Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 12: Robustness and Diffusion Models

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

EE-556 (Fall 2022)

License Information for Mathematics of Data Slides

Recall: Wasserstein GANs formulation

∘ Ingredients:

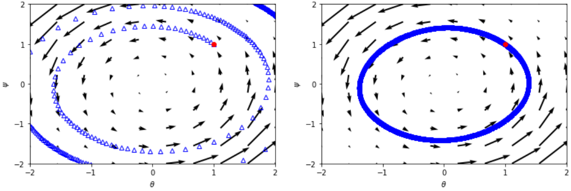
- fixed *noise* distribution p_Ω (e.g., normal)
- target distribution $\hat{\mu}_n$ (natural images)
- $\mathcal{X}$ parameter class inducing a class of functions (generators)
- $\mathcal{Y}$ parameter class inducing a class of functions (dual variables)

Wasserstein GANs formulation [4]

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

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

Difficulties of GAN training

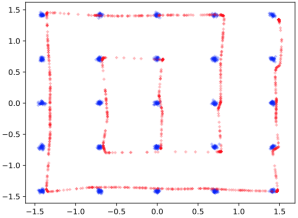


(a) SimGD (b) AltGD

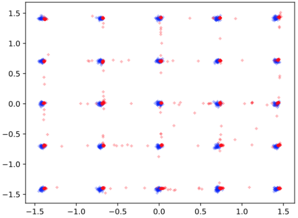
Figure: Mode collapse (left). Simultaneous vs alternating generator/discriminator updates (right).

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

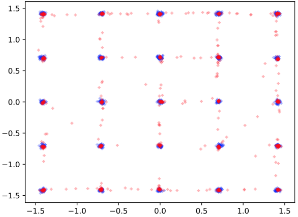
Application to 25 Gaussians: Algorithms matter [33]



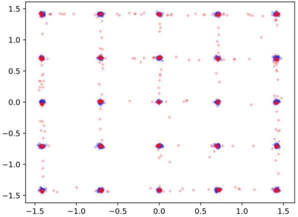
(a) SGD



(b) Adam



(c) Mirror-GAN



(d) Mirror-Prox-GAN

Abstract minmax formulation

Minimax formulation

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

where

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

◦ Key questions:

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

Abstract minmax formulation

Minimax formulation

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

where

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

◦ Key questions:

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

A buffet of negative results [20]

"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."

The difficulty of the nonconvex-nonconcave setting

Minimax formulation

Consider the following problem that captures adversarial training, GANs, and robust reinforcement learning:

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

where Φ is differentiable and nonconvex in $\mathbf{x}$ and nonconcave in $\mathbf{y}$.

From minimax to minimization

Assume $\Phi(\mathbf{x},\mathbf{y}) = f(\mathbf{x})$ for all $\mathbf{y}$. The minimax optimization problem then seeks to find $\mathbf{x}^\star$ such that

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^p,$$

where $\mathbf{x}^\star$ is a global minimum of the nonconvex function f.

- Finding $\mathbf{x}^\star$ is NP-Hard even when f is smooth! (see the complexity supplementary material)
- Finding solutions to a nonconvex-nonconvex min-max problem is harder in general.

Question 1 with a twist: Where do the algorithms want to converge?

Definition (Saddle points & Local Nash equilibria)

The point $(\mathbf{x}^\star, \mathbf{y}^\star)$ is called a saddle-point or a local Nash equilibrium (LNE) if it holds that

$$\Phi(\mathbf{x}^\star, \mathbf{y}) \leq \Phi(\mathbf{x}^\star, \mathbf{y}^\star) \leq \Phi(\mathbf{x}, \mathbf{y}^\star) \qquad \text{(Saddle Point / LNE)}$$

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,

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

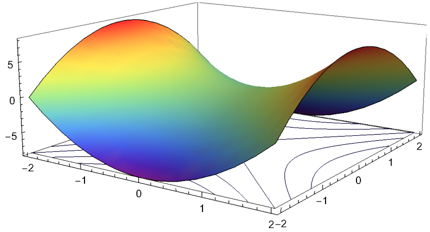


Figure: $\Phi(x, y) = x^2 - y^2$

Saddles of different shapes

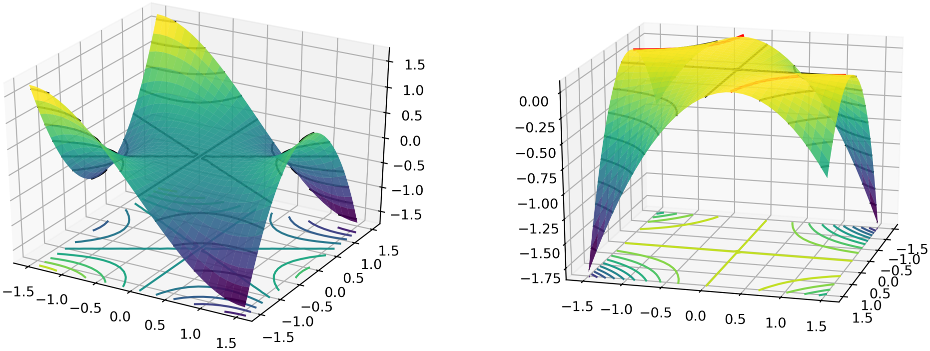


Figure: The monkey saddle $\Phi(x,y) = x^3 - 3xy^2$ (left). The weird saddle $\Phi(\mathbf{x},\mathbf{y}) = -\mathbf{x}^2\mathbf{y}^2 + \mathbf{xy}$ (right) [41].

Recall SGD results from Lecture 11

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

◦ 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 11

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

◦ 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 $\mathbf{z} = [\mathbf{x},\mathbf{y}]$.

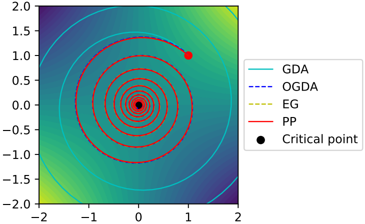


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

- (In)Famous algorithms
  - Gradient Descent Ascent (GDA)
  - Proximal point method (PPM) [59, 28]
  - Extra-gradient (EG) [43]
  - Optimistic GDA (OGDA) [71, 54]
  - Reflected-Forward-Backward-Splitting (RFBS) [13]

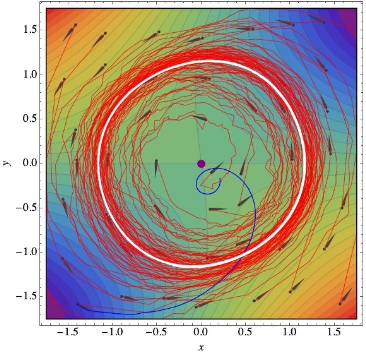
- EG and OGDA are approximations of the PPM
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k)$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^{k+1})$.
  - $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k - \alpha V(\mathbf{z}^{k-1}))$
  - $\mathbf{z}^{k+1} = \mathbf{z}^k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

Minimax is more difficult than just optimization [34]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [8].
  - For optimization, {attracting ICT} ≡ {solutions}
  - For minimax, {attracting ICT} ≡ {solutions} ∪ {spurious sets}

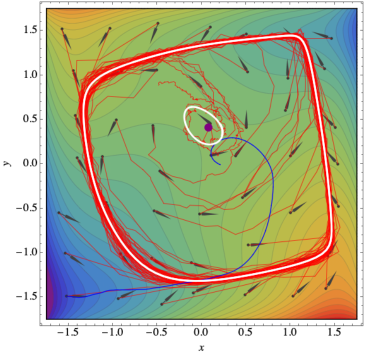
- "Almost" bilinear ≠ bilinear:

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



- The "forsaken" solutions:

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



When do the algorithms converge?

Assumption (weak Minty variational inequality)

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

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

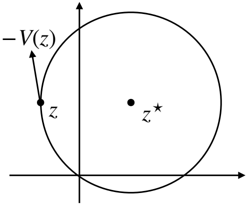
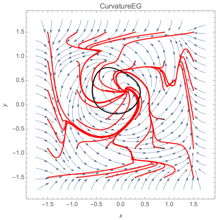


Figure: The operator $V(z)$ is allowed to point away from the solution by some amount when ρ is negative.

- A variant EG+ converges when $\rho > -L/8$
  - Diakonikolas, Daskalakis, Jordan, AISTATS 2021.
- It still cannot handle the examples of [34].

- Complete picture under weak MVI (ICLR Spotlight):
  - Pethick, Lalafat, Patrinos, Fercoq, Cevher; 2021.
  - constrained and regularized settings with $\rho > -L/2$
  - matching lower bounds
  - stochastic variants handling the examples of [34]
  - adaptive variants handling the examples of [34]



An alternative proposal: From pure to mixed Nash equilibrium (NE)

∘ 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

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

- $\mathcal{M}(\mathcal{Z}) := \{$all randomized strategies on $\mathcal{Z}\}$

GAN training as infinite dimensional matrix games

- A different way of looking at GAN objective
  - $\mathbf{I}ph := \int h \, \mathrm{d}p$ for a measure p and function h (Riesz representation)
  - the linear operator G and its adjoint $G^\dagger$:

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

  where $G : \mathcal{M}(\mathcal{Y}) \to \phi(\mathcal{X})$, and $G^\dagger : \mathcal{M}(\mathcal{X}) \to \phi(\mathcal{Y})$

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

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

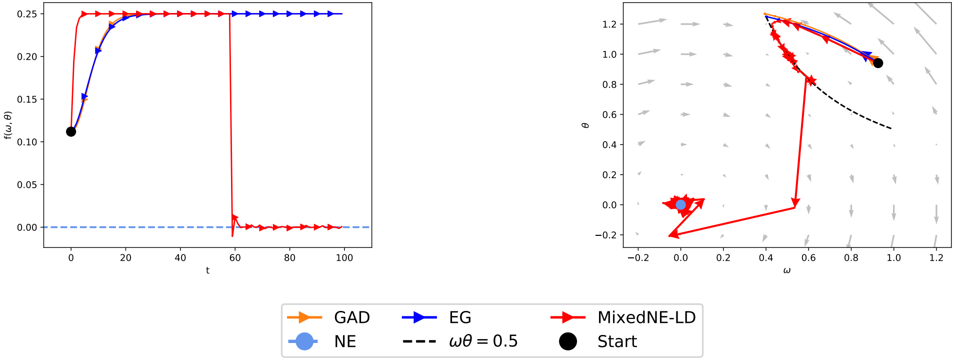
$$
\Updownarrow
$$

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

  - If $\mathcal{X}$ and $\mathcal{Y}$ are finite $\Rightarrow$ mirror descent
  - We can solve this *infinite* dimensional problem via *sampling*: Mirror descent + Langevin dynamics [33]

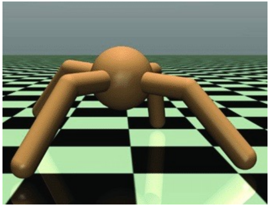
Escaping traps with the mixed-NE concept[1]

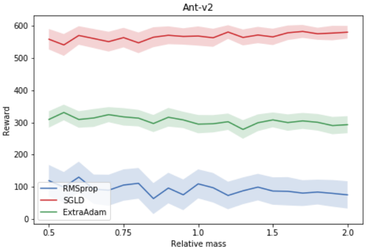
$$\max_{\omega\in[-2,2]} \min_{\mathbf{x}\in[-2,2]} -\omega^2\mathbf{x}^2 + \omega\mathbf{x}$$

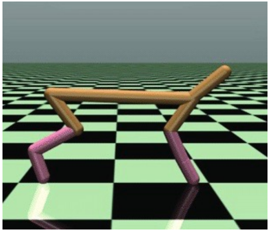


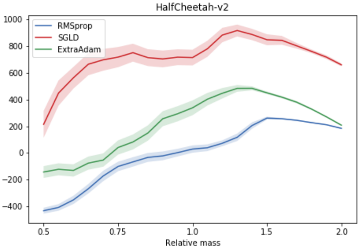
[1] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics" NeurIPS 2020.

Application: Noisy action robust reinforcement learning[1]









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

Two natural questions...

1. Can we learn natural distributions without needing to solve a difficult min-max objective?

2. What is the role of a NN architecture in robustness?

The *sampling* problem

- We assume that a computer can generate a uniform random variable $U \in [0, 1]$.
- We want simulations of random variables with more complicated distributions.

Sampling problem

Let π be a distribution of interest over $\mathbb{R}^p$, with density

$$\pi(\mathbf{a}) = \frac{\exp(-f(\mathbf{a}))}{\int_{\mathbb{R}^p} e^{-f(\mathbf{u})} d\mathbf{u}}.$$

Is it possible to generate samples $\mathbf{a}_i$'s that are approximately distributed according to π $(i = 1, \ldots, n)$?

Remarks:

- The notion of *closeness* to the target π will depend on the application.
- Common metrics are the TV norm, the KL divergence and Wassertein distances.

Definition

The function f is called the potential of π. The gradient of $-f$, i.e, $\nabla_{\mathbf{a}} \log(\pi(\mathbf{a}))$, is called the *score* or the *Stein score* of π.

Iterative refinement like in optimization: MCMC

◦ Just like in optimization, we can iteratively transform an initial guess $\mathbf{a}_0$ to get close to a sample from π.

◦ Given oracle access to ∇f, Langevin Monte Carlo can output a variable close to π.

Langevin Monte Carlo (LMC)

The Langevin Monte Carlo algorithm, or Unadjusted Langevin Algorithm, is defined by the following recursion

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \eta_k \nabla f(\mathbf{a}_k) + \sqrt{2\eta_k}\mathbf{z}_{k+1}$$

where η_k is the step-size and $(\mathbf{z}_k)_k$ is a sequence of i.i.d $\mathcal{N}(0, I_p)$ random variables.

Remarks:

◦ LMC is actually a biased discretization of a gradient flow in the space of measures [40, 68].

◦ LMC is similar to the perturbed SGD we saw in Lecture 11, whose objective is to minimize f.

◦ Sampling cam be faster than optimization in restricted settings [53].

Variants

- LMC (or ULA) is a discretization of an SDE - the *overdamped* Langevin diffusion.
- The *underdamped* Langevin diffusion yields analogues of Nesterov acceleration for sampling [52].
- For constrained distributions, projected [10, 44] and mirrored [32, 1] versions exist.

Langevin Monte Carlo

◦ Extremely well studied: convergence of the algorithm established in the broadest settings [17].

Reference	W_2	TV	KL
[22]	-	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-4})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-2})$
[23]	-	$\widetilde{\mathcal{O}}(L^2p^5\epsilon^{-2})$	-
[19]	$\widetilde{\mathcal{O}}(Lp^9\epsilon^{-6})$	-	-
[17]	-	$\widetilde{\mathcal{O}}(L^2p^4\epsilon^{-2})$	$\widetilde{\mathcal{O}}(L^2p^4\epsilon^{-1})$
[48]	$\widetilde{\mathcal{O}}(L(f)^2p^4C_P^3\epsilon^{-4})$	-	-
[60]	$\widetilde{\mathcal{O}}(Lp^9\epsilon^{-6})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-3})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-\frac{3}{2}})$

Table: Complexity of obtaining an ϵ-close sample. L is the smoothness constant of f, C_p is the Poincare constant[14]. $\widetilde{\mathcal{O}}$ ignores logarithmic terms.

Definition

TV Norm and KL divergence Let p, q be two probability distributions on $(\mathbb{R}^p, \mathcal{B}(\mathbb{R}^p))$,

$$\mathrm{TV}(\mathrm{p}, \mathrm{q}) = \sup_{E \in \mathcal{B}} |\mathrm{p}(E) - \mathrm{q}(E)| \qquad \mathrm{KL}(\mathrm{p}\|\mathrm{q}) = \mathbb{E}_{\mathrm{p}}[\log(\frac{\mathrm{p}}{\mathrm{q}})]$$

Takeaway message

If the score of the target distribution is known, sampling can be provably achieved for a broad class of targets.

Learning the score

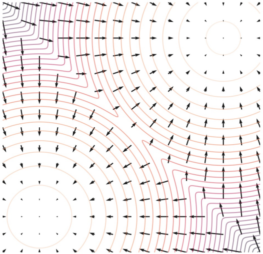


Figure: The score is the vector field pointing to higher density regions [63].

- The key quantity we need is $\nabla_{\mathbf{a}} \log(\pi(\mathbf{a}))$.

Hyvärinen's trick [37]

Given samples from a data distribution π, it is possible to learn $\nabla_{\mathbf{a}} \log(\pi(\mathbf{a}))$ via integration-by-parts.

Remark: ◦ Originally proposed to learn *unnormalized* parametric distributions [37].

Hyvärinen's trick

◦ Approach: Parameterize the vector field with a neural network $h_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}^d$ and approximate the score via

$$\min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_{\mathbf{a}\sim\pi}\left[\|h_{\mathbf{x}}(\mathbf{a}) - \nabla\log\pi(\mathbf{a})\|_2^2\right].$$

Integration by parts.

$$\begin{aligned}
\frac{1}{2}\mathbb{E}_{\mathbf{a}\sim\pi}[\|\nabla_{\mathbf{a}}\log p(\mathbf{a}) - h_{\mathbf{x}}(\mathbf{a})\|_2^2] &= \int\left(\frac{1}{2}\|h_{\mathbf{x}}(\mathbf{a})\|^2 - h_{\mathbf{x}}(\mathbf{a})^T\frac{\nabla_{\mathbf{a}}p(\mathbf{a})}{p(\mathbf{a})} + \frac{1}{2}\|\nabla\log p(\mathbf{a})\|^2\right)p(\mathbf{a})d\mathbf{a} \\
&= \frac{1}{2}\mathbb{E}_{\mathbf{a}\sim\pi}[\|h_{\mathbf{x}}(\mathbf{a})\|^2] - \int h_{\mathbf{x}}(\mathbf{a})^T\nabla_{\mathbf{a}}p(\mathbf{a})d\mathbf{a} + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\mathbf{a}\sim\pi}[\|h_{\mathbf{x}}(\mathbf{a})\|^2] + \int \text{tr}(\nabla_{\mathbf{a}}h_{\mathbf{x}}(\mathbf{a}))p(\mathbf{a})d\mathbf{a} + \text{constant} \\
&= \mathbb{E}_{\mathbf{a}\sim\pi}\left[\frac{1}{2}\|h_{\mathbf{x}}(\mathbf{a})\|^2 + \text{tr}(\nabla_{\mathbf{a}}h_{\mathbf{x}}(\mathbf{a}))\right] + \text{constant} \\
&\simeq \frac{1}{n}\sum_{i=1}^{N}\left(\frac{1}{2}\|h_{\mathbf{x}}(\mathbf{a}_i)\|^2 + \text{tr}(\nabla_{\mathbf{a}}h_{\mathbf{x}}(\mathbf{a}_i))\right) + \text{constant}.
\end{aligned}$$

□

Hyvärinen's trick

Implementable score matching loss

Given independent samples $\mathbf{a}_i$ $(i = 1, \ldots, n)$ of the target distribution π, we can estimate the score by solving

$$\min_{\mathbf{x}\in\mathcal{X}} \boldsymbol{E}_{\mathbf{a}\sim\pi}\left[\mathbf{tr}(J_{h_{\mathbf{x}}}(\mathbf{a})) + \|h_{\mathbf{x}}(\mathbf{a})\|_2^2\right] \simeq \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|h_{\mathbf{x}}(\mathbf{a}_i)\|^2 + \mathbf{tr}(\nabla_{\mathbf{a}} h_{\mathbf{x}}(\mathbf{a}_i))\right), \tag{5}$$

where $J_{h_{\mathbf{x}}}(\mathbf{a})$ denotes the Jacobian of $h_{\mathbf{x}}$ at $\mathbf{a}$.

Remark:

- ◦ Optimizing this loss requires the computation of a neural network hessian.
- ◦ Sliced score matching [64] and Denoising score matching [67] circumvent this expensive step.

Weaknesses of score matching

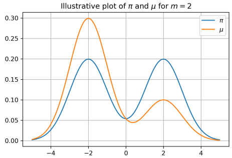


Figure: μ and π are very close in Fisher divergence but do not have the same mode mass [6].

Caveats

Score-matching using Hyvärinen's trick is equivalent to solving

$$\min_{\mathbf{x}\in\mathcal{X}} \mathsf{J}(\pi\|\mu_{\mathbf{x}}). \tag{6}$$

where J is the Fisher divergence [51] and $\mu_{\mathbf{x}}$ is the distribution whose score is given by $h_{\mathbf{x}}$. Unfortunately, closeness in Fisher divergence does not necessarily imply closeness in other divergences.

Remarks:
- ◦ Score matching is not always the most sample efficient [42].
- ◦ The MLE estimator, minimizing KL may be more efficient.

What about natural distributions ?

Natural distributions - Manifold hypothesis

A natural distribution π, like that of images *does not* admit a density of the form e^{-f}. It is assumed to be supported on a low dimensional manifold. Can we still perform sampling when there is no defined score to learn ?

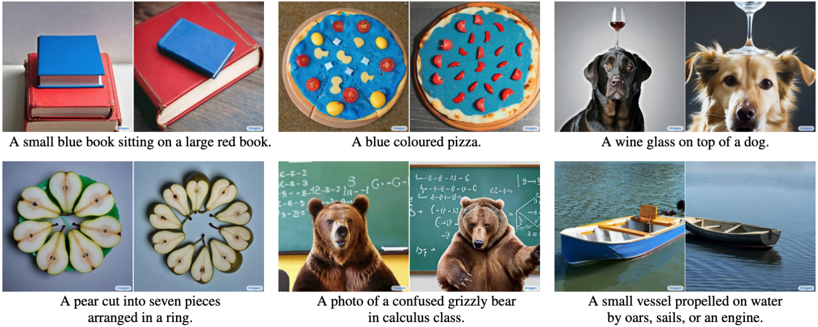
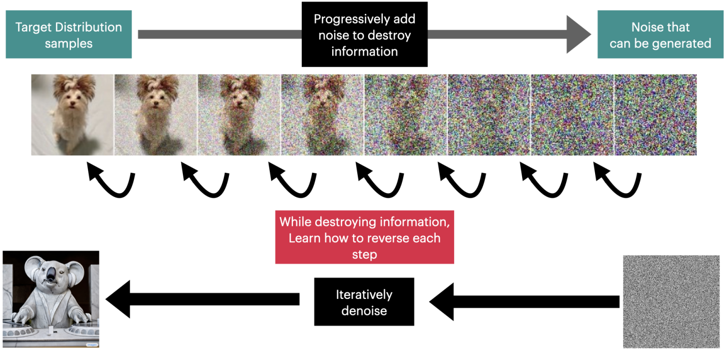


Figure: Images from Imagen [61]

An interpretation of the mathematical foundations for score-based generation



Progressively destroy an image and return back

Diffusion models progressively add noise to an image until it corresponds to pure noise. While doing so they learn the path going in the reverse direction from noise to image.

Stochastic Differential Equations (SDE) formalism

$$d\mathbf{a}_t = f(\mathbf{a}_t, t)dt + g(t)d\mathbf{w}_t$$

Target Distribution - only have samples

Noise that can be generated

$$d\mathbf{a}_t = \left[f(\mathbf{a}_t, t) - g^2(t)\nabla \log \mathrm{p}_t(\mathbf{a_t})\right]dt + g(t)d\bar{\mathbf{w}}_t$$

Score of the intermediate steps

Score-based Generative Models with SDEs - Forward process

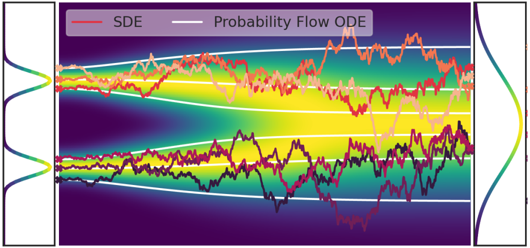


Figure: The forward process: going from data distribution to noise [65].

Forward diffusion

Choose a diffusion process of the form

$$d\mathbf{a}_t = f(\mathbf{a}_t, t)dt + g(t)d\mathbf{w}_t \qquad (7)$$

where f and g are functions of your choice such that $\mathbf{a}_0 \sim \mathrm{p}_0 = \mathrm{p}_{\text{data}}$ and $\mathbf{a}_T$ is easy to sample from for some $T > 0$ (e.g., a Gaussian).

Reverse diffusion

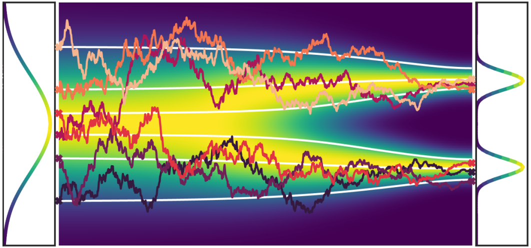


Figure: The reverse process: going from noise to data distribution [65].

Reversing the SDE

The reverse of a diffusion process, as shown by [3], is a diffusion process given by

$$d\mathbf{a}_t = \left[f(\mathbf{a}_t, t) - g^2(t) \nabla_{\mathbf{a}} \log \mathbf{p}_t(\mathbf{a}) \right] dt + g(t) d\bar{\mathbf{w}}_t$$

where $\bar{w}$ flows backward from T to 0 and dt is a negative time step.

Joint training of the score network

Estimating scores for the SDE

Train a time dependent score-based model $h_{\mathbf{x}}(\mathbf{a}, t)$ by solving

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \mathbb{E}_{t\sim\mathsf{Unif}([0,T])} \left[\lambda(t)\mathbb{E}_{\mathbf{a}_0}\mathbb{E}_{\mathbf{a}_t|\mathbf{a}_0}[\|\nabla_{\mathbf{a}_t} \log \mathsf{p}_{0\to t}(\mathbf{a}_t|\mathbf{a}_0) - h_{\mathbf{x}}(\mathbf{a}_t, t)\|^2]\right],$$

where $p_{0\to t}(\mathbf{a}_t|\mathbf{a}_0)$ is the transition kernel from $\mathbf{a}_0$ to $\mathbf{a}_t$ and λ is a positive weight function.

Theoretical guarantees

Sampling is as hard as learning the score [15]

Let q be a bounded data distribution. If the score estimation error in L_2 is at most $O(\epsilon)$, then with an appropriate choice of step size, the reverse diffusion outputs a measure which is ϵ-close in total variation (TV) distance to q in $O(L^2 p/\epsilon^2)$ iterations, where L is the Lipschitz constant of $\nabla \log q$, and p is the dimension of the input.

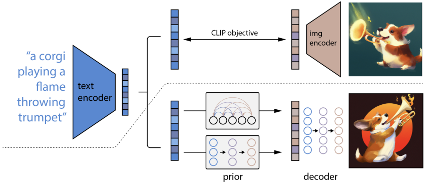
Question: ∘ How hard is it to learn the score ?

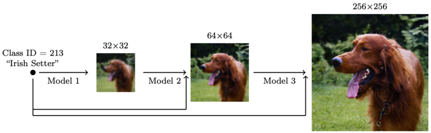
Learning "natural" distributions is hard

No polynomial time algorithm can learn the pushforward of a Gaussian by a single layer neural network.

Remark: ∘ In the statistical query model, no algorithm can learn the score efficiently [16].

Modern tricks to generate appealing images





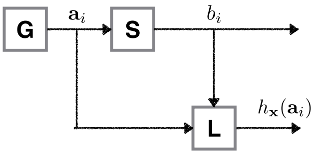
Diffusion models in practice

Additional components are

- Diffusion models conditioned on a text embedding [58].
- Classifier-guidance [58], or classifier free guidance, for better conditional generation [31].
- Sequence or cascade of conditional super-resolution diffusion models to increase resolution [30, 61].

Question: ∘ Is it better than GANs due to Graduate Student Descent?

Recall: from empirical risk minimization to minimax optimization



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

Robustness examples in ML

- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\boldsymbol{\eta}:\|\boldsymbol{\eta}\|_\infty \leq \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}\right), \mathbf{b}_i\right) \right] \right\}$ Adversarial training [36].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\boldsymbol{\eta}:\|\boldsymbol{\eta}\|_2 \leq \epsilon} L(h_{\mathbf{x}+\boldsymbol{\eta}}(\mathbf{a}_i), \mathbf{b}_i) \right] \right\}$ ϵ-stability training [9], Sharpness-aware minimization [26].

Robustness in deep learning: worst-case metric

Definition (Lipschitz constant with respect to the input)

The Lipschitz constant of a differentiable h is $L = \sup_{\mathbf{a}\in\mathbb{R}^p} \|\nabla_{\boldsymbol{a}} h_{\mathbf{x}}(\boldsymbol{a})\|_\star$, where $\|\cdot\|_\star$ is the dual norm.

Remarks:

- Lipschitz constant can be used to describe the worst-case robustness.
- [11, 12] claim that over-parameterization is necessary for the worst-case robustness.
- Lipschitz constant theoretically correlates with the generalization ability of NN classifiers [7].
- There is a trade off between perturbation stability and and approximation ability of NNs [21].

Robustness in deep learning: average-case metric

Definition (Perturbation Stability [70])

The perturbation stability of a neural network $h_{\mathbf{x}}(\mathbf{a})$ is defined as follows:

$$\mathscr{P}(h,\epsilon) = \mathbb{E}_{\mathbf{a},\hat{\mathbf{a}},\boldsymbol{x}} \left\| \nabla_{\boldsymbol{a}} h_{\mathbf{x}}(\mathbf{a})^\top (\mathbf{a} - \hat{\mathbf{a}}) \right\|_2, \quad \forall \boldsymbol{a} \sim \mathcal{D}_A, \quad \hat{\mathbf{a}} \sim \mathsf{Unif}(\mathbb{B}(\epsilon,\mathbf{a})) \,.$$

where $\boldsymbol{x}$ is the neural network parameter, $\mathcal{D}_A$ is the input data distribution, and ϵ is the perturbation radius. $\mathsf{Unif}(\mathbb{B}(\epsilon,\mathbf{a}))$ means the uniform distribution inside the sphere with the center $\mathbf{a}$ and radius ϵ.

Remarks:
- ◦ Average-case robustness may be more meaningful in practice.
- ◦ Perturbation stability can be used to describe the average-case robustness.

Robustness in deep learning: estimation of Lipschitz constant

- Goals: Compute better (tractable) upper bounds on the Lipschitz constant of NNs.
- Applications: Worst-case robustness certification/training.

Table: A comparison of methods for Lipschitz constant estimation.

Bound	layers	norm	quality	method
[57]	single	ℓ_∞	good	SDP
LipSDP [25]	any	ℓ_2	good	SDP
Product	any	$\{1, 2, \cdots, \infty\}$	bad	various
LiPopt [45]	any	$\{1, 2, \cdots, \infty\}$	better	LP/SDP
LipMIP [39]	any	$\{1, 2, \cdots, \infty\}$	exact	LP/IP

Robustness in deep learning: Impact of the NN architecture

◦ It is important to understand the impact of the architecture design choices in NN training

◦ As a running example, let us consider an L-layer fully-connected neural network:

$$h^{(0)}(\mathbf{a}) = \mathbf{a},$$

$$h^{(l)}(\mathbf{a}) = \underbrace{\sigma}_{\text{activation}} \left(\underbrace{\left[\mathbf{X}_l \right]}_{\text{weight}} \underbrace{\left[h^{(l-1)}(\mathbf{a}) \right]}_{\text{input features}} \right), \qquad (L\text{-Layer NN})$$

$$h_{\mathbf{x}}(\mathbf{a}) = h^{(L)}(\mathbf{a}) = \frac{1}{\alpha}\sigma\left(\mathbf{X}_L h^{(L-1)}(\mathbf{a})\right), \qquad \mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_L].$$

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times p}$, $\mathbf{X}_L \in \mathbb{R}^{1 \times m}$, $\mathbf{X}_l \in \mathbb{R}^{m \times m}$ for $l = 2, 3, \cdots, L-1$ (weights).
- Initialization: $\mathbf{X}_1 \sim \mathcal{N}(0, \beta_1^2)$, $\mathbf{X}_L \sim \mathcal{N}(0, \beta_L^2)$, $\mathbf{X}_l \sim \mathcal{N}(0, \beta^2)$ for $l = 2, 3, \cdots, L-1$ (weights).
- Activation function ReLU: $\sigma(\cdot) = \max(\cdot, 0) : \mathbb{R} \to \mathbb{R}$.
- Without loss of generality, we will avoid the bias variables in the sequel.

Robustness in deep learning: initializations

Table: Some commonly used initializations in neural networks.

Initialization name	β_1^2	β^2	β_L^2	α
LeCun [46]	$\frac{1}{p}$	$\frac{1}{m}$	$\frac{1}{m}$	1
He [29]	$\frac{2}{p}$	$\frac{2}{m}$	$\frac{2}{m}$	1
NTK [2]	$\frac{2}{m}$	$\frac{2}{m}$	1	1
Xavier [27]	$\frac{2}{m+p}$	$\frac{1}{m}$	$\frac{2}{m+1}$	1
Mean-field [55]	1	1	1	m
E et al. [24]	1	1	β_c^2	1

Robustness in deep learning: lazy-training

Definition (Lazy-training (Linear) regime [50])

Define an L-layer fully-connected ReLU NN via (L-Layer NN). Let $\mathbf{x}(t) := [\mathbf{X}_1(t), \mathbf{X}_2(t), \dots, \mathbf{X}_L(t)]$ represent the weights of network at training time t. As $m \to \infty$, if the following condition holds

$$\sup_{t \in [0,+\infty)} \frac{\|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_2}{\|\boldsymbol{X}_l(0)\|_2} \to 0, \quad \forall l \in [L] .$$

then the NN training dynamics falls into the lazy-training regime.

Remarks:

- [18] identify the lazy training behavior for $m \to \infty$.
- In the lazy training, NN parameters stay close to initialization during the training.
- The gradient flow of the NN effectively follows the linearization of the NN in lazy training.
- We also refer to the regime with this behavior as the linear regime.
- Lazy training has been extensively studied both empirically and theoretically [38, 47, 5].
- See further the Neural Tangent Kernel Supplementary Lecture.

Robustness in deep learning: initialization and lazy-training

Table: Some commonly used initializations in neural networks.

Initialization name	β_1^2	β^2	β_L^2	α
LeCun [46]	$\frac{1}{p}$	$\frac{1}{m}$	$\frac{1}{m}$	1
He [29]	$\frac{2}{p}$	$\frac{2}{m}$	$\frac{2}{m}$	1
NTK [2]	$\frac{2}{m}$	$\frac{2}{m}$	1	1
Xavier [27]	$\frac{2}{m+p}$	$\frac{1}{m}$	$\frac{2}{m+1}$	1
Mean-field [55]	1	1	1	m
E et al. [24]	1	1	β_c^2	1

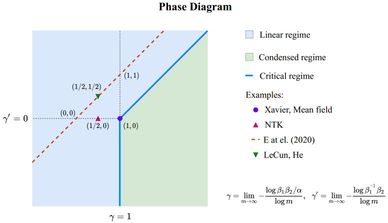


Figure: Phase diagram of two-layer ReLU NNs at infinite-width limit in [50].

Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization)

Table: Comparison of the *perturbation stability* of a deep ReLU neural network under three common Gaussian initializations with different variances in [70].

Initialization name	Bound for $\mathscr{P}(h,\epsilon)/\epsilon^{*}$	Trend w.r.t width	Trend w.r.t depth
LeCun [46]	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)\left(\frac{\sqrt{2}}{2}\right)^{L-2}$	$\nearrow\searrow$	$\searrow$
He [29]	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}$	$\nearrow\searrow$	$\nearrow$
NTK [2]	$\sqrt{L^3 m} e^{-m/L^3} + 1$	$\nearrow\searrow$	$\nearrow$

* The larger perturbation stability means worse average robustness.

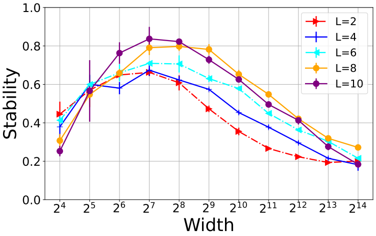
Robustness in deep learning: width and depth & other trade-offs

Table: Comparison of the orders of the bound of three related works under NTK initialization. (The original result of [69] can be reduced to $\sqrt{mL}$ as the $\frac{m}{(\log m)^6} \geq L^{12}$ condition is required).

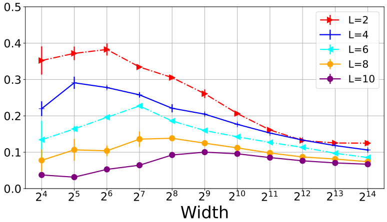
Metrics	[70]	[69]	[35]
$\mathscr{P}(h,\epsilon)/\epsilon$	$\sqrt{L^3 m e^{-m/L^3}} + 1$	$L^2 m^{1/3} \sqrt{\log m} + \sqrt{mL}$	$2^{\frac{3L-5}{2}} \sqrt{m}$

Remarks:
- Consider the over-parameterized regime under NTK initialization [70].
- The width is good but depth is bad for average robustness
- Lipschitz constant directly correlates with the generalization ability of neural network classifiers [7].
- But depth plays a more significant role than width in the expressive power of neural networks [66].

Robustness in deep learning: lazy training experiment for FCN



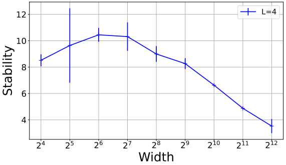
(a) He initialization



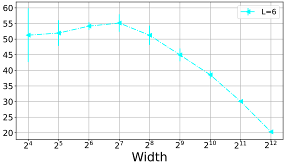
(b) LeCun initialization

Figure: Relationship between the *perturbation stability* and depth of FCN under different depths of $L = 2, 4, 6, 8$ and 10 in [70].

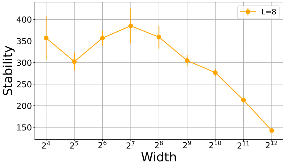
Robustness in deep learning: lazy training experiment for CNN



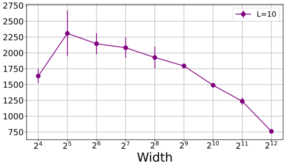
(a) $L = 4$



(b) $L = 6$



(c) $L = 8$



(d) $L = 10$

Figure: Relationship between the *perturbation stability* and width of CNN under He initialization for different depths of $L = 4, 6, 8$ and 10. More experimental results on ResNet can be found in [70].

Wrap up!

◦ Homework 2 continues on Friday!

⋆Robust Reinforcement Learning

◦ Discounted return:

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

◦ State and state-action value functions:

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

◦ Recall the standard performance objective: $J(\mu) := \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \boldsymbol{s} V^{\mu}(s) = \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \boldsymbol{s} Q^{\mu}(s, \mu(s))$

◦ An action robust formulation:

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

◦ See [41] for further details and results.

*Standard Reinforcement Learning

◦ Discounted return:

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

◦ State and state-action value functions:

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

◦ Performance objective:

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

⋆Deterministic Policy Gradient

∘ Deterministic policy parametrization:

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

∘ The off-policy performance objective:

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

∘ The off-policy gradient: [62]

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

- ▷ 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

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

◦ Exploration: Add stochasticity while collecting the episodes

▷ noise injection in the action space [62, 49]

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

▷ noise injection in the parameter space [56]

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

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

◦ 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 [41] for further details and results.

References I

[1] Kwangjun Ahn and Sinho Chewi.
Efficient constrained sampling via the mirror-langevin algorithm.
Advances in Neural Information Processing Systems, 34:28405–28418, 2021.
(Cited on page 22.)

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song.
A convergence theory for deep learning via over-parameterization.
In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.
(Cited on pages 41, 43, and 44.)

[3] Brian D.O. Anderson.
Reverse-time diffusion equation models.
Stochastic Processes and their Applications, 1982.
(Cited on page 32.)

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein generative adversarial networks.
In *International conference on machine learning*, pages 214–223. PMLR, 2017.
(Cited on page 3.)

References II

[5] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang.
On exact computation with an infinitely wide neural net.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 42.)

[6] Krishna Balasubramanian, Sinho Chewi, Murat A Erdogdu, Adil Salim, and Shunshi Zhang.
Towards a theory of non-log-concave sampling: first-order stationarity guarantees for langevin monte carlo.
In *Conference on Learning Theory*, pages 2896–2923. PMLR, 2022.
(Cited on page 27.)

[7] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky.
Spectrally-normalized margin bounds for neural networks.
In *Advances in Neural Information Processing Systems 30*, pages 6240–6249. Curran Associates, Inc., 2017.
(Cited on pages 37 and 45.)

[8] Michel Benaïm and Morris W. Hirsch.
Asymptotic pseudotrajectories and chain recurrent flows, with applications.
Journal of Dynamics and Differential Equations, 8(1):141–176, 1996.
(Cited on page 14.)

References III

[9] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher.
Adversarially robust optimization with gaussian processes.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5765–5775, 2018.
(Cited on page 36.)

[10] Sebastien Bubeck, Ronen Eldan, and Joseph Lehec.
Finite-time analysis of projected langevin monte carlo.
Advances in Neural Information Processing Systems, 28, 2015.
(Cited on page 22.)

[11] Sebastien Bubeck, Yuanzhi Li, and Dheeraj M Nagaraj.
A law of robustness for two-layers neural networks.
In *Annual Conference on Learning Theory*, 2021.
(Cited on page 37.)

[12] Sebastien Bubeck and Mark Sellke.
A universal law of robustness via isoperimetry.
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on page 37.)

References IV

[13] Volkan Cevher and Bang Cong Vu.
A reflected forward-backward splitting method for monotone inclusions involving lipschitzian operators.
Set-Valued and Variational Analysis, pages 1–12, 2020.
(Cited on page 13.)

[14] Djalil Chafai.
Entropies, convexity, and functional inequalities.
Journal of Mathematics of Kyoto University, 44(2):325–363, 2004.
(Cited on page 23.)

[15] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang.
Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions.
arXiv preprint arXiv:2209.11215, 2022.
(Cited on page 34.)

[16] Sitan Chen, Jerry Li, and Yuanzhi Li.
Learning (very) simple generative models is hard.
arXiv preprint arXiv:2205.16003, 2022.
(Cited on page 34.)

References V

[17] Sinho Chewi, Murat A Erdogdu, Mufan Bill Li, Ruoqi Shen, and Matthew Zhang.
Analysis of langevin monte carlo from poincar\'e to log-sobolev.
arXiv preprint arXiv:2112.12662, 2021.
(Cited on page 23.)

[18] Lenaic Chizat, Edouard Oyallon, and Francis Bach.
On lazy training in differentiable programming.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 42.)

[19] Arnak S Dalalyan, Avetik Karagulyan, and Lionel Riou-Durand.
Bounding the error of discretized langevin algorithms for non-strongly log-concave targets.
arXiv preprint arXiv:1906.08530, 2019.
(Cited on page 23.)

[20] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis.
The complexity of constrained min-max optimization.
arXiv preprint arXiv:2009.09623, 2020.
(Cited on pages 6 and 7.)

References VI

[21] Elvis Dohmatob and Alberto Bietti.
On the (non-)robustness of two-layer neural networks in different learning regimes, 2022.
(Cited on page 37.)

[22] Alain Durmus, Szymon Majewski, and Blazej Miasojedow.
Analysis of langevin monte carlo via convex optimization.
Journal of Machine Learning Research, 20:73–1, 2019.
(Cited on page 23.)

[23] Alain Durmus and Eric Moulines.
Nonasymptotic convergence analysis for the unadjusted langevin algorithm.
The Annals of Applied Probability, 27(3):1551–1587, 2017.
(Cited on page 23.)

[24] Weinan E, Chao Ma, and Lei Wu.
A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics.
Science China Mathematics, 2020.
(Cited on pages 41 and 43.)

References VII

[25] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas.
Efficient and accurate estimation of lipschitz constants for deep neural networks.
In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
(Cited on page 39.)

[26] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur.
Sharpness-aware minimization for efficiently improving generalization.
In *International Conference on Learning Representations*, 2021.
(Cited on page 36.)

[27] Xavier Glorot and Yoshua Bengio.
Understanding the difficulty of training deep feedforward neural networks.
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
(Cited on pages 41 and 43.)

[28] Osman Güler.
On the convergence of the proximal point algorithm for convex minimization.
SIAM J. Control Opt., 29(2):403–419, March 1991.
(Cited on page 13.)

References VIII

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.
(Cited on pages 41, 43, and 44.)

[30] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans.
Cascaded diffusion models for high fidelity image generation.
J. Mach. Learn. Res., 23:47–1, 2022.
(Cited on page 35.)

[31] Jonathan Ho and Tim Salimans.
Classifier-free diffusion guidance.
arXiv preprint arXiv:2207.12598, 2022.
(Cited on page 35.)

[32] Ya-Ping Hsieh, Ali Kavis, Paul Rolland, and Volkan Cevher.
Mirrored langevin dynamics.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2883–2892, 2018.
(Cited on page 22.)

References IX

[33] Ya-Ping Hsieh, Chen Liu, and Volkan Cevher.
Finding mixed Nash equilibria of generative adversarial networks.
In *International Conference on Machine Learning*, 2019.
(Cited on pages 5 and 17.)

[34] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher.
The limits of min-max optimization algorithms: Convergence to spurious non-critical sets.
arXiv preprint arXiv:2006.09065, 2020.
(Cited on pages 14 and 15.)

[35] Hanxun Huang, Yisen Wang, Sarah Monazam Erfani, Quanquan Gu, James Bailey, and Xingjun Ma.
Exploring architectural ingredients of adversarially robust deep neural networks.
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on page 45.)

[36] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári.
Learning with a strong adversary.
arXiv preprint arXiv:1511.03034, 2015.
(Cited on page 36.)

References X

[37] Aapo Hyvärinen.
Estimation of Non-Normalized Statistical Models by Score Matching.
Journal of Machine Learning Research, 6(24):695–709, 2005.
(Cited on page 24.)

[38] Arthur Jacot, Franck Gabriel, and Clément Hongler.
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580, 2018.
(Cited on page 42.)

[39] Matt Jordan and Alexandros G Dimakis.
Exactly computing the local lipschitz constant of relu networks.
In *Advances in Neural Information Processing Systems*, 2020.
(Cited on page 39.)

[40] Richard Jordan, David Kinderlehrer, and Felix Otto.
The variational formulation of the fokker–planck equation.
SIAM journal on mathematical analysis, 29(1):1–17, 1998.
(Cited on page 22.)

References XI

[41] Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, and Volkan Cevher.
Robust reinforcement learning via adversarial training with langevin dynamics.
In *NeurIPS*, 2020.
(Cited on pages 10, 49, and 53.)

[42] Frederic Koehler, Alexander Heckett, and Andrej Risteski.
Statistical efficiency of score matching: The view from isoperimetry.
arXiv preprint arXiv:2210.00726, 2022.
(Cited on page 27.)

[43] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
(Cited on page 13.)

[44] Andrew Lamperski.
Projected stochastic gradient langevin algorithms for constrained sampling and non-convex learning.
In *Conference on Learning Theory*, pages 2891–2937. PMLR, 2021.
(Cited on page 22.)

References XII

[45] Fabian Latorre, Paul Rolland, and Volkan Cevher.
Lipschitz constant estimation of neural networks via sparse polynomial optimization.
In *International Conference on Learning Representations*, 2020.
(Cited on page 39.)

[46] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller.
Efficient backprop.
In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
(Cited on pages 41, 43, and 44.)

[47] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington.
Wide neural networks of any depth evolve as linear models under gradient descent.
In *Advances in Neural Information Processing Systems*, 2019.
(Cited on page 42.)

[48] Joseph Lehec.
The langevin monte carlo algorithm in the non-smooth log-concave case.
arXiv preprint arXiv:2101.10695, 2021.
(Cited on page 23.)

References XIII

[49] 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.
(Cited on page 52.)

[50] Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang.
Phase diagram for two-layer relu neural networks at infinite-width limit.
Journal of Machine Learning Research, 2021.
(Cited on pages 42 and 43.)

[51] Siwei Lyu.
Interpretation and generalization of score matching.
arXiv preprint arXiv:1205.2629, 2012.
(Cited on page 27.)

[52] Yi-An Ma, Niladri Chatterji, Xiang Cheng, Nicolas Flammarion, Peter Bartlett, and Michael I Jordan.
Is there an analog of nesterov acceleration for mcmc?
arXiv preprint arXiv:1902.00996, 2019.
(Cited on page 22.)

References XIV

[53] Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan.
Sampling can be faster than optimization.
Proceedings of the National Academy of Sciences, 116(42):20881–20885, 2019.
(Cited on page 22.)

[54] Yura Malitsky and Matthew K Tam.
A forward-backward splitting method for monotone inclusions without cocoercivity.
SIAM Journal on Optimization, 30(2):1451–1472, 2020.
(Cited on page 13.)

[55] Song Mei, Andrea Montanari, and Phan-Minh Nguyen.
A mean field view of the landscape of two-layers neural networks.
Proceedings of the National Academy of Sciences (PNAS), 2018.
(Cited on pages 41 and 43.)

[56] 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.
(Cited on page 52.)

References XV

[57] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang.
Semidefinite relaxations for certifying robustness to adversarial examples.
In *NeurIPS*, 2018.
(Cited on page 39.)

[58] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen.
Hierarchical text-conditional image generation with clip latents.
arXiv preprint arXiv:2204.06125, 2022.
(Cited on page 35.)

[59] R. Tyrrell Rockafellar.
Convex Analysis.
Princeton Univ. Press, Princeton, NJ, 1970.
(Cited on page 13.)

[60] Paul Rolland, Armin Eftekhari, Ali Kavis, and Volkan Cevher.
Double-loop unadjusted langevin algorithm.
In *International Conference on Machine Learning*, pages 8169–8177. PMLR, 2020.
(Cited on page 23.)

References XVI

[61] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al.
Photorealistic text-to-image diffusion models with deep language understanding.
arXiv preprint arXiv:2205.11487, 2022.
(Cited on pages 28 and 35.)

[62] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller.
Deterministic policy gradient algorithms.
In *ICML*, 2014.
(Cited on pages 51 and 52.)

[63] Yang Song and Stefano Ermon.
Generative modeling by estimating gradients of the data distribution.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 24.)

[64] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon.
Sliced score matching: A scalable approach to density and score estimation.
In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.
(Cited on page 26.)

References XVII

[65] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole.
Score-based generative modeling through stochastic differential equations.
arXiv preprint arXiv:2011.13456, 2020.
(Cited on pages 31 and 32.)

[66] Gal Vardi, Gilad Yehudai, and Ohad Shamir.
Width is less important than depth in relu neural networks.
In *Annual Conference on Learning Theory*, 2022.
(Cited on page 45.)

[67] Pascal Vincent.
A connection between score matching and denoising autoencoders.
Neural computation, 23(7):1661–1674, 2011.
(Cited on page 26.)

[68] Andre Wibisono.
Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem.
page 35.
(Cited on page 22.)

References XVIII

[69] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu.
Do wider neural networks really help adversarial robustness?
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on page 45.)

[70] Zhenyu Zhu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher.
Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization).
In *Advances in Neural Information Processing Systems*, 2022.
(Cited on pages 38, 44, 45, 46, and 47.)

[71] Martin Zinkevich.
Online convex programming and generalized infinitesimal gradient ascent.
In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.
(Cited on page 13.)