Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 11: Adversarial machine learning and generative adversarial networks

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-556 (Fall 2025)

License Information for Mathematics of Data Slides

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Outline

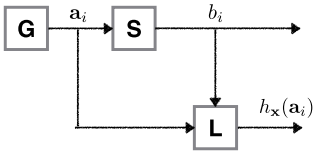
- This class
  - Adversarial Machine Learning (minmax)
    - Adversarial Training
    - Generative Adversarial Networks (GANs)
- Next class
  - Difficulty of minmax
  - Diffusion models

Adversarial machine learning

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$$

- A seemingly simple optimization formulation
- Critical in machine learning with many applications
  - Adversarial examples and training
  - Generative adversarial networks
  - *Robust reinforcement learning (more on this next week)
  - ...

From empirical risk minimization...



Definition (Empirical Risk Minimization (ERM))

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be samples with $b_i \in \{-1, 1\}$ and $\mathbf{a}_i \in \mathbb{R}^p$. The ERM problem reads

$$\min_{\mathbf{x}} \left\{ R_n(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i)$ is the loss on the sample $(\mathbf{a}_i, b_i)$.

Some frequently used loss functions

- $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = \log(1 + \exp(-b_i h_{\mathbf{x}}(\mathbf{a}_i)))$ *Logistic loss.*
- $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = (b_i - h_{\mathbf{x}}(\mathbf{a}_i))^2$ *Squared error.*
- $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = \max(0, 1 - b_i h_{\mathbf{x}}(\mathbf{a}_i))$ *Hinge loss.*

...Into adversarial examples

Definition (Adversarial examples [28])

Let $h_{\mathbf{x}^\star} : \mathbb{R}^p \to \mathbb{R}$ be a model trained through empirical risk minimization, with optimal parameters $\mathbf{x}^\star$. Let $(\mathbf{a}, b)$ be a sample with $b \in \{-1, 1\}$ and $\mathbf{a} \in \mathbb{R}^p$. An **adversarial example** is a perturbation $\boldsymbol{\delta} \in \mathbb{R}^p$ designed to lead the trained model $h_{\mathbf{x}^\star}$ to misclassify a given input $\mathbf{a}$. Given an $\epsilon > 0$, it is constructed by solving

$$\boldsymbol{\delta} \in \underset{\boldsymbol{\delta}:\|\boldsymbol{\delta}\| \leq \epsilon}{\arg\max}\, L(h_{\mathbf{x}^\star}(\mathbf{a} + \boldsymbol{\delta}), b)$$

Example norms frequently used in adversarial attacks

- The most commonly used norm is the ℓ_∞-norm [12, 21].
- The use of ℓ_1-norm leads to sparse attacks.

Figure: (Left) An ℓ_∞-attack: The alteration is hard to perceive. (Right) An ℓ_1-attack: The alteration in this case is obvious.

Adversarial examples and proximal gradient descent

- Target problem:

$$\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)$$

- We can do better than FGSM via proximal gradient methods for composite minimization:

$$\max_{\boldsymbol{\delta}\in\mathbb{R}^p} \underbrace{L(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)}_{f(\boldsymbol{\delta})} + \underbrace{\delta_{\mathcal{N}}(\boldsymbol{\delta})}_{g(\boldsymbol{\delta})},$$

where $\delta_{\mathcal{N}}(\boldsymbol{\delta})$ is the indicator function of the ball $\mathcal{N} := \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$.

Recall: Proximal operator of indicator functions

For the indicator functions of simple sets, e.g., $g(\boldsymbol{\delta}) := \delta_{\mathcal{N}}(\boldsymbol{\delta})$, the prox-operator is the projection operator

$$\text{prox}_{\lambda g}(\boldsymbol{\delta}) := \pi_{\mathcal{N}}(\boldsymbol{\delta}),$$

where $\pi_{\mathcal{N}}(\boldsymbol{\delta})$ denotes the projection of $\boldsymbol{\delta}$ onto $\mathcal{N}$. When $\mathcal{N} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \lambda\}$, $\pi_{\mathcal{N}}(\boldsymbol{\delta}) = \text{clip}(\boldsymbol{\delta}, [-\lambda, \lambda])$.

Adversarial examples and proximal gradient descent (cont'd)

- Target non-convex problem:

$$\max_{\boldsymbol{\delta} \in \mathbb{R}^p} \underbrace{L(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)}_{f(\boldsymbol{\delta})} + \underbrace{\delta_{\mathcal{N}}(\boldsymbol{\delta})}_{g(\boldsymbol{\delta})},$$

where $\delta_{\mathcal{N}}(\boldsymbol{\delta})$ is the indicator function of the ball $\mathcal{N} := \{\mathbf{y} : \|\mathbf{y}\|_\infty \leq \epsilon\}$.

Proximal gradient ascent (PGA)

1. Choose $\boldsymbol{\delta}^0 \in \operatorname{dom} f(\boldsymbol{\delta}) + g(\boldsymbol{\delta})$ as initialization.
2. For $k = 0, 1, \cdots$, generate a sequence $\{\boldsymbol{\delta}^k\}_{k \geq 0}$ as:

$$\boldsymbol{\delta}^{k+1} := \operatorname{prox}_{\alpha_k g}\left(\boldsymbol{\delta}^k + \alpha_k \nabla f(\boldsymbol{\delta}^k)\right).$$

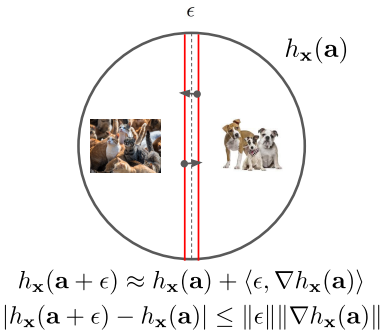
Remarks:

- PGA results in more powerful adversarial "attacks" than FGSM [16].
- The PGA is incorrectly referred to as projected gradient descent in this literature.
- Practitioners prefer to use several steps of FGSM instead of PGA [17, 18, 21]:

$$\boldsymbol{\delta}^{k+1} = \pi_{\mathcal{X}}\left(\boldsymbol{\delta}^k + \alpha_k \ \textbf{sign}\left(\nabla f(\boldsymbol{\delta}^k)\right)\right).$$

- See the appendix for a through study of the FSGM.

Challenge: Adversarial examples are inevitable



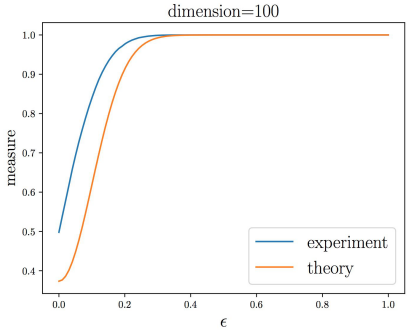
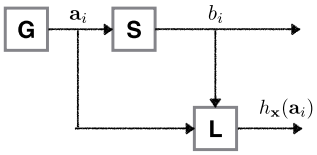


Figure: Understanding the robustness of a classifier in high-dimensional spaces. Shafahi et al. 2019.

Hardness results have never been a barrier for ML researchers



Definition (Empirical Risk Minimization (ERM))

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be samples with $b_i \in \{-1, 1\}$ and $\mathbf{a}_i \in \mathbb{R}^p$. The ERM problem reads

$$\min_{\mathbf{x}} \left\{ R_n(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i)$ is the loss on the sample $(\mathbf{a}_i, b_i)$.

Objectives

- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\|_\infty \leq \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\delta}\right), b_i\right) \right] \right\}$ — Adversarial training [14].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\|_2 \leq \epsilon} L(h_{\mathbf{x}+\boldsymbol{\delta}}\left(\mathbf{a}_i\right), b_i) \right] \right\}$ — ϵ-stability training [4], Sharpness-aware minimization [10].
- $\min_{\mathbf{x}} \max_{b^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \leq \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\delta}\right), b^c\right) \right]$ — Class fairness [25].

Remark: ○ We focus on adversarial training during the lecture. See supplementary material for more.

Towards adversarial training

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$, with the data $\mathbf{a}_i \in \mathbb{R}^p$ and the labels b_i. The problem of adversarial training is the following adversarial optimization problem

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i) \right] \approx \min_{\mathbf{x}} \mathbb{E}_{(\mathbf{a},b)\sim\mathbb{P}} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i) \right].$$

Note the similarity with the template $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$.

Beyond robustness: Adversarial training for better interpretability

- Retinopathy classification problem: Given a retinal image (left), predict whether there is a disease.
- **Zeiss:** How can we interpret the prediction of a model $h_{\mathbf{x}}(\mathbf{a})$?
- **Solution:** Look at $\nabla_{\mathbf{x}} h_{\mathbf{x}}(\mathbf{a})$, called the saliency map [9]. Minimax adversarial training seems to help!

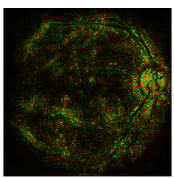
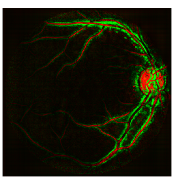




Table: **Left:** Ground truth image, **Middle:** Saliency map, **Right:** Saliency map with adversarial training.

Solving the outer problem

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$, with $\mathbf{a}_i \in \mathbb{R}^p$ and b_i be the corresponding labels. The adversarial training optimization problem is given by

$$\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i) \right]}_{=: f_i(\mathbf{x})} \right\}.$$

Note that L is not continuously differentiable due to ReLU, max-pooling, etc.

Solving the outer problem

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$, with $\mathbf{a}_i \in \mathbb{R}^p$ and b_i be the corresponding labels. The adversarial training optimization problem is given by

$$\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i) \right]}_{=: f_i(\mathbf{x})} \right\}.$$

Note that L is not continuously differentiable due to ReLU, max-pooling, etc.

Question

How can we compute the gradient

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) := \nabla_{\mathbf{x}} \left(\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i) \right)?$$

- **Challenge:** It involves differentiating with respect to a maximization.
- **A solution:** We can use Danskin's theorem under some conditions.

Danskin's Theorem (1966): How do we compute the gradient?

Theorem ([5])

Let $\mathcal{S}$ be compact set, $\Phi : \mathbb{R}^p \times \mathcal{S}$ be continuous such that $\Phi(\cdot, \mathbf{y})$ is differentiable for all $\mathbf{y} \in \mathcal{S}$, and $\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y})$ be continuous on $\mathbb{R}^p \times \mathcal{S}$. Define

$$f(\mathbf{x}) \coloneqq \max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}), \qquad \mathcal{S}^\star(\mathbf{x}) \coloneqq \arg\max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}).$$

Let $\mathbf{d} \in \mathbb{R}^p$, and $\|\mathbf{d}\|_2 = 1$. The directional derivative $D_{\mathbf{d}} f(\bar{\mathbf{x}})$ of f in the direction $\mathbf{d}$ at $\bar{\mathbf{x}}$ is given by

$$D_{\mathbf{d}} f(\bar{\mathbf{x}}) = \max_{\mathbf{y} \in \mathcal{S}^\star(\bar{\mathbf{x}})} \langle \mathbf{d}, \nabla_{\mathbf{x}} \Phi(\bar{\mathbf{x}}, \mathbf{y}) \rangle.$$

An immediate consequence

If $\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b_i)$ is unique, then we have

$$\nabla_{\mathbf{x}} f_i(\mathbf{x}) = \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}^\star), b_i).$$

A practical implementation of adversarial training: Stochastic subgradient descent

Stochastic Adversarial Training [21]

Input: learning rate α_k, iterations T, batch size K.

1. initialize neural network parameters $\mathbf{x}^0$

2. **For** $k = 0, 1, ..., T$:

i. initialize update vector $\mathbf{g}^k := 0$

ii. select a mini-batch of data $B \subset \{1, \dots, n\}$ with $|B| = K$

iii. **For** $i \in B$:

a. Find an attack $\boldsymbol{\delta}^\star$ by (approximately) solving

$$\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}^k}(\mathbf{a}_i + \boldsymbol{\delta}), b_i)$$

b. Store update

$$\mathbf{g}^k := \mathbf{g}^k + \nabla_{\mathbf{x}} L(h_{\mathbf{x}^k}(\mathbf{a}_i + \boldsymbol{\delta}^\star), b_i)$$

iv. Update parameters

$$\mathbf{x}^{k+1} := \mathbf{x}^k - \frac{\alpha_k}{K}\mathbf{g}^k$$

Remarks:

- Expensive!
- Inner problem **iii.a** cannot be solved to optimality (non-convex).
- Practitioners use FGSM or PGA or PGA-ℓ_∞ to approximate the true $\boldsymbol{\delta}^\star$.
- Update in step **iii.b** is motivated by Corollary A.2 in [21]

Optimized perturbations are typically not unique!

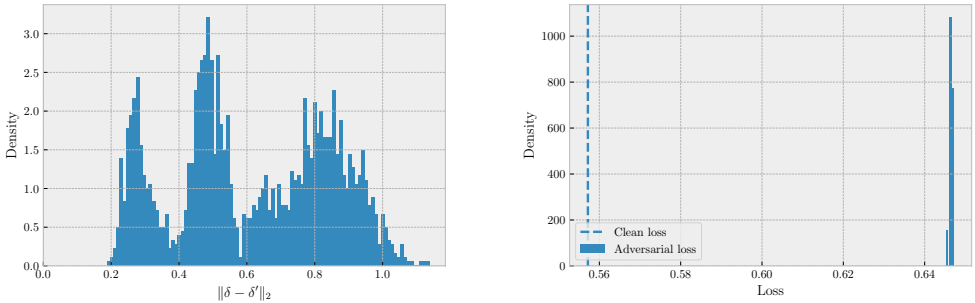


Figure: (*left*) Pairwise ℓ_2-distances between "optimized" perturbations with different initializations are bounded away from zero. (*right*) The losses of multiple perturbations on the same sample concentrate around a value much larger than the clean loss.

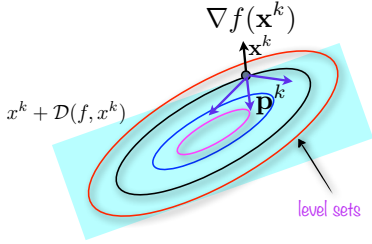
Theoretical foundations

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x},\delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [21]

Published as a conference paper at ICLR 2018

Towards Deep Learning Models Resistant to Adversarial Attacks

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu*
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu



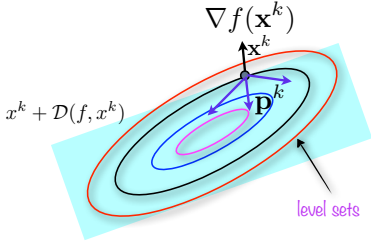
Theoretical foundations ?

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x},\delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [21]

Published as a conference paper at ICLR 2018

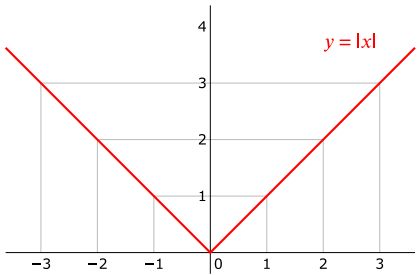
TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu*
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu



A counterexample

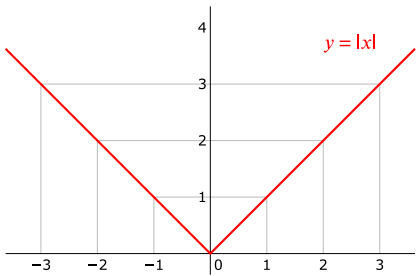
$$f(\mathbf{x}) \coloneqq \max_{\boldsymbol{\delta} \in [-1,1]} \mathbf{x}\boldsymbol{\delta} = |\mathbf{x}| \,.$$



- We have $\mathcal{S} \coloneqq [-1, 1]$ and $\Phi(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{x}\boldsymbol{\delta}$.
- At $\mathbf{x} = 0$, we have $\mathcal{S}^\star(0) = [-1, 1]$.
- We can choose $\boldsymbol{\delta} = 1 \in \mathcal{S}^\star(0)$: $\Phi(\mathbf{x}, 1) = \mathbf{x}$.

A counterexample

$$f(\mathbf{x}) \coloneqq \max_{\boldsymbol{\delta} \in [-1,1]} \mathbf{x}\boldsymbol{\delta} = |\mathbf{x}| \,.$$



- We have $\mathcal{S} \coloneqq [-1, 1]$ and $\Phi(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{x}\boldsymbol{\delta}$.
- At $\mathbf{x} = 0$, we have $\mathcal{S}^\star(0) = [-1, 1]$.
- We can choose $\boldsymbol{\delta} = 1 \in \mathcal{S}^\star(0)$: $\Phi(\mathbf{x}, 1) = \mathbf{x}$.
  - $-\nabla_{\mathbf{x}} \Phi(0, 1) = -1 \neq 0$.
  - Is -1 a descent direction at $\mathbf{x} = 0$?

Descent directions in the non-convex case

General Danskin's Theorem

Assume $\mathcal{Y}$ is compact and $\Phi(\mathbf{x}, \mathbf{y})$ differentiable in $\mathbf{x}$ but not necessarily convex in $\mathbf{x}$. Define $\mathcal{Y}^\star(\mathbf{x}) := \arg\max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ as the set of maximizers. Then $f(\mathbf{x}) := \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ is *directionally differentiable* and its directional derivative is given by

$$Df(\mathbf{x}, \mathbf{d}) = \max_{\mathbf{y}^\star \in \mathcal{Y}^\star(\mathbf{x})} \langle \mathbf{d}, \nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star) \rangle \tag{1}$$

Corollary A.2 in [21] (proven wrong!)

Let $\mathbf{y}_0^\star$ be a maximizer of $\max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$. Then as long as $\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}_0^\star)$ is non-zero, $-\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}_0^\star)$ is a descent direction for $f(\mathbf{x})$.

Remarks:

- The notion of directional derivative is one-sided:

$$Df(\mathbf{x}, \mathbf{d}) \coloneqq \lim_{t \to 0^+} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t} \tag{2}$$

- Only when $\mathcal{Y}^\star(\mathbf{x}) = \{\mathbf{y}^\star\}$ is a singleton, $-\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star)$ is *necessarily* a descent direction f.

Directional derivatives, not descent directions

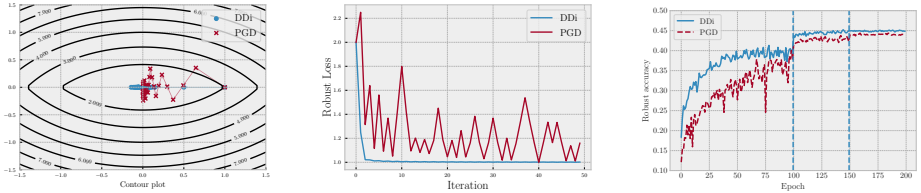


Figure: **(Left and Middle)** Synthetic adversarial training example. **(Right)** Resnet18 on CIFAR10 - Robust accuracy comparison between PGD and DDi.

Solving the inner problem does not yield a descent direction

Danskin's Theorem involves all the maximizers when computing the directional derivative along a direction $\mathbf{d}$. A single maximizer is *not* sufficient.

Remarks:

- A recent approach (DDi) computes many maximizers to find a descent direction [19].
- In practice however, the lack of descent does not seem to matter.

Danskin's theorem

Danskin's theorem (Bertsekas variant)

Let $\Phi(\mathbf{x}, \mathbf{y}) : \mathbb{R}^p \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{Y} \subset \mathbb{R}^m$ is a compact set and define $f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$. Suppose that $\Phi(\mathbf{x}, \mathbf{y})$ is convex for each $\mathbf{y}$ in the compact set $\mathcal{Y}$; the interior of the domain of f is nonempty; and $\Phi(\mathbf{x}, \mathbf{y})$ is continuous.

Define $\mathcal{Y}^\star(\mathbf{x}) := \arg\max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ as the set of maximizers and $\mathbf{y}^\star \in \mathcal{Y}^\star$ as an element of this set. We have

1. $f(\mathbf{x})$ is a convex function.
2. If $\mathcal{Y}^\star(\mathbf{x})$ is a singleton, then the function $f(\mathbf{x}) = \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ is differentiable at $\mathbf{x}$:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \nabla_{\mathbf{x}} \left(\max_{\mathbf{y} \in \mathcal{Y}} \phi(\mathbf{x}, \mathbf{y}) \right) = \nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star).$$

3. If $\mathcal{Y}^\star(\mathbf{x})$ contains more than one element, then the subdifferential $\partial_{\mathbf{x}} f(\mathbf{x})$ of f is given by

$$\partial_{\mathbf{x}} f(\mathbf{x}) = \operatorname{conv} \left\{ \partial_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star) : \mathbf{y}^\star \in \mathcal{Y}^\star(\mathbf{x}) \right\}.$$

Remarks:

- The adversarial problem is not convex in $\mathbf{x}$ in general.
- (Sub)Gradients of f are calculated as $\nabla_{\mathbf{x}} f(\mathbf{x}) = \nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star)$.

Out of the frying pan into the fire

Original Formulation of Adversarial Training (I)

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

Original Formulation of Adversarial Training (I)

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

which loss L?

Original Formulation of Adversarial Training (II)

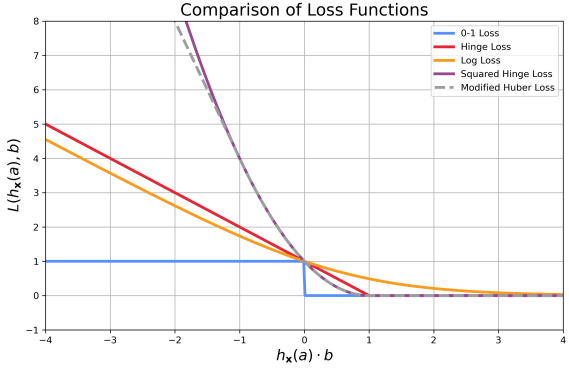
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

Original Formulation of Adversarial Training (II)

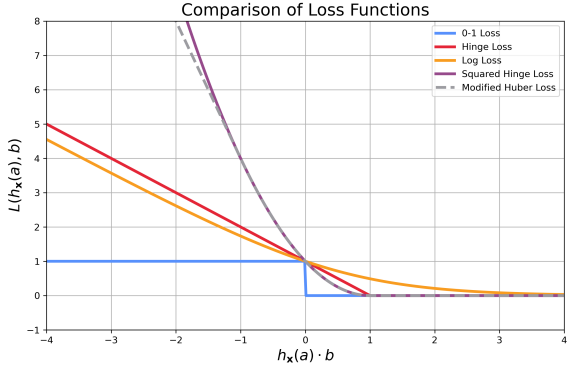
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{\mathsf{CE}}(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

Surrogate-based optimization for Risk Minimization



Surrogate-based optimization for Risk Minimization

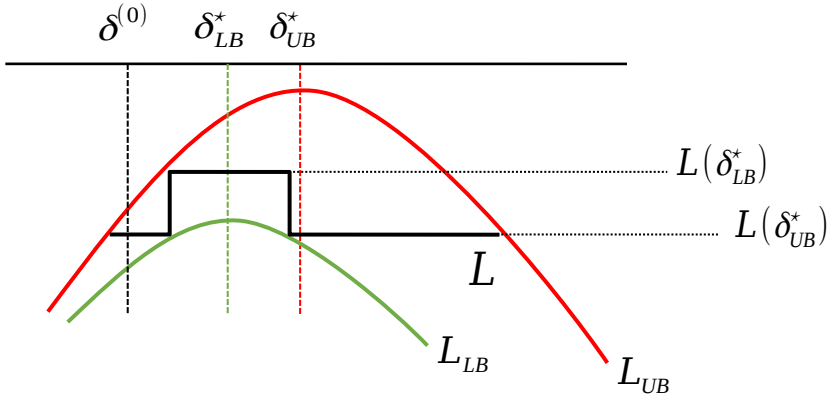


$$\mathbb{E}\left[L_{01}(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)\right] \leq \min_{\mathbf{x}} \mathbb{E}\left[L_{\mathsf{CE}}\left(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b\right)\right]$$

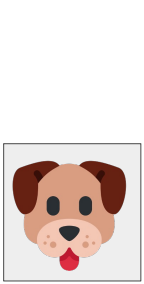
Adversary maximizes an upper bound (I)

$$L_{01}\left(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}^{\star}),b\right) \leq \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{\mathsf{CE}}\left(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}),b\right)$$

Adversary maximizes an upper bound (II)



Why maximizing cross-entropy leads to weak adversaries



$\mathbf{a}$

$+ \quad \delta_A \quad = \quad (.49, .51, \quad 0, \quad 0)$

$L_{\mathrm{CE}}(h_{\mathbf{x}}(\mathbf{a} + \delta_A), 🐶) = 1.18$

$+ \quad \delta_B \quad = \quad (.26, .24, \quad .25, \quad .25)$

$L_{\mathrm{CE}}(h_{\mathbf{x}}(\mathbf{a} + \delta_B), 🐶) = 1.38$

Adversary's problem can be "solved" without using surrogates

Theorem (Reformulation of the Adversary's problem)

$$\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} \max_{j\neq b} h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta})_j - h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta})_b \Rightarrow$$

$$\boldsymbol{\delta}^\star \in \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}\left(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b\right)$$

Bilevel Optimization (BETA)

- Best targeted attack (BETA) optimization formulation [27]:

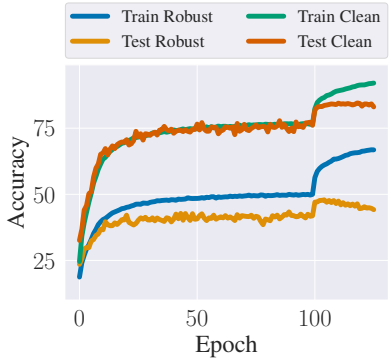
$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} L_{\mathsf{CE}} \left(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}^{\star}_{i,j^{\star}}), b_i \right)$$

$$\text{such that } \boldsymbol{\delta}^{\star}_{i,j} \in \underset{\boldsymbol{\delta}:\, \|\boldsymbol{\delta}\| \leq \epsilon}{\arg\max}\, h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta})_j - h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta})_{b_i}$$

$$j^{\star} \in \underset{j \in [K] - \{b_i\}}{\arg\max}\, h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^{\star}})_j - h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^{\star}})_{b_i}$$

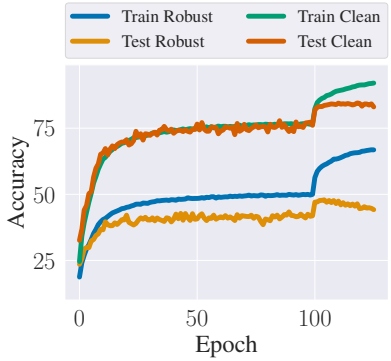
Practical Consequences of the Bilevel Formulation

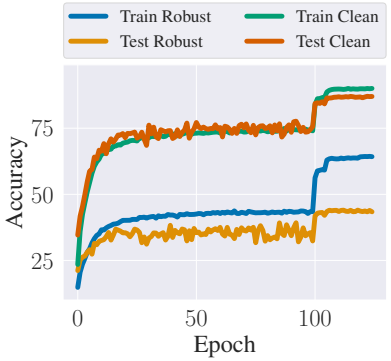
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT



Practical Consequences of the Bilevel Formulation

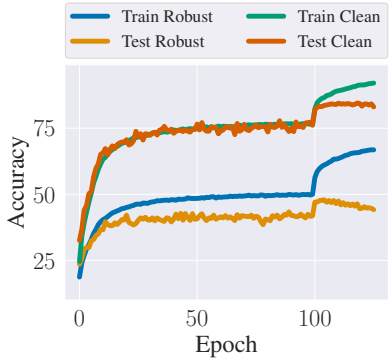
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT (Right). Robust accuracy estimated with PGD^{20}

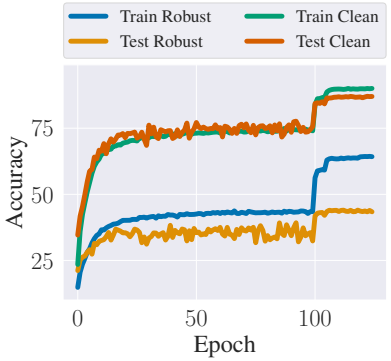




Practical Consequences of the Bilevel Formulation

Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT (Right). Robust accuracy estimated with PGD^{20}





No Robust Overfitting occurs!

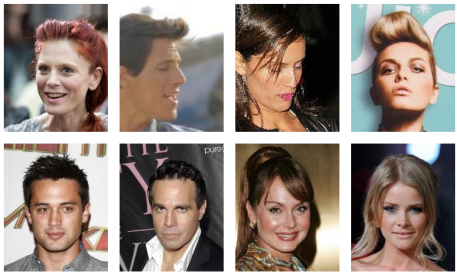
Practical Consequences of the Bilevel Formulation

Table: Adversarial performance on CIFAR-10.

Training algorithm	Test accuracy											
	Clean		FGSM		PGD^{10}		PGD^{40}		$BETA^{10}$		APGD	
	Best	Last	Best	Last	Best	Last	Best	Last	Best	Last	Best	Last
FGSM	81.96	75.43	**94.26**	**94.22**	42.64	1.49	42.66	1.62	40.30	0.04	41.56	0.00
PGD^{10}	83.71	83.21	51.98	47.39	46.74	39.90	45.91	39.45	43.64	40.21	44.36	42.62
$TRADES^{10}$	81.64	81.42	52.40	51.31	47.85	42.31	47.76	42.92	44.31	40.97	43.34	41.33
$MART^{10}$	78.80	77.20	53.84	53.73	49.08	41.12	48.41	41.55	44.81	41.22	45.00	42.90
$BETA\text{-}AT^{5}$	**87.02**	**86.67**	51.22	51.10	44.02	43.22	43.94	42.56	42.62	42.61	41.44	41.02
$BETA\text{-}AT^{10}$	85.37	85.30	51.42	51.11	45.67	45.39	45.22	45.00	44.54	44.36	44.32	44.12
$BETA\text{-}AT^{20}$	82.11	81.72	54.01	53.99	**49.96**	**48.67**	**49.20**	**48.70**	**46.91**	**45.90**	**45.27**	**45.25**

Adversarial machine learning: Introduction to Generative Adversarial Networks (GANs)

- Recall the parametric density estimation setting

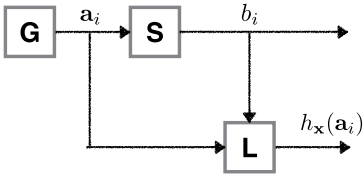


(source: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

$\mathbf{a}_i =$ [...images...]

$b_i =$ [...probability...]

- Goal: Games, denoising, image recovery...



- Generator $\mathbb{P}_{\mathbf{a}}$
  - Nature
- Supervisor $\mathbb{P}_{B|\mathbf{a}}$
  - Frequency data
- Learning Machine $h_{\mathbf{x}}(\mathbf{a}_i)$
  - Data scientist: Mathematics of Data

A way to model complex distributions: The push-forward measure

- Traditionally, we use analytical distributions: Restricts what we could model in real applications.
- Now, we use more expressive probability measures via *push-forward measures* with neural networks

Definition

- Let $\omega \sim \mathsf{p}_\Omega$ be a random variable.
- $h_{\mathbf{x}}(\cdot) : \mathbb{R}^p \to \mathbb{R}^m$ a function parameterized by parameters $\mathbf{x}$.

The pushforward measure of p_Ω under $h_{\mathbf{x}}$, denoted by $h_{\mathbf{x}}\#\mathsf{p}_\Omega$ is the distribution of $h_{\mathbf{x}}(\omega)$.

Example: Chi-square distribution

Let $\omega \sim \mathsf{p}_\Omega := \mathcal{N}(0,1)$ be the normal distribution. Let $h_x : \mathbb{R} \to \mathbb{R}$, $h_x(\omega) = w^x$. Let us fix $x = 2$. Then, $h_x\#\mathsf{p}_\Omega$ is the chi-square distribution with one degree of freedom.

Explanation: Change of variables.

Assume that $h : \mathbb{R}^n \to \mathbb{R}^n$ is monotonic. Given the random variable $\omega \sim \mathsf{p}_\Omega$ with probability density function $\mathsf{p}_\Omega(\omega)$, the density $\mathsf{p}_Y(\mathbf{y})$ of $\mathbf{y} = h_{\mathbf{x}}(\omega)$ reads

$$\mathsf{p}_Y(\mathbf{y}) = \mathsf{p}_\Omega(h_{\mathbf{x}}^{-1}(\mathbf{y}))\det\left(\mathbf{J}_{\mathbf{y}} h_{\mathbf{x}}^{-1}(\mathbf{y})\right)$$

where det denotes the determinant operation.

Towards an optimization problem

Problem (Ideal parametric density estimator)

Given a true distribution $\mu^\natural$, we can solve the following optimization problem,

$$\min_{\mathbf{x}} W_1(\mu^\natural, h_{\mathbf{x}} \# p_\Omega), \tag{3}$$

where the measurable function $h_{\mathbf{x}}$ is parameterized by $\mathbf{x}$ and $\omega \sim p_\Omega$ is "simple" e.g., Gaussian.

Remarks:

- See the appendix for the details of the Wasserstein distance W.
- Issues:
  - We only have access to empirical samples $\hat{\mu}_n$ of $\mu^\natural$.
  - W_1 is non-smooth, it cannot be computed exactly.

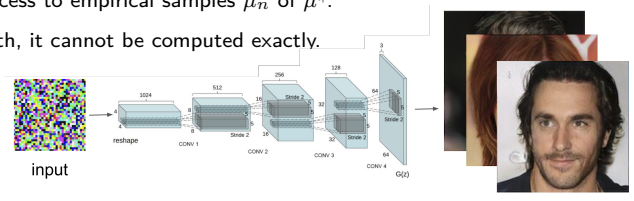
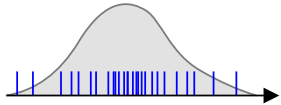


Figure: Schematic of a generative model, $h_{\mathbf{x}} \# \omega$ [11, 15].

Learning without concentration

- We can minimize $W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega\right)$ with respect to $\mathbf{x}$.
- Figure: Empirical distribution (blue), $\hat{\mu}_n = \sum_{i=1}^n \delta_i$



A plug-in empirical estimator

Using the triangle inequality for Wasserstein distances we can upper bound in the follow way,

$$W_1(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega) \leq W_1(\mu^\natural, \hat{\mu}_n) + W_1(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega), \tag{4}$$

where $\hat{\mu}_n$ is the empirical estimator of $\mu^\natural$ obtained from n independent samples from $\mu^\natural$.

Theorem (Slow convergence of empirical measures in 1-Wasserstein [30, 6])

Let $\mu^\natural$ be a measure defined on $\mathbb{R}^p$ and let $\hat{\mu}_n$ be its empirical measure. Then the $\hat{\mu}_n$ converges, in the worst case, at the following rate,

$$W_1(\mu^\natural, \hat{\mu}_n) \gtrsim n^{-1/p}. \tag{5}$$

Remarks:

- Using an empirical estimator in high-dimensions is terrible in the worst case.
- However, it does not directly say that $W_1\left(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega\right)$ will be large.
- So we can still proceed and hope our parameterization interpolates harmlessly.

Duality of 1-Wasserstein

- Instead of computing W_1, we can obtain lower bounds using duality.

Theorem (Kantorovich-Rubinstein duality)

$$W_1(\mu,\nu) = \sup_{\mathtt{d}} \{\langle \mathtt{d},\mu\rangle - \langle \mathtt{d},\nu\rangle : \mathtt{d} \textit{ is 1-Lipschitz}\} \tag{6}$$

Remark:
- $\mathtt{d}$ is the "dual" variable. In the literature, it is commonly referred to as the "discriminator."

Inner product is an expectation

$$\langle \mathtt{d},\mu\rangle = \int \mathtt{d}\mathrm{d}\mu = \int \mathtt{d}(\mathbf{a})\mathrm{d}\mu(\mathbf{a}) = \boldsymbol{E}_{\mathbf{a}\sim\mu}\left[\mathtt{d}(\mathbf{a})\right]. \tag{7}$$

Kantorovich-Rubinstein duality applied to our objective

$$W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\omega\right) = \sup\left\{\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}[\mathtt{d}(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a}\sim h_{\mathbf{x}}\#\omega}[\mathtt{d}(\mathbf{a})] : \mathtt{d} \text{ is 1-Lipschitz}\right\} \tag{8}$$

Integral Probability Metrics

We can define a more general class of (semi)metrics in the space of probability distributions

Definition (Integral Probability Metric)

Let $\mathcal{F}$ be a class of functions from $\mathbb{R}^p$ to $\mathbb{R}$. For two probability measures μ and ν, the IPM associated to $\mathcal{F}$ is defined as:

$$\mathcal{F}(\mu, \nu) \coloneqq \sup_{f \in \mathcal{F}} \langle f, \mu \rangle - \langle f, \nu \rangle = \sup_{f \in \mathcal{F}} \boldsymbol{E}_{\mathbf{a} \sim \mu}[f(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a} \sim \nu}[f(\mathbf{a})] \tag{9}$$

Remarks:

- The 1-Wasserstein distance corresponds to $\mathcal{F} \coloneqq \{f : \mathbb{R}^p \to \mathbb{R}, f \text{ is } 1 - \text{Lipschitz}\}$
- The class cannot be described with finite parameters.

Neural network distances inspired by the 1-Wasserstein distance

- We use neural networks to parametrize a class of functions.
- Constraining the Lipschitz constant of Neural Networks is NP-Hard [29].
- We can constrain upper bounds on the Lipschitz constant [20].

Lemma

Let $h_{\mathbf{X}_1,\mathbf{X}_2}(\mathbf{a}) \coloneqq \mathbf{X}_2^T \sigma(\mathbf{X}_1\mathbf{a})$ be a one-hidden-layer neural network. Then its Lipschitz constant $L_{\mathbf{X}_1,\mathbf{X}_2}$ with respect to the ℓ_2-norm is bounded as:

$$L_{\mathbf{X}_1,\mathbf{X}_2} \leq \|\mathbf{X}_1\|_2 \|\mathbf{X}_2\|_2 \tag{10}$$

Neural Network Distance

Let

$$\mathcal{F} \coloneqq \{h_{\mathbf{X}_1,\mathbf{X}_2}(\mathbf{a}) = \mathbf{X}_2^T \sigma(\mathbf{X}_1\mathbf{a}) : \|\mathbf{X}_2\|_2 \leq 1, \|\mathbf{X}_1\|_2 \leq 1\}. \tag{11}$$

The IPM corresponding to $\mathcal{F}$ is referred to as a *Neural Network Distance*.

Remark: ○ Different network architectures/constraints lead to different Neural Network distance notions.

Wasserstein GANs formulation

- Ingredients:
  - fixed *noise* distribution p_Ω (e.g., normal)
  - target distribution $\hat{\mu}_n$ (natural images)
  - $\mathcal{X}$ parameter class inducing a class of functions (generators)
  - $\mathcal{Y}$ parameter class inducing a class of functions (dual variables)

Wasserstein GANs formulation [3]

Define a parameterized function $\mathsf{d}_{\mathbf{y}}(\mathbf{a})$, where $\mathbf{y} \in \mathcal{Y}$ such that $\mathsf{d}_{\mathbf{y}}(\mathbf{a})$ is 1-Lipschitz. In this case, the Wasserstein GAN optimization problem is given by

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathsf{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\mathsf{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right). \tag{12}$$

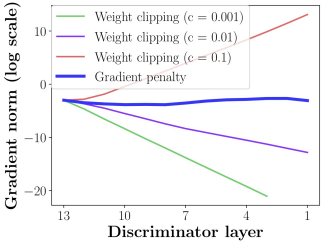
The theory-practice gap: Enforcing 1-Lipschitz of the discriminator

Weight clipping [3]

The "dual" or the "discriminator" $\mathtt{d}_{\mathbf{y}}$ weights $\mathbf{y}$ are constrained by an ℓ_∞-ball with radius $c > 0$, denoted as $\mathcal{B}$, at every iteration with

$$\pi_{\mathcal{B}}(\mathbf{y}) = \mathrm{clip}(\mathbf{y}, [-c, c]). \tag{13}$$

This trick is used to pseudo-enforce the constraint.



Remark: ○ *"Weight clipping is a clearly terrible way to enforce a Lipschitz constraint"* – original authors.

Gradient penalty [13]

Recall that 1-Lipschitz is equivalent to $\|\nabla_{\mathbf{a}} \mathtt{d}_{\mathbf{y}}(\mathbf{a})\|_* \leq 1$. This can be enforced directly through

$$\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}\left[\mathtt{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\Omega}\left[\mathtt{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] + \lambda \boldsymbol{E}_{\mathbf{a}\sim\nu}\left[\left(\|\nabla_{\mathbf{a}} \mathtt{d}_{\mathbf{y}}(\mathbf{a})\|_* - 1\right)^2\right]. \tag{14}$$

Remarks:

○ In practice the distribution ν mimicks uniform (linearly interpolated) sampling as follows: $\mathbf{a} \sim \mathrm{Uniform}(\mathbf{a}_i, h_{\mathbf{x}}(\boldsymbol{\omega}_i))$.

○ Spectral normalization: Divide each weight matrix by their spectral norm [22].

○ Learnable spline activations: both a 1-Lipschitz and more expressive architecture [24].

Practical implementation of GANs

Stochastic training of Wasserstein GANs

Input: primal and "dual" learning rates γ_t and α_m, primal iterations T, "dual" network $\mathsf{d}_{\mathbf{y}}$, generator network $h_{\mathbf{x}}$, noise distribution p_Ω, real distribution $\hat{\mu}_n$, primal and dual batch sizes B, K, "dual" iterations M.

1. initialize $\mathbf{x}^0$
2. **For** $t = 0, 1, ..., T-1$:
   - **For** $m = 0, 1, ..., M-1$:
     - initialize $\mathbf{y}^0$,
     - draw noise sample $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K \sim \mathsf{p}_\Omega$
     - draw real samples $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_K \sim \hat{\mu}_n$
     - "dual" pseudo-loss $L(\mathbf{y}) := K^{-1} \sum_{i=1}^K \mathsf{d}_{\mathbf{y}}(\boldsymbol{r}_i) - \mathsf{d}_{\mathbf{y}}(h_{\mathbf{x}^t}(\boldsymbol{\omega}_i))$
     - $\sharp$update "dual" parameters $\mathbf{y}^{m+1} = \mathbf{y}^m + \gamma_m \nabla_{\mathbf{y}} L(\mathbf{y}^m)$
     - $\sharp$enforce 1-Lipschitz constraint on $\mathsf{d}_{\mathbf{y}^{m+1}}$
   - **end-For**
   - draw noise sample $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_B \sim \mathsf{p}_\Omega$
   - generator pseudo-loss $L(\mathbf{x}) := -B^{-1} \sum_{i=1}^B \mathsf{d}_{\mathbf{y}^M}(h_{\mathbf{x}}(\boldsymbol{\omega}_i))$
   - update generator parameters $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \nabla_{\mathbf{x}} L(\mathbf{x}^t)$
3. **end-For**

$\sharp$: Ideally, should be performed jointly.

Some historical background for a Turing award

Vanilla GAN [11]

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\log \mathtt{d}_{\mathbf{y}}(\mathbf{a})\right] + \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\log\left(1 - \mathtt{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right)\right] \tag{15}$$

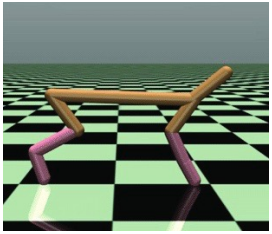
- Binary cross-entropy modeling.
- $\mathtt{d}_{\mathbf{y}}(\mathbf{a}) : \mathcal{Y} \to [0,1]$ represents the probability that $\mathbf{a}$ came from the real data distribution $\mu^\natural$.

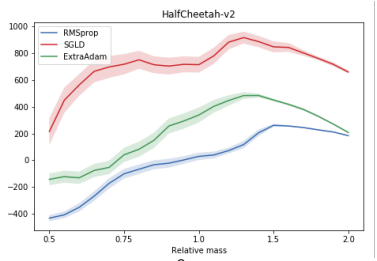
Observation: ○ Minimizes Jensen-Shannon divergence:

$$\mathrm{JSD}(\hat{\mu}_n \| h_{\mathbf{x}} \# \mathsf{p}_\Omega) = \frac{1}{2} D(\hat{\mu}_n \| h_{\mathbf{x}} \# \mathsf{p}_\Omega) + \frac{1}{2} D(h_{\mathbf{x}} \# \mathsf{p}_\Omega \| \hat{\mu}_n).$$

Take home messages

- Even the simplified view of robust & adversarial ML is challenging
- $\min$-$\max$-type has spurious attractors with no equivalent concept in $\min$-type
- Other successful attempts[1] consider "mixed Nash" concepts[2]





- Existing theory and methods for adversarial training is wrong! **... SAM too...**[3]

[1] Y-P. Hsieh, C. Liu, and V. Cevher, "Finding mixed Nash equilibria of generative adversarial networks," International Conference on Machine Learning, 2019.

[2] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics," NeurIPS, 2020.

[3] W. Xie, F. Latorre, K. Antonakopoulos, T. Pethick, and V. Cevher "Improving SAM requires rethinking its optimization formulation," ICLR, 2024.

Wrap up!

- Continuing on Homework 2!

A robustness example: Linear prediction

Linear model

Consider a linear model $h_{\mathbf{x}^\star}(\mathbf{a}) = \langle \mathbf{x}^\star, \mathbf{a} \rangle$ with weights $\mathbf{x}^\star \in \mathbb{R}^p$, for some input $\mathbf{a}$.

An adversarial perturbation

We aim at finding the perturbation $\boldsymbol{\delta} \in \mathbb{R}^p$ subject to $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ that produces the largest change on $h_{\mathbf{x}^\star}(\mathbf{a})$:

$$\begin{aligned}
\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}) &= \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \langle \mathbf{x}^\star, \mathbf{a}+\boldsymbol{\delta} \rangle \\
&= \langle \mathbf{x}^\star, \mathbf{a} \rangle + \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \langle \mathbf{x}^\star, \boldsymbol{\delta} \rangle && \triangleright \text{ As } \mathbf{a} \text{ does not influence the optimization.} \\
&= \langle \mathbf{x}^\star, \mathbf{a} \rangle + \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq 1} \langle \mathbf{x}^\star, \epsilon\boldsymbol{\delta} \rangle && \triangleright \text{ By the change of variables } \boldsymbol{\delta} := \boldsymbol{\delta}/\epsilon \\
&= \langle \mathbf{x}^\star, \mathbf{a} \rangle + \epsilon\|\mathbf{x}^\star\|_1 && \triangleright \text{ Definition of the dual norm } \|\mathbf{x}\|_1 := \max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq 1} \langle \mathbf{x}, \boldsymbol{\delta} \rangle
\end{aligned}$$

Taking $\boldsymbol{\delta}^\star = \operatorname{sign}(\mathbf{x}^\star)$ achieves this maximum: $\langle \mathbf{x}, \epsilon \operatorname{sign}(\mathbf{x}^\star) \rangle = \epsilon \sum_{i=1}^n \operatorname{sign}(x_i^\star) x_i^\star = \epsilon \sum_{i=1}^n |x_i^\star| = \epsilon\|\mathbf{x}^\star\|_1$.

Remarks:

- For the linear model, we have $\nabla_{\mathbf{a}} h_{\mathbf{x}^\star}(\mathbf{a}) = \mathbf{x}^\star$.
- *The gradient sign* of $h_{\mathbf{x}^\star}$ with respect to the input $\mathbf{a}$ achieves the worst perturbation.
- Sparse models are robust in linear prediction.

Adversarial examples in neural networks

- Target problem:

$$\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)$$

- Historically, researchers first tried to find approximate solutions that empirically perform well [12, 21].

Fast Gradient Sign Method (FGSM) [12]

Let $h_{\mathbf{x}^\star} : \mathbb{R}^p \to \mathbb{R}$ be a model trained through empirical risk minimization on the loss L, with optimal parameters $\mathbf{x}^\star$. Let $(\mathbf{a}, b)$ be a sample with $b \in \{-1, 1\}$ and $\mathbf{a} \in \mathbb{R}^p$. The *Fast Gradient Sign Method* computes the adversarial example

$$\boldsymbol{\delta} = \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right) = \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} h_{\mathbf{x}^\star}(\mathbf{a}) \nabla_h L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right)$$

Remarks:

- The FGSM obtains adversarial examples by using *sign of the gradient of the loss*.
- Such an approach can be viewed as a linearization of the objective L around the data $\mathbf{a}$.
- For single output $h_{\mathbf{x}}(\mathbf{a})$, $\nabla_h L(h_{\mathbf{x}^\star}(\mathbf{a}), b)$ is a scalar,
  - $\text{sign}\left(\nabla_{\mathbf{a}} h_{\mathbf{x}^\star}(\mathbf{a})\right)$ pattern is important

Results of FGSM on MNIST

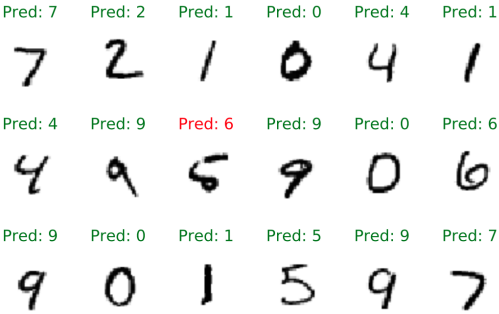


Figure: MNIST images with the predicted digit.

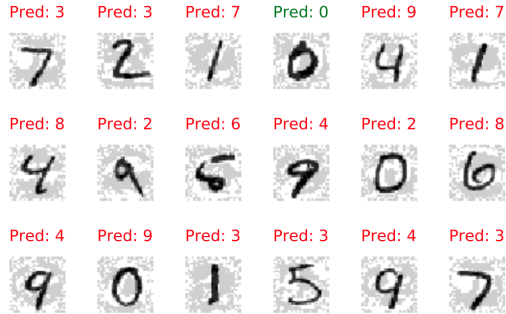


Figure: MNIST images perturbed by a FGSM attack.

Taken from `https://adversarial-ml-tutorial.org/adversarial_examples/`

A proposed link between FGSM and PGA

- Recall
  - A single step of PGA reads $\boldsymbol{\delta}_{\text{PGA}}^{k+1} := \pi_{\mathcal{N}}\left(\boldsymbol{\delta}^k + \alpha \nabla f(\boldsymbol{\delta})\right)$
  - The FGSM attack is defined as $\boldsymbol{\delta}_{\text{FGSM}} := \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right)$
  - When $\mathcal{N} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \lambda\}$, $\pi_{\mathcal{N}}(\boldsymbol{\delta}) = \text{clip}(\boldsymbol{\delta}, [-\lambda, \lambda])$

FGSM as one step of PGA

Let $\boldsymbol{\delta}^0 = \mathbf{0}$ and $\alpha > 0$ such that $(\alpha \left|\nabla f(\mathbf{0})\right|)_i > \epsilon$ for $i = 1, \ldots, n$. Then, one step of PGA yields

$$
\begin{aligned}
\boldsymbol{\delta}_{\text{PGA}}^1 &= \pi_{\mathcal{N}}\left(\boldsymbol{\delta}^0 + \alpha \nabla_{\boldsymbol{\delta}} \nabla f(\boldsymbol{\delta}^0)\right) \\
&= \text{clip}\left(\alpha \nabla f(\mathbf{0}), [-\epsilon, \epsilon]\right) && \triangleright\ \boldsymbol{\delta}^0 = \mathbf{0} \\
&= \epsilon \ \text{sign}\left(\nabla f(\mathbf{0})\right) && \triangleright\ \text{All values are outside of the interval } [-\epsilon, \epsilon] \\
&= \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right) = \boldsymbol{\delta}_{\text{FGSM}} && \triangleright\ \nabla f(\mathbf{0}) = \nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)
\end{aligned}
$$

A proposed link between FGSM and PGA

- Recall
  - A single step of PGA reads $\boldsymbol{\delta}_{\text{PGA}}^{k+1} := \pi_{\mathcal{N}}\left(\boldsymbol{\delta}^k + \alpha \nabla f(\boldsymbol{\delta})\right)$
  - The FGSM attack is defined as $\boldsymbol{\delta}_{\text{FGSM}} := \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right)$
  - When $\mathcal{N} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \lambda\}$, $\pi_{\mathcal{N}}(\boldsymbol{\delta}) = \text{clip}(\boldsymbol{\delta}, [-\lambda, \lambda])$

FGSM as one step of PGA

Let $\boldsymbol{\delta}^0 = \mathbf{0}$ and $\alpha > 0$ such that $(\alpha \left|\nabla f(\mathbf{0})\right|)_i > \epsilon$ for $i = 1, \ldots, n$. Then, one step of PGA yields

$$
\begin{aligned}
\boldsymbol{\delta}_{\text{PGA}}^1 &= \pi_{\mathcal{N}}\left(\boldsymbol{\delta}^0 + \alpha \nabla_{\boldsymbol{\delta}} \nabla f(\boldsymbol{\delta}^0)\right) \\
&= \text{clip}\left(\alpha \nabla f(\mathbf{0}), [-\epsilon, \epsilon]\right) && \triangleright\ \boldsymbol{\delta}^0 = \mathbf{0} \\
&= \epsilon \ \text{sign}\left(\nabla f(\mathbf{0})\right) && \triangleright\ \text{All values are outside of the interval } [-\epsilon, \epsilon] \\
&= \epsilon \ \text{sign}\left(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)\right) = \boldsymbol{\delta}_{\text{FGSM}} && \triangleright\ \nabla f(\mathbf{0}) = \nabla_{\mathbf{a}} L(h_{\mathbf{x}^\star}(\mathbf{a}), b)
\end{aligned}
$$

Multiple steps of FGSM: A connection to majorization-minimization in Lecture 4

Minimization-majorization for concave functions

Let f be a concave function which is smooth in the ℓ_∞-norm with constant L_∞. Our target non-convex problem is given by

$$\max_{\boldsymbol{\delta}} f(\boldsymbol{\delta}) + \delta_{\mathcal{N}}(\boldsymbol{\delta})$$

where $\delta_{\mathcal{N}}(\boldsymbol{\delta})$ is the indicator function of the ball $\mathcal{N} := \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$. Smoothness in ℓ_∞-norm implies

$$f(\boldsymbol{\delta}) + \delta_{\mathcal{N}}(\boldsymbol{\delta}) \geq \underbrace{f(\boldsymbol{\zeta}) + \langle \nabla_{\boldsymbol{\delta}} f(\boldsymbol{\zeta}), \boldsymbol{\delta} - \boldsymbol{\zeta} \rangle - \frac{L_\infty}{2} \|\boldsymbol{\delta} - \boldsymbol{\zeta}\|_\infty^2 + \delta_{\mathcal{X}}(\boldsymbol{\delta})}_{\boldsymbol{\delta}^\star \leftarrow \arg\max_{\boldsymbol{\delta}}} .$$

Maximizing the RHS with respect to $\boldsymbol{\delta}$ leads to the following (non trivial) solution [7]:

$$\boldsymbol{\delta}^\star = \text{clip}\left(\boldsymbol{\zeta} - t^\star \text{sign}(\nabla f(\boldsymbol{\zeta})), [-\epsilon, \epsilon]\right)$$

where $t^\star = \arg\max_{t:\|\boldsymbol{\delta}-\boldsymbol{\zeta}\|_\infty \leq t} \max_{\boldsymbol{\zeta}:\|\boldsymbol{\zeta}\|_\infty \leq \epsilon} \langle \nabla f(\boldsymbol{\zeta}), \boldsymbol{\delta} - \boldsymbol{\zeta} \rangle$ can be found by linear search.

Remarks:

- Setting $\boldsymbol{\zeta} = \boldsymbol{\delta}^k$ and $\boldsymbol{\delta}^\star = \boldsymbol{\delta}^{k+1}$ with a fixed step size $\alpha = t^\star$, we obtain the update in [17, 18, 21]
$$\boldsymbol{\delta}^{k+1} = \text{clip}\left(\boldsymbol{\delta}^k - t^\star \text{sign}(\nabla f(\boldsymbol{\delta}^k)), [-\epsilon, \epsilon]\right).$$
- This proof holds for **concave** and smooth functions, and need further quantification for our setting.

A notion of distance between distributions

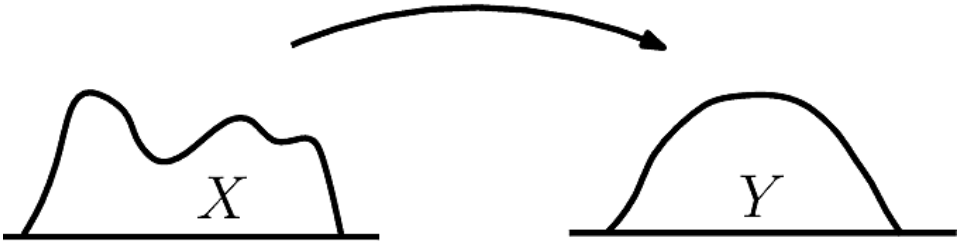


Figure: The Earth Mover's distance

Minimum cost transportation problem (Monge's problem)

Find a *transport map* $T : \mathbb{R}^d \to \mathbb{R}^d$ such that $T(X) \sim Y$, minimizing the cost

$$\text{cost}(T) := \boldsymbol{E}_X \| Y - T(X) \|. \tag{16}$$

The Wasserstein distance

Definition

Let μ and ν be two probability measures on $\mathbb{R}^d$. Their set of couplings is defined as

$$\Gamma(\mu,\nu) := \{\pi \text{ prob. measure on } \mathbb{R}^d \times \mathbb{R}^d \text{ with marginals } \mu,\nu\} \tag{17}$$

Definition (q-Wasserstein distance (Primal))

$$W_q(\mu,\nu) := \left(\inf_{\pi\in\Gamma(\mu,\nu)} \boldsymbol{E}_{(\mathbf{a},\mathbf{a}')\sim\pi} d(\mathbf{a},\mathbf{a}')^q\right)^{1/q} \tag{18}$$

where $q = 1, 2$ and d is a distance.

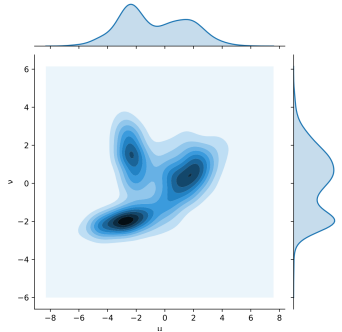


Figure: Two one-dimensional distributions plotted on the x and y axes, and one possible joint distribution that defines a transport plan between them (`https://en.wikipedia.org/wiki/Wasserstein_metric`).

Properties of the Wasserstein distance

- For any $q \geq 1$, the q-Wasserstein distance *is* a distance:
  - $W_q(\mu, \nu) = 0$ if and only if μ, ν have the same density almost everywhere (identity).
  - $W_q(\mu, \nu) = W_q(\nu, \mu)$ (symmetry).
  - $W_q(\mu, \rho) \leq W_q(\mu, \nu) + W_q(\nu, \rho)$ (triangle inequality).

Problem (Wasserstein Projection)

Given a target probability measure μ on $\mathbb{R}^d$ we are interested in solving the following optimization problem:

$$\min_{\nu \in \Delta} W_q(\mu, \nu), \tag{19}$$

where Δ is a set of probability measures on $\mathbb{R}^d$, and q is often selected as 1 or 2.

General diagram of GANs

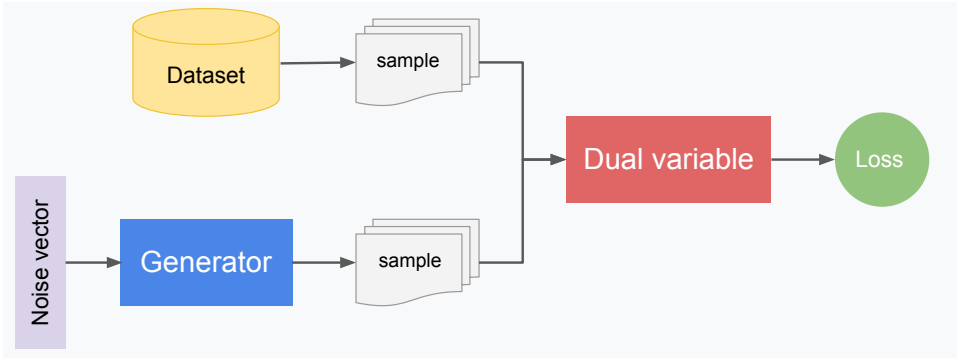


Figure: Generator/dual variable/dataset relation in GANs

*Sharpness-aware minimization (SAM) [10]

- Intuition: Flat minima usually generalizes better than sharp minima.

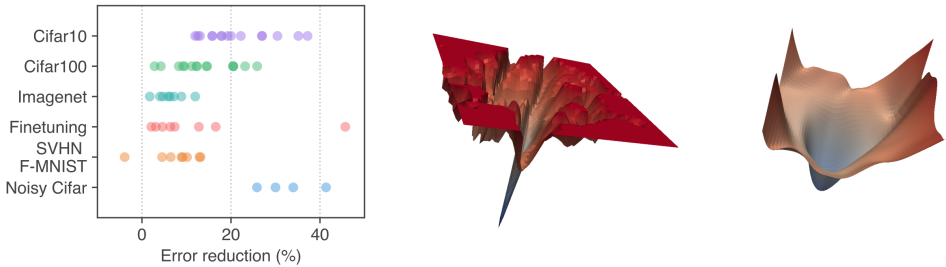


Figure: ResNet trained via SAM converges to a flatter minima **(Right)** compared with the one trained via SGD **(Middle)**, and thus leads to considerable error rate reduction **(Left)** [10].

*Sharpness-aware minimization (SAM) [10]

- Efficient approximation to the objective $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \leq \epsilon} L(h_{\mathbf{x}+\boldsymbol{\delta}}(\mathbf{a}_i), b_i) \right] \right\}$:
  - Let's first consider the the inner maximization problem. By first-order Taylor expansion, we have:

$$\boldsymbol{\delta}^\star = \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \leq \epsilon} L\left(h_{\mathbf{x}+\boldsymbol{\delta}}(\mathbf{a}_i), b_i\right) \approx \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \leq \epsilon} \left[L\left(h_{\mathbf{x}}(\mathbf{a}_i), b_i\right) + \boldsymbol{\delta}^\top \nabla_{\mathbf{x}} L\left(h_{\mathbf{x}}(\mathbf{a}_i), b_i\right) \right]$$

$$= \arg\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \leq \epsilon} \boldsymbol{\delta}^\top \nabla_{\mathbf{x}} L\left(h_{\mathbf{x}}(\mathbf{a}_i), b_i\right) = \epsilon \frac{\nabla_{\mathbf{x}} L\left(h_{\mathbf{x}}(\mathbf{a}_i), b_i\right)}{\|\nabla_{\mathbf{x}} L\left(h_{\mathbf{x}}(\mathbf{a}_i), b_i\right)\|_2}.$$

  - Plugging $\boldsymbol{\delta}^\star$ back the original objective and take the derivative:

$$\nabla_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2 \leq \epsilon} L(h_{\mathbf{x}+\boldsymbol{\delta}}(\mathbf{a}_i), b_i) \right] \right\} = \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_{\mathbf{x}} L(h_{\mathbf{x}+\boldsymbol{\delta}^\star}(\mathbf{a}_i), b_i) \right]$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left[(1 + \frac{d\boldsymbol{\delta}^\star}{d\boldsymbol{w}}) \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \mid_{\mathbf{x}+\boldsymbol{\delta}^\star} \right] \approx \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \mid_{\mathbf{x}+\boldsymbol{\delta}^\star} \right],$$

    where in the last equation the second-order term is dropped for accelerating the computation.

  - Thus, the parameters are updated by: $\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_{\mathbf{x}^k} L(h_{\mathbf{x}^k}(\mathbf{a}_i), b_i) \mid_{\mathbf{x}^k + \boldsymbol{\delta}^{\star k}} \right]$, where γ_k is a step-size.

SAM's update rule

SAM

The SAM update rule is given by:

$$\tilde{\mathbf{x}}^k = \mathbf{x}^k + \epsilon \frac{\nabla L(\mathbf{x}^k)}{\|\nabla L(\mathbf{x}^k)\|} \qquad \text{[Perturb weights]}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \nabla L(\tilde{\mathbf{x}}^k) \qquad \text{[Update step]}$$

where γ_k is the step-size, and $L(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i\right), b_i\right)$.

Remarks:

- SAM requires two gradient computations per update, which are sequentially dependent.
- The computational time of SAM is doubled compared with base optimizers (e.g. SGD).

Question:

- Can we run it as fast as base optimizers?

  Yes! Parallelize two gradient computations.

SAM Parallelized (SAMPa) [31]

SAMPa-λ

An auxiliary sequence $\mathbf{y}$ is introduced for parallel computation. The SAMPa update rule is given by:

$$\tilde{\mathbf{x}}^k = \mathbf{x}^k + \epsilon \frac{\nabla L(\mathbf{y}^k)}{\|\nabla L(\mathbf{y}^k)\|} \qquad \text{[Perturb weights]}$$

$$\mathbf{y}^{k+1} = \mathbf{x}^k - \gamma_k \nabla L(\mathbf{y}^k) \qquad \text{[Auxiliary sequence]}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (1-\lambda)\gamma_k \nabla L(\tilde{\mathbf{x}}^k) - \lambda \gamma_k \nabla L(\mathbf{y}^{k+1}) \qquad \text{[Update step]}$$

where $\lambda \in [0, 1]$. Note that $\nabla L(\tilde{\mathbf{x}}^k)$ and $\nabla L(\mathbf{y}^{k+1})$ are computed in parallel, incorporating optimistic gradient descent as follows:

$$y_{t+1} = x_t - \eta_t \nabla f(y_t), \quad x_{t+1} = x_t - \eta_t \nabla f(y_{t+1})$$

Table: **Resnet-56 with Efficient SAM variants on CIFAR-10.** The best result is in bold and the second best is underlined.

	SAM	SAMPa-0.2	LookSAM	AE-SAM	SAF	MESA	ESAM
Accuracy	94.26	**94.62**	91.42	<u>94.46</u>	93.89	94.23	94.21
Time/Epoch (s)	18.81	<u>10.94</u>	16.28	13.47	**10.09**	15.43	15.97

Fast gradient sign method (FGSM) [12]

Projected gradient descent (PGD) attack: A misnomer

Let $\boldsymbol{\eta}^{(0)} = \mathbf{0}$, the PGD update rule is given by:

$$\hat{\boldsymbol{\eta}}^{(t)} = \boldsymbol{\eta}^{(t-1)} + \alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{\eta}} L\left(h_{\mathbf{x}}\left(\mathbf{a} + \boldsymbol{\eta}^{(t-1)}\right), b\right)\right) \qquad \text{[Gradient step]}$$

$$\boldsymbol{\eta}^{(t)} = \max\left\{\min\left\{\hat{\boldsymbol{\eta}}^{(t)}, \epsilon\right\}, -\epsilon\right\}, \qquad \text{[Projection step]}$$

where α is the step-size and the procedure is ran for T steps. If $T = 1$ and $\alpha = \epsilon$ we recover the FGSM:

$$\boldsymbol{\eta}_{\text{FGSM}} = \epsilon \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{\eta}} L\left(h_{\mathbf{x}}\left(\mathbf{a}\right), b\right)\right)$$

Problems:

- In Adversarial Training: $\times T$ overhead in training time.
- If $T = 1$, we can observe **Catastrophic Overfitting (CO)**

Fast gradient sign method (FGSM) [12]

Projected gradient descent (PGD) attack: A misnomer

Let $\boldsymbol{\eta}^{(0)} = \mathbf{0}$, the PGD update rule is given by:

$$\hat{\boldsymbol{\eta}}^{(t)} = \boldsymbol{\eta}^{(t-1)} + \alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{\eta}} L\left(h_{\mathbf{x}}\left(\mathbf{a} + \boldsymbol{\eta}^{(t-1)}\right), b\right)\right) \qquad \text{[Gradient step]}$$

$$\boldsymbol{\eta}^{(t)} = \max\left\{\min\left\{\hat{\boldsymbol{\eta}}^{(t)}, \epsilon\right\}, -\epsilon\right\}, \qquad \text{[Projection step]}$$

where α is the step-size and the procedure is ran for T steps. If $T = 1$ and $\alpha = \epsilon$ we recover the FGSM:

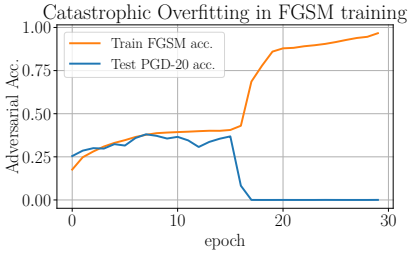
$$\boldsymbol{\eta}_{\text{FGSM}} = \epsilon \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{\eta}} L\left(h_{\mathbf{x}}\left(\mathbf{a}\right), b\right)\right)$$

Problems:

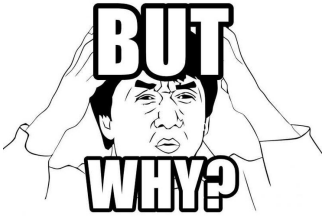
- In Adversarial Training: $\times T$ overhead in training time.
- If $T = 1$, we can observe **Catastrophic Overfitting (CO)**

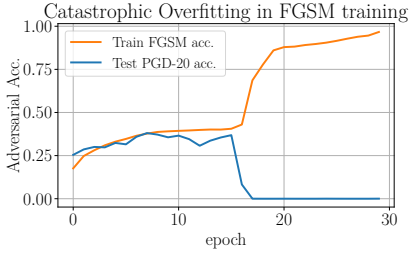
Example:

- PreActResNet18 on CIFAR10 at $\epsilon = 8/255$.
- Outcome: 100% robust to FGSM attacks and 0% robust to PGD-20 attacks.



More on CO





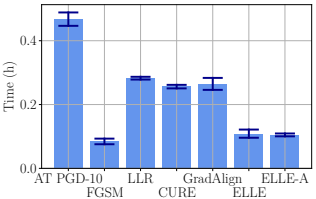
Why?

The single step solution $\boldsymbol{\eta}_{\text{FGSM}}$ only makes sense if our loss is locally linear, i.e.:

$$L\left(h_{\mathbf{x}^k}(\mathbf{a}+\boldsymbol{\eta}),b\right) \approx L\left(h_{\mathbf{x}^k}(\mathbf{a}),b\right) + \boldsymbol{\eta}^\top \nabla_{\mathbf{a}} L\left(h_{\mathbf{x}^k}(\mathbf{a}),b\right)\ , \quad \forall \boldsymbol{\eta} : ||\boldsymbol{\eta}||_\infty \leq \epsilon\,. \quad \text{[1st order Taylor expansion]}$$

Observation: This property is lost during AT with FGSM and CO appears [2].

The ELLE way [Abad Rocamora, Liu, Chrysos, Olmos and Cevher, ICLR 2024]



(a) Training time comparison

ϵ	8		16	
Method	AutoAttack	Clean	AutoAttack	Clean
LLR	$42.18 \pm (0.20)$	$75.02 \pm (0.09)$	$16.92 \pm (0.20)$	$42.81 \pm (9.62)$
CURE	$43.60 \pm (0.17)$	$77.74 \pm (0.11)$	$\underline{18.25} \pm (0.45)$	$52.49 \pm (0.04)$
GradAlign	$\mathbf{44.66} \pm (0.21)$	$\mathbf{80.50} \pm (0.07)$	$17.46 \pm (1.71)$	$44.35 \pm (15.32)$
ELLE	$42.78 \pm (0.95)$	$\underline{80.13} \pm (0.32)$	$\mathbf{18.28} \pm (0.17)$	$\mathbf{59.73} \pm (0.16)$
ELLE-A	$\underline{44.32} \pm (0.04)$	$79.81 \pm (0.10)$	$18.03 \pm (0.15)$	$\underline{59.21} \pm (1.23)$
AT PGD-10	$46.95 \pm (0.11)$	$79.11 \pm (0.08)$	$24.77 \pm (0.26)$	$59.64 \pm (0.46)$

(b) PreActResNet18 in CIFAR10

Algorithmic approaches:

- Local linearization (LLR) [26]
- Curvature regularization (CURE) [23]
- Gradient alignment (GradAlign) [2]
- Efficient local linearity regularization (ELLE) [1]

Overcoming CO with local linearity regularization [2, 1]

Why?

The single step solution $\boldsymbol{\eta}_{\text{FGSM}}$ only makes sense if our loss is locally linear, i.e.:

$$L\left(h_{\mathbf{x}^k}(\mathbf{a}+\boldsymbol{\eta}), b\right) \approx L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right) + \boldsymbol{\eta}^\top \nabla_{\mathbf{a}} L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right) \ , \quad \forall \boldsymbol{\eta} : ||\boldsymbol{\eta}||_\infty \leq \epsilon \, . \quad \text{[1st order Taylor expansion]}$$

Observation: This property is lost during AT with FGSM and CO appears [2].

Overcoming CO with local linearity regularization [2, 1]

Why?

The single step solution $\boldsymbol{\eta}_{\text{FGSM}}$ only makes sense if our loss is locally linear, i.e.:

$$L\left(h_{\mathbf{x}^k}(\mathbf{a}+\boldsymbol{\eta}), b\right) \approx L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right) + \boldsymbol{\eta}^\top \nabla_{\mathbf{a}} L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right) , \quad \forall \boldsymbol{\eta} : ||\boldsymbol{\eta}||_\infty \leq \epsilon . \quad \text{[1st order Taylor expansion]}$$

Observation: This property is lost during AT with FGSM and CO appears [2].

- We can measure the how locally linear a model is with the gradient missalignment.

Gradient Missalignment [2]

Let the point $\tilde{\mathbf{a}}$ be sampled uniformly such that $||\mathbf{a} - \tilde{\mathbf{a}}||_\infty \leq \epsilon$ and the gradients $\mathbf{g} = \nabla_{\mathbf{a}} L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right)$ and $\tilde{\mathbf{g}} = \nabla_{\tilde{\mathbf{a}}} L\left(h_{\mathbf{x}^k}(\tilde{\mathbf{a}}), b\right)$. The gradient missalignment is defined as:

$$\text{Grad.Miss.}(\mathbf{x}^k, \mathbf{a}) = 1 - \frac{\mathbf{g}^\top \tilde{\mathbf{g}}}{||\mathbf{g}||_2 ||\tilde{\mathbf{g}}||_2} , \tag{20}$$

with a locally linear model at $\mathbf{a}$ having $\text{Grad.Miss.}(\mathbf{x}^k, \mathbf{a}) = 0$.

Overcoming CO with local linearity regularization [2, 1]

- **Observation:** We can regularize the Gradient Missalignment during training to avoid CO, i.e., GradAlign [2]:

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}_{\mathsf{FGSM}}\right), b_i\right) + \lambda \cdot \mathsf{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i)\,.$$

- **Remark:** differentiating $\nabla_{\mathbf{x}}\mathsf{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i)$ is an expensive operation due to *Double Backpropagation* [8].
- **Question:** Can we do better?.

Overcoming CO with local linearity regularization [2, 1]

- **Observation:** We can regularize the Gradient Missalignment during training to avoid CO, i.e., GradAlign [2]:

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}_{\mathsf{FGSM}}\right), b_i\right) + \lambda \cdot \mathsf{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i)\,.$$

- **Remark:** differentiating $\nabla_{\mathbf{x}}\mathsf{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i)$ is an expensive operation due to *Double Backpropagation* [8].
- **Question:** Can we do better?. **Yes!**

ELLE [1]

Let the point $\tilde{\mathbf{a}}$ be sampled uniformly such that $||\mathbf{a} - \tilde{\mathbf{a}}||_\infty \leq \epsilon$ and $\hat{\mathbf{a}} = \alpha \cdot \mathbf{a} + (1-\alpha) \cdot \tilde{\mathbf{a}}$ with α sampled uniformly from $[0, 1]$. The ELLE regularization term is defined as:

$$\mathtt{ELLE}(\mathbf{x}^k, \mathbf{a}) = \left(L\left(h_{\mathbf{x}^k}(\hat{\mathbf{a}}), b\right) - \alpha \cdot L\left(h_{\mathbf{x}^k}(\mathbf{a}), b\right) - (1-\alpha) \cdot L\left(h_{\mathbf{x}^k}(\tilde{\mathbf{a}}), b\right)\right)^2\,, \tag{21}$$

with a locally linear model at $\mathbf{a}$ having $\mathtt{ELLE}(\mathbf{x}^k, \mathbf{a}) = 0$.

- **Advantage:** Regularizing ELLE does not involve *Double Backpropagation* and can as well overcome CO [1].

Is the training “fair”?

- Another grand challenge in ML: Fairness & bias
- A concrete example: Adversarial training may sacrifice subset of classes in favor of consensus
  - CIFAR10: 51% average robust accuracy while the worst class is 23.5%
  - CIFAR100: the worst class has *zero* accuracy while the best has 76%

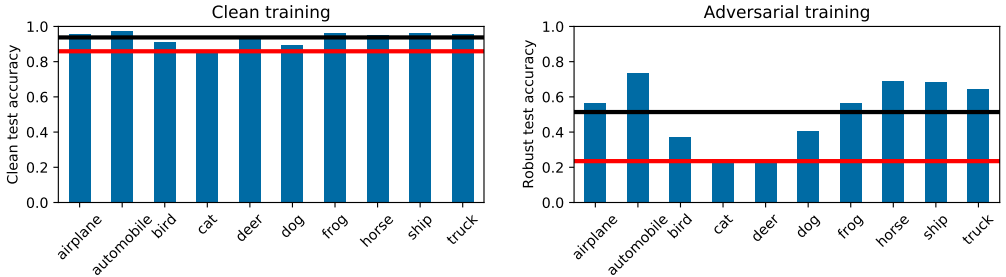


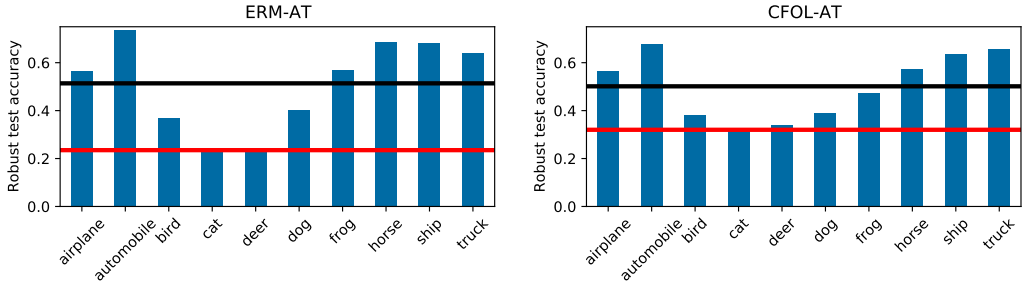
Figure: Clean accuracy and robust accuracy on CIFAR10 after clean training and adversarial training respectively.

Key challenges in ML demand much more than ERM

- Protect the weak: Class-focused online learning for adversarial training [25]

$$\min_{\mathbf{x}} \max_{b^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}), b^c) \right]$$

- Great potential via the minimax formulation: the average does not suffer much or can even improve!



References I

[1] Elias Abad Rocamora, Fanghui Liu, Grigorios G. Chrysos, Pablo M. Olmos, and Volkan Cevher.
Efficient local linearity regularization to overcome catastrophic overfitting.
In *Submitted to The Twelfth International Conference on Learning Representations*, 2024.
under review.
(Cited on pages 71, 72, 73, 74, and 75.)

[2] Maksym Andriushchenko and Nicolas Flammarion.
Understanding and improving fast adversarial training.
Advances in Neural Information Processing Systems, 2020.
(Cited on pages 70, 71, 72, 73, 74, and 75.)

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein generative adversarial networks.
In *International conference on machine learning*, pages 214–223. PMLR, 2017.
(Cited on pages 48 and 49.)

[4] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher.
Adversarially robust optimization with gaussian processes.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5765–5775, 2018.
(Cited on page 10.)

References II

[5] J. Danskin.
The theory of max-min, with applications.
SIAM Journal on Applied Mathematics, 14(4):641–664, 1966.
(Cited on page 15.)

[6] Richard Mansfield Dudley.
The speed of mean glivenko-cantelli convergence.
The Annals of Mathematical Statistics, 40(1):40–50, 1969.
(Cited on page 44.)

[7] Marwa EL HALABI.
Learning with Structured Sparsity: From Discrete to Convex and Back.
PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2018.
(Cited on page 59.)

[8] Christian Etmann.
A closer look at double backpropagation, 2019.
(Cited on pages 74 and 75.)

References III

[9] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola-Bibiane Schönlieb.
On the connection between adversarial robustness and saliency map interpretability.
In *International conference on machine learning*. PMLR, 2019.
(Cited on page 12.)

[10] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur.
Sharpness-aware minimization for efficiently improving generalization.
In *International Conference on Learning Representations*, 2021.
(Cited on pages 10, 64, and 65.)

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative Adversarial Networks.
ArXiv e-prints, June 2014.
(Cited on pages 43 and 51.)

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572, 2014.
(Cited on pages 6, 55, 68, and 69.)

References IV

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville.
Improved training of wasserstein gans.
In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.
(Cited on page 49.)

[14] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári.
Learning with a strong adversary.
arXiv preprint arXiv:1511.03034, 2015.
(Cited on pages 10, 11, 13, and 14.)

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of gans for improved quality, stability, and variation.
In *International Conference on Learning Representations*, 2018.
(Cited on page 43.)

[16] Ziko Kolter and Aleksander Madry.
Adversarial robustness - theory and practice.
NeurIPS 2018 tutorial: `https://adversarial-ml-tutorial.org/`.
(Cited on page 8.)

References V

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial examples in the physical world.
arXiv preprint arXiv:1607.02533, 2016.
(Cited on pages 8 and 59.)

[18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial machine learning at scale.
arXiv preprint arXiv:1611.01236, 2016.
(Cited on pages 8 and 59.)

[19] Fabian Latorre, Igor Krawczuk, Leello Tadesse Dadi, Thomas Pethick, and Volkan Cevher.
Finding actual descent directions for adversarial training.
In *International Conference on Learning Representations*, 2023.
(Cited on page 23.)

[20] Fabian Latorre, Paul Rolland, and Volkan Cevher.
Lipschitz constant estimation of neural networks via sparse polynomial optimization.
arXiv preprint arXiv:2004.08688, 2020.
(Cited on page 47.)

References VI

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu.
Towards deep learning models resistant to adversarial attacks.
In *ICLR '18: Proceedings of the 2018 International Conference on Learning Representations*, 2018.
(Cited on pages 6, 8, 16, 18, 19, 22, 55, and 59.)

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
arXiv preprint arXiv:1802.05957, 2018.
(Cited on page 49.)

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard.
Robustness via curvature regularization, and vice versa.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9078–9086, 2019.
(Cited on page 71.)

[24] Sebastian Neumayer, Alexis Goujon, Pakshal Bohra, and Michael Unser.
Approximation of Lipschitz Functions Using Deep Spline Neural Networks.
SIAM Journal on Mathematics of Data Science, 5(2):306–322, June 2023.
Publisher: Society for Industrial and Applied Mathematics.
(Cited on page 49.)

References VII

[25] Thomas Pethick, Grigorios G Chrysos, and Volkan Cevher.
Revisiting adversarial training for the worst-performing class.
Transactions on Machine Learning Research, 2023.
(Cited on pages 10 and 77.)

[26] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli.
Adversarial robustness through local linearization.
Advances in neural information processing systems, 32, 2019.
(Cited on page 71.)

[27] Alexander Robey, Fabian Latorre, George J Pappas, Hamed Hassani, and Volkan Cevher.
Adversarial training should be cast as a non-zero-sum game.
arXiv preprint arXiv:2306.11035, 2023.
(Cited on page 36.)

[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.
Intriguing properties of neural networks.
In *International Conference on Learning Representations*, 2014.
(Cited on page 6.)

References VIII

[29] Aladin Virmaux and Kevin Scaman.
Lipschitz regularity of deep neural networks: analysis and efficient estimation.
Advances in Neural Information Processing Systems, 31, 2018.
(Cited on page 47.)

[30] Jonathan Weed, Francis Bach, et al.
Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance.
Bernoulli, 25(4A):2620–2648, 2019.
(Cited on page 44.)

[31] Wanyun Xie, Thomas Pethick, and Volkan Cevher.
Sampa: Sharpness-aware minimization parallelized.
In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
(Cited on page 67.)