Optimization Challenges in Adversarial Machine Learning

Volkan Cevher
https://lions.epfl.ch

DS3@École Polytechnique

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

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

Acknowledgements

- Material from EE-556 Mathematics of Data course at EPFL
  - https://www.epfl.ch/labs/lions/teaching/ee-556-mathematics-of-data-from-theory-to-computation/slides-2020/
- LIONS group members (current & alumni): https://lions.epfl.ch
  - 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, Ilija Bogunovic, Yen-Huan Li, Anastasios Kyrillidis, Ya-Ping Hsieh, Bang Cong Vu, Kamal Parameswaran, Jonathan Scarlett, Luca Baldassarre, Bubacarr Bah, Alp Yurtsever.
- Many talented collaborators
  - Panayotis Mertikopoulos, Kfir Levy, Francis Bach, Joel Tropp, Madeleine Udell, Stephen Becker, Suvrit Sra, Mark Schmidt, Larry Carin, Olivier Fercoq

Today: 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

Warm up

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

Warm up

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

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

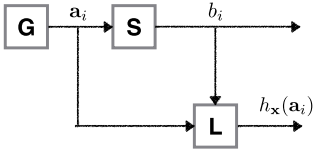
Warm up

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

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

- In the sequel,
  - the set $\mathcal{X}$ is convex and has a tractable projection operator $\pi_{\mathcal{X}}$
  - all convergence characterizations are with feasible iterates $\mathbf{x}^k \in \mathcal{X}$
  - gradient mapping means $G_\alpha(\mathbf{x}^k) = \frac{1}{\alpha}[\mathbf{x}^k - \pi_{\mathcal{X}}(\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k))]$, where α is the step-size
  - 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, and $\delta_{\mathcal{X}}$ refers to the indicator function for the set $\mathcal{X}$

A deep learning optimization problem in supervised learning



Definition (Optimization formulation)

The deep-learning training problem is given by

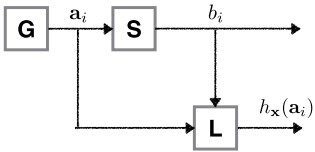
$$\mathbf{x}^\star_{\mathsf{DL}} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints on the parameters.

Some frequently used loss functions

- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \log(1 + \exp(-b h_{\mathbf{x}}(\mathbf{a})))$ *logistic loss*
- $L(h_{\mathbf{x}}(\mathbf{a}), b) = (b - h_{\mathbf{x}}(\mathbf{a}))^2$ *squared error*
- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \max(0, 1 - b h_{\mathbf{x}}(\mathbf{a}))$ *hinge loss*

A deep learning optimization problem in supervised learning



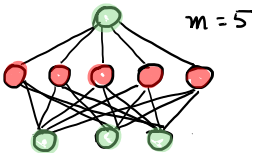
Definition (Optimization formulation)

The deep-learning training problem is given by

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

where $\mathcal{X}$ denotes the constraints on the parameters.

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



$$h_{\mathbf{x}}(\mathbf{a}) := \begin{bmatrix} \mathbf{X}_2 \end{bmatrix} \sigma \left(\begin{bmatrix} \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \end{bmatrix} + \begin{bmatrix} \mu_1 \end{bmatrix} \right) + \begin{bmatrix} \mu_2 \end{bmatrix}$$

activation (over σ); weight (over $\mathbf{X}_1$); input (over $\mathbf{a}$); bias (over μ_1); bias (over μ_2)

hidden layer = learned features

An approximation theoretic motivation: Why neural networks?

Theorem (Universal approximation [7])

Let $\sigma(\cdot)$ be a nonconstant, bounded, and increasing continuous function. Let $I_d = [0,1]^d$. The space of continuous functions on I_d is denoted by $\mathcal{C}(I_d)$.

Given $\epsilon > 0$ and $g \in \mathcal{C}(I_d)$ there exists a 1-hidden-layer network h with m neurons such that h is an ϵ-approximation of g, i.e.,

$$\sup_{\mathbf{a} \in I_d} |g(\mathbf{a}) - h(\mathbf{a})| \leq \epsilon$$

Caveat

The number of neurons m needed to approximate some function g can be arbitrarily large!

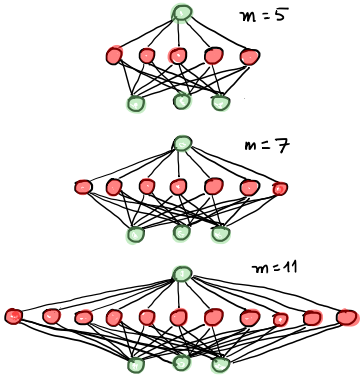
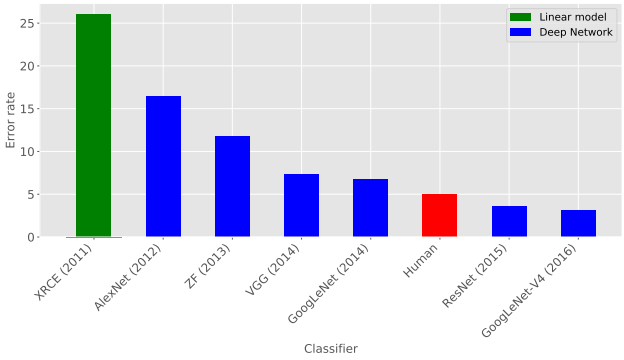
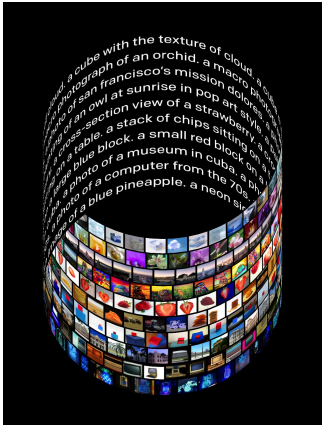


Figure: Neural networks of increasing width

A more realistic motivation: Why neural networks?

- Beyond human performance in applications





Size of neural networks grows exponentially!

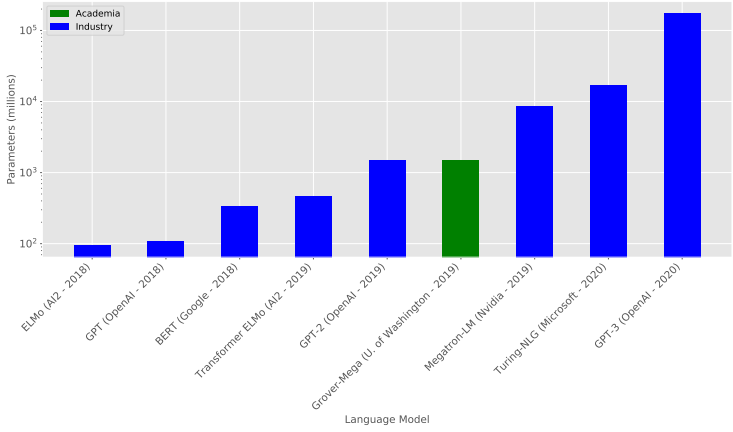


Figure: Number of parameters in language models based on deep learning.

Towards training with neural networks

- What do we have at hand?
  1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
  2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

Towards training with neural networks

- What do we have at hand?
  1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
  2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

- Barriers to training of neural networks:
  1. Curse-of-dimensionality
  2. Non-convexity
  3. Ill-conditioning

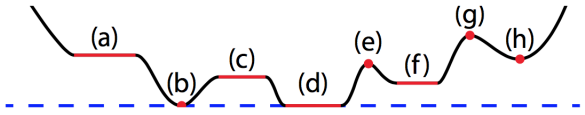


Figure: A non-convex function. (a) and (c) are plateaus, (b) and (d) are global minima, (f) and (h) are local minima, (e) and (g) are local maxima. [19]

Towards training with neural networks

- What do we have at hand?
  1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
  2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

- Barriers to training of neural networks:
  1. Curse-of-dimensionality → first-order methods
  2. Non-convexity → stochasticity + momentum
  3. Ill-conditioning → adaptive gradient methods

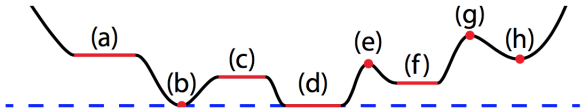


Figure: A non-convex function. (a) and (c) are plateaus, (b) and (d) are global minima, (f) and (h) are local minima, (e) and (g) are local maxima. [19]

Basic first-order methods: GD and SGD

- Consider the finite sum (e.g., ERM) setting

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

Algorithms in the finite sum setting

Gradient Descent	**Stochastic Gradient Descent**
$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)$	$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \theta_k)$
$\nabla f(\mathbf{x}^k) = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\mathbf{x}^k)$	$G(\mathbf{x}^k, \theta_k) = \nabla f_j(\mathbf{x}^k)$, $j \sim \text{Uniform}(\{1, \cdots, n\})$
α_k can be constant	$\mathbb{E}[G(\mathbf{x}^k, \theta_k)] = \nabla f(\mathbf{x}^k)$

- We will mostly focus on SGD in the sequel due to its scalability and generalization performance.

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size α_k
1. For $k = 0, 1, \ldots$:   obtain the (minibatch) stochastic gradient $\mathbf{g}^k$ update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k \mathbf{g}^k$

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size α_k
1. For $k = 0, 1, \ldots$:   obtain the (minibatch) stochastic gradient $\mathbf{g}^k$ update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k \mathbf{g}^k$

Perturbed Stochastic Gradient Descent [12]
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size α_k
1. For $k = 0, 1, \ldots$:   sample noise ξ uniformly from unit sphere   update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k(\mathbf{g}^k + \xi)$

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD

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

1. For $k = 0, 1, \ldots$:

obtain the (minibatch) stochastic gradient $\mathbf{g}^k$

update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k \mathbf{g}^k$

Perturbed Stochastic Gradient Descent [12]

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

1. For $k = 0, 1, \ldots$:

sample noise ξ uniformly from unit sphere

update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k(\mathbf{g}^k + \xi)$

Stochastic Gradient Langevin Dynamics [53]

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

1. For $k = 0, 1, \ldots$

sample noise ξ standard Gaussian

update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k \mathbf{g}^k + \sqrt{2\alpha_k}\xi$

Basic questions:

1. Does SGD converge with probability 1?
2. Does SGD avoid non-minimum points with probability 1?
3. How fast does SGD converge to local minimizers?

Critical points

Recall (Classification of critical points)

Let $f : \mathbb{R}^p \to \mathbb{R}$ be twice differentiable and let $\bar{\mathbf{x}}$ be a critical point, i.e., $\nabla f(\bar{\mathbf{x}}) = 0$. Let $\{\lambda_i\}_{i=1}^d$ be the eigenvalues of the hessian $\nabla^2 f(\bar{\mathbf{x}})$, then

- $\lambda_i > 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a local minimum
- $\lambda_i < 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a local maximum
- $\lambda_i > 0$, $\lambda_j < 0$ for some i, j and $\lambda_i \neq 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a saddle point
- Other cases $\Rightarrow$ inconclusive

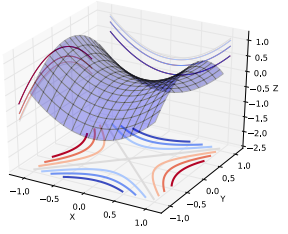


Figure: Minmax saddle ($\lambda_i \neq 0$ for all i)

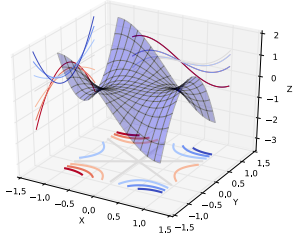


Figure: Monkey saddle ($\lambda_i = 0$ for some i)

Q1: Does SGD converge?

- SGD converges to the critical points of f as $k \to \infty$.

1. GD converges from any intialization with constant step-size and full gradients
2. With probability 1, (P)SGD does not converge with constant step-size α [2, 45]
3. With probability 1, SGD converges with vanishing step-size if $\mathbf{x}^k$ is bounded with probability 1 [34, 2]

Boundedness is not required (Theorem 1 of [37])

Assume Lipschitzness, sublevel regularity, $\mathbb{E}\|\mathbf{g}\|^q \leq \sigma^q$ and $\sum_k \alpha_k^{1+q/2} < \infty$ ($q \geq 2$). Then, $\mathbf{x}^k$ converges with probability 1.

Q2: Does SGD avoid saddle points?

- SGD avoids strict saddles ($\lambda_{\min}(\nabla^2 f(\bar{\mathbf{x}})) < 0$)

1. GD avoids strict saddles from almost all initializations [30]
2. With probability $1-\zeta$, PSGD with constant α escapes strict saddles after $\Omega\left(\log(1/\zeta)/\alpha^2\right)$ iterations [13]
   - However, SGD does not converge with constant α
   - We cannot take $\zeta = 0$

SGD avoids traps almost surely (Theorem 3 of [37])

Assume bounded uniformly exciting noise and $\alpha_k = \mathcal{O}\left(\frac{1}{k^\kappa}\right)$ for $\kappa \in (0, 1]$. Then, SGD avoids strict saddles from any initial condition with probability 1.

Q3: How fast does SGD converge to local minimizers?

- SGD remains close to Hurwicz minimizers (i.e., $\mathbf{x}^* : \lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) > 0$)

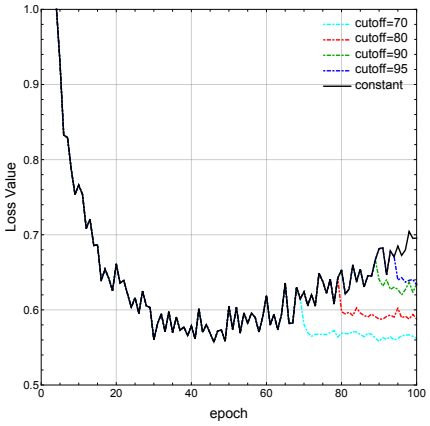
1. SGD with constant α can obtain objective value ϵ-close to a Hurwicz minimizer in $\mathcal{O}(1/\epsilon^2)$-iterations [13, 14]
   - However, SGD does not converge with constant α
   - Need averaging which is problematic in non-convex optimization

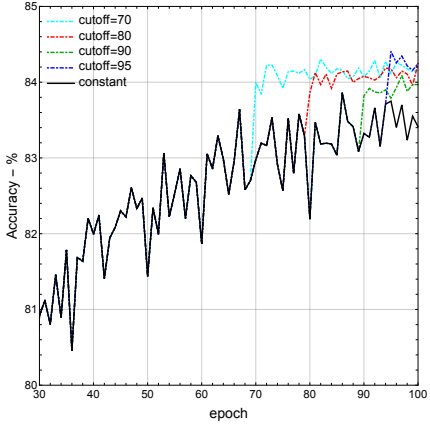
Using a vanishing step-size helps! (Theorem 4 of [37])

Using $\alpha_k = \mathcal{O}\left(\frac{1}{k}\right)$, SGD enjoys a $\mathcal{O}\left(\frac{1}{k}\right)$ convergence rate in objective value.

Using $1/k$ step-size decrease helps in practice

- ResNet training at different cool-down cut-offs





Worst-case iteration complexities of classical projected first-order methods[1][2]

$f(\mathbf{x})$	gradient oracle	L-smooth	Stationarity measure	GD/SGD	Accelerated GD/SGD
Convex	stochastic	yes	$f(\mathbf{x}^k) - f^\star =$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$
Convex	deterministic	yes	$f(\mathbf{x}^k) - f^\star =$	$\mathcal{O}\left(\frac{1}{k}\right)$	$\mathcal{O}\left(\frac{1}{k^2}\right)$
Convex	stochastic	no	$f(\mathbf{x}^k) - f^\star =$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$
Nonconvex	stochastic	yes	$\|G_\alpha(\mathbf{x}^k)\|^2 =$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$
Nonconvex	deterministic	yes	$\|G_\alpha(\mathbf{x}^k)\|^2 =$	$\mathcal{O}\left(\frac{1}{k}\right)^4$	$\mathcal{O}\left(\frac{1}{k}\right)^4$
Nonconvex	stochastic	no	$\text{dist}(0, \partial(f(\mathbf{x}^k) + \delta_{\mathcal{X}}(\mathbf{x}^k)))^2 =$	?[3][5][6]	?[3][5][6]

- Basic structures, such as smoothness or strong convexity, help, but there are more structures that can be used:
  - $\max$-form, metric subregularity, Polyak-Lojasiewicz, Kurdyka-Lojasiewicz, weak convexity,[3] growth cond...

[1] Y. Nesterov, "Introductory lectures on convex optimization: A basic course," Springer Science, 2013.

[2] Y. Carmon, J.C. Duchi, O. Hinder, and A. Sidford, "Lower bounds for finding stationary points I–II." Mathematical Programming, 2019.

[3] D. Davis and D. Drusvyatskiy, "Stochastic model-based minimization of weakly convex functions," SIOPT, 2019.

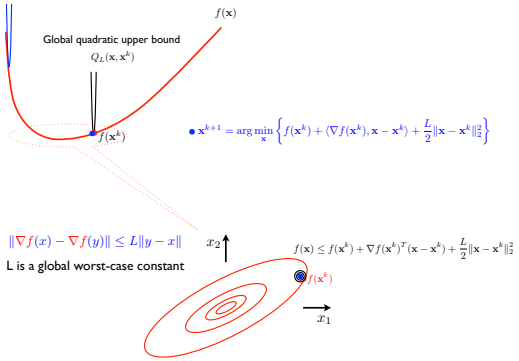
[4] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," MathProg, 2016.

[5] J. Zhang, et al., "On complexity of finding stationary points of nonsmooth nonconvex functions," arXiv:2002.04130, 2020.

[6] O. Shamir, "Can We Find Near-Approximately-Stationary Points of Nonsmooth Nonconvex Functions?" arXiv:2002.11962, 2020.

Worst-case is often too pessimistic

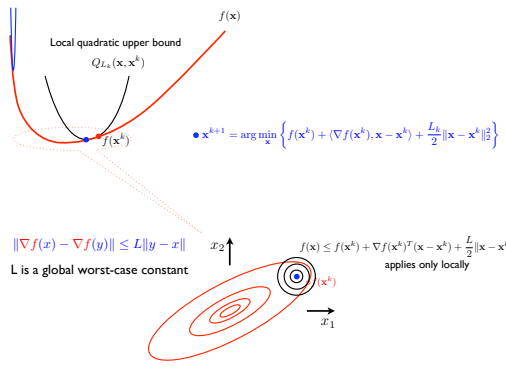
- GD: $x^{k+1} = x^k - \frac{1}{L}\nabla f(x^k)$



- Rates are not everything!
  - overall computational effort is what matters
  - constants &implementations are key
- Knowledge of smoothness, the value of L,...
  - challenging
- Must "somehow" adapt to a "different" function
  - online and without knowing L
  - can reduce overall computational effort!

A classical approach: Line-search

- Long history: Backtracking, Armijo, steepest descent...



- Universal accelerated gradient method[1]

$$f(x^k) - f^\star = \mathcal{O}\left(\frac{L_\nu \|x^0 - x^\star\|^{1+\nu}}{k^{\frac{1+3\nu}{2}}} \right)$$

  - adapts to Hölder smoothness ($\nu \in [0, 1]$)

$$\boxed{\|\nabla f(x) - \nabla f(y)\|_2 \leq L_\nu \|x - y\|_2^\nu}$$

  - has extensions to primal-dual optimization[2]
  - sets accuracy a priori & monotonic step-sizes

- Not as universal as we wish it to be
  - different procedures for stochastic gradients[3]

[1] Y. Nesterov, "Universal Gradient Methods for Convex Optimization Problems," Mathematical Programming, 2015.
[2] A. Yurtsever, Q. Tran-Dinh, and V. Cevher, "A Universal Primal-Dual Convex Optimization Framework," NeurIPS, 2015.
[3] S. Vaswani et al., "Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates," NeurIPS, 2019.

Stochastic adaptive first-order methods

Adaptive methods

Stochastic adaptive methods converge without knowing the smoothness constant.

They do so by making use of the information from stochastic gradients and their norms.

Stochastic adaptive first-order methods

Adaptive methods

Stochastic adaptive methods converge without knowing the smoothness constant.

They do so by making use of the information from stochastic gradients and their norms.

- Additional resources:
  - My OPT-ML talk on universality and adaptation at NeurIPS: https://t.co/sw9qfiXYfW?amp=1
  - Eusipco tutorial with Ahmet Alacaoglu, Ali Kavis, and Kfir Levy: https://tinyurl.com/y2nu6m5c

Variable metric stochastic gradient descent algorithm

Variable metric stochastic gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\mathbf{g}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Variable metric stochastic gradient descent algorithm

Variable metric stochastic gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\mathbf{g}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Common choices of the variable metric $\mathbf{H}_k$

- $\mathbf{H}_k := \lambda_k \mathbf{I}$ $\implies$ stochastic gradient descent method.
- $\mathbf{H}_k := \mathbf{D}_k$ (a positive diagonal matrix) $\implies$ stochastic adaptive gradient methods.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past stochastic gradient information.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past stochastic gradient information.

- Roughly speaking, $\mathbf{H}_k = \text{function}(\mathbf{g}^1, \mathbf{g}^2, \cdots, \mathbf{g}^k)$
- Some well-known examples:

AdaGrad [9]

$$\mathbf{H}_k = \sqrt{\sum_{t=1}^{k} \mathbf{g}^k \mathbf{g}^{k\top}}$$

RmsProp [50]

$$\mathbf{H}_k = \sqrt{\beta \mathbf{H}_{k-1} + (1-\beta)\text{diag}(\mathbf{g}^k)^2}$$

ADAM [25]

$$\hat{\mathbf{H}}_k = \beta \hat{\mathbf{H}}_{k-1} + (1-\beta)\text{diag}(\mathbf{g}^k)^2$$
$$\mathbf{H}_k = \sqrt{\hat{\mathbf{H}}_k/(1-\beta^k)}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

- If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes stochastic gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How does the step-size adapt?

If the stochastic gradient $\|\mathbf{g}^k\|$ is large/small $\rightarrow$ AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [31]

1. Set $Q^0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} Q^k & = Q^{k-1} + \|\mathbf{g}^k\|^2 \\ \mathbf{H}_k & = \sqrt{Q^k} I \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

- If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes stochastic gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How does the step-size adapt?

If the stochastic gradient $\|\mathbf{g}^k\|$ is large/small $\rightarrow$ AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [31]

1. Set $Q^0 = 0$.

2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} Q^k & = Q^{k-1} + \|\mathbf{g}^k\|^2 \\ \mathbf{H}_k & = \sqrt{Q^k} I \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

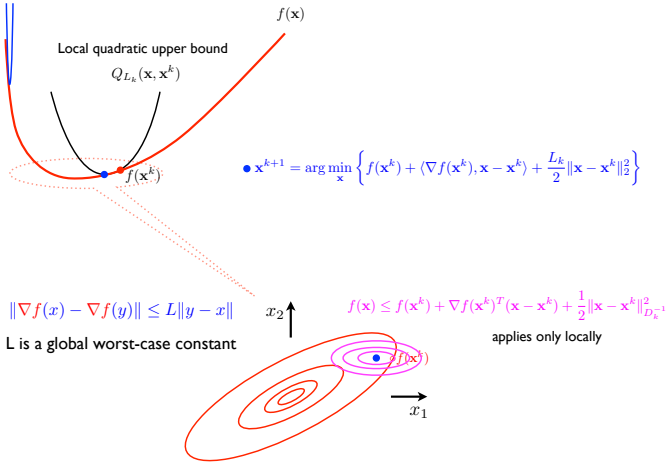
Adaptation through first-order information

- When $H_k = \lambda_k I$, AdaGrad estimates local geometry through stochastic gradient norms.
- Akin to estimating a local quadratic upper bound (majorization / minimization) using gradient history.

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

Adaptation strategy with a positive diagonal matrix $\mathbf{D}_k$

Adaptive step-size + coordinate-wise extension = adaptive step-size for each coordinate



AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

- Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

- For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)

1. Set $\mathbf{Q}^0 = 0$.

2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \operatorname{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

- Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

- For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)

1. Set $\mathbf{Q}^0 = 0$.

2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \operatorname{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

Adaptation across each coordinate

- When $\mathbf{H}_k = \mathbf{D}_k$, we adapt across each coordinate individually.
- Essentially, we have a finer treatment of the function we want to optimize.

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

RMSProp

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{Q}^k & = \beta \mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

RMSProp

1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{Q}^k & = \beta \mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

Remarks:

- RMSProp uses weighted averaging with constant β
- Recent gradients have greater importance

Example: AdaGrad vs. RMSProp

Setting:

- $f(x) = x^4$ (one-dimensional function)
- $x^0 = 10$, $x^\star = 0$

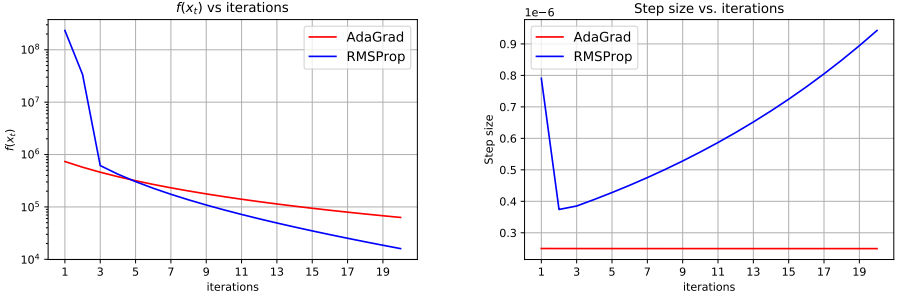


Figure: RMSProp vs. AdaGrad

ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

RMSProp + 2nd order moment estimation = ADAM

ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

RMSProp + 2nd order moment estimation = ADAM

ADAM

Input. Step size α, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$

1. Set $\mathbf{m}_0, \mathbf{v}_0 = 0$

2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{g}_k & = \nabla f(\mathbf{x}^{k-1}) \\ \mathbf{m}_k & = \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^{\cdot 2} \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{m}}_k & = \mathbf{m}_k/(1-\beta_1^k) \leftarrow \text{Bias correction} \\ \hat{\mathbf{v}}_k & = \mathbf{v}_k/(1-\beta_2^k) \leftarrow \text{Bias correction} \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} + \epsilon \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha \hat{\mathbf{m}}_k ./ \mathbf{H}_k \end{cases}$$

Output : $\mathbf{x}^k$

(Every vector operation is an element-wise operation)

Non-convergence of ADAM and a new method: AmsGrad

- It has been shown that ADAM may not converge for *some* objective functions [56].
- An ADAM alternative is proposed that is proved to be convergent [43].

AmsGrad

Input. Step size $\{\gamma_k\}_{k\in\mathbb{N}}$, exponential decay rates $\{\beta_{1,k}\}_{k\in\mathbb{N}}$, $\beta_2 \in [0,1)$

1. Set $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$ and $\hat{\mathbf{v}}_0 > 0$

2. For $k = 1, 2, \ldots$, iterate

$$\begin{cases} \mathbf{g}_k & = G(\mathbf{x}^k, \theta) \\ \mathbf{m}_k & = \beta_{1,k}\mathbf{m}_{k-1} + (1-\beta_{1,k})\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2\mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^{\cdot 2} \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{v}}_k & = \max\{\hat{\mathbf{v}}_{k-1}, \mathbf{v}_k\} \text{ and } \hat{\mathbf{V}}_k = \mathsf{diag}(\hat{\mathbf{v}}_k) \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} \\ \mathbf{x}^{k+1} & = \Pi_{\mathcal{X}}^{\sqrt{\hat{\mathbf{V}}_k}}(\mathbf{x}^k - \gamma_k \hat{\mathbf{m}}_k ./ \mathbf{H}_k) \end{cases}$$

Output : $\mathbf{x}^k$

where $\pi_{\mathcal{K}}^{\mathbf{A}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \langle (\mathbf{x}-\mathbf{y}), \mathbf{A}(\mathbf{x}-\mathbf{y})\rangle$ (weighted Euclidean projection onto $\mathcal{K}$).
(Every vector operation is an element-wise operation)

Example: Least squares with synthetic data

Setting:

- $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$
- $\mathbf{A} \in \mathbb{R}^{n \times p}$, $\mathbf{A} \sim N(\mu, \sigma^2 I)$
- $n = 1000$, $p = 1000$

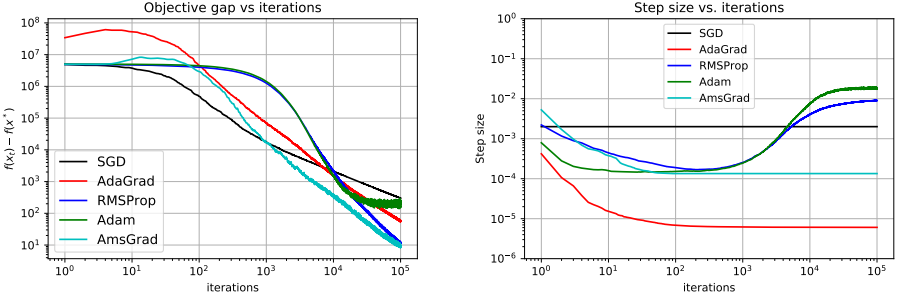


Figure: Comparison of convergence rate and stepsize evolution. Mini-batch stochastic gradients with a batch size of 20

A comparison of algorithms

	GD/SGD	Accelerated GD/SGD	AdaGrad	AcceleGrad/UniXgrad	Adam/AMSGrad
Convex, stochastic	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[1]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[1]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[2]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[3,4]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[5]
Convex, deterministic, L-smooth	$\mathcal{O}\left(\frac{1}{k}\right)$[1]	$\mathcal{O}\left(\frac{1}{k2}\right)$[1]	$\mathcal{O}\left(\frac{1}{k}\right)$[3]	$\mathcal{O}\left(\frac{1}{k2}\right)$[3,4]	$\mathcal{O}\left(\frac{1}{k}\right)$[6]
Nonconvex, stochastic, L-smooth	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[1]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[1]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[7]	?	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$[8]
Nonconvex, deterministic, L-smooth	$\mathcal{O}\left(\frac{1}{k}\right)$[1]	$\mathcal{O}\left(\frac{1}{k}\right)$[1]	$\mathcal{O}\left(\frac{1}{k}\right)$[7]	?	$\mathcal{O}\left(\frac{1}{k}\right)$[6]

[1] Lan, First-order and Stochastic Optimization Methods for Machine Learning. Springer Nature, 2020.

[2] Duchi, Hazan, Singer, Adaptive subgradient methods for online learning and stochastic optimization, JMLR, 2011

[3] Levy, Yurtsever, Cevher, Online adaptive methods, universality and acceleration, NeurIPS 2018

[4] Kavis, Levy, Bach, Cevher, UniXGrad: A Universal, Adaptive Algorithm with Optimal Guarantees for Constrained Optimization, NeurIPS, 2019

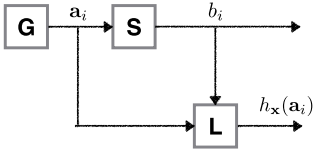
[5] Reddi, Kale, Kumar, On the convergence of adam and beyond, ICLR, 2018.
Alacaoglu, Malitsky, Mertikopoulos, Cevher, A new regret analysis for Adam-type algorithms, ICML 2020.

[6] Barakat, Bianchi, Convergence Rates of a Momentum Algorithm with Bounded Adaptive Step Size for Nonconvex Optimization, ACML, 2020

[7] Ward, Xu, Bottou, AdaGrad stepsizes: Sharp convergence over nonconvex landscapes, ICML 2019.

[8] Alacaoglu, Malitsky, Cevher, Convergence of adaptive algorithms for weakly convex constrained optimization, arXiv, 2020.
Chen, Zhou, Tang, Yang, Cao, Gu, Closing the generalization gap of adaptive gradient methods in training deep neural networks, IJCAI 2020.
Chen, Liu, Sun, Hong, On the convergence of a class of adam-type algorithms for non-convex optimization, ICLR 2018.

From empirical risk minimization...



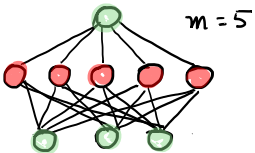
Definition (Optimization formulation)

The deep-learning training problem is given by

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

where $\mathcal{X}$ denotes the constraints on the parameters.

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



$$h_{\mathbf{x}}(\mathbf{a}) := \begin{bmatrix} \mathbf{X}_2 \end{bmatrix} \sigma \left(\begin{bmatrix} \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \end{bmatrix} + \begin{bmatrix} \mu_1 \end{bmatrix} \right) + \begin{bmatrix} \mu_2 \end{bmatrix}$$

activation (over σ); weight (over $\mathbf{X}_1$); input (over $\mathbf{a}$); bias (over μ_1); bias (over μ_2)

hidden layer = learned features

...Into adversarial examples

Definition (Adversarial examples [49])

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

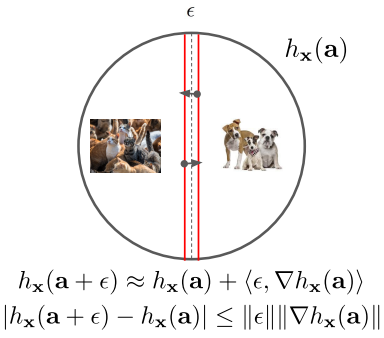
$$\boldsymbol{\eta} \in \underset{\boldsymbol{\eta}:\|\boldsymbol{\eta}\| \leq \epsilon}{\arg\max}\, L(h_{\mathbf{x}^\star}(\mathbf{a} + \boldsymbol{\eta}), \mathbf{b})$$

Example norms frequently used in adversarial attacks

- The most commonly used norm is the ℓ_∞-norm [17, 36].
- 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.

Challenge: Robustness to adversarial examples



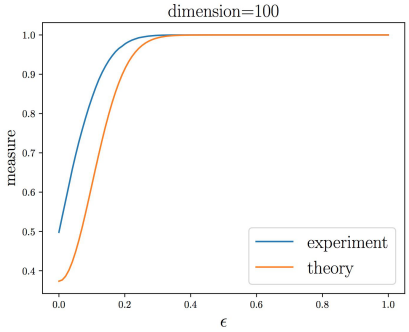


Figure: Understanding the robustness of a classifier in high-dimensional spaces. Shafahi et al. 2019.

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 $\boldsymbol{\eta} \in \mathbb{R}^n$ 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 [17, 36].

Fast Gradient Sign Method (FGSM) [17]

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

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

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,
  - $\operatorname{sign}\left(\nabla_{\mathbf{a}} h_{\mathbf{x}^\star}(\mathbf{a})\right)$ pattern is important

Results of FGSM on MNIST

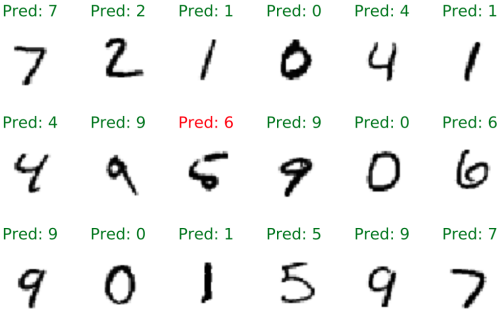


Figure: MNIST images with the predicted digit.

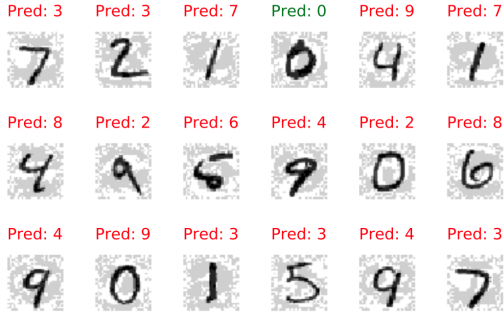


Figure: MNIST images perturbed by a FGSM attack.

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

Adversarial examples and proximal gradient ascent

- 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

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

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 ascent (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 [26].
- The PGA is incorrectly referred to as projected gradient descent in this literature.
- Practitioners prefer to use several steps of FGSM instead of PGA [28, 29, 36]:

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

A proposed link between FGSM and PGD

- 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

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

A proposed link between FGSM and PGD

- 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

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

Multiple steps of FGSM: A connection to majorization-minimization

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 [11]:

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

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 [28, 29, 36]
$$\boldsymbol{\eta}^{k+1} = \text{clip}\left(\boldsymbol{\eta}^k - t^\star \text{sign}(\nabla f(\boldsymbol{\eta}^k)), [-\epsilon, \epsilon]\right).$$
- This proof holds for **concave** and smooth functions, and need further quantification for our setting.

Towards adversarial training

Adversarial Training

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

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 $\mathbf{a}_i \in \mathbb{R}^p$ 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

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 $\mathbf{a}_i \in \mathbb{R}^p$ 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 $f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$. Let $\Phi(\mathbf{x}, \mathbf{y})$ is an extended real-valued closed proper convex function for each $\mathbf{y}$ in the compact set $\mathcal{Y}$; the interior of the domain of f is nonempty; $\Phi(\mathbf{x}, \mathbf{y})$ is jointly continuous on the relative interior of the domain of f and $\mathcal{Y}$.

Define $\mathcal{Y}^\star := \arg\max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ 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 $\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ is unique, 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 $\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ is not unique, then the subdifferential $\partial_{\mathbf{x}} f(\mathbf{x})$ of f is given by

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

Remarks:

- The adversarial problem is not convex in $\mathbf{x}$ in general.
- With proper initialization, overparameterization works argue that it is effectively convex.
- (Sub)Gradients of f_i are calculated as $\partial f_i(\mathbf{x}) = \nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\eta}^\star(\mathbf{x})), \mathbf{b}_i)$.

A corollary to Danskin's theorem

Adversarial Training

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 $\mathbf{a}_i \in \mathbb{R}^p$ 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 continuously differentiable due to ReLU, max-pooling, etc.

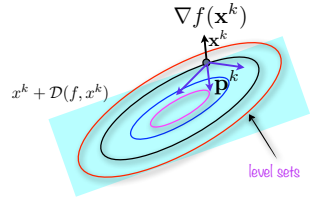


Figure: Descent directions in 2D should be an element of the cone of descent directions $\mathcal{D}(f, \cdot)$.

Descent directions [36]

Define $\mathcal{Y}^\star := \arg\max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$ as the set of maximizers, $\mathbf{y}^\star \in \mathcal{Y}^\star$, and $f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$. As long as $\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}^\star)$ is non-zero, it is a descent direction (and not a subgradient!) for $f(\mathbf{x})$.

Remarks:

- $\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\eta}^\star(\mathbf{x})), \mathbf{b}_i)$ is a descent direction for $f_i(\mathbf{x})$.
- We cannot find global maximizers $\mathcal{Y}^\star$.
- Only when $\mathbf{y}^\star$ is a singleton, $\nabla_{\mathbf{x}} L(h_{\mathbf{x}}(\mathbf{a}_i + \boldsymbol{\eta}^\star(\mathbf{x})), \mathbf{b}_i)$ is a (sub)gradient [4].

A practical implementation of adversarial training: Stochastic subgradient descent

Stochastic Adversarial Training [36]

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 (sub)gradient vector $\mathbf{g}^k := 0$

ii. select a mini-batch of data $B \subset \{1, \ldots, n\}$ 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 optimal (sub)gradient

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

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 $\boldsymbol{\eta}^\star$.
- (Sub)Gradient computation in step **iii.b** is motivated by Danskin's theorem.

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 [48]. Adversarial training helps!

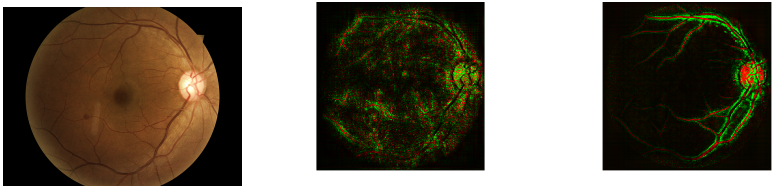
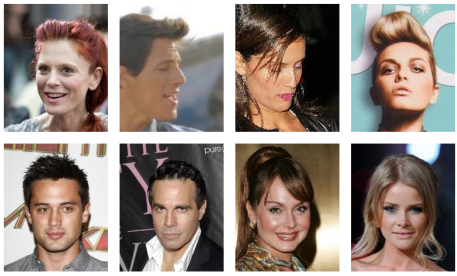


Table: **Left:** Ground truth image, **Middle:** Saliency map, **Right:** Saliency map with adversarial training.

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

- Recall the parametric density estimation setting

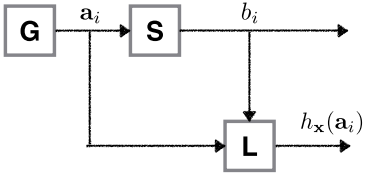


(source: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

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

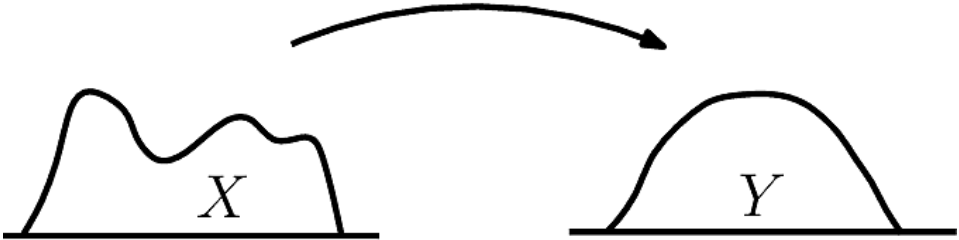


Figure: The Earth Mover's distance

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

$$\text{cost}(T) := \boldsymbol{E}_X \left\| X - T(X) \right\| . \tag{1}$$

The Wasserstein distance

Definition

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

$$\Gamma(\mu,\nu) := \{\pi \text{ probability measure on } \mathbb{R}^d \times \mathbb{R}^d \text{ with marginals } \mu,\nu\} \tag{2}$$

Definition (Primal form of the q-Wasserstein distance)

$$W_q(\mu,\nu) := \left(\inf_{\pi\in\Gamma(\mu,\nu)} \boldsymbol{E}_{(\mathbf{a},\mathbf{a}')\sim\pi} \left\|\mathbf{a}-\mathbf{a}'\right\|^q \right)^{1/q}, \tag{3}$$

where $q = 1, 2$.

Problem

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

$$\min_{\nu\in\Delta(\mathbb{R}^d)} W_q(\mu,\nu), \tag{4}$$

where $\Delta(\mathbb{R}^d)$ is the set of all probability measures on $\mathbb{R}^d$ and q is often selected as 1.

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, $h\#\mathsf{p}_\Omega$ 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,

$$\min_{\mathbf{x}} W_1(\mu^\natural, h_{\mathbf{x}} \# p_\Omega), \tag{5}$$

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

- Issues:
  - We only have access to empirical samples $\hat{\mu}_n$ of $\mu^\natural$.
  - W_1 is non-smooth.
  - $h_{\mathbf{x}}$ is non-convex.

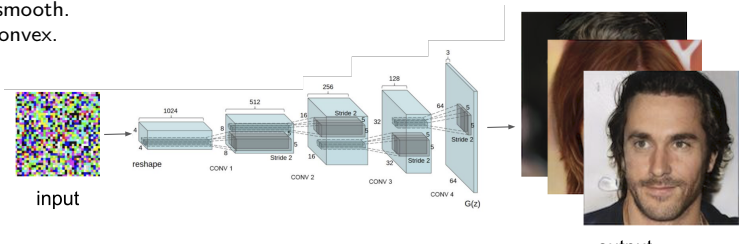
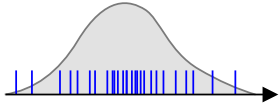


Figure: Schematic of a generative model, $h_{\mathbf{x}} \# \omega$ [16, 24].

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,

$$W_1(\mu^\natural, h_{\mathbf{x}}\#\mathsf{p}_\Omega) \leq W_1(\mu^\natural, \hat{\mu}_n) + W_1(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega), \tag{6}$$

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 [52, 10])

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,

$$W_1(\mu^\natural, \hat{\mu}_n) \gtrsim n^{-1/p}. \tag{7}$$

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

- How do we get a sub-gradient of $W_1\left(\hat{\mu}_n, h_{\mathbf{x}} \# \mathsf{p}_\Omega\right)$ with respect to $\mathbf{x}$?

Theorem (Kantorovich-Rubinstein duality)

$$W_1(\mu, \nu) = \sup_{\mathtt{d}} \{\langle \mathtt{d}, \mu \rangle - \langle \mathtt{d}, \nu \rangle : \mathtt{d} \textit{ is 1-Lipschitz}\} \tag{8}$$

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

Inner product is an expectation

$$\langle \mathtt{d}, \mu \rangle = \int \mathtt{d} \mathrm{d}\mu = \int \mathtt{d}(\mathbf{a}) \mathrm{d}\mu(\mathbf{a}) = \boldsymbol{E}_{\mathbf{a} \sim \mu} \left[\mathtt{d}(\mathbf{a})\right]. \tag{9}$$

Kantorovich-Rubinstein duality applied to our objective

$$W_1\left(\hat{\mu}_n, h_{\mathbf{x}} \# \omega\right) = \sup \left\{ \boldsymbol{E}_{\mathbf{a} \sim \hat{\mu}_n} [\mathtt{d}(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a} \sim h_{\mathbf{x}} \# \omega} [\mathtt{d}(\mathbf{a})] : \mathtt{d} \text{ is 1-Lipschitz} \right\} \tag{10}$$

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 [1]

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

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathsf{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\mathsf{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right) \tag{11}$$

Obtaining a stochastic sub-gradient with respect to $\mathbf{x}$

- Recall Danskin's theorem
- For fixed $\mathbf{x}$, we obtain an optimal solution $\mathbf{y}^\star$ for the inner problem, e.g., with gradient ascent.
- Then, we can use the (sub)gradient for $\mathbf{x}$ at $(\mathbf{x}, \mathbf{y}^\star)$ in the outer problem.

General diagram of GANs

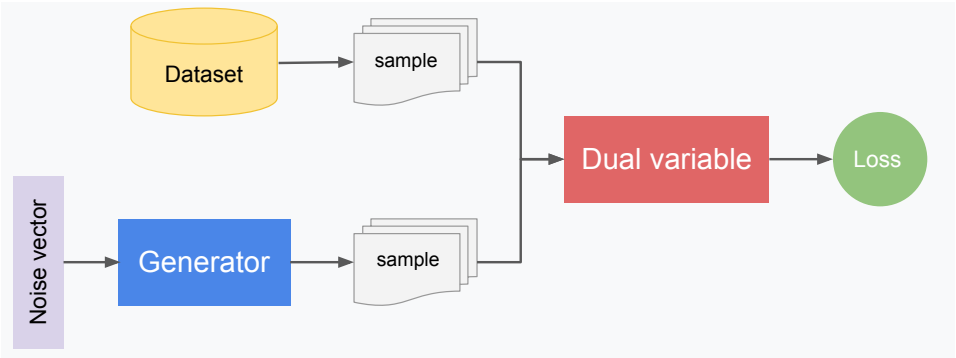


Figure: Generator/dual variable/dataset relation in GANs

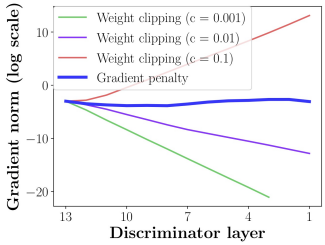
The theory-practice gap: Enforcing 1-Lipschitz of the discriminator

Weight clipping [1]

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

$$\pi_{\mathcal{B}}(\mathbf{y}) = \mathrm{clip}(\mathbf{y}, [-c, c]). \tag{12}$$

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 [18]

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

$$\boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}\left[\mathrm{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\Omega}\left[\mathrm{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] + \lambda \boldsymbol{E}_{\mathbf{a}\sim\nu}\left[\left(\|\nabla_{\mathbf{a}} \mathrm{d}_{\mathbf{y}}(\mathbf{a})\|_* - 1\right)^2\right]. \tag{13}$$

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 [38].

Practical implementation of GANs

Stochastic training of Wasserstein GANs

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

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

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

Some historical background for a Turing award

Vanilla GAN [16]

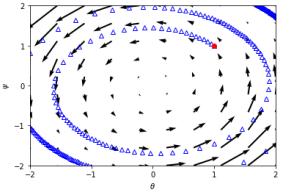
$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\log \mathtt{d}_{\mathbf{y}}(\mathbf{a})\right] + \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathtt{p}_\Omega} \left[\log\left(1-\mathtt{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right)\right] \tag{14}$$

- 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 $\mu^\natural$.

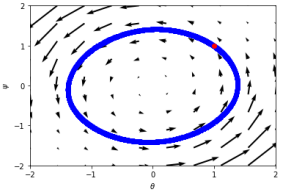
Observation: ○ Minimizes Jensen-Shannon divergence:

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

Difficulties of GAN training



(a) SimGD



(b) AltGD

Figure: Mode collapse (left). Simultaneous vs alternating generator/discriminator updates (right).

- Heuristics galore!
- Difficult to enforce 1-Lipschitz constraint
- Overall a difficult minimax problem: Scalability, mode collapse, periodic cycling...
- Privacy concerns due to memorization

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{15}$$

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?

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{15}$$

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."

Solution concept

- Like for nonconvex problems in minimization we try to find a *local* solution.

Definition (Local Nash Equilibrium)

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

$$\Phi(\mathbf{x}^\star, \mathbf{y}) \leq \Phi(\mathbf{x}^\star, \mathbf{y}^\star) \leq \Phi(\mathbf{x}, \mathbf{y}^\star) \tag{LNE}$$

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

Necessary conditions

Through a Taylor expansion around $\mathbf{x}^\star$ and $\mathbf{y}^\star$ one can show that a LNE implies,

$$\nabla_{\mathbf{x}} \Phi(\mathbf{x}, \mathbf{y}), -\nabla_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) = 0$$
$$\nabla_{\mathbf{xx}} \Phi(\mathbf{x}, \mathbf{y}), -\nabla_{\mathbf{yy}} \Phi(\mathbf{x}, \mathbf{y}) \succeq 0$$

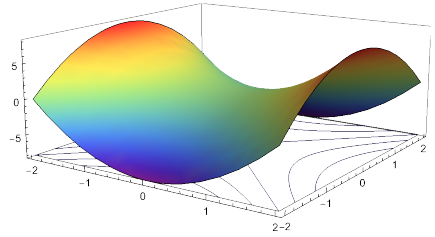


Figure: $\Phi(x, y) = x^2 - y^2$

Recall SGD results

$$\min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- For a non-convex, smooth f, we have that

1. SGD converges to the critical points of f as $N \to \infty$.
2. SGD avoids strict saddles/traps ($\lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) < 0$) almost surely.
3. SGD remains close to Hurwicz minimizers (i.e., $\mathbf{x}^* : \lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) > 0$ almost surely.

Recall SGD results

$$\min_{\mathbf{x}:\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- For a non-convex, smooth f, we have that
  1. SGD converges to the critical points of f as $N \to \infty$.
  2. SGD avoids strict saddles/traps ($\lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) < 0$) almost surely.
  3. SGD remains close to Hurwicz minimizers (i.e., $\mathbf{x}^* : \lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) > 0$ almost surely.

- Nail in the coffin:
  - not even sure if we obtain stochastic descent directions by approximately solving inner problems in GANs.
  - GANs are fundamentally different from adversarial training!
- Need more direct approaches with the stochastic gradient estimates.

Generalized Robbins-Monro schemes

- 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: [21]

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.

Basic algorithms for minimax

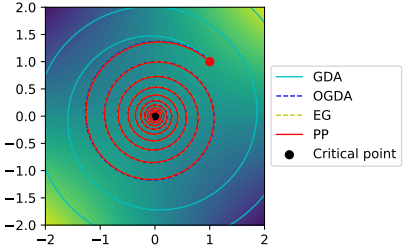


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

- (In)Famous algorithms
  - Gradient Descent Ascent (GDA)
  - Proximal point method (PPM)
  - Extra-gradient (EG)
  - Optimistic Gradient Descent Ascent (OGDA)
  - Reflected-Forward-Backward-Splitting (RFBS)

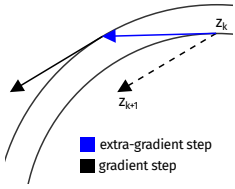
- EG and OGDA are approximations of the PPM
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha \mathbf{V}(\mathbf{z}^k)$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha \mathbf{V}(\mathbf{z}^{k+1})$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha \mathbf{V}(\mathbf{z}^k - \alpha \mathbf{V}(\mathbf{z}^{k-1}))$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

Extra-gradient algorithm (EG)[1]

EG method for saddle point problems

1. Choose $\mathbf{x}^0, \mathbf{y}^0$ and α.
2. For $k = 0, 1, \cdots$, perform:

$$\tilde{\mathbf{x}}^k := \mathbf{x}^k - \alpha \nabla_{\mathbf{x}} \Phi(\mathbf{x}^k, \mathbf{y}^k),$$
$$\tilde{\mathbf{y}}^k := \mathbf{y}^k + \alpha \nabla_{\mathbf{y}} \Phi(\mathbf{x}^k, \mathbf{y}^k).$$
$$\mathbf{x}^{k+1} := \mathbf{x}^k - \alpha \nabla_{\mathbf{x}} \Phi(\tilde{\mathbf{x}}^k, \tilde{\mathbf{y}}^k).$$
$$\mathbf{y}^{k+1} := \mathbf{y}^k + \alpha \nabla_{\mathbf{y}} \Phi(\tilde{\mathbf{x}}^k, \tilde{\mathbf{y}}^k).$$



- Idea: Predict the gradient at the next point

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha \mathbf{V}\Big(\underbrace{\mathbf{z}^k - \alpha \mathbf{V}(\mathbf{z}^k)}_{\text{prediction of } \mathbf{z}^{k+1}} \Big) \qquad \text{(EG)}$$

Remark: - 1-extra-gradient computation per iteration

[1] G.M. Korpelevic, "An extragradient method for finding saddle points and for other problems," Ekonomika i Matematischeskie Metody, vol. 12, no. 4, pp. 747-756, 1976.

ExtraAdam

ExtraAdam for GANs [15]

Input. Step size γ, exponential decay rates $\eta_1, \eta_2 \in [0, 1)$

1. Set $\mathbf{m}_0, \mathbf{v}_0 = 0$
2. For $k = 0, 1, \ldots,$ iterate

$$
\begin{cases}
\mathbf{g}_k & = V(\mathbf{z}^k, \zeta^k) \\
\mathbf{m}_{k-1/2} & = \eta_1 \mathbf{m}_{k-1} + (1-\eta_1)\mathbf{g}_k \leftarrow \text{1st order estimate} \\
\mathbf{v}_{k-1/2} & = \eta_2 \mathbf{v}_{k-1} + (1-\eta_2)\mathbf{g}_k^2 \leftarrow \text{2nd order estimate} \\
\hat{\mathbf{m}}_{k-1/2} & = \mathbf{m}_{k-1/2}/(1-\eta_1^k) \leftarrow \text{Bias correction} \\
\hat{\mathbf{v}}_{k-1/2} & = \mathbf{v}_{k-1/2}/(1-\eta_2^k) \leftarrow \text{Bias correction} \\
\mathbf{z}^{k+1/2} & = \mathbf{z}^k - \gamma \hat{\mathbf{m}}_{k-1/2}/(\sqrt{\hat{\mathbf{v}}_{k-1/2}} + \epsilon) \leftarrow \text{Extrapolation step} \\
\mathbf{g}_{k+1/2} & = V(\mathbf{z}^{k+1/2}, \zeta^{k+1/2}) \\
\mathbf{m}_k & = \eta_1 \mathbf{m}_{k-1/2} + (1-\eta_1)\mathbf{g}_{k+1/2} \leftarrow \text{1st order estimate} \\
\mathbf{v}_k & = \eta_2 \mathbf{v}_{k-1/2} + (1-\eta_2)\mathbf{g}_{k+1/2}^2 \leftarrow \text{2nd order estimate} \\
\hat{\mathbf{m}}_k & = \mathbf{m}_k/(1-\eta_1^k) \leftarrow \text{Bias correction} \\
\hat{\mathbf{v}}_k & = \mathbf{v}_k/(1-\eta_2^k) \leftarrow \text{Bias correction} \\
\mathbf{z}^{k+1} & = \mathbf{z}^k - \gamma \hat{\mathbf{m}}_k/(\sqrt{\hat{\mathbf{v}}_k} + \epsilon) \leftarrow \text{Update step}
\end{cases}
$$

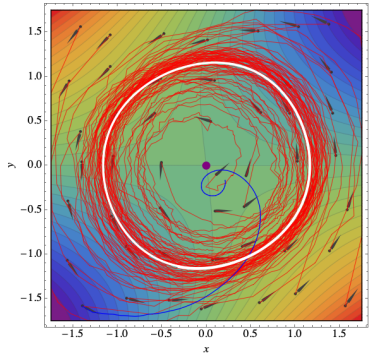
Output : $\mathbf{z}^k$

Minimax is more difficult than just optimization [21]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [3].
  - For optimization, {attracting ICT} $\equiv$ {solutions}
  - For minimax, {attracting ICT} $\equiv$ {solutions} $\cup$ {spurious sets}

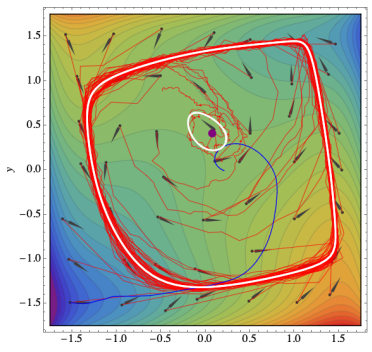
- "Almost" bilinear $\neq$ bilinear:

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



- The "forsaken" solutions:

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

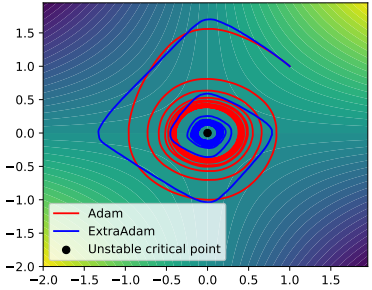


Minimax is more difficult than just optimization [21]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [3].
  - For optimization, {attracting ICT} $\equiv$ {solutions}
  - For minimax, {attracting ICT} $\equiv$ {solutions} $\cup$ {spurious sets}

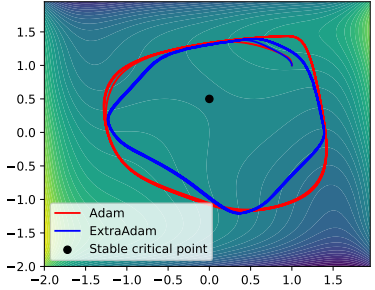
- "Almost" bilinear $\neq$ bilinear:

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



- The "forsaken" solutions:

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



Comparison of convergence rates for smooth convex-concave minimax

Method	Assumption on $\Phi(\cdot,\cdot)$	Convergence rate	Reference	Note
PP	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[44]	
PP	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[44]	Implicit algorithm
PP	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[44]	
EG	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[39]	
EG	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[40, 39]	1 extra-gradient evaluation per iteration
EG	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[40, 39]	
OGDA	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[39]	
OGDA	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[40, 39]	no obvious downside
OGDA	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[40, 39]	

An abridged collection of algorithms for convex-concave minimax optimization

A restricted minimax formulation

Let us consider

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{16}$$

where $\Phi(\mathbf{x},\mathbf{y})$ is convex in $\mathbf{x}$ and concave in $\mathbf{y}$.

1. $\mathcal{X}\subset\mathbb{R}^p$ and $\mathcal{Y}\subset\mathbb{R}^n$; and $\Phi(\mathbf{x},\mathbf{y})$ is smooth, or bilinear, or strongly convex/strongly concave
   - Algorithms: Proximal-Point [44], Extra-gradient [27, 40, 39], OGDA [40, 39]
2. $\mathcal{X}\subset\mathbb{R}^p$ and $\mathcal{Y}\subset\mathbb{R}^n$ with tractable "mirror maps"; and $\Phi(\mathbf{x},\mathbf{y})$ is smooth and continuously differentiable
   - Algorithm: Mirror-Prox [41]
3. $\mathcal{X}=\mathbb{R}^p$ and $\mathcal{Y}=\mathbb{R}^n$; and $\Phi(\mathbf{x},\mathbf{y})=h(\mathbf{x})+f(\mathbf{x})+\langle\mathbf{A}\mathbf{x},\mathbf{y}\rangle-g^*(\mathbf{y})$
   - Algorithms: Chambolle-Pock [5], Condat-Vu [6, 51], PD3O [55]

Between convex-concave and nonconvex-nonconcave

Nonconvex-concave problems

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

- $\Phi(\mathbf{x},\mathbf{y})$ is nonconvex in $\mathbf{x}$, concave in $\mathbf{y}$, smooth in $\mathbf{x}$ and $\mathbf{y}$.

Recall

Define $f(\mathbf{x}) = \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y})$.

- Gradient descent applied to nonconvex f requires $\mathcal{O}(\epsilon^{-2})$ iterations to give an ϵ-stationary point.
- (Sub)gradient of f can be computed using Danskin's theorem:

$$\nabla_{\mathbf{x}}\Phi(\cdot, y^\star(\cdot)) \in \partial f(\cdot), \text{ where } y^\star(\cdot) \in \arg\max_{\mathbf{y}\in\mathcal{Y}} \Phi(\cdot,\mathbf{y}),$$

which is tractable since Φ is concave in $\mathbf{y}$ [33].

Remark: ○ "Conceptually" much easier than nonconvex-nonconcave case.

A summary of results for nonconvex-concave setting

○ A summary of gradient complexities to reach $\epsilon-$first order stationary point in terms of **gradient mapping.**

Method	Assumption on $\Phi(\cdot,\cdot)$	Convergence rate	Reference
GDA	noconvex-concave	$\tilde{\mathcal{O}}\left(\epsilon^{-6}\right)$	[33]
GDA	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[33]
GDmax	nonconvex-concave	$\tilde{\mathcal{O}}\left(\epsilon^{-6}\right)$	[22]
GDmax	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[22]
HiBSA, AGP	nonconvex-concave	$\tilde{\mathcal{O}}\left(\epsilon^{-4}\right)$	[35], [54]
HiBSA, AGP	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[35], [54]

The elephant in the room: Nonsmooth, nonconvex optimization

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

- Finding a stationary point of nonsmooth nonconvex minimization problems are hard [57]
  - A traditional $\epsilon-$stationarity can not be obtained in finite time
- Even the relax notions are hard [46]
- Really puzzling how deep learning approaches with ReLu etc. work...

Detour: Weak convexity (WeCo) & approximate stationarity[1]

- Smooth: Gradient mapping norm
  - $\|G_\alpha(\mathbf{x}^k)\|^2 = \frac{1}{\alpha^2}\|x^k - \pi_{\mathcal{X}}(\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k))\|^2$
  - possible to compute
- Non-smooth: Generalized subdifferential distance
  - $\text{dist}(0, \partial(f(\mathbf{x}^k) + \delta_{\mathcal{X}}(\mathbf{x}^k)))^2$
  - hard in general (even approximately)[2][3]

- f is ρ-weakly convex if $f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x}\|^2$ is convex.

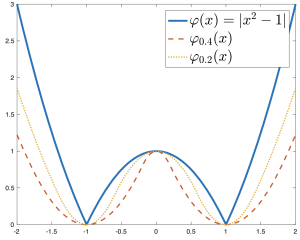


Figure: ME with $f(x) = |x^2 - 1|$, $\mathcal{X} = \mathbb{R}$, and $\hat{v}_t = \mathbb{I}$.[1]

- Moreau envelope (ME):

$$\varphi_{1/\rho}(\mathbf{x}) = \min_{y\in\mathcal{X}} \left\{ f(\mathbf{y}) + \frac{\rho}{2}\|\mathbf{y} - \mathbf{x}\|^2 \right\}$$

$$\hat{\mathbf{x}} \leftarrow \arg\min$$

$$\nabla\varphi_{1/\rho}(x) = \rho(\mathbf{x} - \hat{\mathbf{x}})$$

- Small $\|\nabla\phi_{1/\rho}(\mathbf{x})\|$ implies near-stationarity:[1]

$$\text{dist}(0, \partial(f(\mathbf{x}^k) + \delta_{\mathcal{X}}(\mathbf{x}^k)))^2 \leq \|\nabla\phi_{1/\rho}(\mathbf{x}^k)\|^2$$

  - also implies small $\|G_\alpha(\mathbf{x}^k)\|^2$ if f is smooth

[1] D. Davis and D. Drusvyatskiy, "Stochastic model-based minimization of weakly convex functions," SIOPT, 2019.

[3] J. Zhang, et al., "On complexity of finding stationary points of nonsmooth nonconvex functions," arXiv:2002.04130, 2020.

[3] O. Shamir, "Can We Find Near-Approximately-Stationary Points of Nonsmooth Nonconvex Functions?" arXiv:2002.11962, 2020.

Lifting minimax optimization: From pure to mixed Nash equilibrium

- Rethinking minimax problem as pure strategy game formulation

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

- A corresponding **mixed** strategy formulation

$$\min_{p\in\mathcal{M}(\mathcal{X})} \max_{q\in\mathcal{M}(\mathcal{Y})} \mathbb{E}_{\mathbf{x}\sim p}\mathbb{E}_{\mathbf{y}\sim q}\left[\Phi(\mathbf{x},\mathbf{y})\right]$$

  - $\mathcal{M}(\mathcal{Z}) \coloneqq \{\text{all randomized strategies on } \mathcal{Z}\}$
  - In the sequel, a way to solve this infinite dimensional problem: Mirror descent + Langevin dynamics [20]

From pure to mixed Nash equilibrium

- Key ingredients
  - $\langle p, h \rangle \coloneqq \int h \, \mathrm{d}p$ for a measure p and function h (Riesz representation)
  - the linear operator G and its adjoint $G^\dagger$:

$$\begin{aligned} (Gq)(\mathbf{x}) &\coloneqq \mathbb{E}_{\mathbf{y}\sim q}\left[\Phi(\mathbf{x}, \mathbf{y})\right] \\ (G^\dagger p)(\mathbf{y}) &\coloneqq \mathbb{E}_{\mathbf{x}\sim p}\left[\Phi(\mathbf{x}, \mathbf{y})\right], \end{aligned}$$

- Mixed NE formulation $\simeq$ finite two-player games

$$\min_{p\in\mathcal{M}(\mathcal{X})} \max_{q\in\mathcal{M}(\mathcal{Y})} \mathbb{E}_{\mathbf{x}\sim p}\mathbb{E}_{\mathbf{y}\sim q}\left[\Phi(\mathbf{x}, \mathbf{y})\right]$$

$$\Updownarrow$$

$$\min_{p\in\mathcal{M}(\mathcal{X})} \max_{q\in\mathcal{M}(\mathcal{Y})} \langle p, Gq \rangle$$

  - If $\mathcal{X}$ and $\mathcal{Y}$ are finite $\Rightarrow$ mirror descent
  - Caveat: **infinite dimension!!!**

Mirror descent in infinite dimension

Theorem (Informal, Main result in [20])

For a learning rate η, a probability measure p, and an arbitrary function f, define

$$p_+ = \mathsf{MD}_\eta(p, f) \quad \equiv \quad \mathrm{d}p_+ = \frac{e^{-\eta f}\,\mathrm{d}p}{\int e^{-\eta f}\,\mathrm{d}p}.$$

Then (inf-dimensional) entropic mirror descent solves the mixed Nash equilibrium.

- **Issue:** Cannot update probability measures in practice...
  - But maybe we can sample by Langevin dynamics!

Langevin dynamics for inf-dimensional mirror descent

- Pick p_1, q_1 (say Gaussians), need to draw samples from:

$$\mathrm{d}p_2(\mathbf{x}) \propto e^{-\eta f(\mathbf{x})}\mathrm{d}p_1, \quad f(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim q_1}[\Phi(\mathbf{x},\mathbf{y})]$$

  - LD solves the problem with (stochastic) gradients of f:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma\tilde{\nabla} f(\mathbf{x}_k) + \sqrt{2\gamma}\mathcal{N}(0, I)$$

  - Query $\tilde{\nabla} f$ by drawing n samples $\{\mathbf{y}_i\}_{i=1}^n$ from q_1 and estimate

$$\tilde{\nabla} f(\mathbf{x}) = \nabla\left(\hat{\mathbb{E}}_{\mathbf{y}\sim q_1}[\Phi(\mathbf{x},\mathbf{y})]\right) \equiv \frac{1}{n}\sum_{i=1}^n \nabla\Phi(\mathbf{x},\mathbf{y}_i)$$

- Therefore,

$$\text{samples from } (p_1, q_1) \overset{\text{LD}}{\Rightarrow} \text{samples from } (p_2, q_2) \overset{\text{LD}}{\Rightarrow} \text{samples from } (p_3, q_3) \overset{\text{LD}}{\Rightarrow} \cdots$$

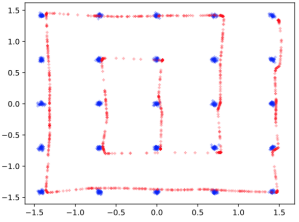
Langevin dynamics for inf-dimensional mirror descent

- With additional simplifications, we can write

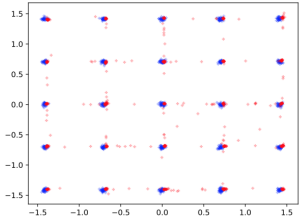
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \tilde{\nabla}_{\mathbf{x}} \Phi(\mathbf{x}_k, \mathbf{y}_k) + \sqrt{2\gamma_k}\mathcal{N}(0, I)$$
$$\mathbf{y}_{k+1} = \mathbf{y}_k + \gamma_k \tilde{\nabla}_{\mathbf{y}} \Phi(\mathbf{x}_k, \mathbf{y}_k) + \sqrt{2\gamma_k}\mathcal{N}(0, I) \qquad \text{(SGLD)}$$

- Per-iteration cost $\simeq$ SGDA

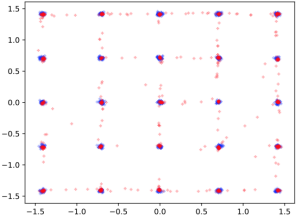
Application to 25 Gaussians: Algorithms matter [20]



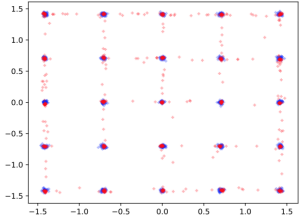
(a) SGD



(b) Adam

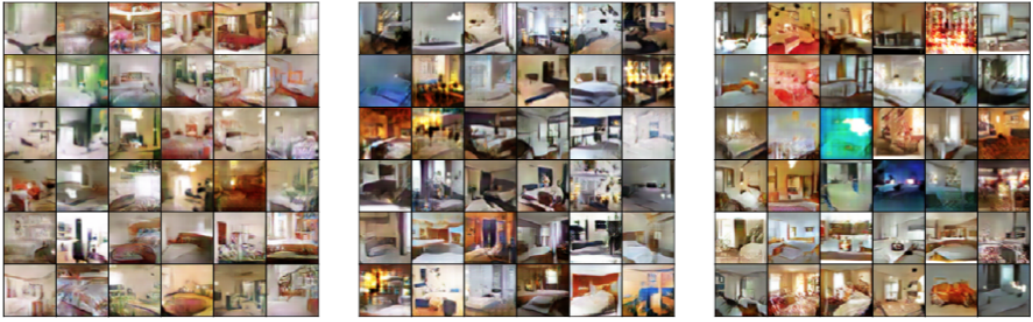


(c) Mirror-GAN



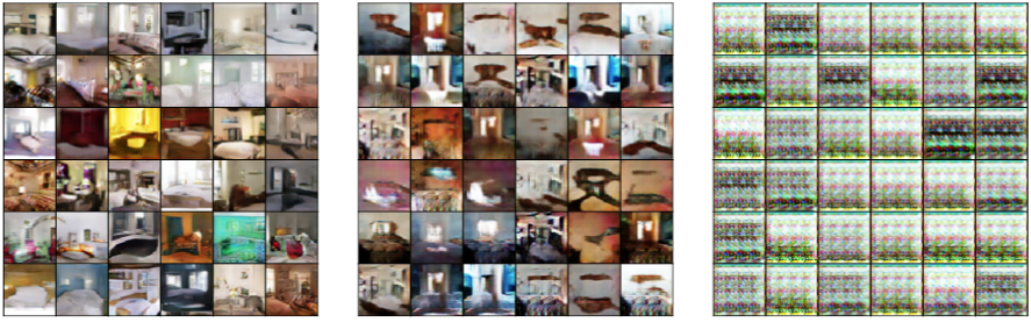
(d) Mirror-Prox-GAN

Real LSUN Dataset: RMSProp, $4 \times 10^4, 8 \times 10^4, \times 10^5$ iterations [20]



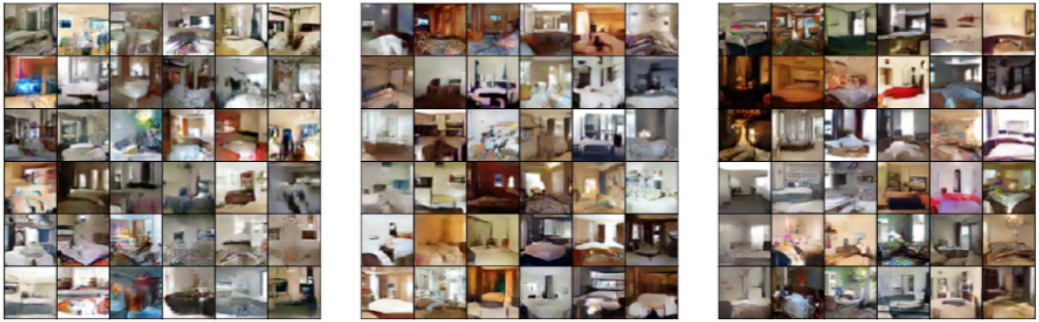
(a) RMSProp

Real LSUN Dataset: Adam, $4 \times 10^4, 8 \times 10^4, \times 10^5$ iterations [20]



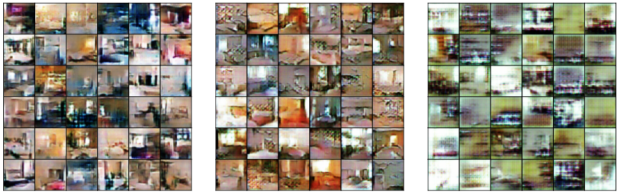
(b) Adam

Real LSUN Dataset: Mirror-GAN, $4 \times 10^4, 8 \times 10^4, \times 10^5$ iterations [20]



(c) Mirror-GAN, **Algorithm 3**

Real LSUN Dataset: Extra-Adam, $4 \times 10^4, 8 \times 10^4, \times 10^5$ iterations [20]



(d) Simultaneous Extra-Adam

(e) Alternated Extra-Adam

Application: Noisy action robust reinforcement learning[1]

- Train RL agent in the presence of an adversary
- Adversary budget: $\alpha \in [0, 0.5]$

Noisy action robust MDP game

for $t = 1, 2, \ldots$ do:

both players observe state $S_t \in \mathcal{S}$

both players choose actions $A_t = \mu(S_t) \in \mathcal{A}$, and $A'_t = \nu(S_t) \in \mathcal{A}$

execute the noisy action $\bar{A}_t = (1 - \alpha)A_t + \alpha A'_t$

agent gets reward $R_{t+1} = R(S_t, \bar{A}_t)$, adversary gets $-R_{t+1}$

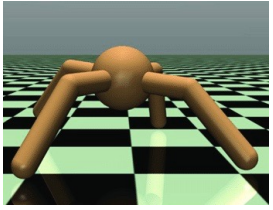
both players enter new state S_{t+1}

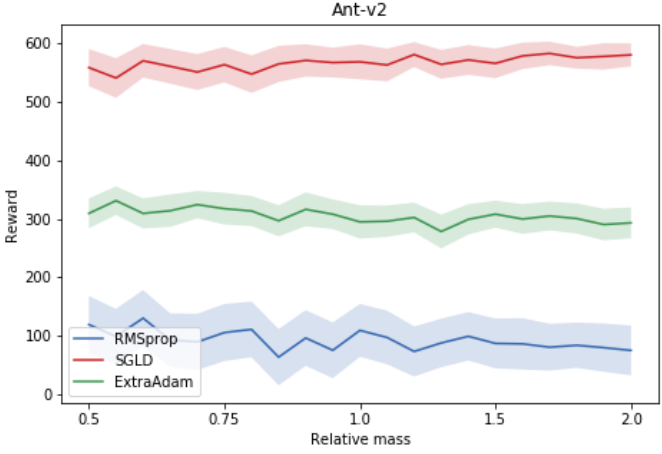
- Hope: Train in one environment, generalize to others

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

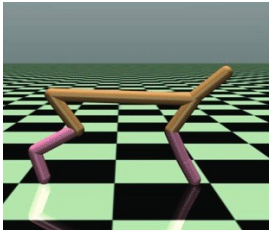
Experimental evaluation via MuJoCo

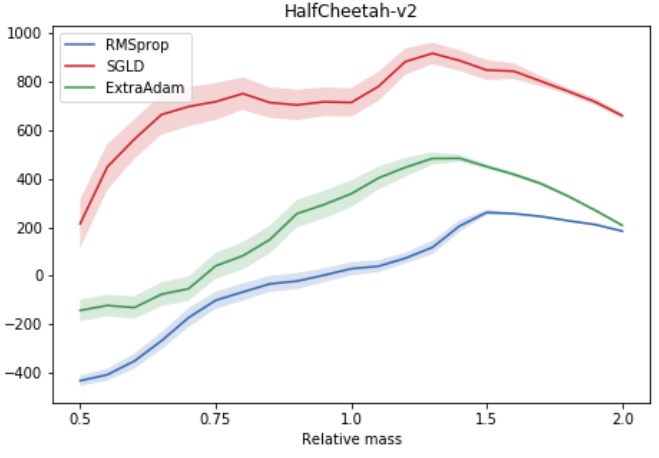
- Standard MuJoCo datasets





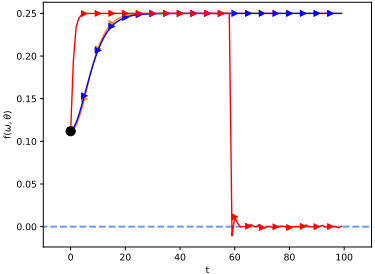
Experimental evaluation via MuJoCo

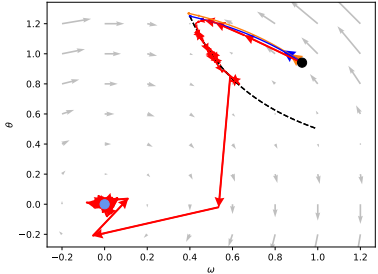


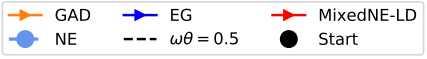


Simple Nash Equilibrium Test

$$\max_{\omega\in[-2,2]} \min_{\theta\in[-2,2]} -\omega^2\theta^2 + \omega\theta$$







Thank you for your attention!

- Minimax is more difficult than just optimization!
- Universal adaptation is an open research topic
- Interpretability, robustness, and adversarial training are all entangled...

References I

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein gan.
arXiv preprint arXiv:1701.07875, 2017.

[2] Michel Benaïm.
Dynamics of stochastic approximation algorithms.
In Jacques Azéma, Michel Émery, Michel Ledoux, and Marc Yor, editors, *Séminaire de Probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pages 1–68. Springer Berlin Heidelberg, 1999.

[3] Michel Benaïm and Morris W. Hirsch.
Journal of Dynamics and Differential Equations.

[4] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo.
Robust nonconvex optimization for simulation-based problems.
Operations Research, 2007.

[5] A. Chambolle and T. Pock.
A first-order primal-dual algorithm for convex problems with applications to imaging.
Journal of Mathematical Imaging and Vision, 40(1):120–145, 2011.

[6] L. Condat.
A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms.
J. Optim. Theory Appl., 158:460–479, 2013.

References II

[7] George Cybenko.
Approximation by superpositions of a sigmoidal function.
Mathematics of control, signals and systems, 2(4):303–314, 1989.

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

[9] John Duchi, Elad Hazan, and Yoram Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

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

[11] Marwa El Halabi.
Learning with structured sparsity: From discrete to convex and back.
Technical report, EPFL, 2018.

[12] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points—online stochastic gradient for tensor decomposition.
In *Conference on Learning Theory*, pages 797–842, 2015.

References III

[13] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points — Online stochastic gradient for tensor decomposition.
In *COLT '15: Proceedings of the 28th Annual Conference on Learning Theory*, 2015.

[14] Saeed Ghadimi and Guanghui Lan.
Stochastic first- and zeroth-order methods for nonconvex stochastic programming.
SIAM Journal on Optimization, 23(4):2341–2368, 2013.

[15] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien.
A variational inequality perspective on generative adversarial networks.
In *International Conference on Learning Representations*, 2018.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.
Generative adversarial nets.
In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572, 2014.

[18] 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.

References IV

[19] Benjamin D Haeffele and René Vidal.
Global optimality in neural network training.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017.

[20] Ya-Ping Hsieh, Chen Liu, and Volkan Cevher.
Finding mixed Nash equilibria of generative adversarial networks.
volume 97 of *Proceedings of Machine Learning Research*, pages 2810–2819, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[21] 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.

[22] Chi Jin, Praneeth Netrapalli, and Michael I Jordan.
What is local optimality in nonconvex-nonconcave minimax optimization?
arXiv preprint arXiv:1902.00618, 2019.

[23] Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, and Volkan Cevher.
Robust reinforcement learning via adversarial training with langevin dynamics.
arXiv preprint arXiv:2002.06063, 2020.

[24] 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.

References V

[25] Diederik Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

[26] Ziko Kolter and Aleksander Madry.
Adversarial robustness - theory and practice.
NeurIPS 2018 tutorial: `https://adversarial-ml-tutorial.org/`.

[27] G. M. Korpelevic.
An extragradient method for finding saddle-points and for other problems.
Ėkonom. i Mat. Metody., 12(4):747–756, 1976.

[28] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial examples in the physical world.
arXiv preprint arXiv:1607.02533, 2016.

[29] Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial machine learning at scale.
arXiv preprint arXiv:1611.01236, 2016.

[30] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht.
First-order methods almost always avoid strict saddle points.
Mathematical Programming, 176(1):311–337, February 2019.

References VI

[31] Kfir Levy.
Online to offline conversions, universality and adaptive minibatch sizes.
In *Advances in Neural Information Processing Systems*, pages 1613–1622, 2017.

[32] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
arXiv preprint arXiv:1509.02971, 2015.

[33] Tianyi Lin, Chi Jin, and Michael I Jordan.
On gradient descent ascent for nonconvex-concave minimax problems.
arXiv preprint arXiv:1906.00331, 2019.

[34] Lennart Ljung.
Analysis of recursive stochastic algorithms.
22(4):551–575, August 1977.

[35] Songtao Lu, Ioannis Tsaknakis, Mingyi Hong, and Yongxin Chen.
Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications.
IEEE Transactions on Signal Processing, 2020.

[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu.
Towards deep learning models resistant to adversarial attacks.
In *International Conference on Learning Representations*, 2018.

References VII

[37] Panayotis Mertikopoulos, Nadav Hallak, Ali Kavis, and Volkan Cevher.
On the almost sure convergence of stochastic gradient descent in non-convex problems, 2020.

[38] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
arXiv preprint arXiv:1802.05957, 2018.

[39] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil.
Convergence rate of $\mathcal{O}(1/k)$ for optimistic gradient and extra-gradient methods in smooth convex-concave saddle point problems.
arXiv preprint arXiv:1906.01115, 2019.

[40] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil.
A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach.
In *International Conference on Artificial Intelligence and Statistics*, pages 1497–1507. PMLR, 2020.

[41] A. Nemirovskii.
Prox-method with rate of convergence $\mathcal{O}(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems.
SIAM J. Op, 15(1):229–251, 2004.

[42] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz.
Parameter space noise for exploration.
arXiv preprint arXiv:1706.01905, 2017.

References VIII

[43] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
arXiv preprint arXiv:1904.09237, 2019.

[44] R. T. Rockafellar.
Augmented lagrangians and applications of the proximal point algorithm in convex programming.
Mathemathics of Operations Research, 1:97–116, 1976.

[45] Gregory Roth and W. Sandholm.
Stochastic approximations with constant step size and differential inclusions.
SIAM J. Control. Optim., 51:525–555, 2013.

[46] Ohad Shamir.
Can we find near-approximately-stationary points of nonsmooth nonconvex functions?
arXiv preprint arXiv:2002.11962, 2020.

[47] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller.
Deterministic policy gradient algorithms.
In *ICML*, 2014.

[48] Mukund Sundararajan, Ankur Taly, and Qiqi Yan.
Axiomatic attribution for deep networks.
arXiv preprint arXiv:1703.01365, 2017.

References IX

[49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.
Intriguing properties of neural networks.
arXiv preprint arXiv:1312.6199, 2013.

[50] Tijmen Tieleman and Geoffrey Hinton.
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, 4(2):26–31, 2012.

[51] Bang Cong Vu.
A splitting algorithm for dual monotone inclusions involving cocoercive operators.
Advances in Computational Mathematics, 38(3):667–681, 2013.

[52] 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.

[53] Max Welling and Yee W Teh.
Bayesian learning via stochastic gradient langevin dynamics.
In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[54] Zi Xu, Huiling Zhang, Yang Xu, and Guanghui Lan.
A unified single-loop alternating gradient projection algorithm for nonconvex-concave and convex-nonconcave minimax problems.
arXiv preprint arXiv:2006.02032, 2020.

References X

[55] Ming Yan.

A new primal–dual algorithm for minimizing the sum of three functions with a linear operator.

Journal of Scientific Computing, 76(3):1698–1717, 2018.

[56] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar.

Adaptive methods for nonconvex optimization.

In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9793–9803. Curran Associates, Inc., 2018.

[57] Jingzhao Zhang, Hongzhou Lin, Suvrit Sra, and Ali Jadbabaie.

On complexity of finding stationary points of nonsmooth nonconvex functions.

arXiv preprint arXiv:2002.04130, 2020.