Optimization Challenges in Adversarial Machine Learning

EPFL CIS - RIKEN AIP Joint Seminar Series

Volkan Cevher

https://lions.epfl.ch
@CevherLIONS

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

Acknowledgements

- LIONS group members (current & alumni): https://lions.epfl.ch
  - EE-556 (Mathematics of Data): Course material
- Optimization for Machine Learning (OPT-ML) 2020 Workshop at NeurIPS: https://opt-ml.org/
- Eusipco tutorial with Ahmet Alacaoglu, Ali Kavis, and Kfir Levy on adaptive methods
- DS3 Summer School Lectures

Today: “Basic” adversarial machine learning in 45mins

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

- A seemingly simple optimization formulation
- Critical in machine learning with many applications
  - Adversarial examples and training
  - Generative adversarial networks
  - Robust reinforcement learning

Warm up: Flexibility of the template

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

Warm up: Flexibility of the template

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

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

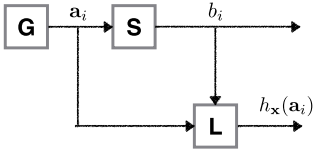
Warm up: Flexibility of the template

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

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

- (eula) In the sequel,
  - the set $\mathcal{X}$ is convex 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

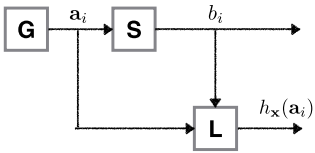
$$\mathbf{x}_{\mathsf{DL}}^{\star} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints 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



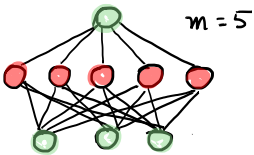
Definition (Optimization formulation)

The deep-learning training problem is given by

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

where $\mathcal{X}$ denotes the constraints 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 ↓ σ | weight ↓ $\mathbf{X}_1$ | input ↓ $\mathbf{a}$ | bias ↓ μ_1 | bias ↓ μ_2

hidden layer = learned features

An approximation theoretic motivation: Why neural networks?

Theorem (Universal approximation [4])

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!

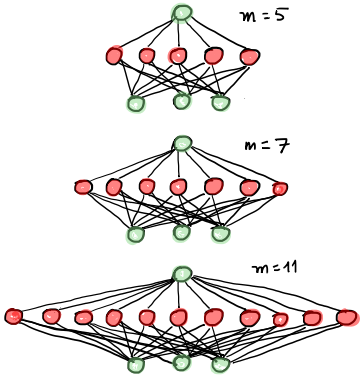
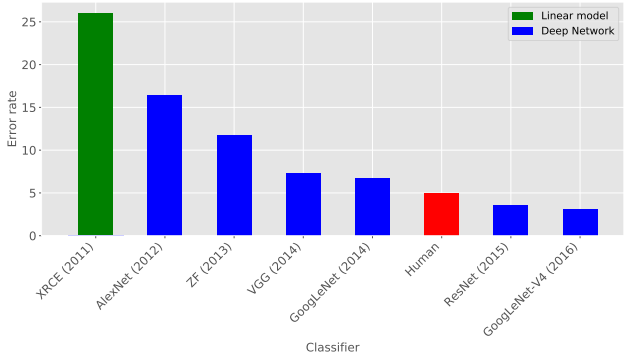


Figure: Neural networks of increasing width

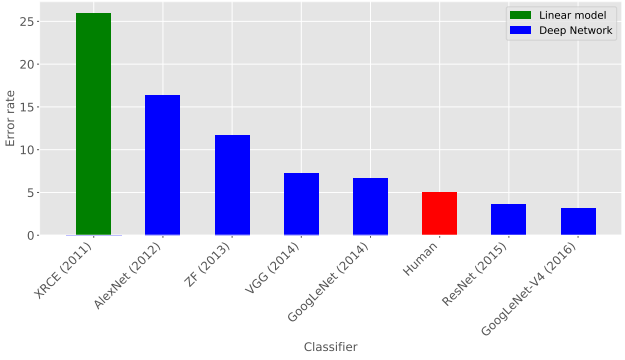
A more realistic motivation: Why neural networks?

- Practical performance in applications



A more realistic motivation: Why neural networks?

- Practical performance in applications



- The human: Andrej Karpathy[1]

[1] https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

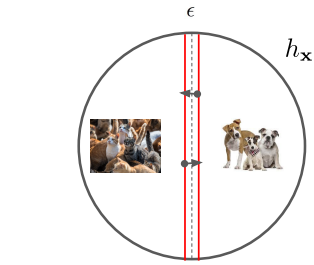
A grand challenge in DL/ML applications: Robustness (I)

(a) Turtle classified as rifle. Athalye et al. 2018.

(b) Stop sign classified as 45 mph sign. Eykholt et al. 2018

Figure: Natural or human-crafted modifications that trick neural networks used in computer vision tasks

A grand challenge in DL/ML applications: Robustness (II)



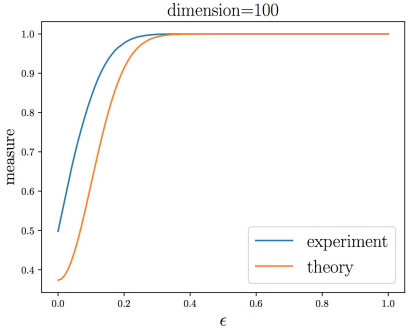


Figure: Understanding the robustness of a classifier in high-dimensional spaces. Shafahi et al. 2019.

Towards adversarial training for robustness

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

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

This problem can be formulated within 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

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

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

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

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

Question

How can we compute the gradient

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

- **Challenge:** It involves differentiating with respect to a maximization.
- **A solution:** We can use Danskin's theorem under some conditions. See EPFL EE-556 (Lectures 9-10).

Basic first-order methods: GD and SGD

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

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

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

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 [7]
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 [7]

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 [25]

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

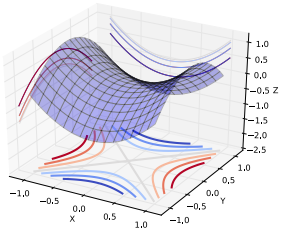


Figure: Minmax saddle ($\lambda_i \neq 0$ for all i)

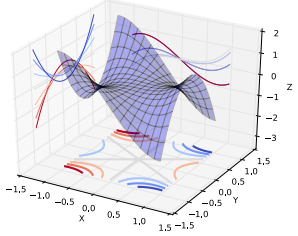


Figure: Monkey saddle ($\lambda_i = 0$ for some i)

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, 23]
3. With probability 1, SGD converges with vanishing step-size if $\mathbf{x}^k$ is bounded with probability 1 [16, 2]

Boundedness is not required (Theorem 1 of [18])

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 [14]
2. With probability $1-\zeta$, PSGD with constant α escapes strict saddles after $\Omega\left(\log(1/\zeta)/\alpha^2\right)$ iterations [8]
   - However, SGD does not converge with constant α
   - We cannot take $\zeta = 0$

SGD avoids traps almost surely (Theorem 3 of [18])

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., $\mathbf{x}^* : \lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) > 0$)

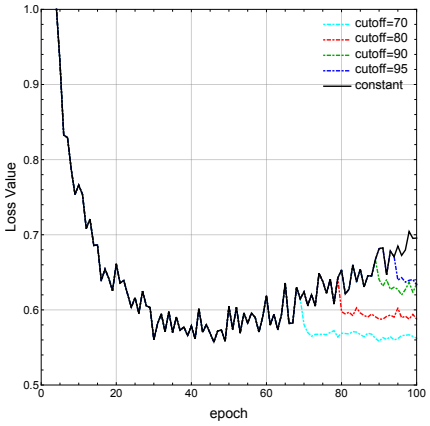
1. SGD with constant α can obtain objective value ϵ-close to a Hurwicz minimizer in $\mathcal{O}(1/\epsilon^2)$-iterations [8, 9]
   - However, SGD does not converge with constant α
   - Need averaging which is problematic in non-convex optimization

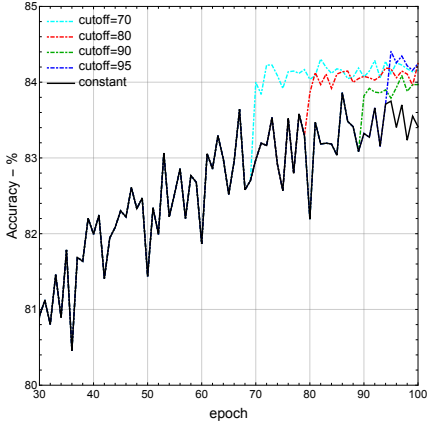
Using a vanishing step-size helps! (Theorem 4 of [18])

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	$\mathrm{dist}(0, \partial(f(\mathbf{x}^k) + \delta_{\mathcal{X}}(\mathbf{x}^k)))^2 =$	?[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-Łojasiewicz, Kurdyka-Łojasiewicz, 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.

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.

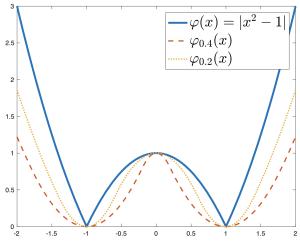


Figure: ME with $f(x) = |x^2 - 1|$, $\mathcal{X} = \mathbb{R}$, and $\hat{v}_t = \mathbb{I}$.[1]

- 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.
[2] 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.

A comparison of adaptive algorithms

	GD/SGD	Accelerated GD/SGD	AdaGrad	AcceleGrad/UniXgrad	Adam/AMSGrad
Convex, stochastic	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^2$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^2$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^{4,5}$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^6$
Convex, deterministic, L-smooth	$\mathcal{O}\left(\frac{1}{k}\right)^2$	$\mathcal{O}\left(\frac{1}{k^2}\right)^2$	$\mathcal{O}\left(\frac{1}{k}\right)^4$	$\mathcal{O}\left(\frac{1}{k^2}\right)^{4,5}$	$\mathcal{O}\left(\frac{1}{k}\right)^7$
Nonconvex, stochastic, L-smooth	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^2$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^2$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^8$	?	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^9$
Nonconvex, deterministic, L-smooth	$\mathcal{O}\left(\frac{1}{k}\right)^2$	$\mathcal{O}\left(\frac{1}{k}\right)^2$	$\mathcal{O}\left(\frac{1}{k}\right)^8$	?	$\mathcal{O}\left(\frac{1}{k}\right)^7$

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

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

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

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

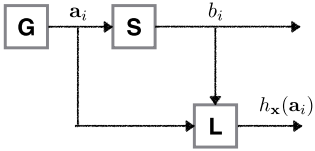
[6] Reddi, Kale, Kumar, On the convergence of adam and beyond, ICLR, 2018.
Alacaoglu, Malitsky, Mertikopoulos, Cevher, A new regret analysis for Adam-type algorithms, ICML 2020.

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

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

[9] Alacaoglu, Malitsky, Cevher, Convergence of adaptive algorithms for weakly convex constrained optimization, NeurIPS, 2021.
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...



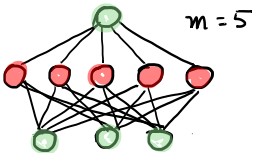
Definition (Optimization formulation)

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 $\mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$



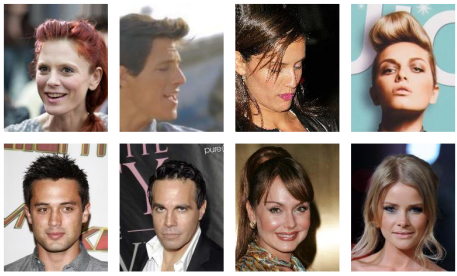
$$h_{\mathbf{x}}(\mathbf{a}) := \begin{bmatrix} \mathbf{X}_2 \end{bmatrix} \sigma \left(\begin{bmatrix} \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \end{bmatrix} + \begin{bmatrix} \mu_1 \end{bmatrix} \right) + \begin{bmatrix} \mu_2 \end{bmatrix}$$

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 generative adversarial networks (GANs)

- Key parametric density estimation setting

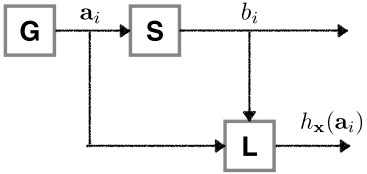


(source: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

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

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

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



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

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

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

Definition

- Let $\omega \sim \mathrm{p}_\Omega$ 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 $h_{\mathbf{x}} \# \mathrm{p}_\Omega$ is the distribution of $h_{\mathbf{x}}(\omega)$.

Towards an optimization problem

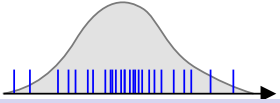
Problem (Ideal parametric density estimator)

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

$$\min_{\mathbf{x}} W_1(\mu^\natural, h_{\mathbf{x}} \# \mathsf{p}_\Omega), \tag{1}$$

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

- 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,

$$W_1(\mu^\natural, h_{\mathbf{x}} \# \mathsf{p}_\Omega) \leq W_1(\mu^\natural, \hat{\mu}_n) + W_1(\hat{\mu}_n, h_{\mathbf{x}} \# \mathsf{p}_\Omega), \tag{2}$$

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

Remarks:

- Using an empirical estimator in high-dimensions is terrible in the worst case [24, 6].
- However, it does not directly say that $W_1\left(\mu^\natural, h_{\mathbf{x}} \# \mathsf{p}_\Omega\right)$ will be large.

Duality of 1-Wasserstein: Towards a minimax formulation

Theorem (Kantorovich-Rubinstein duality)

$$W_1(\mu, \nu) = \sup_{\mathtt{d}} \{\langle \mathtt{d}, \mu \rangle - \langle \mathtt{d}, \nu \rangle : \mathtt{d} \textit{ is 1-Lipschitz}\} \tag{3}$$

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

Inner product is an expectation

$$\langle \mathtt{d}, \mu \rangle = \int \mathtt{d} \mathrm{d}\mu = \int \mathtt{d}(\mathbf{a}) \mathrm{d}\mu(\mathbf{a}) = \boldsymbol{E}_{\mathbf{a} \sim \mu} \left[\mathtt{d}(\mathbf{a}) \right]. \tag{4}$$

Kantorovich-Rubinstein duality applied to our objective

$$W_1\left(\hat{\mu}_n, h_{\mathbf{x}} \# \omega\right) = \sup_{\mathtt{d}} \left\{ \boldsymbol{E}_{\mathbf{a} \sim \hat{\mu}_n} [\mathtt{d}(\mathbf{a})] - \boldsymbol{E}_{\mathbf{a} \sim h_{\mathbf{x}} \# \omega} [\mathtt{d}(\mathbf{a})] : \mathtt{d} \text{ is 1-Lipschitz} \right\} \tag{5}$$

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

$$\min_{\mathbf{x}\in\mathcal{X}} \left(\max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n} \left[\mathsf{d}_{\mathbf{y}}(\mathbf{a})\right] - \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega} \left[\mathsf{d}_{\mathbf{y}}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right] \right). \tag{6}$$

This problem can be formulated within the template $\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x}, \mathbf{y})$.

Remarks:
- Cannot solve in a manner similar to adversarial training a la Danskin. Need a direct approach.
- Scalability, model collapse, catastrophic forgetting. Heuristics galore!
- Enforce Lipschitz constraint weight clipping, gradient penalty, spectral normalization [1, 10, 19].

Abstract minmax formulation

Minimax formulation

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{7}$$

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

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \Phi(\mathbf{x},\mathbf{y}), \tag{7}$$

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 [5]

"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,

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

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,

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

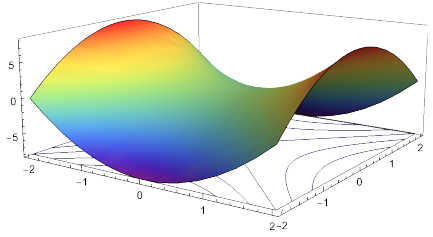


Figure: $\Phi(x, y) = x^2 - y^2$

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.

Basic algorithms for minimax

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

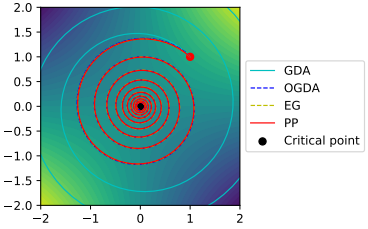


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

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

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

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: [12]

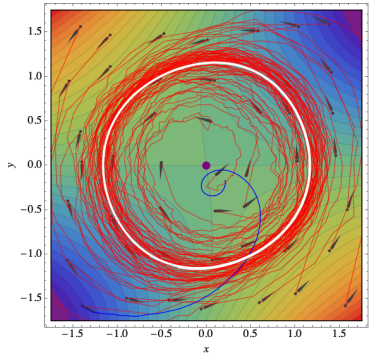
1. Bounded trajectories of $\mathcal{A}$ always converge to an internally chain-transitive (ICT) set.
2. Trajectories of $\mathcal{A}$ may converge with arbitrarily high probability to spurious attractors that contain no critical point of Φ.

Minimax is more difficult than just optimization [12]

- 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}

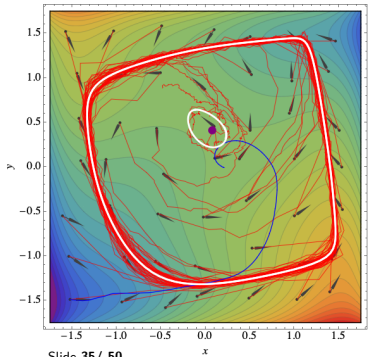
- "Almost" bilinear $\neq$ bilinear:

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



- The "forsaken" solutions:

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

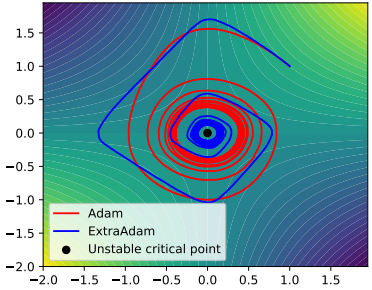


Minimax is more difficult than just optimization [12]

- 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}

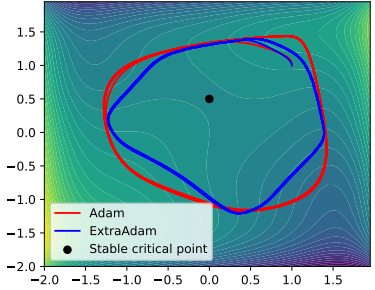
- "Almost" bilinear $\neq$ bilinear:

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



- The "forsaken" solutions:

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



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	$\tilde{\mathcal{O}}\left(\epsilon^{-6}\right)$	[15]
GDA	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[15]
GDmax	nonconvex-concave	$\tilde{\mathcal{O}}\left(\epsilon^{-6}\right)$	[13]
GDmax	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[13]
HiBSA, AGP	nonconvex-concave	$\tilde{\mathcal{O}}\left(\epsilon^{-4}\right)$	[17], [26]
HiBSA, AGP	nonconvex- strongly concave	$\mathcal{O}\left(\epsilon^{-2}\right)$	[17], [26]

Comparison of convergence rates for smooth convex-concave minimax

Method	Assumption on $\Phi(\cdot,\cdot)$	Convergence rate	Reference	Note
PP	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[22]	
PP	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[22]	Implicit algorithm
PP	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[22]	
EG	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[20]	
EG	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[21, 20]	1 extra-gradient evaluation per iteration
EG	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[21, 20]	
OGDA	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[20]	
OGDA	strongly convex- strongly concave	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[21, 20]	no obvious downside
OGDA	Bilinear	$\mathcal{O}\left(\kappa \log(\epsilon^{-1})\right)$	[21, 20]	

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

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

  - $\mathcal{M}(\mathcal{Z}) \coloneqq \{\text{all randomized strategies on } \mathcal{Z}\}$
  - There is a way to solve this infinite dimensional problem: Mirror descent + Langevin dynamics [11]

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!!!** See solution details in [11].

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 $-R_{t+1}$

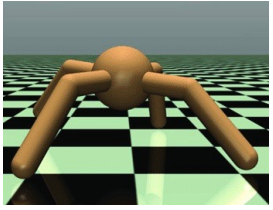
both players enter new state S_{t+1}

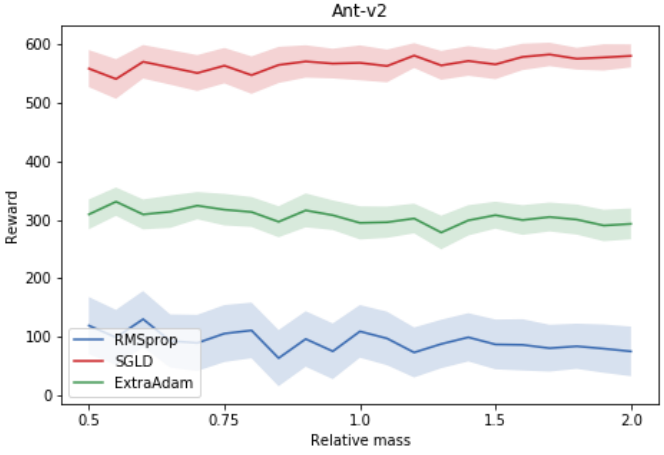
- Hope: Train in one environment, generalize to others

[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.

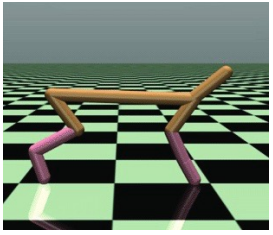
Experimental evaluation via MuJoCo

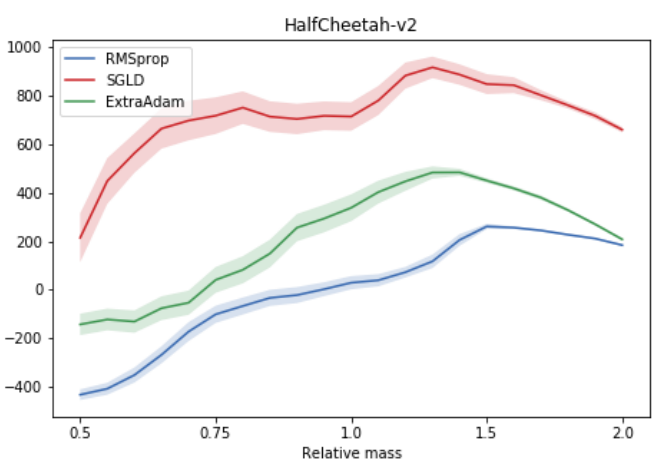
- Standard MuJoCo datasets





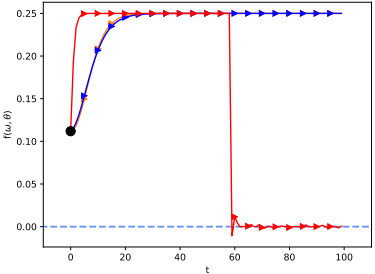
Experimental evaluation via MuJoCo

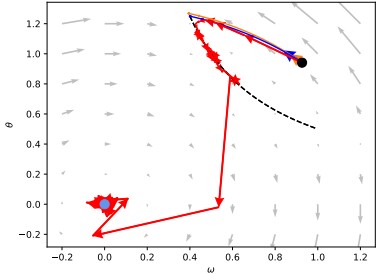


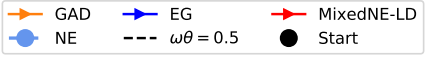


Escaping traps with the mixed-NE concept

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







Thank you for your attention![10]

- Minimax is more difficult than just optimization!
- It is possible to avoid limit cycles under right settings (upcoming work!)
- Universal adaptation is an open research topic

[10] Postdoc positions are available in my group!

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

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

References II

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

[7] 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.

[8] 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.

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

[10] 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 III

[11] 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.

[12] 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.

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

[14] 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.

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

References IV

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

[17] 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.

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

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

[20] 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.

References V

[21] 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.

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

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

[24] 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.

[25] 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.

References VI

[26] 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.