacks**)



Felix Mujkanovic, Simon Geisler, Stephan Günnemann, Aleksandar Bojchevski.
Are Defenses for Graph Neural Networks Robust? NeurIPS 2022

Roadmap

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

Introduction

- Most ML models (trained and/or applied in the traditional way) are vulnerable to adversarial examples
- How to defend against adversarial examples?
- Can we prevent them?
- Can we improve the robustness of our models?

What does not seem to work

- **Post-hoc prevention of attacks**
E.g. gradient obfuscation, i.e. randomizing or shattering gradients of the model in order to prevent gradient-based attacks. So far, these defenses have all been broken by stronger attacks.
- **Detection of adversarial examples**
E.g. out-of-distribution shift: since the adversarial examples come from a different distribution than the natural images, we could try to distinguish the two distributions or perform outlier / anomaly detection. While some of these methods work against “vanilla” PGD attacks, targeted attacks are very successful against these defenses.
- Fixing a “bad” model seems not to be the solution

Athalye, Anish, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." ICML 2018.

Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." AISec 2017.

Robust Training

Robust training refers to training procedures aimed at producing models that are robust to adversarial (and/or other) perturbations.

A **common theme** is to optimize a ‘worst-case’ loss (also called robust loss), i.e. the loss achieved under the worst-case perturbation.

- Let $\ell(\hat{y}, y)$ be some loss, e.g. cross-entropy loss
- The (non-robust) training tries to find an f that minimizes the expected loss

$$R = \mathbb{E}_{(\mathbf{x}, y) \in \mathbb{P}_{\text{data}}} [\ell(f(\mathbf{x}), y)]$$

- The robust version of this problem is

$$R_{\text{rob}} = \mathbb{E}_{(\mathbf{x}, y) \in \mathbb{P}_{\text{data}}} \left[\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f(\tilde{\mathbf{x}}), y) \right]$$

Loss achieved by the worst-case perturbation in $\mathcal{P}(\mathbf{x})$

Robust Training

Robust training refers to training procedures aimed at producing models that are robust to adversarial (and/or other) perturbations.

A **common theme** is to optimize a ‘worst-case’ loss (also called robust loss), i.e. the loss achieved under the worst-case perturbation.

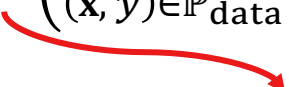
- **Adversarial training** is an easy-to-implement robust training procedure that uses adversarial examples as a proxy for the ‘worst-case’ perturbation.
- In the next chapter, we will cover **robustness certification** techniques. Some of these can also be used for robust training.

Adversarial Training

Idea: perform stochastic gradient descent (SGD) on the **robust loss** R_{rob} :

$$R_{\text{rob}} = \mathbb{E}_{(\mathbf{x}, y) \in \mathbb{P}_{\text{data}}} \left[\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f(\tilde{\mathbf{x}}), y) \right]$$

For $f = f_{\theta}$ being a **neural network** parameterized by weights θ , we can write

$$\begin{aligned} \nabla_{\theta} R_{\text{rob}} &= \nabla_{\theta} \left(\mathbb{E}_{(\mathbf{x}, y) \in \mathbb{P}_{\text{data}}} \left[\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f_{\theta}(\tilde{\mathbf{x}}), y) \right] \right) \\ &= \mathbb{E}_{(\mathbf{x}, y) \in \mathbb{P}_{\text{data}}} \left[\nabla_{\theta} \left(\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f_{\theta}(\tilde{\mathbf{x}}), y) \right) \right] \end{aligned}$$


How to take the gradient of the **worst-case loss** w.r.t. the weights θ ?

Adversarial Training: Danskin's Theorem

- How to obtain $\nabla_{\theta} L = \nabla_{\theta} \left(\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f_{\theta}(\tilde{\mathbf{x}}), y) \right)$?
- That is, the **gradient** of the **worst-case loss** w.r.t. the model parameters.
- **Danskin's Theorem***: Let $\Delta(\theta)$ be the set of $\tilde{\mathbf{x}}$ for which the supremum is obtained. If $\Delta(\theta)$ contains only a single element, i.e. $\Delta(\theta) = \{\tilde{\mathbf{x}}_{\theta}^*\}$, then the sup is differentiable at θ and

$$\nabla_{\theta} \left(\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell(f_{\theta}(\tilde{\mathbf{x}}), y) \right) = \nabla_{\theta} \ell(f_{\theta}(\tilde{\mathbf{x}}_{\theta}^*), y)$$

* Technically, the theorem requires some conditions which might not hold in our case, see [Madry et al., 2017]

Madry, Aleksander, et al. "Towards Deep Learning Models Resistant to Adversarial Attacks", ICLR 2018.

Adversarial Training: Algorithm

- Using **Danskin's theorem** we can compute the gradient of the worst-case loss given the corresponding perturbation.
- **Problem:** finding the **worst-case** perturbed example $\tilde{\mathbf{x}}$ is **intractable**; if we could find it efficiently, we would have solved the **exact verification** problem.
- **Idea:** Create any adversarial example as a proxy of the worst-case perturbation, e.g. via the fast gradient-sign method (FGSM).
- Adversarial training algorithm outline:
 1. Sample $(\mathbf{x}_i, y_i) \sim \mathbb{P}_{\text{data}}$
 2. Using an adversarial attack procedure, find an $\tilde{\mathbf{x}}_i$ with high loss $\ell(f_{\theta}(\tilde{\mathbf{x}}_i), y_i)$
 3. Update weights via gradient descent: $\theta \leftarrow \theta - \eta \nabla_{\theta} \ell(f_{\theta}(\tilde{\mathbf{x}}_i), y_i)$
- In step 2, we must trade off the strength of the attack with its computational cost.

Adversarial Training: Summary

Pro:

- It empirically increases robustness of the resulting models.
- It is easy to implement.

Con:

- If we want to use a powerful attack on the inner optimization, the slowdown is about 10x compared to standard training.
- The resulting models typically have lower accuracy on clean data.
- We don't get any theoretical guarantees of the model's robustness

Improving Robustness for GNNs

GNN robustness is a highly **active research area**

- To date there exists **no defense** against **structure attacks** that consistently improves results; standard methods such as **adversarial training** do not seem to work well

Heuristic defenses

- E.g. adjacency low-rank approximation via truncated Singular Value Decomposition [Entezari 2020]; filtering of malicious edges via attribute similarity [Wu 2019]
- However: equivalent/similar defenses for CNNs have been proven to be non-robust against worst-case perturbations (see before: slide on adaptive attacks)

Robust Training

- In form of Adversarial Training, e.g., via Projected Gradient Descent [Xu 2019]
- Or proposed together with a certification technique (upcoming topic)

Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis.

All you need is Low (rank): Defending against adversarial attacks on graphs. WSDM 2020.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu.

Adversarial examples for graph data: Deep insights into attack and defense. IJCAI 2019.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin Yu Chen, Tsui Wei Weng, Mingyi Hong, and Xue Lin.

Topology attack and defense for graph neural networks: An optimization perspective. IJCAI 2019.

Questions – Robustness

1. Given an arbitrary binary classifier f for an input domain \mathbb{R}^d and the perturbation set $\mathcal{P}_{\epsilon,p}(\mathbf{x})$ as defined before. Is it possible that every $\mathbf{x} \in \mathbb{R}^d$ is “robust”, i.e. no adversarial example exists?
2. Will the fast gradient-sign method (FGSM) always find an adversarial example (assuming there exist some in the set of perturbations $\mathcal{P}_{\epsilon,\infty}(\mathbf{x})$)?
3. Is a projected-gradient-descent (PGD) attack on a GNN via the graph structure a good idea? Why or why not?

Recommended Reading

- Lecture 09: Introduction to adversarial examples and
Lecture 10: Empirical defenses for adversarial examples
of Jerry Li's course on Robustness in Machine Learning (CSE 599-M),
<https://jerryzli.github.io/robust-ml-fall19.html>