Adversarial training should be cast as a non-zero sum game

Volkan Cevher

volkan.cevher@epfl.ch

Applied Algorithms for Machine Learning

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

Paris, France

Acknowledgements

- LIONS group members (current & alumni): https://lions.epfl.ch
  - Anastasios Kyrillidis, Quoc Tran Dinh, Fabian Latorre, Ahmet Alacaoglu, Maria Vladarean, Chaehwan Song, Ali Kavis, Mehmet Fatih Sahin, Thomas Sanchez, Thomas Pethick, Igor Krawczuk, Leello Dadi, Paul Rolland, Junhong Lin, Marwa El Halabi, Baran Gozcu, Quang Van Nguyen, Yurii Malitskyi, Armin Eftekhari, Ilija Bogunovic, Yen-Huan Li, Anastasios Kyrillidis, Ya-Ping Hsieh, Bang Cong Vu, Kamal Parameswaran, Jonathan Scarlett, Luca Baldassarre, Bubacarr Bah, Grigorios Chrysos, Stratis Skoulakis, Fanghui Liu, Kimon Antonakopoulos, Andrej Janchevski, Pedro Abranches, Luca Viano, Zhenyu Zhu, Yongtao Wu, Wanyun Xie, Elias Abad Rocamora, Alp Yurtsever.
  - EE-556 (Mathematics of Data): Course material
- Many talented faculty collaborators
  - Panayotis Mertikopoulos, Georgios Piliouras, Kfir Levy, Francis Bach, Joel Tropp, Madeleine Udell, Stephen Becker, Suvrit Sra, Mark Schmidt, Larry Carin, Michael Kapralov, Martin Jaggi, David Carlson, Adrian Weller, Adish Singla, Lorenzo Rosasco, Alessandro Rudi, Stefanie Jegelka, Panos Patrinos, Andreas Krause, Niao He, Bernhard Schölkopf, Olivier Fercoq, George Karypis, Shoham Sabach, Mingyi Hong, Francesco Locatello, Chris Russell, Hamed Hassani, George J. Pappas...
- Many talented collaborators
  - Matthaeus Kleindessner, Puya Latafat, Andreas Loukas, Yu-Guan Hsieh, Samson Tan, Parameswaran Raman, Leena Vankadara

Today: “Basic” robust 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

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}) \quad (\mathrm{argmin}, \mathrm{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\mathrm{argmin}, \mathrm{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\mathrm{argmin} \to \mathbf{x}^\star)$$

Warm up: Flexibility of the template

$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\text{argmin}, \text{argmax} \rightarrow \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{argmin} \rightarrow \mathbf{x}^\star)$$

○ (eula) In the sequel,

- the set $\mathcal{X}$ is convex
- all convergence characterizations are with feasible iterates $\mathbf{x}^k \in \mathcal{X}$
- L-smooth means $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \forall \mathbf{x},\mathbf{y} \in \mathcal{X}$
- ∇ may refer to the generalized subdifferential

Warm up: Flexibility of the template

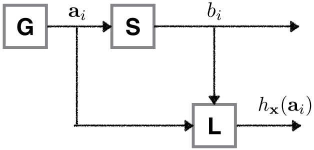
$$\Phi^\star = \min_{\mathbf{x}\in\mathcal{X}} \underbrace{\max_{\mathbf{y}:\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})}_{f(\mathbf{x})} \quad (\text{argmin}, \text{argmax} \to \mathbf{x}^\star, \mathbf{y}^\star)$$

$$f^\star = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{argmin} \to \mathbf{x}^\star)$$

○ (eula) In the sequel,

- the set $\mathcal{X}$ is convex
- all convergence characterizations are with feasible iterates $\mathbf{x}^k \in \mathcal{X}$
- L-smooth means $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \forall \mathbf{x},\mathbf{y} \in \mathcal{X}$
- ∇ may refer to the generalized subdifferential

A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

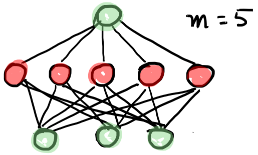
The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

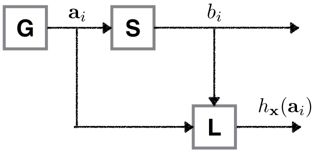
where $\mathcal{X}$ denotes the constraints and L is a loss function.

- ○ A single hidden layer neural network with params $\mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$

$$h_{\mathbf{x}}(\mathbf{a}) := \begin{bmatrix} \mathbf{X}_2 \end{bmatrix} \underbrace{\overset{\text{activation}}{\sigma}\left(\begin{bmatrix} \overset{\text{weight}}{\mathbf{X}_1} \end{bmatrix} \overset{\text{input}}{\begin{bmatrix} \mathbf{a} \end{bmatrix}} + \overset{\text{bias}}{\begin{bmatrix} \mu_1 \end{bmatrix}} \right) + \overset{\text{bias}}{\begin{bmatrix} \mu_2 \end{bmatrix}}}_{\text{hidden layer} \,=\, \text{learned features}}$$



A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and L is a loss function.

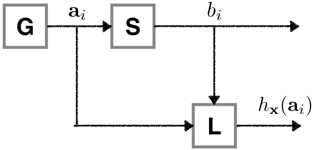
Adversarial Training

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

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

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

A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

The "deep-learning" problem with a neural network $h_{\mathbf{x}}(\mathbf{a})$ is given by

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and L is a loss function.

Example objectives in different tasks

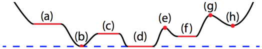
- $\min_{\mathbf{x}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_\infty\le\epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i+\boldsymbol{\delta}\right),\mathbf{b}_i\right)\right]\right\}$ Adversarial training [14].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_2\le\epsilon} L(h_{\mathbf{x}+\boldsymbol{\delta}}\left(\mathbf{a}_i\right),\mathbf{b}_i)\right]\right\}$ ϵ-stability training [5], Sharpness-aware minimization [9].
- $\min_{\mathbf{x}} \max_{\mathbf{b}^c\in[C]} \frac{1}{n_c}\sum_{i=1}^{n_c}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\le\epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i+\boldsymbol{\delta}\right),\mathbf{b}_i^c\right)\right]$ Class fairness [20].

Basic questions on solution concepts

- Consider the finite sum setting

$$\boxed{f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\}.}$$

- Goal: Find $\mathbf{x}^\star$ such that $\nabla f(\mathbf{x}^\star) = 0$.

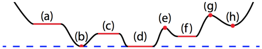


Basic questions on solution concepts

- Consider the finite sum setting

$$\boxed{f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{x}) \right\}.}$$

- Goal: Find $\mathbf{x}^\star$ such that $\nabla f(\mathbf{x}^\star) = 0$.



Vanilla (Minibatch) SGD

Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\boldsymbol{x}^0$, step size α_k

```
1. For $k = 0, 1, \ldots$:
       obtain the (minibatch) stochastic gradient $\mathbf{g}^k$
       update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k \mathbf{g}^k$
```

Solving the outer problem: Gradient computation

Adversarial Training

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

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

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

Question

How can we compute the following stochastic gradient (i.e., $\mathbb{E}_i \nabla_{\mathbf{x}} f_i(\mathbf{x}) = \nabla_{\mathbf{x}} f_i(\mathbf{x})$ for $i \sim \text{Uniform}\{1, \ldots, n\}$):

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

○ **Challenge:** It involves differentiating with respect to a maximization.

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

Theorem ([7])

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

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

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

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

An immediate consequence

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

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

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

Theorem ([7])

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

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

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

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

An immediate consequence

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

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

Observation: ○ Solving the inner problem can be expensive!

A cheap alternative: Fast gradient sign method (FGSM) [10]

Projected gradient descent (PGD) attack: A misnomer

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

$$\begin{aligned}
\hat{\boldsymbol{\delta}}^{(t)} &= \boldsymbol{\delta}^{(t-1)} + \alpha \cdot \mathsf{sign}\left(\nabla_{\boldsymbol{\delta}} L\left(h_{\mathbf{x}}\left(\mathbf{a} + \boldsymbol{\delta}^{(t-1)}\right), b\right)\right) && \text{[Gradient step]} \\
\boldsymbol{\delta}^{(t)} &= \max\left\{\min\left\{\hat{\boldsymbol{\delta}}^{(t)}, \epsilon\right\}, -\epsilon\right\}, && \text{[Projection step]}
\end{aligned}$$

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

$$\boldsymbol{\delta}_{\mathsf{FGSM}} = \epsilon \cdot \mathsf{sign}\left(\nabla_{\boldsymbol{\delta}} L\left(h_{\mathbf{x}}(\mathbf{a}), b\right)\right).$$

Problems:

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

A cheap alternative: Fast gradient sign method (FGSM) [10]

Projected gradient descent (PGD) attack: A misnomer

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

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

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

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

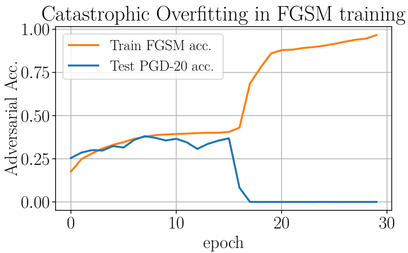
$$\boldsymbol{\delta}_{\mathsf{FGSM}} = \epsilon \cdot \mathsf{sign}\left(\nabla_{\boldsymbol{\delta}} L\left(h_{\mathbf{x}}\left(\mathbf{a}\right), b\right)\right).$$

Problems:

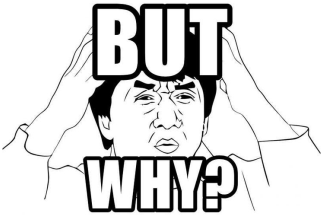
- In Adversarial Training: $\times T$ overhead in training time.
- If $T = 1$, we can observe **Catastrophic Overfitting (CO)**.

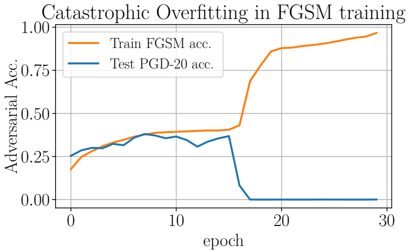
Example:

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



More on CO





Linearizations may not be accurate

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

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

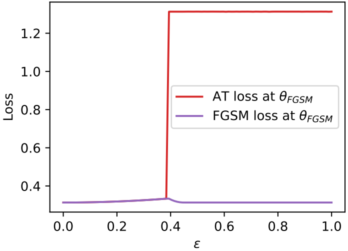
Observation: This property is lost during AT with FGSM and CO appears [2].

A phase transition in adversarial training

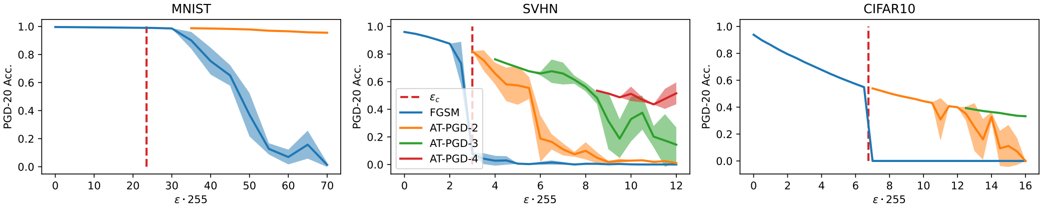
- There is a qualitative increase in difficulty in computation

A toy model for CO (Levi, Abad Rocamora and Cevher, 2024)

Let $\mathrm{a}_2 = \pi/2 = -\mathrm{a}_1$ with labels $b_1 = -1$ and $b_2 = 1$. Let the classifier $h_{\mathrm{x}}(\mathrm{a}) = \sin(\mathrm{x} \cdot \mathrm{a})$ with a single trainable parameter x. CO happens in FGSM AT for $\epsilon > \epsilon_c = \pi/8$.

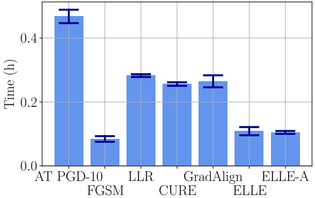


- We provide ϵ_c estimates for the MNIST, SVHN and CIFAR10 datasets.



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

- If it does not succeed by itself, enforce it...



(a) Training time comparison

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

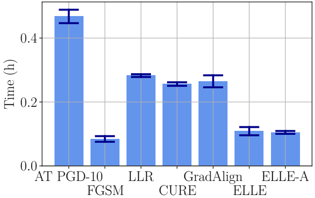
(b) PreActResNet18 in CIFAR10

Algorithmic approaches:

- Local linearization (LLR) [22]
- Curvature regularization (CURE) [19]
- Gradient alignment (GradAlign) [2]
- Efficient local linearity regularization (ELLE) [1]

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

- If it does not succeed by itself, enforce it...



(c) Training time comparison

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

(d) PreActResNet18 in CIFAR10

Algorithmic approaches:

- Local linearization (LLR) [22]
- Curvature regularization (CURE) [19]
- Gradient alignment (GradAlign) [2]
- Efficient local linearity regularization (ELLE) [1]

Question:

- Does the ultimate robustness lie in increasing the inner iterations T (e.g., PGD-10)?

Optimized perturbations are typically not unique!

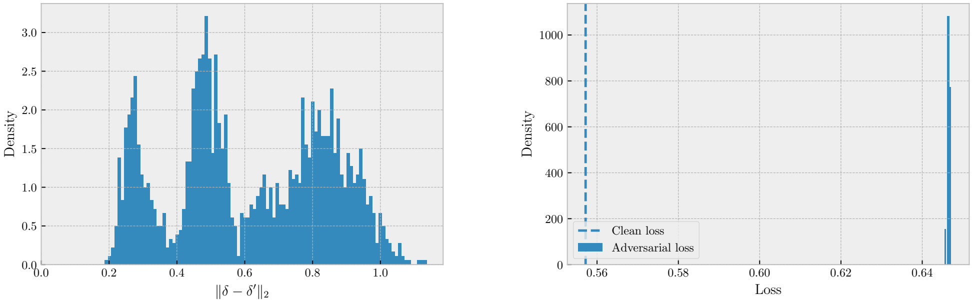
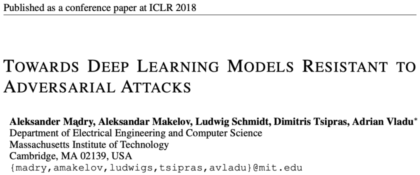


Figure: (*left*) Pairwise ℓ_2-distances between "optimized" perturbations with different initializations are bounded away from zero. (*right*) The losses of multiple perturbations on the same sample concentrate around a value much larger than the clean loss.

Theoretical foundations

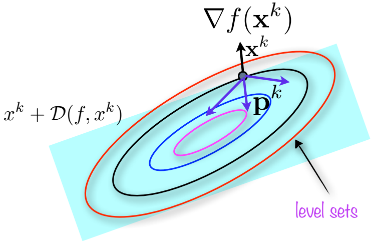
	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x},\delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [16]



Published as a conference paper at ICLR 2018

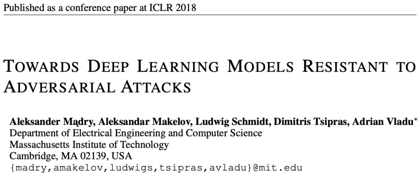
Towards Deep Learning Models Resistant to Adversarial Attacks

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



Theoretical foundations ?

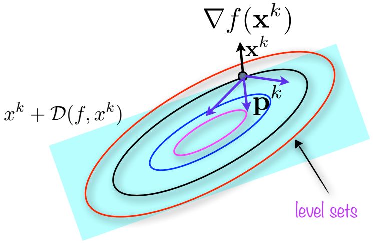
	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x},\delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	descent direction [16]



Published as a conference paper at ICLR 2018

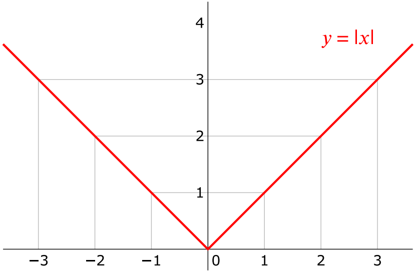
TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

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



A counterexample

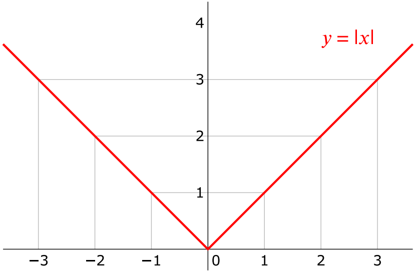
$$f(\mathbf{x}) := \max_{\boldsymbol{\delta} \in [-1,1]} \mathbf{x}\boldsymbol{\delta} = |\mathbf{x}| \, .$$



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

A counterexample

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



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

Our understanding [Latorre, Krawczuk, Dadi, Pethick, Cevher, ICLR (2023)]

- The corollary in [16] is false (it is subtle!).
- We constructed a counter example & proposed an alternative way (DDi) of computing “the gradient”:

	unique $\delta^\star$	non-unique $\delta^\star$
$\nabla_{\mathbf{x}}\Phi(\mathbf{x},\delta^\star)$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	could be ascent direction!

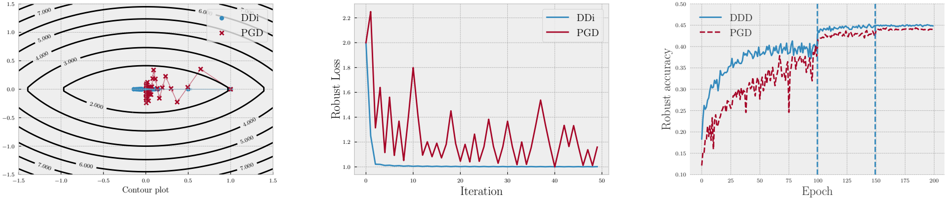


Figure: Left and middle pane: comparison DDi and PGD ([16]) on a synthetic problem. Right pane: DDi vs PGD on CIFAR10.

Comparison with the state-of-the-art

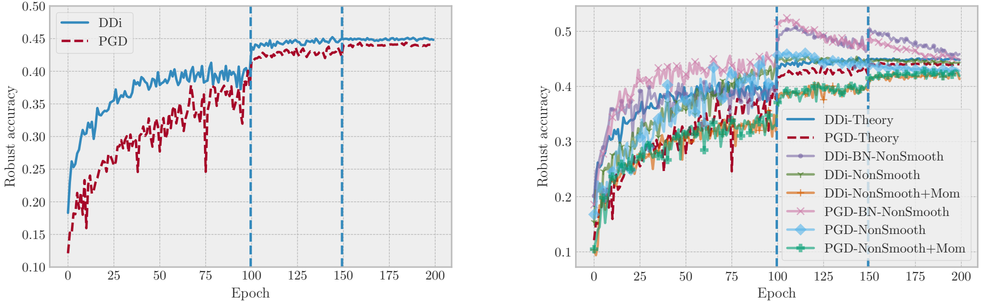


Figure: (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN,ReLU,momentum).

Comparison with the state-of-the-art

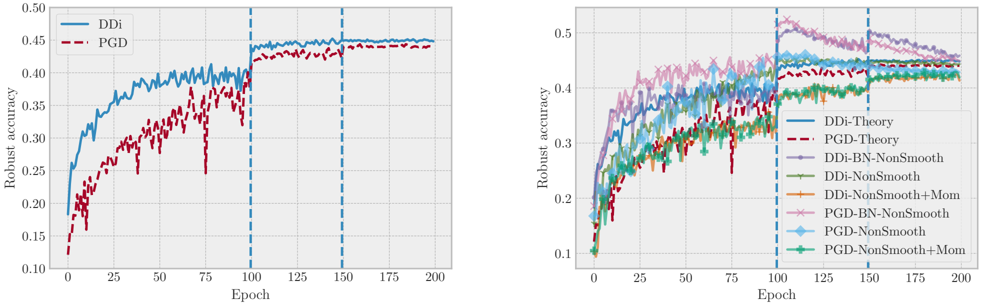


Figure: (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN,ReLU,momentum).

DDi + Graduate Student Descent may improve things (performance or catastrophic overfitting)?

Out of the frying pan into the fire

Original Formulation of Adversarial Training (I)

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

Original Formulation of Adversarial Training (I)

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

which loss L?

Original Formulation of Adversarial Training (II)

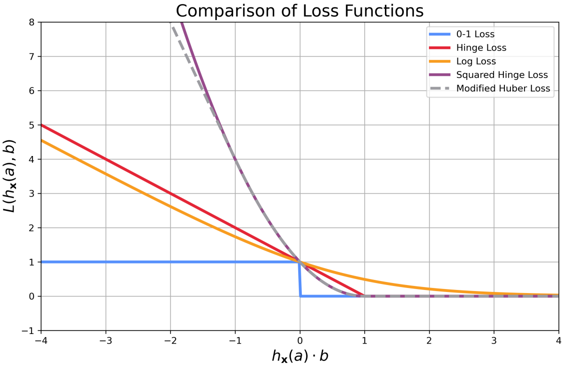
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(h_{\mathbf{x}}\left(\mathbf{a}+\boldsymbol{\delta}\right), b)\right]$$

Original Formulation of Adversarial Training (II)

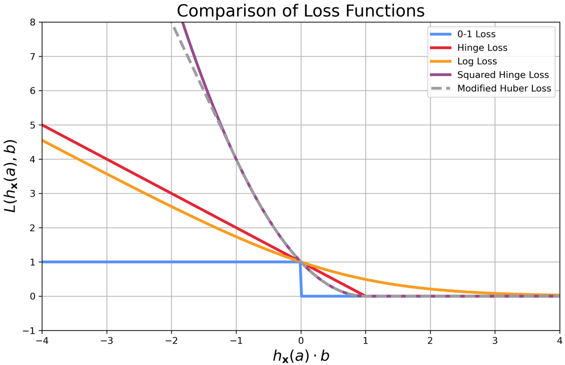
$$\min_{\mathbf{x}} \mathbb{E}\left[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|\leq\epsilon} L_{01}(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

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

Surrogate-based optimization for Risk Minimization



Surrogate-based optimization for Risk Minimization

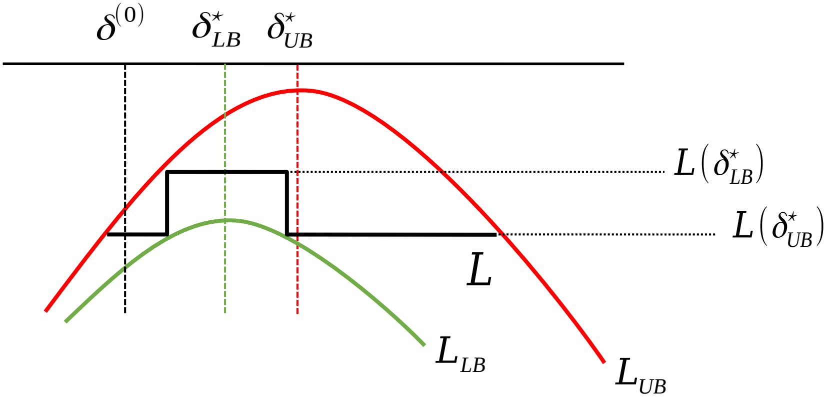


$$\mathbb{E}\left[L_{01}(h_{\mathbf{x}^\star}(\mathbf{a}+\boldsymbol{\delta}), b)\right] \leq \min_{\mathbf{x}} \mathbb{E}\left[L_{\mathsf{CE}}(h_{\mathbf{x}}(\mathbf{a}+\boldsymbol{\delta}), b)\right]$$

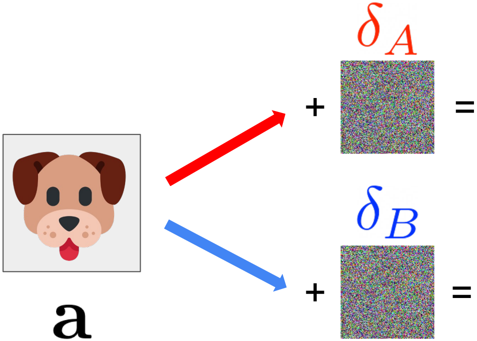
Adversary maximizes an upper bound (I)

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

Adversary maximizes an upper bound (II)



Why maximizing cross-entropy leads to weak adversaries



🐶 🐱 🐸 🤡

(.49, .51, 0, 0)

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

(.26, .24, .25, .25)

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

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

Theorem (Reformulation of the Adversary's problem)

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

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

Bilevel Optimization [Robey,* Latorre,* Pappas, Hassani, Cevher(2023)][1]

- Best targeted attack (BETA) optimization formulation:

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

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

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

[1] https://infoscience.epfl.ch/record/302995 or https://tinyurl.com/33yup77v

Bilevel Optimization [Robey,* Latorre,* Pappas, Hassani, Cevher(2023)][1]

- Best targeted attack (BETA) optimization formulation:

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

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

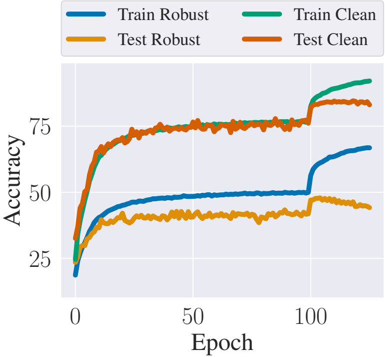
$$j^\star \in \underset{j\in[K]-\{\mathbf{b}_i\}}{\arg\max}\, h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^\star})_j - h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\delta}_{i,j^\star})_{\mathbf{b}_i}$$

Best paper award at ICML AdvML 2023

[1] https://infoscience.epfl.ch/record/302995 or https://tinyurl.com/33yup77v

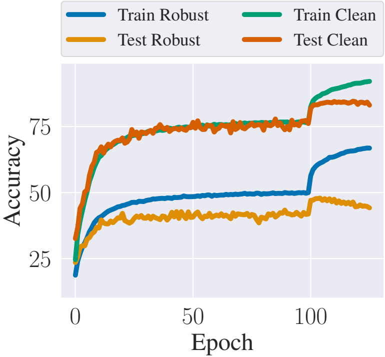
Practical Consequences of the Bilevel Formulation (I)

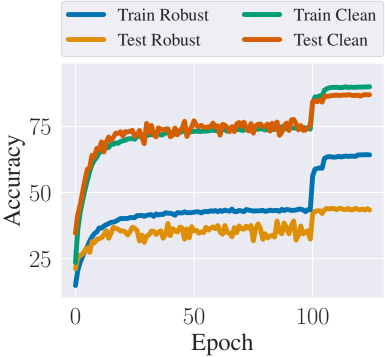
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT



Practical Consequences of the Bilevel Formulation (I)

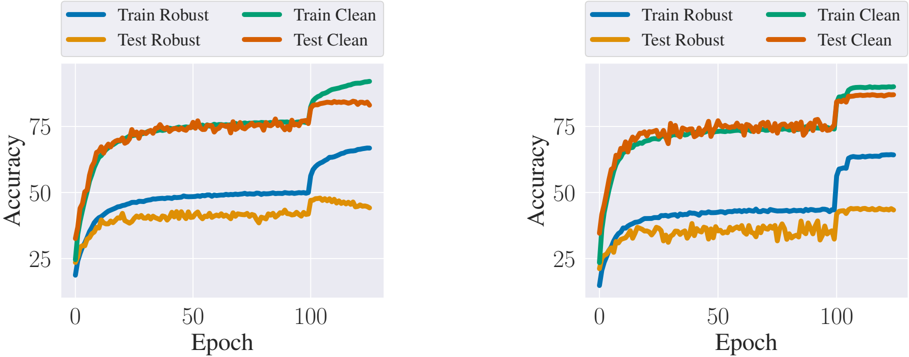
Figure: Learning curves of PGD^{10}-AT (Left) and BETA^{10}-AT (Right). Robust accuracy estimated with PGD^{20}





Practical Consequences of the Bilevel Formulation (I)

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



No Robust Overfitting occurs!

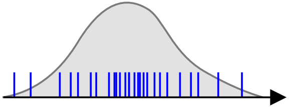
Practical Consequences of the Bilevel Formulation

Table: Adversarial performance on CIFAR-10.

Training algorithm	Test accuracy											
	Clean		FGSM		PGD^{10}		PGD^{40}		BETA^{10}		APGD	
	Best	Last	Best	Last	Best	Last	Best	Last	Best	Last	Best	Last
FGSM	81.96	75.43	**94.26**	**94.22**	42.64	1.49	42.66	1.62	40.30	0.04	41.56	0.00
PGD^{10}	83.71	83.21	51.98	47.39	46.74	39.90	45.91	39.45	43.64	40.21	44.36	42.62
TRADES^{10}	81.64	81.42	52.40	51.31	47.85	42.31	47.76	42.92	44.31	40.97	43.34	41.33
MART^{10}	78.80	77.20	53.84	53.73	49.08	41.12	48.41	41.55	44.81	41.22	45.00	42.90
BETA-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**

Another minimax example: Generative adversarial networks (GANs)

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

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathbf{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathrm{p}_\Omega} \left[\mathbf{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right). \qquad (1)$$

This problem is already captured by the template $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$. Note that the original problem is a direct non-smooth minimization problem and the Rubinstein-Kantarovic duality results in the minimax template.

Remarks:
- Cannot solve in a manner similar to adversarial training a la Danskin. Need a direct approach.
- Scalability, mode collapse, catastrophic forgetting. Heuristics galore!
- Enforce Lipschitz constraint weight clipping, gradient penalty, spectral normalization [3, 12, 18].

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{2}$$

where

- Φ is differentiable and nonconvex in $\mathbf{x}$ and nonconcave in $\mathbf{y}$,
- The domain is unconstrained, specifically $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^n$.

○ Key questions:

1. Where do the algorithms converge?
2. When do the algorithm converge?

Solving the minimax problem: Solution concepts

- Consider the unconstrained setting:

$$\boxed{\Phi^\star = \min_{\mathbf{x}} \max_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y})}$$

- Goal: Find an LNE point $(\mathbf{x}^\star, \mathbf{y}^\star)$.

Definition (Local Nash Equilibrium)

A pure strategy $(\mathbf{x}^\star, \mathbf{y}^\star)$ is called a local Nash equilibrium if

$$\Phi\left(\mathbf{x}^\star, \mathbf{y}\right) \leq \Phi\left(\mathbf{x}^\star, \mathbf{y}^\star\right) \leq \Phi\left(\mathbf{x}, \mathbf{y}^\star\right) \qquad \text{(LNE)}$$

for all $\mathbf{x}$ and $\mathbf{y}$ within some neighborhood of $\mathbf{x}^\star$ and $\mathbf{y}^\star$, i.e., $\|\mathbf{x} - \mathbf{x}^\star\| \leq \varepsilon$ and $\|\mathbf{y} - \mathbf{y}^\star\| \leq \varepsilon$ for some $\varepsilon > 0$.

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{3}$$

where

- Φ is differentiable and nonconvex in $\mathbf{x}$ and nonconcave in $\mathbf{y}$,
- The domain is unconstrained, specifically $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^n$.

○ Key questions:

1. Where do the algorithms converge?
2. When do the algorithm converge?

A buffet of negative results [8]

"Even when the objective is a Lipschitz and smooth differentiable function, deciding whether a min-max point exists, in fact even deciding whether an approximate min-max point exists, is NP-hard. More importantly, an approximate local min-max point of large enough approximation is guaranteed to exist, but finding one such point is PPAD-complete. The same is true of computing an approximate fixed point of the (Projected) Gradient Descent/Ascent update dynamics."

Basic algorithms for minimax

- Given $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$, define $V(\mathbf{z}) = [\nabla_{\mathbf{x}}\Phi(\mathbf{x},\mathbf{y}), -\nabla_{\mathbf{y}}\Phi(\mathbf{x},\mathbf{y})]$ with $\mathbf{z} = [\mathbf{x},\mathbf{y}]$.

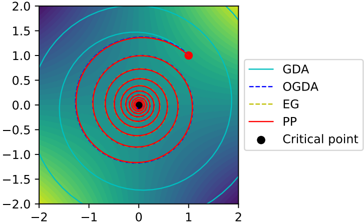


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

- (In)Famous algorithms
  - Gradient Descent Ascent (GDA)
  - Proximal point method (PPM) [23, 11]
  - Extra-gradient (EG) [15]
  - Optimistic GDA (OGDA) [24, 17]
  - Reflected-Forward-Backward-Splitting (RFBS) [6]

- EG and OGDA are approximations of the PPM
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k)$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^{k+1})$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k - \alpha V(\mathbf{z}^k))$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha[2V(\mathbf{z}^k) - V(\mathbf{z}^{k-1})]$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(2\mathbf{z}^k - \mathbf{z}^{k-1})$.

Where do the algorithms converge?

- Recall: Given $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$, define $V(\mathbf{z}) = [\nabla_{\mathbf{x}}\Phi(\mathbf{x},\mathbf{y}), -\nabla_{\mathbf{y}}\Phi(\mathbf{x},\mathbf{y})]$ with $\mathbf{z} = [\mathbf{x},\mathbf{y}]$.

- Given $V(\mathbf{z})$, define stochastic estimates of $V(\mathbf{z},\zeta) = V(\mathbf{z}) + U(\mathbf{z},\zeta)$, where
  - $U(\mathbf{z},\zeta)$ is a bias term,
  - We often have unbiasedness: $\boldsymbol{E}U(\mathbf{z},\zeta) = 0$,
  - The bias term can have bounded moments,
  - We often have bounded variance: $P(\| U(\mathbf{z},\zeta) \| \geq t) \leq 2\exp -\frac{t^2}{2\sigma^2}$ for $\sigma > 0$.

- An abstract template for generalized Robbins-Monro schemes, dubbed as $\mathcal{A}$:

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha_k V(\mathbf{z}^k, \zeta^k).$$

The dessert section in the buffet of negative results: [13]

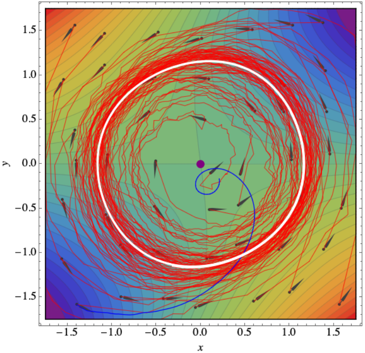
1. Bounded trajectories of $\mathcal{A}$ always converge to an internally chain-transitive (ICT) set.
2. Trajectories of $\mathcal{A}$ may converge with arbitrarily high probability to spurious attractors that contain no critical point of Φ.

Minimax is more difficult than just optimization [13]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [4].
  - For optimization, {attracting ICT} $\equiv$ {solutions}
  - For minimax, {attracting ICT} $\equiv$ {solutions} $\cup$ {spurious sets}

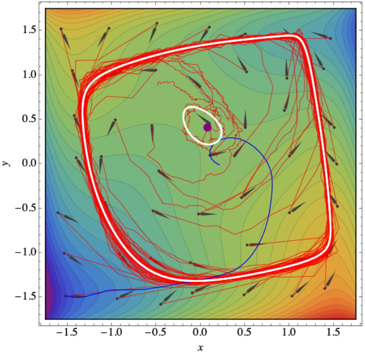
- "Almost" bilinear $\neq$ bilinear:

$$\Phi(x,y) = xy + \epsilon\phi(x), \phi(x) = \frac{1}{2}x^2 - \frac{1}{4}x^4$$



- The "forsaken" solutions:

$$\Phi(y,x) = y(x-0.5) + \phi(y) - \phi(x), \phi(u) = \frac{1}{4}u^2 - \frac{1}{2}u^4 + \frac{1}{6}u^6$$



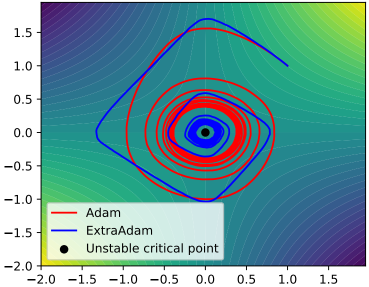
Minimax is more difficult than just optimization [13]

○ Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [4].

- For optimization, $\{\text{attracting ICT}\} \equiv \{\text{solutions}\}$
- For minimax, $\{\text{attracting ICT}\} \equiv \{\text{solutions}\} \cup \{\text{spurious sets}\}$

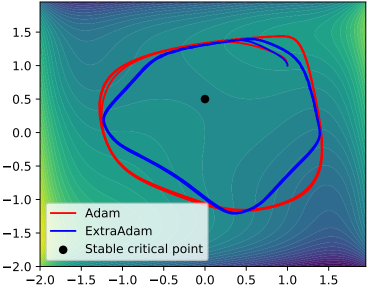
○ "Almost" bilinear ≠ bilinear:

$$\Phi(x, y) = xy + \epsilon\phi(x), \phi(x) = \frac{1}{2}x^2 - \frac{1}{4}x^4$$



○ The "forsaken" solutions:

$$\Phi(y, x) = y(x-0.5) + \phi(y) - \phi(x), \phi(u) = \frac{1}{4}u^2 - \frac{1}{2}u^4 + \frac{1}{6}u^6$$



When do the algorithms converge?

Assumption (weak Minty variational inequality)

For some $\rho \in \mathbb{R}$, weak MVI implies

$$\langle V(\mathbf{z}), \mathbf{z} - \mathbf{z}^\star \rangle \geqslant \rho \|V(\mathbf{z})\|^2, \quad \textit{for all } \mathbf{z} \in \mathbb{R}^n. \qquad (4)$$

- A variant EG+ converges when $\rho > -\frac{1}{8L}$
  - Diakonikolas, Daskalakis, Jordan, AISTATS 2021.
- It still cannot handle the examples of [13].

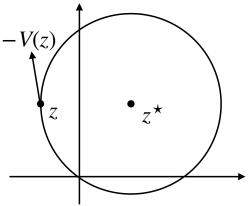
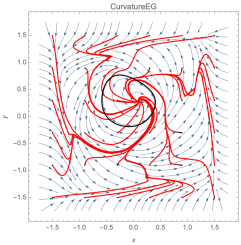


Figure: The operator $V(z)$ is allowed to point away from the solution by some amount when ρ is negative.

- Complete picture under weak MVI (ICLR'22 and '23)
  - Pethick, Lalafat, Patrinos, Fercoq, and Cevher.
  - constrained and regularized settings with $\rho > -\frac{1}{2L}$
  - matching lower bounds
  - stochastic variants handling the examples of [13]
  - adaptive variants handling the examples of [13]



GANs with SEG+ [21]

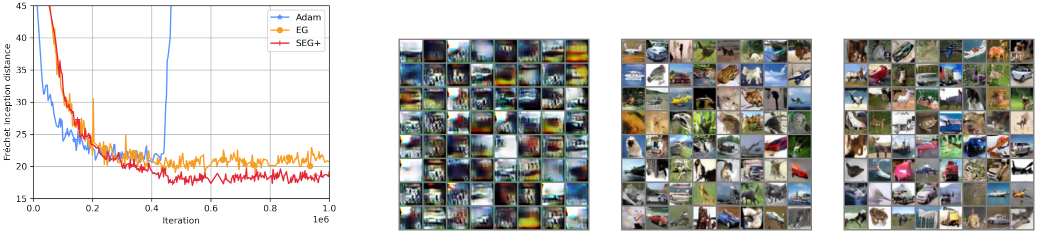
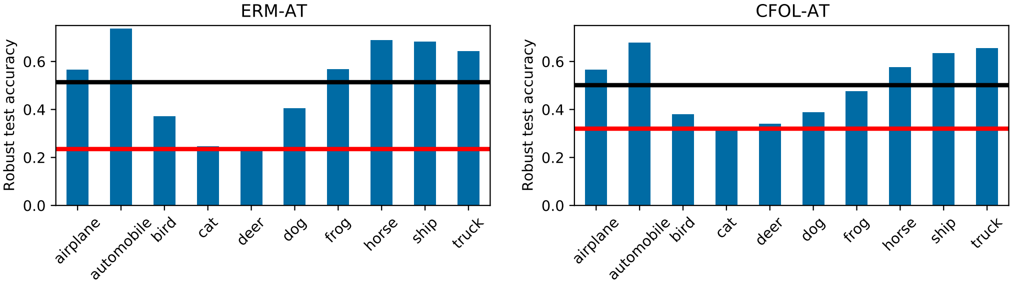


Figure: A performance comparison of GAN training by Adam, EG with stochastic gradients, and SEG+.

Robustness of the worst-performing class [20]



(a)

(b)

Figure: Robust test accuracy of (a) Empirical Risk Minimization and (b) the class focused online learning.

Code: `https://github.com/LIONS-EPFL/class-focused-online-learning-code`

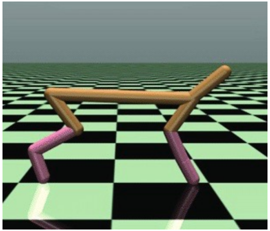
Take home messages

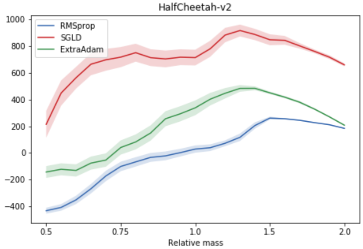
- Even the simplified view of robust & adversarial ML is challenging

- $\min$-$\max$-type has spurious attractors with no equivalent concept in $\min$-type

- Not all step-size schedules are considered in our work: Possible to "converge" under some settings

- Other successful attempts[1] consider "mixed Nash" concepts[2]





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

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

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

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

References I

[1] Elias Abad Rocamora, Fanghui Liu, Grigorios G Chrysos, Pablo M. Olmos, and Volkan Cevher.
Efficient local linearity regularization to overcome catastrophic overfitting.
In *International Conference on Learning Representations*, 2024.
(Cited on pages 20 and 21.)

[2] Maksym Andriushchenko and Nicolas Flammarion.
Understanding and improving fast adversarial training.
Advances in Neural Information Processing Systems, 33:16048–16059, 2020.
(Cited on pages 18, 20, and 21.)

[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 page 47.)

[4] Michel Benaïm and Morris W. Hirsch.
Asymptotic pseudotrajectories and chain recurrent flows, with applications.
Journal of Dynamics and Differential Equations, 8(1):141–176, 1996.
(Cited on pages 53 and 54.)

References II

[5] 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.)

[6] Volkan Cevher and Bang Cong Vu.
A reflected forward-backward splitting method for monotone inclusions involving lipschitzian operators.
Set-Valued and Variational Analysis, pages 1–12, 2020.
(Cited on page 51.)

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

[8] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis.
The complexity of constrained min-max optimization.
arXiv preprint arXiv:2009.09623, 2020.
(Cited on page 50.)

References III

[9] 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 page 10.)

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572, 2014.
(Cited on pages 16 and 17.)

[11] Osman Güler.
On the convergence of the proximal point algorithm for convex minimization.
SIAM J. Control Opt., 29(2):403–419, March 1991.
(Cited on page 51.)

[12] 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 47.)

References IV

[13] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher.
The limits of min-max optimization algorithms: Convergence to spurious non-critical sets.
arXiv preprint arXiv:2006.09065, 2020.
(Cited on pages 52, 53, 54, and 55.)

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

[15] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
(Cited on page 51.)

[16] 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 23, 24, and 27.)

References V

[17] Yura Malitsky and Matthew K Tam.
A forward-backward splitting method for monotone inclusions without cocoercivity.
SIAM Journal on Optimization, 30(2):1451–1472, 2020.
(Cited on page 51.)

[18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
arXiv preprint arXiv:1802.05957, 2018.
(Cited on page 47.)

[19] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard.
Robustness via curvature regularization, and vice versa.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9078–9086, 2019.
(Cited on pages 20 and 21.)

[20] 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 57.)

References VI

[21] Thomas Pethick, Olivier Fercoq, Puya Latafat, Panagiotis Patrinos, and Volkan Cevher.
Solving stochastic weak minty variational inequalities without increasing batch size.
In *The Eleventh International Conference on Learning Representations*, 2023.
(Cited on page 56.)

[22] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli.
Adversarial robustness through local linearization.
Advances in neural information processing systems, 32, 2019.
(Cited on pages 20 and 21.)

[23] R. Tyrrell Rockafellar.
Convex Analysis.
Princeton Univ. Press, Princeton, NJ, 1970.
(Cited on page 51.)

[24] Martin Zinkevich.
Online convex programming and generalized infinitesimal gradient ascent.
In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.
(Cited on page 51.)