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 2022)

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

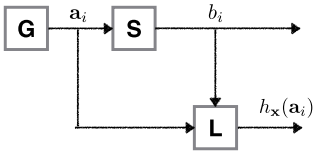
- This class
  - Adversarial Machine Learning (minmax)
    - Adversarial training
    - Generative adversarial networks
    - Difficulty of minmax
- Next class
  - Primal-dual optimization (Part 1)

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

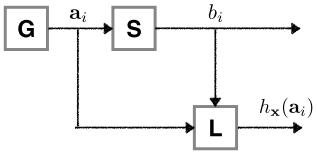
$$\min_{\mathbf{x}} \left\{ R_n(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

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

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

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 $(\mathbf{a}_i, b_i)$.

Objectives in other tasks

- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\eta}:\|\boldsymbol{\eta}\|_\infty \leq \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}\right), \mathbf{b}_i\right) \right] \right\}$ — Adversarial training [12].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{\eta}:\|\boldsymbol{\eta}\|_2 \leq \epsilon} L(h_{\mathbf{x}+\boldsymbol{\eta}}\left(\mathbf{a}_i\right), \mathbf{b}_i) \right] \right\}$ — ϵ-stability training [4], Sharpness-aware minimization [8].
- $\min_{\mathbf{x}} \max_{\mathbf{b}^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\boldsymbol{\eta}:\|\boldsymbol{\eta}\| \leq \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}\right), \mathbf{b}_i^c\right) \right]$ — Class fairness [1].

...Into adversarial examples

Definition (Adversarial examples [20])

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 $\boldsymbol{\eta} \in \mathbb{R}^p$ 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

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

Example norms frequently used in adversarial attacks

- The most commonly used norm is the ℓ_∞-norm [10, 18].
- 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.

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

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

Taking $\boldsymbol{\eta}^\star = \operatorname{sign}(\mathbf{x}^\star)$ 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:

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

- Historically, researchers first tried to find approximate solutions that empirically perform well [10, 18].

Fast Gradient Sign Method (FGSM) [10]

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

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

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

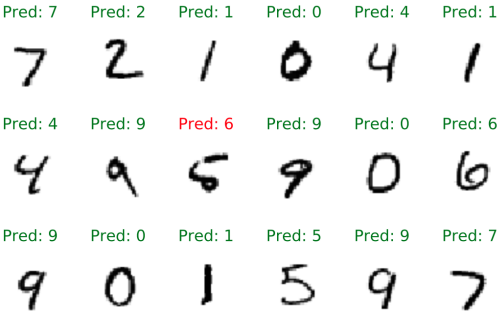


Figure: MNIST images with the predicted digit.

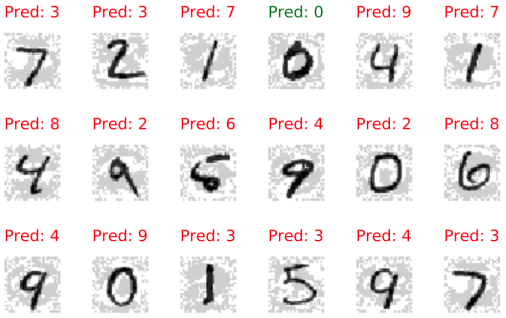


Figure: MNIST images perturbed by a FGSM attack.

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

Adversarial examples and proximal gradient descent

- Target problem:

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

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

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

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

Recall: Proximal operator of indicator functions

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

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

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

Adversarial examples and proximal gradient descent (cont'd)

- Target non-convex problem:

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

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

Proximal gradient ascent (PGA)

1. Choose $\boldsymbol{\eta}^0 \in \operatorname{dom} f(\boldsymbol{\eta}) + g(\boldsymbol{\eta})$ as initialization.

2. For $k = 0, 1, \cdots$, generate a sequence $\{\boldsymbol{\eta}^k\}_{k\geq 0}$ as:

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

Remarks:

- PGA results in more powerful adversarial "attacks" than FGSM [14].
- The PGA is incorrectly referred to as projected gradient descent in this literature.
- Practitioners prefer to use several steps of FGSM instead of PGA [15, 16, 18]:

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

A proposed link between FGSM and PGA

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

FGSM as one step of PGA

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

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

A proposed link between FGSM and PGA

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

FGSM as one step of PGA

Let $\boldsymbol{\eta}^0 = \mathbf{0}$ 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

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

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

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

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

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

Maximizing the RHS with respect to $\boldsymbol{\eta}$ leads to the following (non trivial) solution [6]:

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

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

Remarks:

- Setting $\boldsymbol{\zeta} = \boldsymbol{\eta}^k$ and $\boldsymbol{\eta}^\star = \boldsymbol{\eta}^{k+1}$ with a fixed step size $\alpha = t^\star$, we obtain the update in [15, 16, 18]
$$\boldsymbol{\eta}^{k+1} = \text{clip}\left(\boldsymbol{\eta}^k - t^\star \text{sign}(\nabla f(\boldsymbol{\eta}^k)), [-\epsilon, \epsilon]\right).$$
- This proof holds for **concave** and smooth functions, and need further quantification for our setting.

Towards adversarial training

Adversarial Training [12]

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

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

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

Solving the outer problem

Adversarial Training [12]

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

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

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

Solving the outer problem

Adversarial Training [12]

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

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

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

Question

How can we compute the gradient

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

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

Danskin's theorem

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 $\nabla_{\mathbf{x}} f(\mathbf{x}) = \nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star)$.

The adversarial training formulation

Adversarial Training

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

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

L is not differentiable due to non-smooth activation functions (ReLU), nor convex in $\mathbf{x}$ because of the neural network structure.

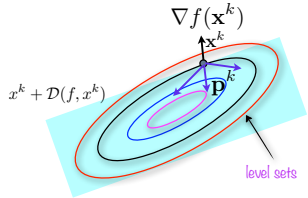


Figure: Descent directions in 2D should be an element of the cone of descent directions $\mathcal{D}(f, \cdot)$.

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 [18] (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

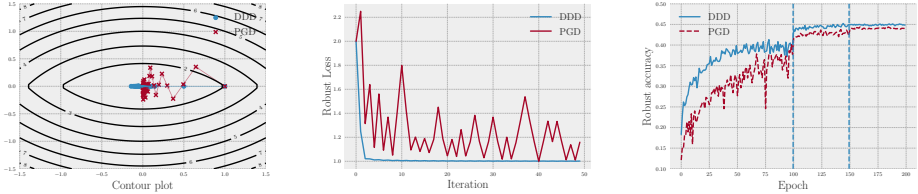


Figure: **(Left and Middle)** Synthetic adversarial training example. **(Right)** Resnet18 on CIFAR10 - Robust accuracy comparison between PGD and DDD.

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 (DDD) computes many maximizers to find a descent direction [2].
- In practice however, the lack of descent does not seem to matter.

A practical implementation of adversarial training: Stochastic subgradient descent

Stochastic Adversarial Training [18]

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 $\boldsymbol{\eta}^\star$ by (approximately) solving

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

b. Store update

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

iv. Update parameters

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

Remarks:

- Expensive but worth it!
- Inner problem **iii.a** cannot be solved to optimality (non-convex).
- Practitioners use FGSM or PGA or PGA-ℓ_∞ to approximate the true $\boldsymbol{\eta}^\star$.
- Update in step **iii.b** is motivated by Corollary A.2 in [18]

Application: 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 [7]. Adversarial training helps!

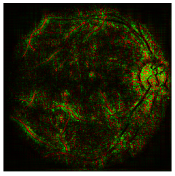
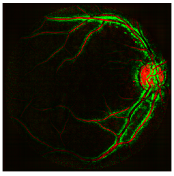




Table: **Left:** Ground truth image, **Middle:** Saliency map, **Right:** Saliency map with adversarial training.

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%

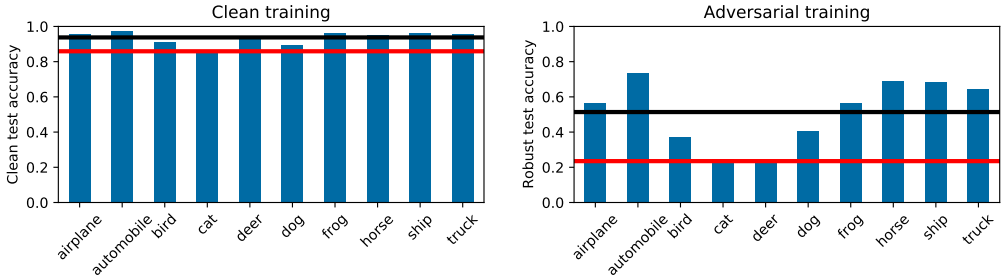


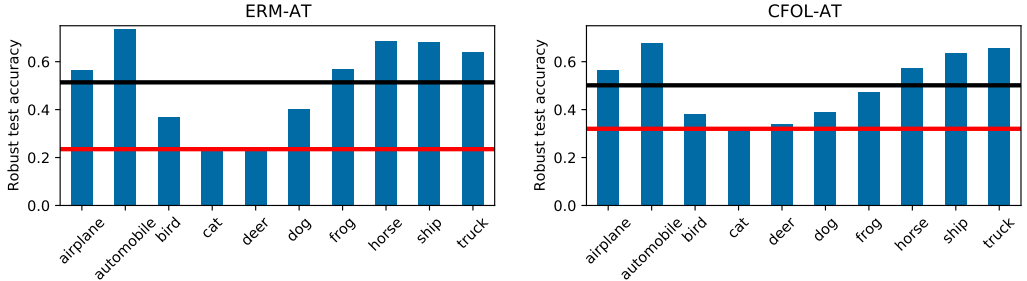
Figure: Clean accuracy and robust accuracy on CIFAR10 after clean training and adversarial training respectively.

Key challenges in ML demand much more than ERM

- Protect the weak: Class-focused online learning for adversarial training [1]

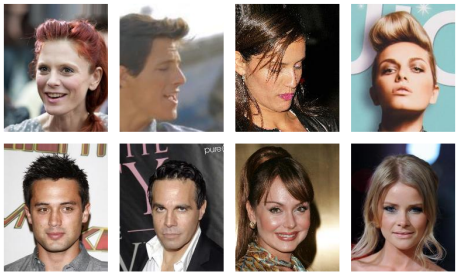
$$\min_{\mathbf{x}} \max_{\mathbf{b}^c \in [C]} \frac{1}{n_c} \sum_{i=1}^{n_c} \left[\max_{\boldsymbol{\eta}: \|\boldsymbol{\eta}\| \leq \epsilon} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\eta}), \mathbf{b}_i^c) \right]$$

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



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

- Recall the parametric density estimation setting

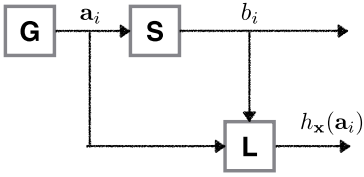


(source: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

$\mathbf{a}_i = [$ ...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 notion of distance between distributions

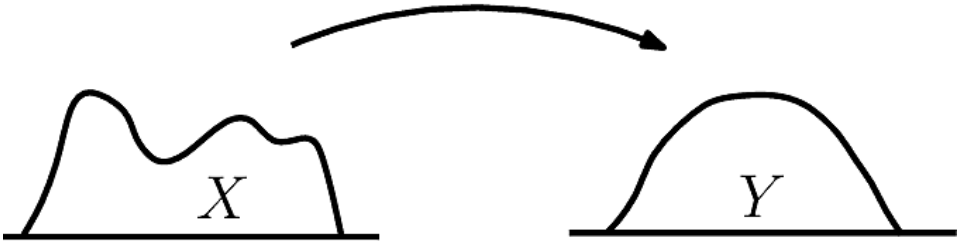


Figure: The Earth Mover's distance

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

$$\text{cost}(T) := \boldsymbol{E}_X \| Y - T(X) \|. \tag{3}$$

The Wasserstein distance

Definition

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

$$\Gamma(\mu,\nu) := \{\pi \text{ prob. measure on } \mathbb{R}^d \times \mathbb{R}^d \text{ with marginals } \mu,\nu\} \quad (4)$$

Definition (q-Wasserstein distance (Primal))

$$W_q(\mu,\nu) := \left(\inf_{\pi\in\Gamma(\mu,\nu)} \boldsymbol{E}_{(\mathbf{a},\mathbf{a}')\sim\pi} d(\mathbf{a},\mathbf{a}')^q \right)^{1/q} \quad (5)$$

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

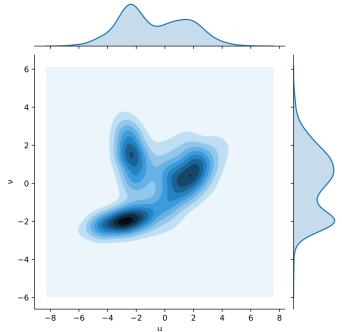


Figure: Two one-dimensional distributions plotted on the x and y axes, and one possible joint distribution that defines a transport plan between them (`https://en.wikipedia.org/wiki/Wasserstein_metric`).

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:

$$\min_{\nu \in \Delta} W_q(\mu, \nu), \tag{6}$$

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

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,

$$\min_{\mathbf{x}} W_1(\mu^\natural, h_{\mathbf{x}} \# p_\Omega), \tag{7}$$

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

- Issues:
  - We only have access to empirical samples $\hat{\mu}_n$ of $\mu^\natural$.
  - W_1 is non-smooth, it cannot be computed exactly.

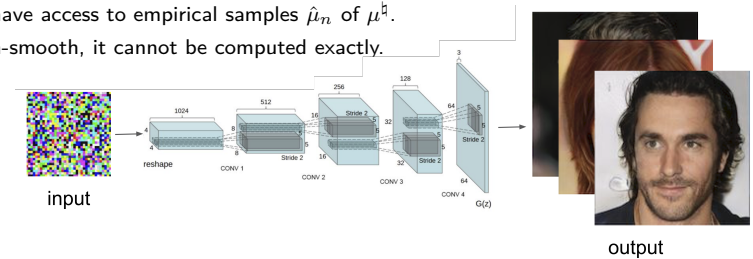
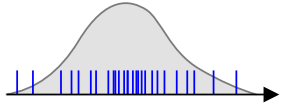


Figure: Schematic of a generative model, $h_{\mathbf{x}} \# \omega$ [9, 13].

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,

$$W_1(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega) \leq W_1(\mu^\natural, \hat{\mu}_n) + W_1(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega), \tag{8}$$

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 [22, 5])

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,

$$W_1(\mu^\natural, \hat{\mu}_n) \gtrsim n^{-1/p}. \tag{9}$$

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)

$$W_1(\mu,\nu) = \sup_{\mathtt{d}} \{\langle \mathtt{d},\mu\rangle - \langle \mathtt{d},\nu\rangle : \mathtt{d} \textit{ is 1-Lipschitz}\} \tag{10}$$

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

Inner product is an expectation

$$\langle \mathtt{d},\mu\rangle = \int \mathtt{d}\mathrm{d}\mu = \int \mathtt{d}(\mathbf{a})\mathrm{d}\mu(\mathbf{a}) = \boldsymbol{E}_{\mathbf{a}\sim\mu}\left[\mathtt{d}(\mathbf{a})\right]. \tag{11}$$

Kantorovich-Rubinstein duality applied to our objective

$$W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\omega\right) = \sup\left\{\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}[\mathtt{d}(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a}\sim h_{\mathbf{x}}\#\omega}[\mathtt{d}(\mathbf{a})] : \mathtt{d} \text{ is 1-Lipschitz}\right\} \tag{12}$$

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:

$$\mathcal{F}(\mu,\nu) \coloneqq \sup_{f\in\mathcal{F}} \langle f,\mu\rangle - \langle f,\nu\rangle = \sup_{f\in\mathcal{F}} \boldsymbol{E}_{\mathbf{a}\sim\mu}[f(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a}\sim\nu}[f(\mathbf{a})] \tag{13}$$

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 [21].
- We can constrain upper bounds on the Lipschitz constant [17].

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:

$$L_{\mathbf{X}_1,\mathbf{X}_2} \leq \|\mathbf{X}_1\|_2 \|\mathbf{X}_2\|_2 \tag{14}$$

Neural Network Distance

Let

$$\mathcal{F} \coloneqq \{h_{\mathbf{X}_1,\mathbf{X}_2}(\mathbf{a}) = \mathbf{X}_2^T \sigma(\mathbf{X}_1\mathbf{a}) : \|\mathbf{X}_2\|_2 \leq 1, \|\mathbf{X}_1\|_2 \leq 1\}. \tag{15}$$

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

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathsf{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\mathsf{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right). \tag{16}$$

General diagram of GANs

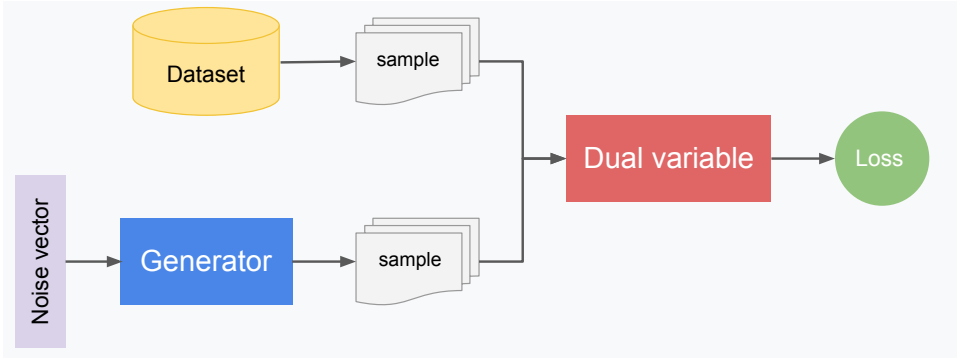


Figure: Generator/dual variable/dataset relation in GANs

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

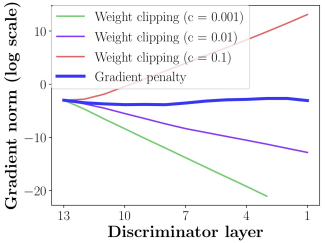
Weight clipping [3]

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

$$\pi_{\mathcal{B}}(\mathbf{y}) = \mathrm{clip}(\mathbf{y}, [-c, c]). \tag{17}$$

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 [11]

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

$$\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}\left[\mathtt{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\Omega}\left[\mathtt{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] + \lambda \boldsymbol{E}_{\mathbf{a}\sim\nu}\left[\left(\|\nabla_{\mathbf{a}} \mathtt{d}_{\mathbf{y}}(\mathbf{a})\|_* - 1\right)^2\right]. \tag{18}$$

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 [19].

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

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

Some historical background for a Turing award

Vanilla GAN [9]

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\log \mathtt{d}_{\mathbf{y}}(\mathbf{a})\right] + \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\log\left(1 - \mathtt{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right)\right] \tag{19}$$

- 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:

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

Wrap up!

- Continuing on Homework 2!

*Sharpness-aware minimization (SAM) [8]

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

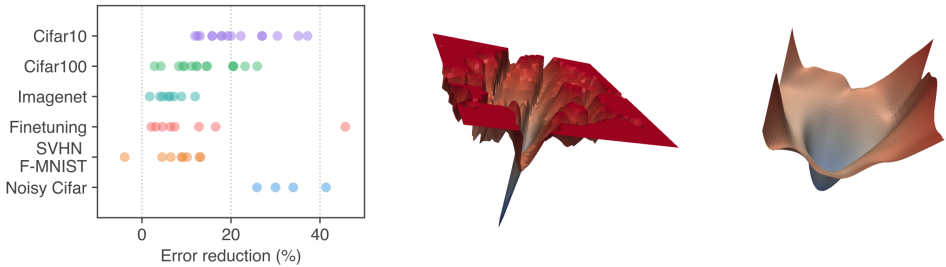


Figure: ResNet trained via SAM converges to a flatter minima **(Right)** compared with the one trained via SGD **(Middle)**, and thus leads to considerable error rate reduction **(Left)** [8].

*Sharpness-aware minimization (SAM) [8]

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

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

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

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

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

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

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

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

References I

[1] Anonymous.
Revisiting adversarial training for the worst-performing class.
Submitted to Transactions of Machine Learning Research, 2022.
Under review.
(Cited on pages 6 and 26.)

[2] Anonymous.
Adversarial training descends without descent: Finding actual descent directions based on danskin's theorem.
In *Submitted to The Eleventh International Conference on Learning Representations*, 2023.
under review.
(Cited on page 22.)

[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 37 and 39.)

References II

[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 6.)

[5] Richard Mansfield Dudley.
The speed of mean glivenko-cantelli convergence.
The Annals of Mathematical Statistics, 40(1):40–50, 1969.
(Cited on page 33.)

[6] Marwa EL HALABI.
Learning with Structured Sparsity: From Discrete to Convex and Back.
PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2018.
(Cited on page 15.)

[7] 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 24.)

References III

[8] 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 6, 43, and 44.)

[9] 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 32 and 41.)

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572, 2014.
(Cited on pages 7 and 9.)

[11] 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 39.)

References IV

[12] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári.
Learning with a strong adversary.
arXiv preprint arXiv:1511.03034, 2015.
(Cited on pages 6, 16, 17, and 18.)

[13] 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 32.)

[14] Ziko Kolter and Aleksander Madry.
Adversarial robustness - theory and practice.
NeurIPS 2018 tutorial: `https://adversarial-ml-tutorial.org/`.
(Cited on page 12.)

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial examples in the physical world.
arXiv preprint arXiv:1607.02533, 2016.
(Cited on pages 12 and 15.)

References V

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial machine learning at scale.
arXiv preprint arXiv:1611.01236, 2016.
(Cited on pages 12 and 15.)

[17] 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 36.)

[18] 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 7, 9, 12, 15, 21, and 23.)

[19] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
arXiv preprint arXiv:1802.05957, 2018.
(Cited on page 39.)

References VI

[20] 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 7.)

[21] 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 36.)

[22] 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 33.)