Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 10: Deep learning IV

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

EE-556 (Fall 2020)

License Information for Mathematics of Data Slides

Outline

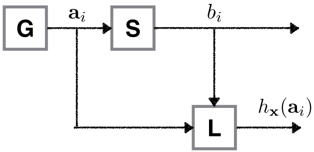
- This class
  - Adversarial Machine Learning (minmax)
    - Adversarial training
    - Generative adversarial networks
    - Difficulty of minmax
- Next class
  - Primal-dual optimization (Part 1)

Adversarial machine learning

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

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

From empirical risk minimization...



Definition (Empirical Risk Minimization (ERM))

Let $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a model with parameters $\mathbf{x}$ and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be samples with $b_i \in \{-1, 1\}$ and $\mathbf{a}_i \in \mathbb{R}^p$. The ERM problem reads

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

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

Some frequently used loss functions

- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \log(1 + \exp(-b h_{\mathbf{x}}(\mathbf{a})))$ *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*

...Into adversarial examples

Definition (Adversarial examples [23])

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

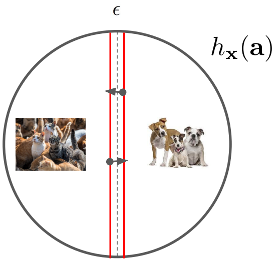
$$\boldsymbol{\eta} \in \arg\max_{\boldsymbol{\eta} : \|\boldsymbol{\eta}\| \leq \epsilon} L(h_{\mathbf{x}^\star}(\mathbf{a} + \boldsymbol{\eta}), \mathbf{b})$$

Example norms frequently used in adversarial attacks

- The most commonly used norm is the ℓ_∞-norm [8, 18].
- The use of ℓ_1-norm leads to sparse attacks.

Figure: (Left) An ℓ_∞-attack: The alteration is hard to perceive. (Right) An ℓ_1-attack: The alteration in this case is obvious.

Challenge: Robustness to adversarial examples



$$h_{\mathbf{x}}(\mathbf{a}+\epsilon) \approx h_{\mathbf{x}}(\mathbf{a}) + \langle \epsilon, \nabla h_{\mathbf{x}}(\mathbf{a}) \rangle$$
$$|h_{\mathbf{x}}(\mathbf{a}+\epsilon) - h_{\mathbf{x}}(\mathbf{a})| \leq \|\epsilon\| \|\nabla h_{\mathbf{x}}(\mathbf{a})\|$$

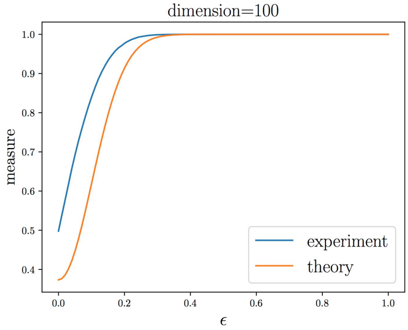


Figure: Understanding the robustness of a classifier in high-dimensional spaces. Shafahi et al. 2019.

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

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 [8, 18].

Fast Gradient Sign Method (FGSM) [8]

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

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

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,

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

Results of FGSM on MNIST

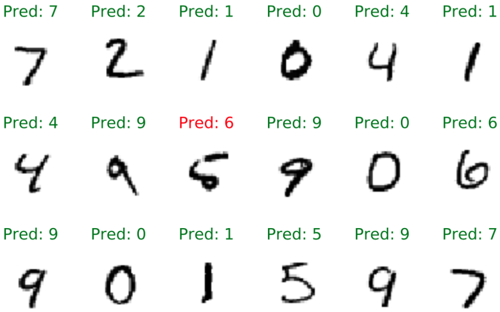


Figure: MNIST images with the predicted digit.

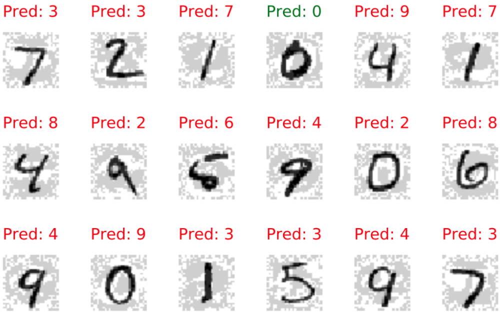


Figure: MNIST images perturbed by a FGSM attack.

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

Adversarial examples and proximal gradient descent

- Target problem:

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

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

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

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

Recall: Proximal operator of indicator functions

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

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

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

Adversarial examples and proximal gradient descent (cont'd)

◦ Target non-convex problem:

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

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

Proximal gradient ascent (PGA)
1. Choose $\boldsymbol{\eta}^0 \in \operatorname{dom} f(\boldsymbol{\eta}) + g(\boldsymbol{\eta})$ as initialization.
2. For $k = 0, 1, \cdots$, generate a sequence $\{\boldsymbol{\eta}^k\}_{k\geq 0}$ as: $$\boldsymbol{\eta}^{k+1} := \operatorname{prox}_{\alpha_k g}\left(\boldsymbol{\eta}^k + \alpha_k \nabla f(\boldsymbol{\eta}^k)\right).$$

Remarks:

◦ PGA results in more powerful adversarial "attacks" than FGSM [14].

◦ The PGA is incorrectly referred to as projected gradient descent in this literature.

◦ Practitioners prefer to use several steps of FGSM instead of PGA [15, 16, 18]:

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

A proposed link between FGSM and PGD

◦ Recall

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

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

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

A proposed link between FGSM and PGD

◦ Recall

- A single step of PGA reads $\boldsymbol{\eta}^{k+1}_{\text{PGA}} := \pi_{\mathcal{N}}\left(\boldsymbol{\eta}^k + \alpha\nabla f(\boldsymbol{\eta})\right)$
- 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 $(\alpha\,|\nabla f(\mathbf{0})|)_i > \epsilon$ for $i = 1, \ldots, n$. Then, one step of PGA yields

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

Multiple steps of FGSM: A connection to majorization-minimization in Lecture 3

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 [5]:

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

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

Remarks:

- Setting $\boldsymbol{\zeta} = \boldsymbol{\eta}^k$ and $\boldsymbol{\eta}^\star = \boldsymbol{\eta}^{k+1}$ with a fixed step size $\alpha = t^\star$, we obtain the update in [15, 16, 18]
$$\boldsymbol{\eta}^{k+1} = \mathsf{clip}\left(\boldsymbol{\eta}^k - t^\star \mathsf{sign}(\nabla f(\boldsymbol{\eta}^k)), [-\epsilon, \epsilon]\right).$$
- 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
$$\partial_{\mathbf{x}} f(\mathbf{x}) = \mathrm{conv}\left\{ \partial_{\mathbf{x}} \Phi(\mathbf{x},\mathbf{y}^\star) : \mathbf{y}^\star \in \mathcal{Y}^\star \right\}.$$

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

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

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

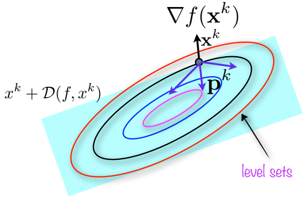


Figure: Descent directions in 2D should be an element of the cone of descent directions $\mathcal{D}(f, \cdot)$.

Descent directions [18]

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 [2].

A practical implementation of adversarial training: Stochastic subgradient descent

Stochastic Adversarial Training [18]
Input: learning rate α_k, iterations T, batch size K.

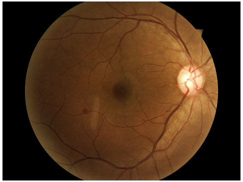
1. initialize neural network parameters $\mathbf{x}^0$
2. For $k = 0, 1, ..., T$:
  i. initialize (sub)gradient vector $\mathbf{g}^k := 0$
  ii. select a mini-batch of data $B \subset \{1, \ldots, n\}$ with $\lvert B\rvert = K$
  iii. For $i \in B$:
    a. Find an attack $\boldsymbol{\eta}^\star$ by (approximately) solving
      $\boldsymbol{\eta}^\star \in \arg\max_{\boldsymbol{\eta}:\lVert\boldsymbol{\eta}\rVert_\infty \leq \epsilon} L(h_{\mathbf{x}^k}(\mathbf{a}_i + \boldsymbol{\eta}), \mathbf{b}_i)$
    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 [22]. Adversarial training helps!



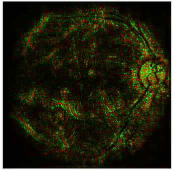
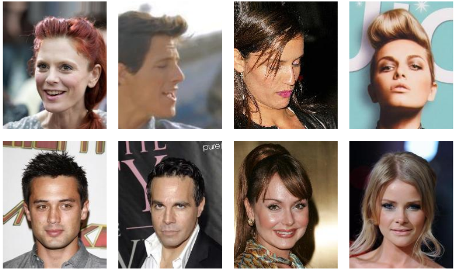


Table: **Left:** Ground truth image, **Middle:** Saliency map, **Right:** Saliency map with adversarial training.

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

- Recall the parametric density estimation setting

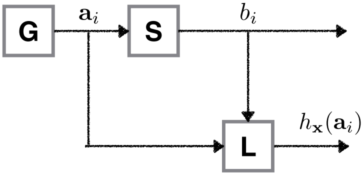


(source: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

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

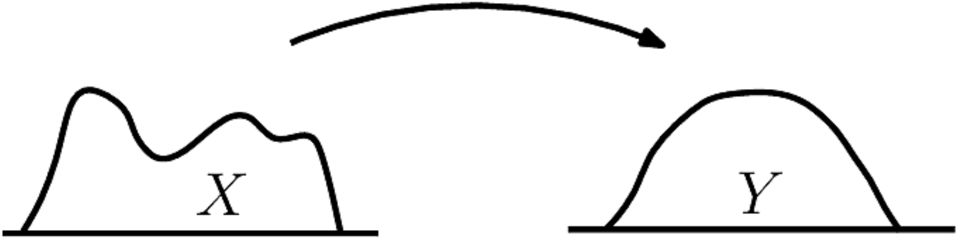


Figure: The Earth Mover's distance

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

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

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 $\omega \sim \mathrm{p}_\Omega$ 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 $h_{\mathbf{x}}(\omega)$.

Example: Chi-square distribution

Let $\omega \sim \mathrm{p}_\Omega := \mathcal{N}(0,1)$ be the normal distribution. Let $h_x : \mathbb{R} \to \mathbb{R}$, $h_x(\omega) = w^x$. Let us fix $x = 2$. Then, $h\#\mathrm{p}_\Omega$ 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 $\omega \sim \mathrm{p}_\Omega$ with probability density function $\mathrm{p}_\Omega(\omega)$, the density $\mathrm{p}_Y(\mathbf{y})$ of $\mathbf{y} = h_{\mathbf{x}}(\omega)$ reads

$$\mathrm{p}_Y(\mathbf{y}) = \mathrm{p}_\Omega(h_{\mathbf{x}}^{-1}(\mathbf{y}))\mathrm{det}\left(\mathbf{J}_{\mathbf{y}} h_{\mathbf{x}}^{-1}(\mathbf{y})\right)$$

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.

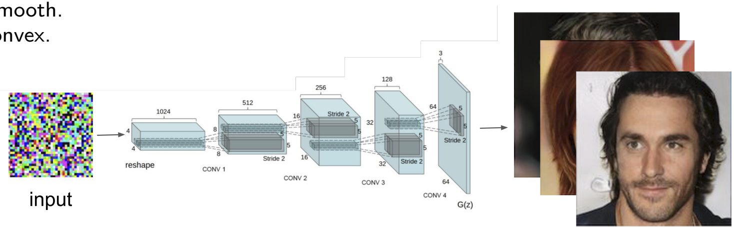
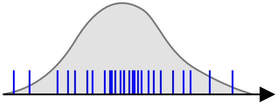


Figure: Schematic of a generative model, $h_{\mathbf{x}} \# \omega$ [7, 13].

Learning without concentration

- We can minimize $W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\mathsf{p}_\Omega\right)$ with respect to $\mathbf{x}$.
- Figure: Empirical distribution (blue), $\hat{\mu}_n = \sum_{i=1}^n \delta_i$



A plug-in empirical estimator

Using the triangle inequality for Wasserstein distances we can upper bound in the follow way,

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

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 [24, 4])

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 $W_1\left(\hat{\mu}_n, h_{\mathbf{x}}\#\mathrm{p}_\Omega\right)$ 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 $\mathbf{d}_{\mathbf{y}}(\mathbf{a})$, where $\mathbf{y} \in \mathcal{Y}$ such that $\mathbf{d}_{\mathbf{y}}(\mathbf{a})$ is 1-Lipschitz. In this case, the Wasserstein GAN optimization problem is given by

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

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

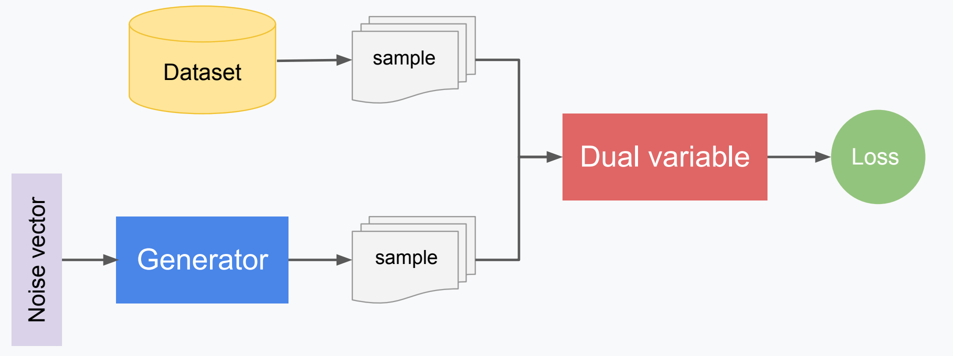


Figure: Generator/dual variable/dataset relation in GANs

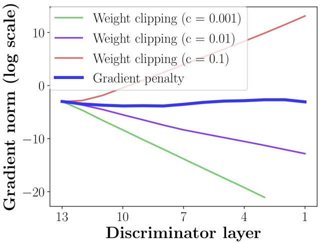
The theory-practice gap: Enforcing 1-Lipschitz of the discriminator

Weight clipping [1]

The "dual" or the "discriminator" $\mathbf{d_y}$ 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 [9]

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

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

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 [19].

Practical implementation of GANs

Stochastic training of Wasserstein GANs
Input: primal and "dual" learning rates γ_t and α_m, primal iterations T, "dual" network $\mathtt{d}_{\mathbf{y}}$, generator network $h_{\mathbf{x}}$, noise distribution p_{Ω}, real distribution $\hat{\mu}_n$, primal and dual batch sizes B, K, "dual" iterations M.

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

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

Some historical background for a Turing award

Vanilla GAN [7]

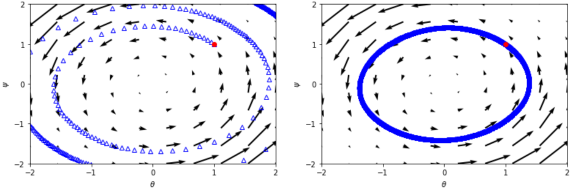
$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{E}_{\mathbf{a}\sim\hat{\mu}_n}\left[\log \mathbf{d_y}(\mathbf{a})\right] + \boldsymbol{E}_{\boldsymbol{\omega}\sim\mathsf{p}_\Omega}\left[\log\left(1-\mathbf{d_y}(h_{\mathbf{x}}(\boldsymbol{\omega}))\right)\right] \tag{14}$$

- Binary cross-entropy modeling.
- $\mathbf{d_y}(\mathbf{a}) : \mathcal{Y} \to [0,1]$ represents the probability that $\mathbf{a}$ came from the real data distribution $\mu^\natural$.

Observation: ∘ Minimizes Jensen-Shannon divergence:

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

Difficulties of GAN training

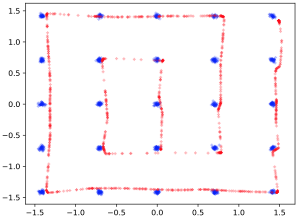


(a) SimGD (b) AltGD

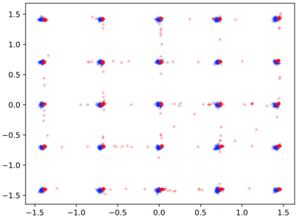
Figure: Mode collapse (left). Simultaneous vs alternating generator/discriminator updates (right).

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

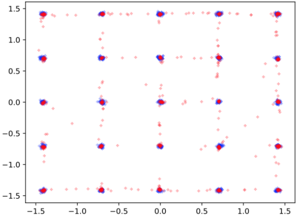
Application to 25 Gaussians: Algorithms matter [10]



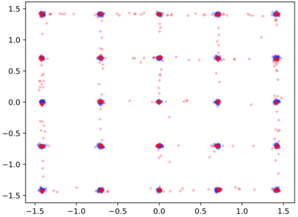
(a) SGD



(b) Adam



(c) Mirror-GAN



(d) Mirror-Prox-GAN

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

A buffet of negative results [3]

"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,

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

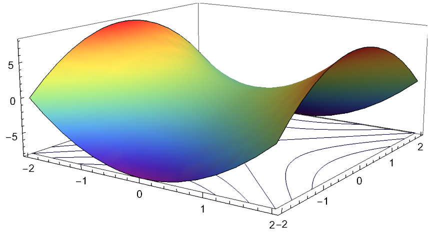


Figure: $\Phi(x, y) = x^2 - y^2$

Recall SGD results from Lecture 9

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

◦ 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 from Lecture 9

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

◦ 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: [11]

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.

A deterministic, simple example beyond convex-concave

- Extragradient method: $\mathbf{z}^{k+1/2} = \mathbf{z}^k + \alpha_k V(\mathbf{z}^k), \mathbf{z}^{k+1} = \mathbf{z}^k + \alpha_k V(\mathbf{z}^{k+1/2})$.

Example (Almost bilinear)

$$\Phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y} + \varepsilon\phi(\mathbf{y}) \tag{16}$$

where $\varepsilon > 0$ and $\phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^2 - \frac{1}{4}\mathbf{y}^4$.

Example (Forsaken)

$$\Phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}(\mathbf{y} - 0.1) + \phi(\mathbf{x}) - \phi(\mathbf{y}) \tag{17}$$

where $\phi(\mathbf{z}) = \frac{1}{4}\mathbf{z}^2 - \frac{1}{2}\mathbf{z}^4 + \frac{1}{6}\mathbf{z}^6$.

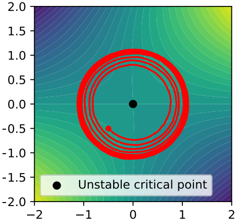


Figure: Extra-gradient on (Almost bilinear) with $\epsilon = 0.1$ converges to a stable limit cycle near an unstable critical point.

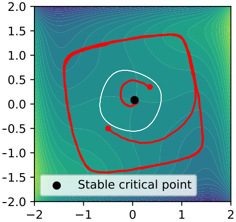


Figure: Extra-gradient on (Forsaken) can converge to a stable limit cycle. the white contour indicates the unstable limit cycle.

ExtraAdam

ExtraAdam for GANs [6]

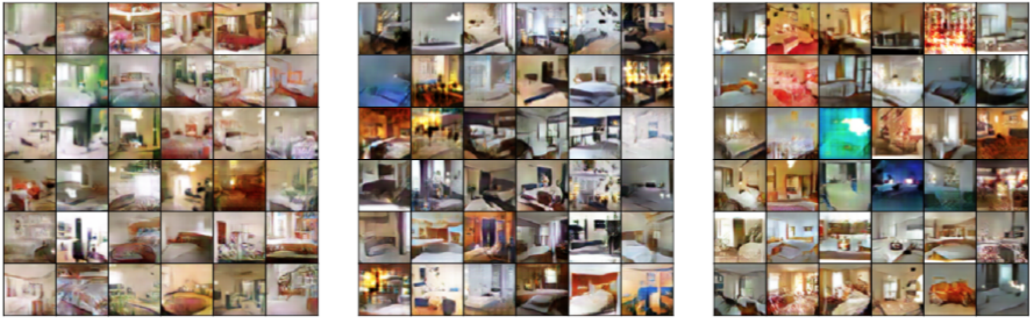
Input. Step size γ, exponential decay rates $\eta_1, \eta_2 \in [0, 1)$

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

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

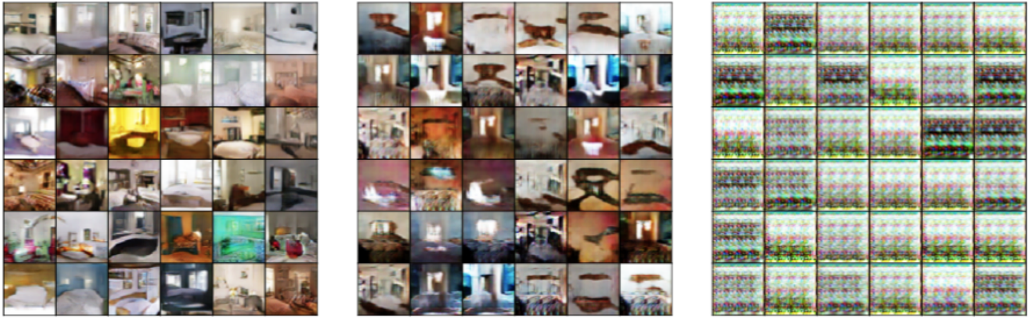
Output : $\mathbf{z}^k$

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



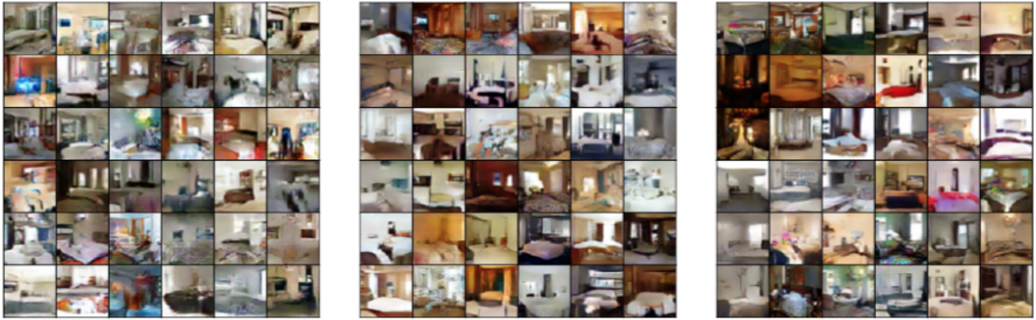
(a) RMSProp

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



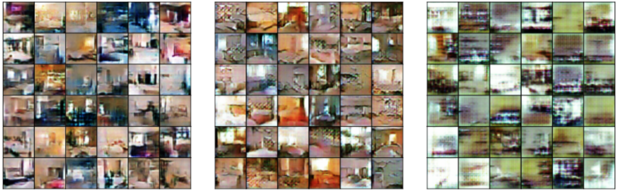
(b) Adam

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

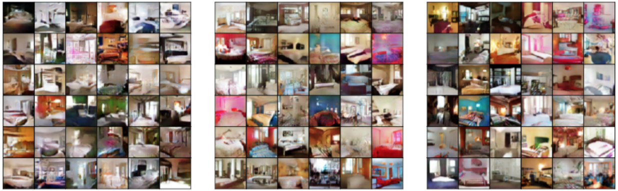


(c) Mirror-GAN, **Algorithm 3**

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



(d) Simultaneous Extra-Adam

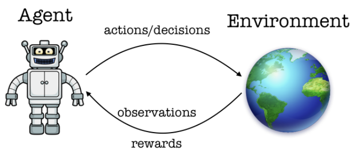


(e) Alternated Extra-Adam

Wrap up!

- Homework 1 is due on Friday at the beginning of the recitation
- Homework 2 will be posted on Friday at the beginning of the recitation

*Reinforcement Learning Game



- Environment: Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, R)$
- Agent: Parameterized deterministic policy $\mu_\theta : \mathcal{S} \to \mathcal{A}$, where $\theta \in \Theta$

Beyond supervised learning: Reinforcement Learning

At time step $t = 0$: $S_0 \sim P_0(\cdot)$

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

agent observes the environment's state $S_t \in \mathcal{S}$

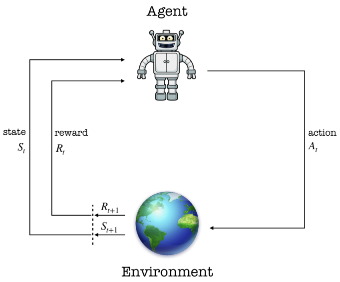
agent chooses an action $A_t = \mu_\theta(S_t) \in \mathcal{A}$

agent receives a reward $R_{t+1} = R(S_t, A_t)$

agent finds itself in a new state $S_{t+1} \sim T(\cdot \mid S_t, A_t)$

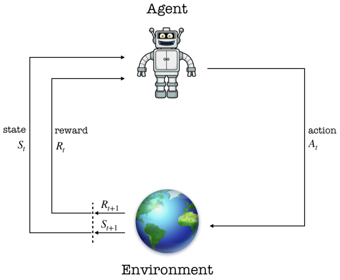
⋆Exploration vs. Exploitation in RL

◦ Challenge: Exploration vs. exploitation!



*Exploration vs. Exploitation in RL

∘ Challenge: Exploration vs. exploitation!

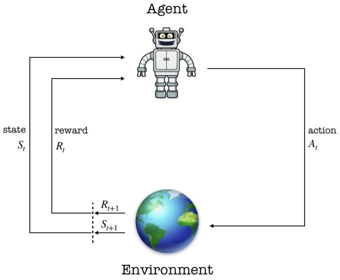


∘ Objective (non-concave): $$\max_{\theta \in \Theta} \; J(\theta) \; := \; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \;\middle|\; \mu_\theta, \mathcal{M}\right]$$

- ▷ The environment only reveals the rewards after actions
- ▷ Exploitation: Maximize objective by choosing the appropriate action

*Exploration vs. Exploitation in RL

- Challenge: Exploration vs. exploitation!



- Objective (non-concave): $$\max_{\theta\in\Theta} \; J(\theta) \; := \; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \;\middle|\; \mu_\theta, \mathcal{M}\right]$$
  - ▷ The environment only reveals the rewards after actions
  - ▷ Exploitation: Maximize objective by choosing the appropriate action
  - ▷ Exploration: Gather information on other actions

*Standard Reinforcement Learning

- Markov Decision Process (MDP): $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, R)$
  - $\mathcal{S}$: state space
  - $\mathcal{A}$: action space
  - $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$: state transition dynamics
  - $\gamma \in (0,1)$: discounting factor
  - $P_0 : \mathcal{S} \to [0,1]$: initial state distribution
  - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: reward function
- Agent's (deterministic) policy: $\mu : \mathcal{S} \to \mathcal{A}$

Reinforcement Learning Game

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

agent observes the environment's state $S_t \in \mathcal{S}$

agent chooses an action $A_t = \mu(S_t) \in \mathcal{A}$

agent receives a reward $R_{t+1} = R(S_t, A_t)$, and finds itself in a new state S_{t+1}

*Standard Reinforcement Learning

◦ Discounted return:

$$Z = \sum_{t=1}^{\infty} \gamma^{t-1} R_t$$

◦ State and state-action value functions:

$$\begin{aligned} V^{\mu}(s) &:= \mathbb{E}\left[Z \mid S_1 = s; \mu, \mathcal{M}\right] \\ Q^{\mu}(s,a) &:= \mathbb{E}\left[Z \mid S_1 = s, A_1 = a; \mu, \mathcal{M}\right] \end{aligned}$$

◦ Performance objective:

$$\max_{\mu} J(\mu) := \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[V^{\mu}(s)\right] = \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[Q^{\mu}(s,\mu(s))\right]$$

*Deterministic Policy Gradient

- Deterministic policy parametrization:

$$\{\mu_\theta : \theta \in \Theta\}$$

- The off-policy performance objective:

$$\max_{\theta \in \Theta} J(\theta) \; := \; J(\mu_\theta) \; = \; \mathop{\mathbb{E}}_{s \sim \mathcal{D}} [Q^{\mu_\theta}(s, \mu_\theta(s))]$$

- The off-policy gradient: [21]

$$\begin{aligned} \nabla_\theta J(\theta) \; &\approx \; \mathbb{E}_{s \sim \mathcal{D}} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s, a)|_{a = \mu_\theta(s)} \right] \\ &\approx \; \frac{1}{N} \sum \nabla_a Q^\phi(s, a) \nabla_\theta \mu_\theta(s) \end{aligned}$$

  - ▷ biased gradient estimate
  - ▷ function approximation Q^ϕ for critic

*An optimization interpretation

- Objective (non-concave): $$\max_{\theta\in\Theta} \ J(\theta) \ := \ \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \ \middle| \ \mu_\theta, \mathcal{M}\right]$$

- Exploitation: Progress in the gradient direction

$$\theta_{t+1} \ \leftarrow \ \theta_t + \eta_t \nabla_\theta \widehat{J(\theta_t)}$$

- Exploration: Add stochasticity while collecting the episodes
  - ▷ noise injection in the action space [21, 17]

$$a = \mu_\theta(s) + \mathcal{N}(0, \sigma^2 I)$$

  - ▷ noise injection in the parameter space [20]

$$\tilde{\theta} = \theta + \mathcal{N}(0, \sigma^2 I)$$

*Robust Reinforcement Learning

- Discounted return:

$$Z = \sum_{t=1}^{\infty} \gamma^{t-1} R_t$$

- State and state-action value functions:

$$\begin{aligned} V^\mu(s) &:= \mathbb{E}\left[Z \mid S_1 = s; \mu, \mathcal{M}\right] \\ Q^\mu(s,a) &:= \mathbb{E}\left[Z \mid S_1 = s, A_1 = a; \mu, \mathcal{M}\right] \end{aligned}$$

- Recall the standard performance objective: $J(\mu) := \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[V^\mu(s)\right] = \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[Q^\mu(s,\mu(s))\right]$

- An action robust formulation:

$$\max_{\mu} \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[\max_{\nu\in\mathcal{N}} Q^\mu(s,\mu(s)+\nu)\right]$$

- See [12] for further details and results.

References I

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

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

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

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

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

References II

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

References III

[11] 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.

[12] 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.

[13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of GANs for improved quality, stability, and variation.
In *International Conference on Learning Representations*, 2018.

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

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

References IV

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

[17] 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.

[18] 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.

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

[20] 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 V

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

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

[23] 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.

[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.