

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 2023)



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

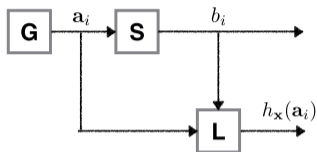
- ▶ This class
 - ▶ Adversarial Machine Learning (minmax)
 - ▶ Adversarial Training
 - ▶ Generative Adversarial Networks (GANs)
- ▶ Next class
 - ▶ Difficulty of minmax
 - ▶ Diffusion models

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 $h_{\mathbf{x}} : \mathbb{R}^p \rightarrow \mathbb{R}$ 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

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

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

- ▶ $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = \log(1 + \exp(-b_i h_{\mathbf{x}}(\mathbf{a}_i)))$
- ▶ $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = (b_i - h_{\mathbf{x}}(\mathbf{a}_i))^2$
- ▶ $L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = \max(0, 1 - b_i h_{\mathbf{x}}(\mathbf{a}_i))$

Logistic loss.

Squared error.

Hinge loss.

...Into adversarial examples

Definition (Adversarial examples [26])

Let $h_{\mathbf{x}^*} : \mathbb{R}^p \rightarrow \mathbb{R}$ be a model trained through empirical risk minimization, with optimal parameters \mathbf{x}^* . 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 $\delta \in \mathbb{R}^p$ designed to lead the trained model $h_{\mathbf{x}^*}$ to misclassify a given input \mathbf{a} . Given an $\epsilon > 0$, it is constructed by solving

$$\delta \in \arg \max_{\delta: \|\delta\| \leq \epsilon} L(h_{\mathbf{x}^*}(\mathbf{a} + \delta), b)$$

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:

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

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

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

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

Recall: Proximal operator of indicator functions

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

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

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

Adversarial examples and proximal gradient descent (cont'd)

- Target non-convex problem:

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

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

Proximal gradient ascent (PGA)

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

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

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]:

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

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

Challenge: Adversarial examples are inevitable

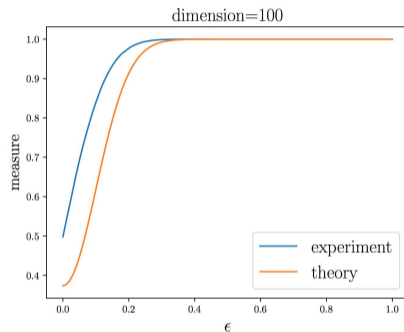
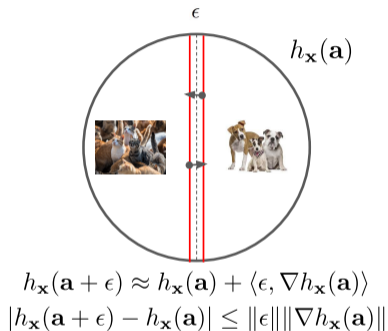
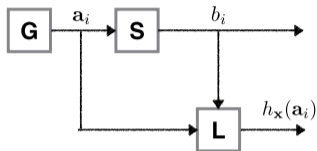


Figure: Understanding the robustness of a classifier in high-dimensional spaces. Shafahi et al. 2019.

Hardness results have never been a barrier for ML researchers



Definition (Empirical Risk Minimization (ERM))

Let $h_{\mathbf{x}} : \mathbb{R}^p \rightarrow \mathbb{R}$ 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

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

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

Objectives

- ▶ $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\delta: \|\delta\|_{\infty} \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \delta), b_i) \right] \right\}$ Adversarial training [14].
- ▶ $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\delta: \|\delta\|_2 \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \delta), b_i) \right] \right\}$ ϵ -stability training [4],
Sharpness-aware minimization [10].
- ▶ $\min_{\mathbf{x}} \max_{b^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\delta: \|\delta\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \delta), b^c) \right]$ Class fairness [24].

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

Towards adversarial training

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^n \rightarrow \mathbb{R}$ 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

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

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 $h_x(\mathbf{a})$?
- **Solution:** Look at $\nabla_x h_x(\mathbf{a})$, called the saliency map [9]. Minimax adversarial training seems to help!

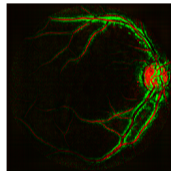
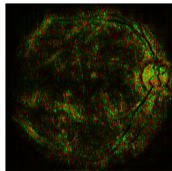


Table: **Left:** Ground truth image, **Middle:** Saliency map, **Right:** Saliency map with adversarial training.

Solving the outer problem

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^p \rightarrow \mathbb{R}$ 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

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

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

Solving the outer problem

Adversarial Training [14]

Let $h_{\mathbf{x}} : \mathbb{R}^p \rightarrow \mathbb{R}$ 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

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

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

Question

How can we compute the gradient

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

- **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

$$f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}), \quad \mathcal{S}^*(\mathbf{x}) := \arg \max_{\mathbf{y} \in \mathcal{S}} \Phi(\mathbf{x}, \mathbf{y}).$$

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

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

An immediate consequence

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

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

A practical implementation of adversarial training: Stochastic subgradient descent

Stochastic Adversarial Training [21]
Input: learning rate α_k , iterations T , batch size K .
<ol style="list-style-type: none">1. initialize neural network parameters \mathbf{x}^02. For $k = 0, 1, \dots, T$:<ol style="list-style-type: none">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 $i \in B$:<ol style="list-style-type: none">a. Find an attack δ^* by (approximately) solving$\delta^* \in \arg \max_{\delta: \ \delta\ _\infty \leq \epsilon} L(h_{\mathbf{x}^k}(\mathbf{a}_i + \delta), b_i)$b. Store update$\mathbf{g}^k := \mathbf{g}^k + \nabla_{\mathbf{x}} L(h_{\mathbf{x}^k}(\mathbf{a}_i + \delta^*), b_i)$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 δ^* .
- Update in step **iii.b** is motivated by Corollary A.2 in [21]

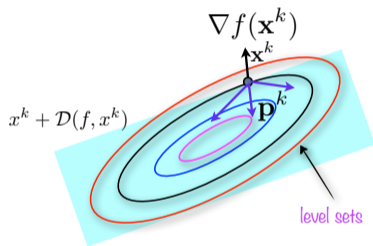
Theoretical foundations

$$\frac{\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \delta^*)}{\nabla_{\mathbf{x}} f(\mathbf{x})} \quad \begin{array}{l} \text{unique } \delta^* \\ \text{non-unique } \delta^* \end{array} \quad \text{descent direction [21]}$$

Published as a conference paper at ICLR 2018

TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

Aleksander Madry, 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



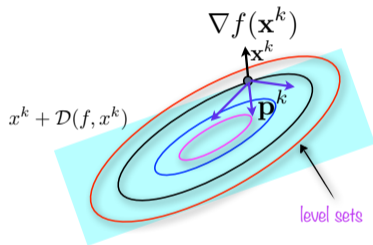
Theoretical foundations ?

$$\frac{\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \delta^*)}{\nabla_{\mathbf{x}} f(\mathbf{x})} \quad \begin{array}{l} \text{unique } \delta^* \\ \text{non-unique } \delta^* \end{array} \quad \text{descent direction [21]}$$

Published as a conference paper at ICLR 2018

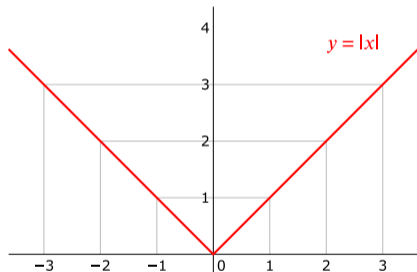
TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

Aleksander Madry, 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

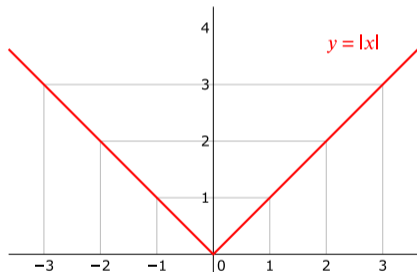
$$f(\mathbf{x}) := \max_{\delta \in [-1, 1]} \mathbf{x}\delta = |\mathbf{x}|.$$



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

A counterexample

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



- We have $\mathcal{S} := [-1, 1]$ and $\Phi(\mathbf{x}, \delta) = \mathbf{x}\delta$.
- At $\mathbf{x} = 0$, we have $\mathcal{S}^*(0) = [-1, 1]$.
- We can choose $\delta = 1 \in \mathcal{S}^*(0)$: $\Phi(\mathbf{x}, 1) = \mathbf{x}$.
 - ▶ $-\nabla_{\mathbf{x}}\Phi(0, 1) = -1 \neq 0$.
 - ▶ 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 $\mathcal{Y}^*(\mathbf{x}) := \arg \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ 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

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

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

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

Remarks: ◦ The notion of directional derivative is one-sided:

$$Df(\mathbf{x}, \mathbf{d}) := \lim_{t \rightarrow 0^+} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t} \quad (2)$$

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

Directional derivatives, not descent directions

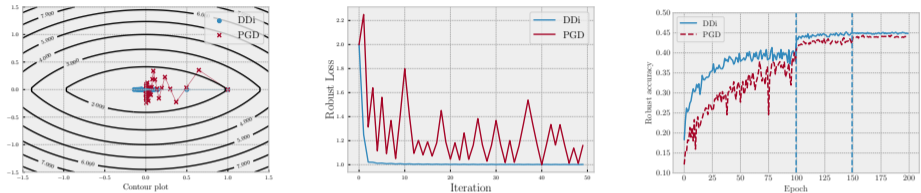


Figure: (Left and Middle) Synthetic adversarial training example. (Right) Resnet18 on CIFAR10 - Robust accuracy comparison between PGD and DDi.

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 $\Phi(\mathbf{x}, \mathbf{y}) : \mathbb{R}^p \times \mathcal{Y} \rightarrow \mathbb{R}$, 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 $\mathcal{Y}^*(\mathbf{x}) := \arg \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ as the set of maximizers and $\mathbf{y}^* \in \mathcal{Y}^*$ as an element of this set. We have

1. $f(\mathbf{x})$ is a convex function.
2. If $\mathcal{Y}^*(\mathbf{x})$ 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} :

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

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

$$\partial_{\mathbf{x}} f(\mathbf{x}) = \text{conv} \{ \partial_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^*) : \mathbf{y}^* \in \mathcal{Y}^*(\mathbf{x}) \}.$$

Remarks:

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

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)

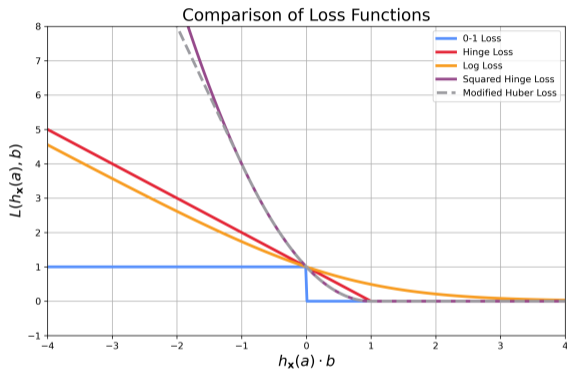
$$\min_{\mathbf{x}} \mathbb{E} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \leq \epsilon} L_{01}(h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}), b) \right]$$

Original Formulation of Adversarial Training (II)

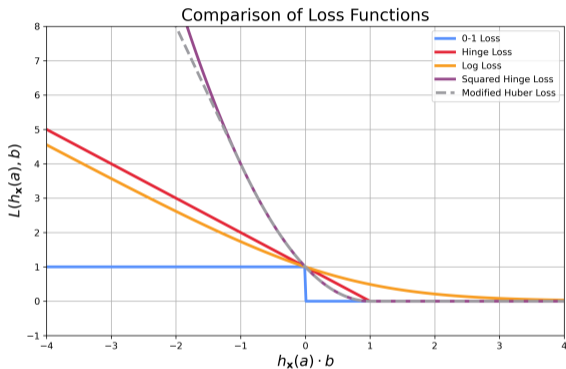
$$\min_{\mathbf{x}} \mathbb{E} \left[\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\| \leq \epsilon} L_{01}(h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}), b) \right]$$

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

Surrogate-based optimization for Risk Minimization



Surrogate-based optimization for Risk Minimization

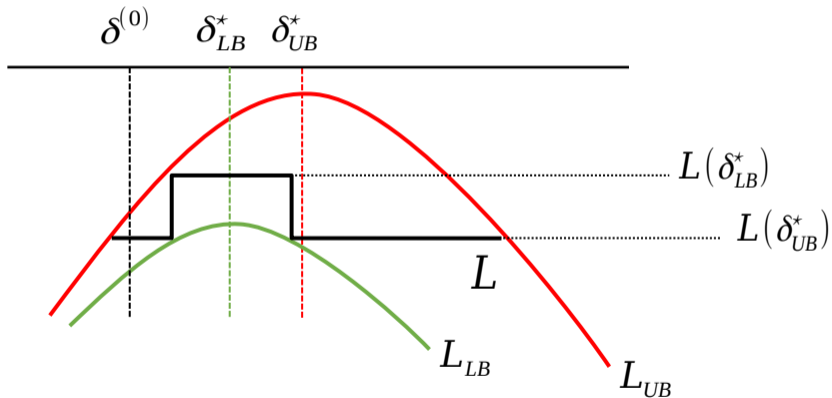


$$\mathbb{E} [L_{01}(h_{\mathbf{x}^*}(\mathbf{a} + \boldsymbol{\delta}), b)] \leq \min_{\mathbf{x}} \mathbb{E} [L_{\text{CE}}(h_{\mathbf{x}}(\mathbf{a} + \boldsymbol{\delta}), b)]$$

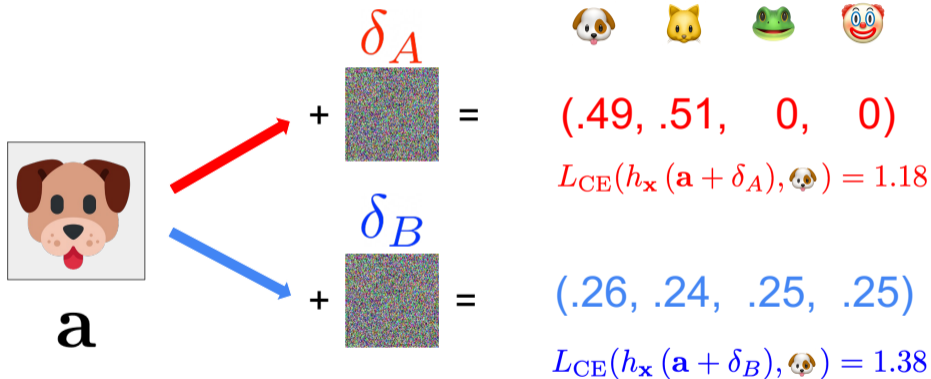
Adversary maximizes an upper bound (I)

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

Adversary maximizes an upper bound (II)



Why maximizing cross-entropy leads to weak adversaries



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

Theorem (Reformulation of the Adversary's problem)

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

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

Bilevel Optimization (BETA)

- Best targeted attack (BETA) optimization formulation [25]:

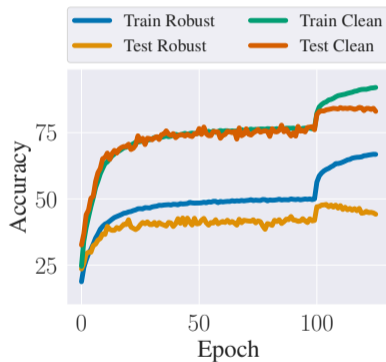
$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^n L_{\text{CE}} \left(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^*}^*), b_i \right)$$

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

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

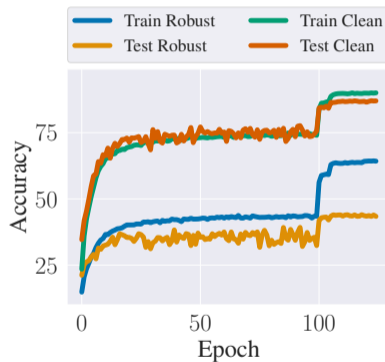
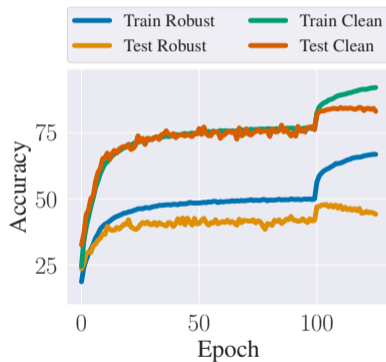
Practical Consequences of the Bilevel Formulation

Figure: Learning curves of PGD¹⁰-AT (Left) and BETA¹⁰-AT



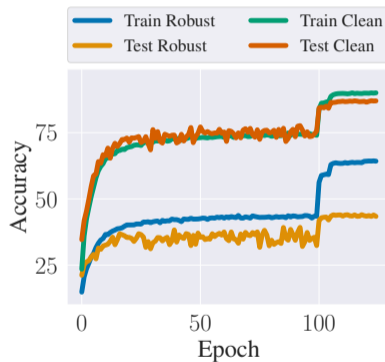
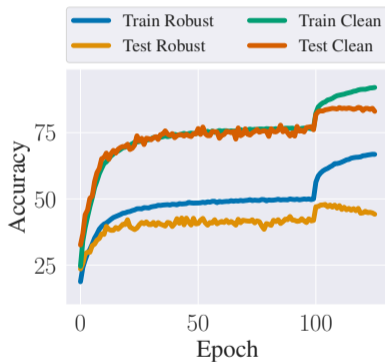
Practical Consequences of the Bilevel Formulation

Figure: Learning curves of PGD¹⁰-AT (Left) and BETA¹⁰-AT (Right). Robust accuracy estimated with PGD²⁰



Practical Consequences of the Bilevel Formulation

Figure: Learning curves of PGD¹⁰-AT (Left) and BETA¹⁰-AT (Right). Robust accuracy estimated with PGD²⁰



No Robust Overfitting occurs!

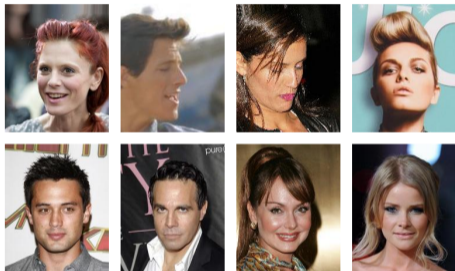
Practical Consequences of the Bilevel Formulation

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-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-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-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)

- o Recall the parametric density estimation setting

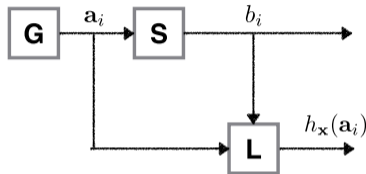


(source: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>)

$\mathbf{a}_i = [\dots \text{images} \dots]$

$\mathbf{b}_i = [\dots \text{probability} \dots]$

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



- o Generator $\mathbb{P}_{\mathbf{a}}$
 - ▶ Nature
- o Supervisor $\mathbb{P}_{B|\mathbf{a}}$
 - ▶ Frequency data
- o 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 $\omega \sim p_\Omega$ be a random variable.
- $h_{\mathbf{x}}(\cdot) : \mathbb{R}^p \rightarrow \mathbb{R}^m$ a function parameterized by parameters \mathbf{x} .

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

Example: Chi-square distribution

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

Explanation: Change of variables.

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

$$p_Y(\mathbf{y}) = p_\Omega(h_{\mathbf{x}}^{-1}(\mathbf{y})) \det(\mathbf{J}_{\mathbf{y}} h_{\mathbf{x}}^{-1}(\mathbf{y}))$$

where \det denotes the determinant operation.

Towards an optimization problem

Problem (Ideal parametric density estimator)

Given a true distribution μ^{\natural} , we can solve the following optimization problem,

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

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 μ^{\natural} .
 - ▶ W_1 is non-smooth, it cannot be computed exactly.

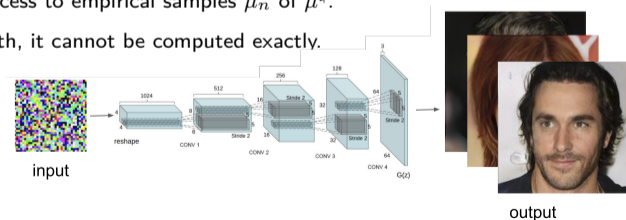
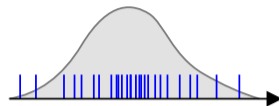


Figure: Schematic of a generative model, $h_{\mathbf{x}} \# \omega$ [11, 15].

Learning without concentration

- We can minimize $W_1(\hat{\mu}_n, h_{\mathbf{x}}\#\mathbf{p}_\Omega)$ 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,

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

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

Theorem (Slow convergence of empirical measures in 1-Wasserstein [28, 6])

Let μ^\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,

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

Remarks:

- Using an empirical estimator in high-dimensions is terrible in the worst case.
- However, it does not directly say that $W_1(\mu^\natural, h_{\mathbf{x}}\#\mathbf{p}_\Omega)$ 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)

$$W_1(\mu, \nu) = \sup_{\mathbf{d}} \{ \langle \mathbf{d}, \mu \rangle - \langle \mathbf{d}, \nu \rangle : \mathbf{d} \text{ is 1-Lipschitz} \} \quad (6)$$

Remark: ◦ \mathbf{d} is the “dual” variable. In the literature, it is commonly referred to as the “discriminator.”

Inner product is an expectation

$$\langle \mathbf{d}, \mu \rangle = \int \mathbf{d} d\mu = \int \mathbf{d}(\mathbf{a}) d\mu(\mathbf{a}) = \mathbf{E}_{\mathbf{a} \sim \mu} [\mathbf{d}(\mathbf{a})]. \quad (7)$$

Kantorovich-Rubinstein duality applied to our objective

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

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:

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

- Remarks:**
- The 1-Wasserstein distance corresponds to $\mathcal{F} := \{f : \mathbb{R}^p \rightarrow \mathbb{R}, f \text{ is } 1\text{-Lipschitz}\}$
 - 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 [27].
- We can constrain upper bounds on the Lipschitz constant [20].

Lemma

Let $h_{\mathbf{X}_1, \mathbf{X}_2}(\mathbf{a}) := \mathbf{X}_2^T \sigma(\mathbf{X}_1 \mathbf{a})$ 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:

$$L_{\mathbf{X}_1, \mathbf{X}_2} \leq \|\mathbf{X}_1\|_2 \|\mathbf{X}_2\|_2 \quad (10)$$

Neural Network Distance

Let

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

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

o 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 $d_{\mathbf{y}}(\mathbf{a})$, where $\mathbf{y} \in \mathcal{Y}$ such that $d_{\mathbf{y}}(\mathbf{a})$ is 1-Lipschitz. In this case, the Wasserstein GAN optimization problem is given by

$$\min_{\mathbf{x} \in \mathcal{X}} \left(\max_{\mathbf{y} \in \mathcal{Y}} E_{\mathbf{a} \sim \hat{\mu}_n} [d_{\mathbf{y}}(\mathbf{a})] - E_{\omega \sim p_{\Omega}} [d_{\mathbf{y}}(h_{\mathbf{x}}(\omega))] \right). \quad (12)$$

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

Weight clipping [3]

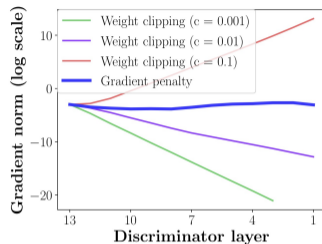
The “dual” or the “discriminator” $d_{\mathbf{y}}$ weights \mathbf{y} are constrained by an ℓ_{∞} -ball with radius $c > 0$, denoted as \mathcal{B} , at every iteration with

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

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 $\|\nabla_{\mathbf{a}} d_{\mathbf{y}}(\mathbf{a})\|_* \leq 1$. This can be enforced directly through

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

Remarks:

- In practice the distribution ν mimicks uniform (linearly interpolated) sampling as follows: $\mathbf{a} \sim \text{Uniform}(\mathbf{a}_i, h_{\mathbf{x}}(\omega_i))$.
- Spectral normalization: Divide each weight matrix by their spectral norm [22].
- Learnable spline activations: both a 1-Lipschitz and more expressive architecture [23].

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 d_y , generator network h_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 $t = 0, 1, \dots, T - 1$:
 - For $m = 0, 1, \dots, M - 1$:
 - initialize \mathbf{y}^0 ,
 - draw noise sample $\omega_1, \dots, \omega_K \sim p_\Omega$
 - draw real samples $\mathbf{r}_1, \dots, \mathbf{r}_K \sim \hat{\mu}_n$
 - “dual” pseudo-loss $L(\mathbf{y}) := K^{-1} \sum_{i=1}^K d_y(\mathbf{r}_i) - d_y(h_{\mathbf{x}^t}(\omega_i))$
 - #update “dual” parameters $\mathbf{y}^{m+1} = \mathbf{y}^m + \gamma_m \nabla_{\mathbf{y}} L(\mathbf{y}^m)$
 - #enforce 1-Lipschitz constraint on $d_{\mathbf{y}^{m+1}}$
 - end-For
 - draw noise sample $\omega_1, \dots, \omega_B \sim p_\Omega$
 - generator pseudo-loss $L(\mathbf{x}) := -B^{-1} \sum_{i=1}^B d_{\mathbf{y}^M}(h_{\mathbf{x}}(\omega_i))$
 - update generator parameters $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \nabla_{\mathbf{x}} L(\mathbf{x}^t)$
- end-For

#: Ideally, should be performed jointly.

Some historical background for a Turing award

Vanilla GAN [11]

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

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

Observation: ◦ Minimizes Jensen-Shannon divergence:

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

Wrap up!

- Continuing on Homework 2!

A robustness example: Linear prediction

Linear model

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

An adversarial perturbation

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

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

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

- Remarks:**
- For the linear model, we have $\nabla_{\mathbf{a}} h_{\mathbf{x}^*}(\mathbf{a}) = \mathbf{x}^*$.
 - *The gradient sign* of $h_{\mathbf{x}^*}$ 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:

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

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

Fast Gradient Sign Method (FGSM) [12]

Let $h_{\mathbf{x}^*} : \mathbb{R}^p \rightarrow \mathbb{R}$ be a model trained through empirical risk minimization on the loss L , with optimal parameters \mathbf{x}^* . 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

$$\delta = \epsilon \operatorname{sign}(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^*}(\mathbf{a}), b)) = \epsilon \operatorname{sign}(\nabla_{\mathbf{a}} h_{\mathbf{x}^*}(\mathbf{a}) \nabla_h L(h_{\mathbf{x}^*}(\mathbf{a}), b))$$

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 $h_{\mathbf{x}}(\mathbf{a})$, $\nabla_h L(h_{\mathbf{x}^*}(\mathbf{a}), b)$ is a scalar,
 - ▶ $\operatorname{sign}(\nabla_{\mathbf{a}} h_{\mathbf{x}^*}(\mathbf{a}))$ pattern is important

Results of FGSM on MNIST

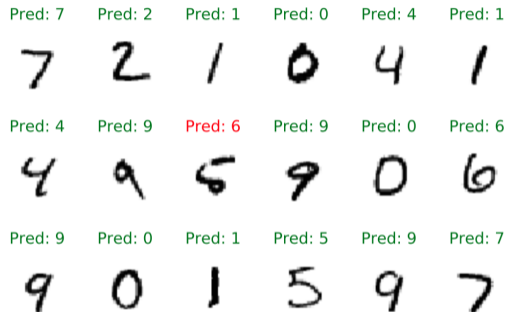


Figure: MNIST images with the predicted digit.



Figure: MNIST images perturbed by a FGSM attack.

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

A proposed link between FGSM and PGA

o Recall

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

FGSM as one step of PGA

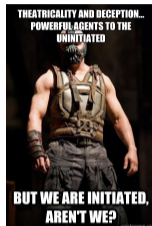
Let $\delta^0 = \mathbf{0}$ and $\alpha > 0$ such that $(\alpha |\nabla f(\mathbf{0})|)_i > \epsilon$ for $i = 1, \dots, n$. Then, one step of PGA yields

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

A proposed link between FGSM and PGA

o Recall

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



FGSM as one step of PGA

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

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

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

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

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

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

Maximizing the RHS with respect to δ leads to the following (non trivial) solution [7]:

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

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

Remarks: \circ Setting $\zeta = \delta^k$ and $\delta^* = \delta^{k+1}$ with a fixed step size $\alpha = t^*$, we obtain the update in [17, 18, 21]

$$\delta^{k+1} = \text{clip}(\delta^k - t^* \text{sign}(\nabla f(\delta^k)), [-\epsilon, \epsilon]).$$

\circ 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* $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that $T(X) \sim Y$, minimizing the cost

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

The Wasserstein distance

Definition

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

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

Definition (q -Wasserstein distance (Primal))

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

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

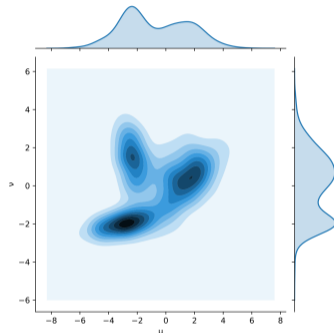


Figure: Two one-dimensional distributions plotted on the x and y axes, and one possible joint distribution that defines a transport plan between them (https://en.wikipedia.org/wiki/Wasserstein_metric).

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:

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

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

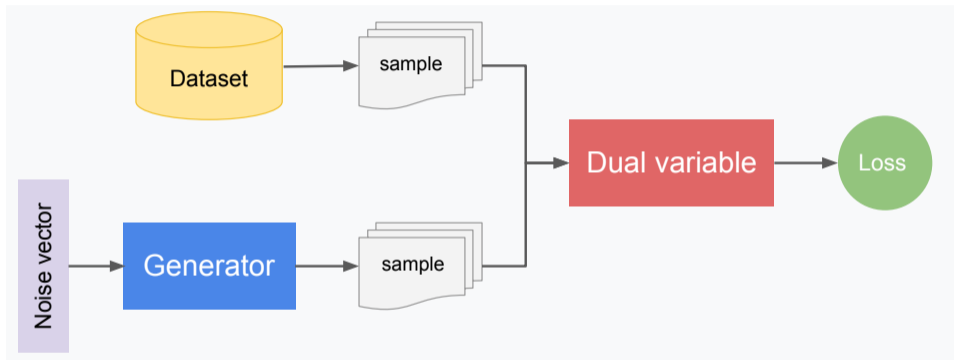


Figure: Generator/dual variable/dataset relation in GANs

*Sharpness-aware minimization (SAM) [10]

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

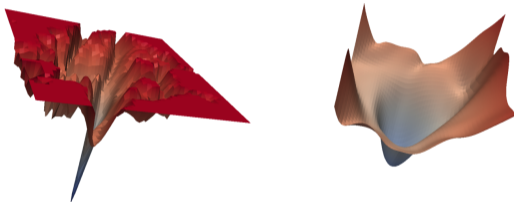
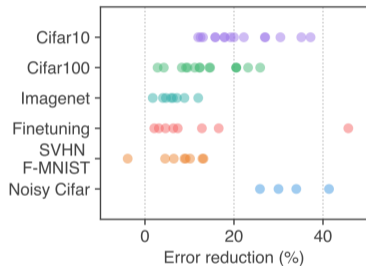


Figure: ResNet trained via SAM converges to a flatter minima (Right) compared with the one trained via SGD (Middle), and thus leads to considerable error rate reduction (Left) [10].

*Sharpness-aware minimization (SAM) [10]

o Efficient approximation to the objective $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\delta: \|\delta\|_2 \leq \epsilon} L(h_{\mathbf{x}+\delta}(\mathbf{a}_i), b_i) \right] \right\}$:

► Let's first consider the the inner maximization problem. By first-order Taylor expansion, we have:

$$\begin{aligned} \delta^* &= \arg \max_{\delta: \|\delta\|_2 \leq \epsilon} L(h_{\mathbf{x}+\delta}(\mathbf{a}_i), b_i) \approx \arg \max_{\delta: \|\delta\|_2 \leq \epsilon} \left[L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) + \delta^\top \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right] \\ &= \arg \max_{\delta: \|\delta\|_2 \leq \epsilon} \delta^\top \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) = \epsilon \frac{\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i)}{\|\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i)\|_2}. \end{aligned}$$

► Plugging δ^* back the original objective and take the derivative:

$$\begin{aligned} \nabla_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\delta: \|\delta\|_2 \leq \epsilon} L(h_{\mathbf{x}+\delta}(\mathbf{a}_i), b_i) \right] \right\} &= \frac{1}{n} \sum_{i=1}^n [\nabla_{\mathbf{x}} L(h_{\mathbf{x}+\delta^*}(\mathbf{a}_i), b_i)] \\ &= \frac{1}{n} \sum_{i=1}^n \left[\left(1 + \frac{d\delta^*}{dw} \right) \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \Big|_{\mathbf{x}+\delta^*} \right] \approx \frac{1}{n} \sum_{i=1}^n \left[\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \Big|_{\mathbf{x}+\delta^*} \right], \end{aligned}$$

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

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

Fast Adversarial Training [29] and Catastrophic Overfitting (CO) [1]

- Obtaining δ^* with 10 iterations of PGA is an expensive procedure.
- **Questions:**
 - ▶ How can we make AT faster?
 - ▶ Can we use $\delta_{\text{FGSM}} = \epsilon \text{sign}(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^k}(\mathbf{a}), b))$ for solving the inner max in AT?

Fast Adversarial Training [29] and Catastrophic Overfitting (CO) [1]

○ Obtaining δ^* with 10 iterations of PGA is an expensive procedure.

○ **Questions:**

▶ How can we make AT faster?

▶ Can we use $\delta_{\text{FGSM}} = \epsilon \text{ sign}(\nabla_{\mathbf{a}} L(h_{\mathbf{x}^k}(\mathbf{a}), b))$ for solving the inner max in AT? **Not Naively!**

○ **Observations:**

▶ Catastrophic Overfitting (CO) appears, i.e., multi-step PGA accuracy drops to zero and FGSM accuracy spikes to 1.

▶ CO occurs when the network becomes highly non-linear around \mathbf{a} [1].

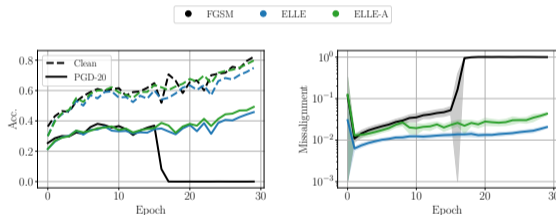


Figure: (Left) PGA-20 adversarial accuracy. (Right) Gradient missalignment.

Overcoming CO with local linearity regularization [1, 2]

Why?

The single step solution δ_{FGSM} only makes sense if our loss is locally linear, i.e.:

$$L(h_{\mathbf{x}^k}(\mathbf{a} + \boldsymbol{\delta}), b) \approx L(h_{\mathbf{x}^k}(\mathbf{a}), b) + \boldsymbol{\delta}^\top \nabla_{\mathbf{a}} L(h_{\mathbf{x}^k}(\mathbf{a}), b), \forall \boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon.$$

Observation: This property is lost during training and CO appears.

Overcoming CO with local linearity regularization [1, 2]

Why?

The single step solution δ_{FGSM} only makes sense if our loss is locally linear, i.e.:

$$L(h_{\mathbf{x}^k}(\mathbf{a} + \delta), b) \approx L(h_{\mathbf{x}^k}(\mathbf{a}), b) + \delta^\top \nabla_{\mathbf{a}} L(h_{\mathbf{x}^k}(\mathbf{a}), b), \forall \delta : \|\delta\|_\infty \leq \epsilon.$$

Observation: This property is lost during training and CO appears.

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

Gradient Missalignment [1]

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

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

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

Overcoming CO with local linearity regularization [1, 2]

- **Observation:** We can regularize the Gradient Missalignment during training to avoid CO, i.e., GradAlign [1]:

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^n L(h_{\mathbf{x}}(\mathbf{a}_i + \delta_{\text{FGSM}}), b_i) + \lambda \cdot \text{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i).$$

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

Overcoming CO with local linearity regularization [1, 2]

- **Observation:** We can regularize the Gradient Missalignment during training to avoid CO, i.e., GradAlign [1]:

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^n L(h_{\mathbf{x}}(\mathbf{a}_i + \delta_{\text{FGSM}}), b_i) + \lambda \cdot \text{Grad.Miss.}(\mathbf{x}, \mathbf{a}_i).$$

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

ELLE [2]

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:

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

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

- **Advantage:** Regularizing ELLE does not involve *Double Backpropagation* and can as well overcome CO [2].

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%

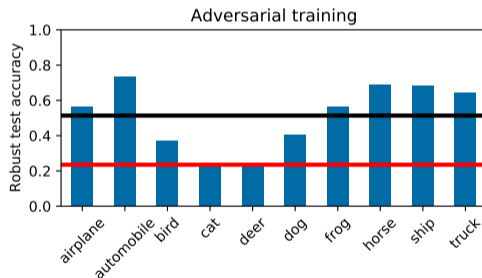
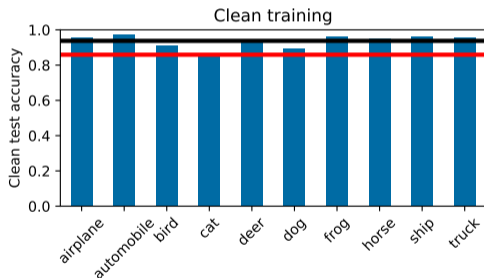


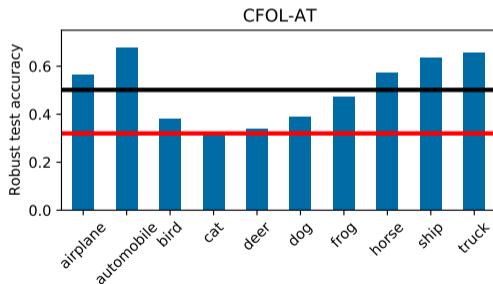
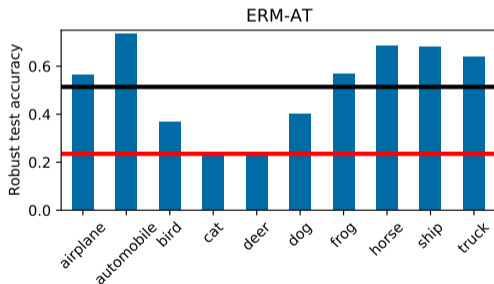
Figure: Clean accuracy and robust accuracy on CIFAR10 after clean training and adversarial training respectively.

Key challenges in ML demand much more than ERM

- Protect the weak: Class-focused online learning for adversarial training [24]

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

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



References I

- [1] Maksym Andriushchenko and Nicolas Flammarion.
Understanding and improving fast adversarial training.
Advances in Neural Information Processing Systems, 2020.
(Cited on pages 64, 65, 66, 67, 68, and 69.)
- [2] Anonymous.
Efficient local linearity regularization to overcome catastrophic overfitting.
In *Submitted to The Twelfth International Conference on Learning Representations*, 2024.
under review.
(Cited on pages 66, 67, 68, and 69.)
- [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 47 and 48.)
- [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 43.)
- [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 57.)
- [8] Christian Etmann.
A closer look at double backpropagation, 2019.
(Cited on pages 68 and 69.)

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, 62, and 63.)
- [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 42 and 50.)
- [12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572, 2014.
(Cited on pages 6 and 53.)

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 48.)
- [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 42.)
- [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 57.)
- [18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial machine learning at scale.
arXiv preprint arXiv:1611.01236, 2016.
(Cited on pages 8 and 57.)
- [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 22.)
- [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 46.)

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, 17, 18, 21, 53, and 57.)
- [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 48.)
- [23] 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 48.)
- [24] 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 71.)

References VII

- [25] 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 35.)
- [26] 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.)
- [27] 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 46.)
- [28] 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 43.)

References VIII

- [29] Eric Wong, Leslie Rice, and J Zico Kolter.
Fast is better than free: Revisiting adversarial training.
International Conference on Learning Representations, 2020.
(Cited on pages 64 and 65.)