Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 12: Diffusion Models and Robustness

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

EE-556 (Fall 2024)

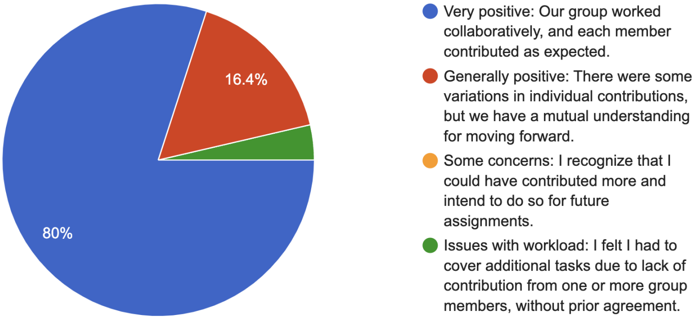
License Information for Mathematics of Data Slides

Teamwork feedback for homework 1

Please provide your feedback on teamwork for Homework 1 below.

Copy chart

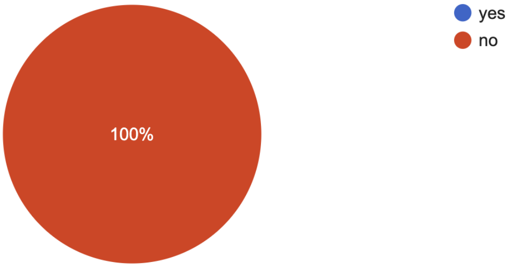
55 responses



Teamwork feedback for homework 1

If you selected the last option in the previous question, would you like a follow-up to address this issue? We may consider adjusting final scores based on each student's contribution if needed.

8 responses

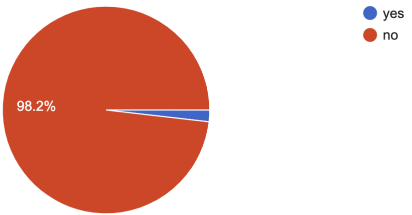


Teamwork feedback for homework 1

Do you plan to change teammates for Homework 2?

Copy chart

55 responses



If you plan to change teammates for Homework 2 and have found new teammates, please indicate their name and sciper number below

1 response

/

Recall: Wasserstein GANs formulation

∘ Ingredients:

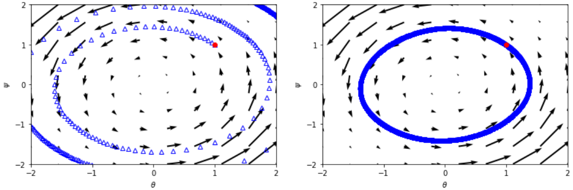
- fixed *noise* distribution p_Ω (e.g., normal)
- target distribution $\hat{\mu}_n$ (natural images)
- $\mathcal{X}$ parameter class inducing a class of functions (generators)
- $\mathcal{Y}$ parameter class inducing a class of functions (dual variables)

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

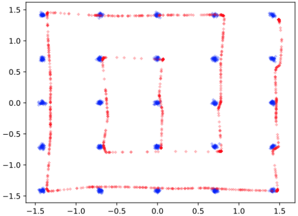


(a) SimGD (b) AltGD

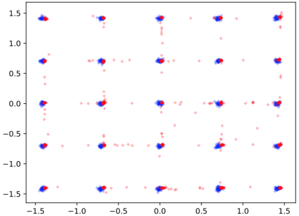
Figure: Mode collapse (left). Simultaneous vs alternating generator/discriminator updates (right).

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

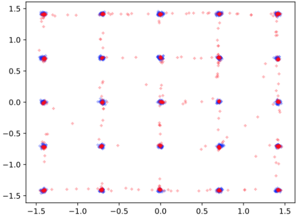
Application to 25 Gaussians: Algorithms matter [39]



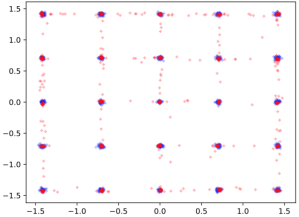
(a) SGD



(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 [22]

"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,

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

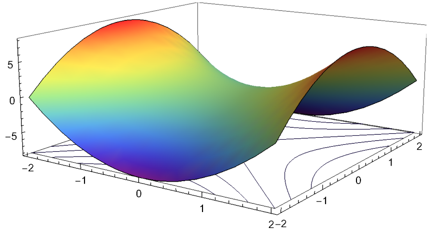


Figure: $\Phi(x, y) = x^2 - y^2$

Saddles of different shapes

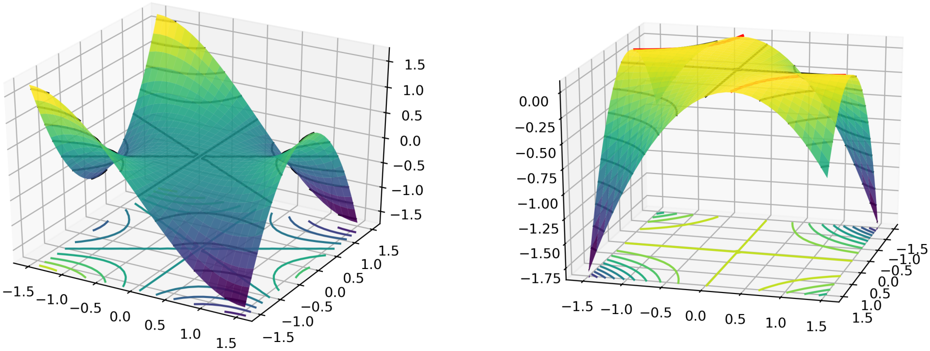


Figure: The monkey saddle $\Phi(x, y) = x^3 - 3xy^2$ (left). The weird saddle $\Phi(\mathbf{x}, \mathbf{y}) = -\mathbf{x}^2\mathbf{y}^2 + \mathbf{x}\mathbf{y}$ (right) [47].

Recall SGD results from Lecture 10

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

◦ 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 10

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

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

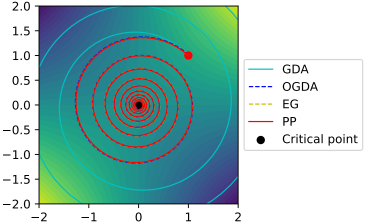


Figure: Trajectory of different algorithms for a simple bilinear game $\min_x \max_y xy$.

∘ (In)Famous algorithms

- Gradient Descent Ascent (GDA)
- Proximal point method (PPM) [69, 33]
- Extra-gradient (EG) [49]
- Optimistic GDA (OGDA) [85, 61]
- Reflected-Forward-Backward-Splitting (RFBS) [13]

∘ EG and OGDA are approximations of the PPM

- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k)$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^{k+1})$.
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(\mathbf{z}^k - \alpha V(\mathbf{z}^k))$
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha[2V(\mathbf{z}^k) - V(\mathbf{z}^{k-1})]$
- $\mathbf{z}^{k+1} = \mathbf{z}^k - \alpha V(2\mathbf{z}^k - \mathbf{z}^{k-1})$

Comparison of convergence rates for smooth convex-concave minimax

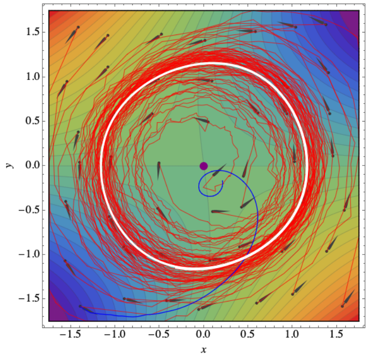
Method	Assumption on $\Phi(\cdot,\cdot)$	Convergence rate	Reference	Note
PP	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[70]	
PP	strongly convex- strongly concave	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[70]	Implicit algorithm
PP	Bilinear	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[70]	
EG	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[29]	
EG	strongly convex- strongly concave	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[64, 29]	1 extra-gradient evaluation per iteration
EG	Bilinear	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[64, 29]	
OGDA	convex-concave	$\mathcal{O}\left(\epsilon^{-1}\right)$	[29]	
OGDA	strongly convex- strongly concave	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[64, 29]	no obvious downside
OGDA	Bilinear	$\mathcal{O}\left(\kappa\log(\epsilon^{-1})\right)$	[64, 29]	

Minimax is more difficult than just optimization [40]

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

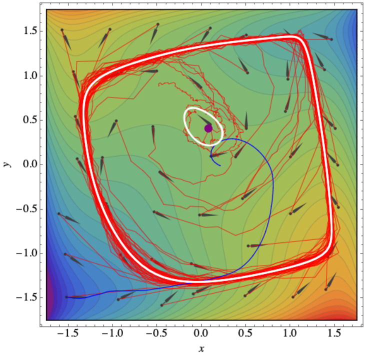
- "Almost" bilinear $\neq$ bilinear:

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



- The "forsaken" solutions:

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



When do the algorithms converge?

Assumption (weak Minty variational inequality)

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

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

- A variant EG+ converges when $\rho > -1/8L$
  - Diakonikolas, Daskalakis, Jordan, AISTATS 2021.
- It still cannot handle the examples of [40].

- Complete picture under weak MVI (ICLR Spotlight):
  - Pethick, Lalafat, Patrinos, Fercoq, Cevher; 2021.
  - constrained and regularized settings with $\rho > -1/2L$
  - matching lower bounds
  - stochastic variants handling the examples of [40]
  - adaptive variants handling the examples of [40]

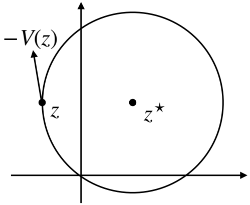
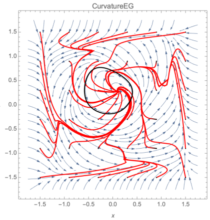


Figure: The operator $V(z)$ is allowed to point away from the solution by some amount when ρ is negative.



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

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

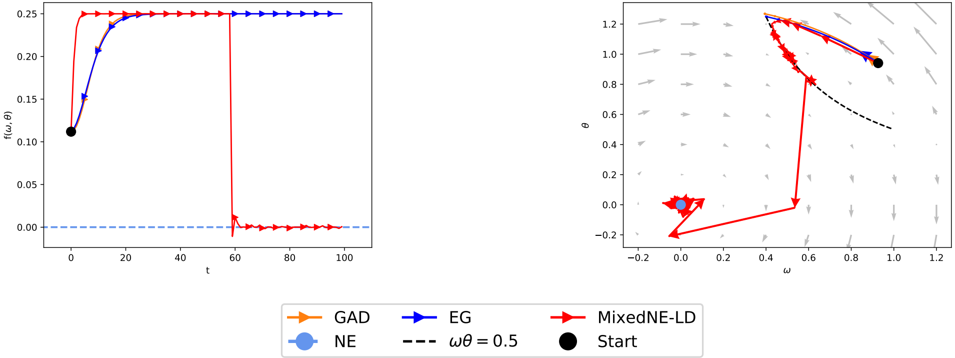
$$
\Updownarrow
$$

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

- If $\mathcal{X}$ and $\mathcal{Y}$ are finite $\Rightarrow$ mirror descent
- We can solve this *infinite* dimensional problem via *sampling*: Mirror descent + Langevin dynamics [39]

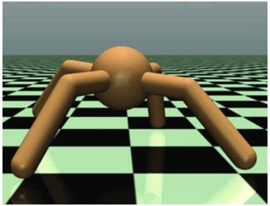
Escaping traps with the mixed-NE concept[1]

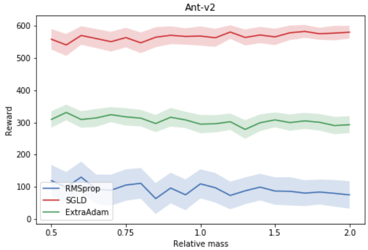
$$\max_{\omega\in[-2,2]} \min_{\mathbf{x}\in[-2,2]} -\omega^2\mathbf{x}^2 + \omega\mathbf{x}$$

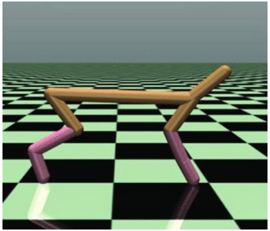


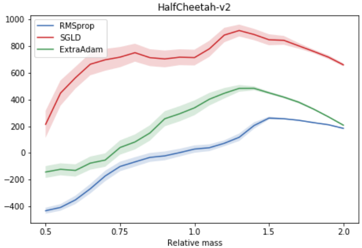
[1] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics" NeurIPS 2020.

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, $-\nabla_{\mathbf{a}} f(\mathbf{a}) = \nabla_{\mathbf{a}} \log(\pi(\mathbf{a}))$, 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 [46, 80].

◦ LMC is similar to the perturbed SGD we saw in Lecture 10, whose objective is to minimize f.

◦ Sampling can be faster than optimization in restricted settings [60].

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 [59].
- For constrained distributions, projected [10, 50] and mirrored [38, 1] versions exist.

Langevin Monte Carlo

∘ Extremely well studied: convergence of the algorithm established in the broadest settings [18].

Reference	W_2	TV	KL
[24]	-	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-4})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-2})$
[25]	-	$\widetilde{\mathcal{O}}(L^2p^5\epsilon^{-2})$	-
[21]	$\widetilde{\mathcal{O}}(Lp^9\epsilon^{-6})$	-	-
[18]	-	$\widetilde{\mathcal{O}}(L^2p^4\epsilon^{-2})$	$\widetilde{\mathcal{O}}(L^2p^4\epsilon^{-1})$
[54]	$\widetilde{\mathcal{O}}(L(f)^2p^4C_P^3\epsilon^{-4})$	-	-
[71]	$\widetilde{\mathcal{O}}(Lp^9\epsilon^{-6})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-3})$	$\widetilde{\mathcal{O}}(Lp^3\epsilon^{-\frac{3}{2}})$

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 $(\mathbb{R}^p, \mathcal{B}(\mathbb{R}^p))$,

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

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

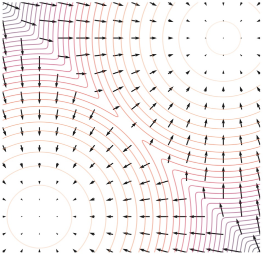


Figure: The score is the vector field pointing to higher density regions [74].

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

Hyvärinen's trick [43]

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 [43].

Hyvärinen's trick

◦ Approach: Parameterize the vector field with a neural network $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}^p$ 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.

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

□

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 [75] and denoising score matching [79] circumvent this expensive step.

Weaknesses of score matching

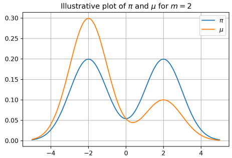


Figure: μ and π are very close in Fisher divergence but do not have the same mode mass [6].

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 [58] 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 distances.

Remarks:

- Score matching is not always the most sample efficient [48].
- The MLE estimator or 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: Samples from stable diffusion 3 [28].

A denoising perspective on sampling

- Denoising corresponds to outputting the best guess for a random variable X given a noisy version X_{noisy}.
- The optimal denoising function `denoise` can be defined as the one minimizing the mean squared error:

$$\texttt{denoise}^\star = \arg \min_{\texttt{denoise}} \mathbb{E}[(X - \texttt{denoise}(X_{\text{noisy}}))^2].$$

Optimal denoising and the score function (Tweedie's formula [27])

Let X be a random variable and define its noisy version $X_\sigma = X + \sigma Z$, where Z is a standard Gaussian. Then, the optimal denoising function that outputs the best guess for X given X_σ is the conditional expectation $\mathbb{E}[X|X_\sigma]$. Moreover, the conditional expectation can be computed as follows:

$$\mathbb{E}[X|X_\sigma] = X_\sigma + \sigma^2 \nabla \log \pi_\sigma(X_\sigma).$$

where π_σ is the density of X_σ.

Remarks:
- Denoising corresponds to taking a gradient step along the score!
- Learning the score is the same as learning how to denoise.

Examples of optimal denoising on MNIST

Figure: Denoising MNIST from various noise levels

Observations:
- Notice that denoising from high noise levels is an averaging of many samples.
- The denoising outputs a true digit as the noise level decreases.

Takeaway message

The conditional expectation $\mathbb{E}[X|X_\sigma]$ is a good denoiser when σ is small. When σ is large, the conditional expectation is an average of too many samples. Denoising from a high noise level is hard.

How to sample with a denoiser

- Given a denoiser, the basic recipe for generating a new sample is the following:
  - Generate a noisy sample $X_{\text{noisy}} = X + tZ$.
  - Denoise it to obtain X.

Remarks:

- How do we obtain a noisy sample if we do not even know how to obtain X?
- If t is large enough then X_{noisy} is very close to being a Gaussian.
- Step 1 then becomes easy but recall that denoising from high noise levels is hard.
- How do we balance these tradeoffs ?

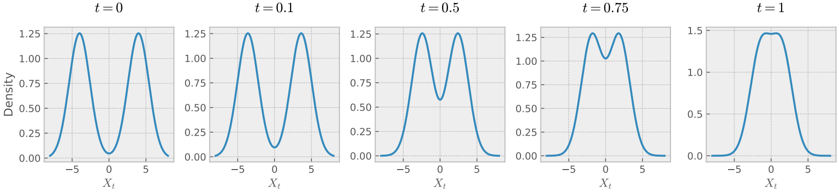


Figure: As t increases the distribution of X_{noisy} becomes more Gaussian.

The answer: Diffusion models

- Diffusion models breakdown the denoising task into several smaller denoising steps.

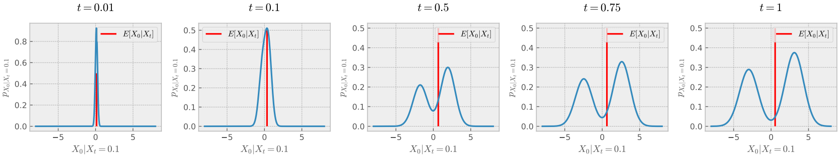


Figure: The vertical red line corresponds to the output of the denoiser, the blue density line is the actual distribution that needs to be approximated. Denoising from a large t to 0 directly leads to misalignment of the denoiser and the true target.

- Instead of denoising from large noise levels, diffusion models denoise progressively:

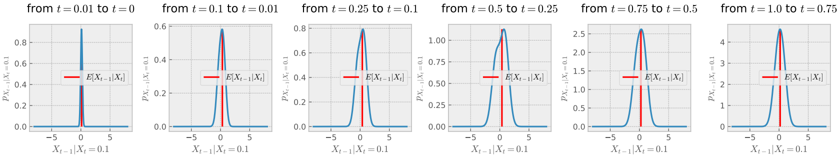
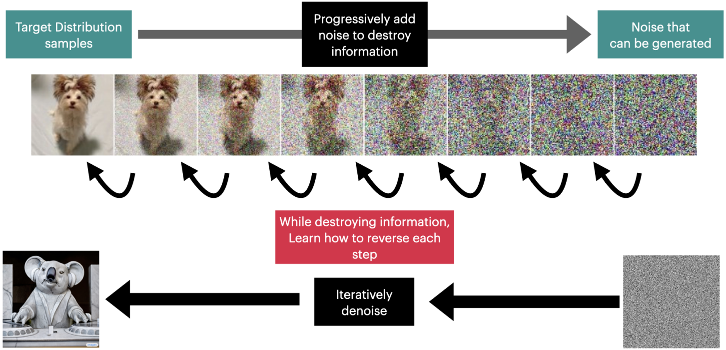


Figure: The denoiser (red vertical line) aligns with the distribution when making smaller denoising steps.

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.

Analyzing what happens during the progressive denoising

- We can gain more understanding about the denoising process
  - The Fourier spectrum of the image provides a revealing insight

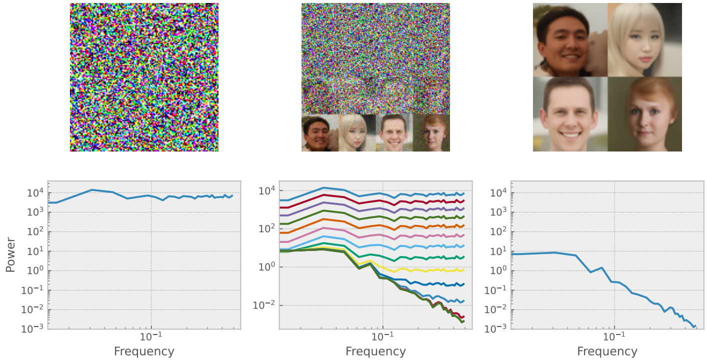


Figure: Notice that the images start of with a flat spectrum (white noise) and the frequeny profile takes shape progressively.

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

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

Score of the intermediate steps

Score-based Generative Models with SDEs - Forward process

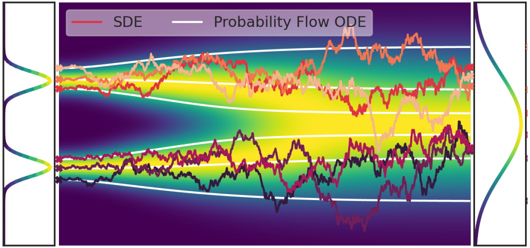


Figure: The forward process: going from data distribution to noise [76].

Forward diffusion

Choose a diffusion process of the form

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

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

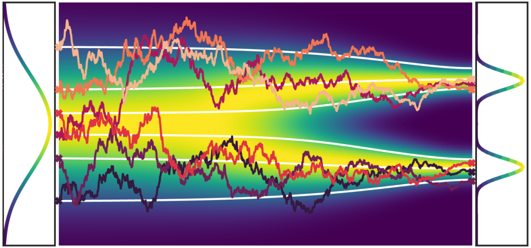


Figure: The reverse process: going from noise to data distribution [76].

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 $\nabla \log \mathrm{q}$, and p is the dimension of the input.

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

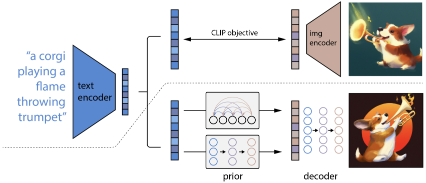
Learning "natural" distributions is hard

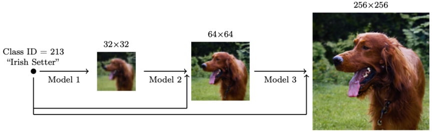
No polynomial time algorithm can learn the pushforward of a Gaussian by a single layer neural network.

Remark:

- ◦ In the statistical query model, no algorithm can learn the score efficiently [17].
- ◦ Neural network required at least two hidden layers and poly(p) neurons [16].
- ◦ A sample complexity bound for score-matching with rate $\frac{1}{\sqrt{n}}$ [84].

Modern tricks to generate appealing images





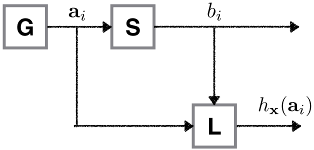
Diffusion models in practice

Additional components are

- Diffusion models conditioned on a text embedding [68].
- Classifier-guidance [68], or classifier free guidance, for better conditional generation [37].
- Sequence or cascade of conditional super-resolution diffusion models to increase resolution [36, 72].

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

- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\boldsymbol{\eta}: \|\boldsymbol{\eta}\|_\infty \le \epsilon} L\left(h_{\mathbf{x}}\left(\mathbf{a}_i + \boldsymbol{\eta}\right), \mathbf{b}_i\right) \right] \right\}$ Adversarial training [42].
- $\min_{\mathbf{x}} \left\{ \frac{1}{n} \sum_{i=1}^n \left[\max_{\boldsymbol{\eta}: \|\boldsymbol{\eta}\|_2 \le \epsilon} L(h_{\mathbf{x}+\boldsymbol{\eta}}\left(\mathbf{a}_i), \mathbf{b}_i\right) \right] \right\}$ ϵ-stability training [9], Sharpness-aware minimization [31].

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 [23].

Robustness in deep learning: average-case metric

Definition (Perturbation Stability [83])

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
[67]	single	ℓ_∞	good	SDP
LipSDP [30]	any	ℓ_2	good	SDP
Product	any	$\{1, 2, \cdots, \infty\}$	bad	various
LiPopt [51]	any	$\{1, 2, \cdots, \infty\}$	better	LP/SDP
LipMIP [45]	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:

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

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

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

- 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: lazy-training

Definition (Lazy-training (Linear) regime [56])

Define an L-layer fully-connected ReLU NN via (L-Layer NN). Let $\mathbf{x}(t) := [\mathbf{X}_1(t), \mathbf{X}_2(t), \ldots, \mathbf{X}_L(t)]$ 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:

- [19] 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 [44, 53, 5].
- See further the Neural Tangent Kernel Supplementary Lecture.

Robustness in deep learning: visualization for lazy training regime

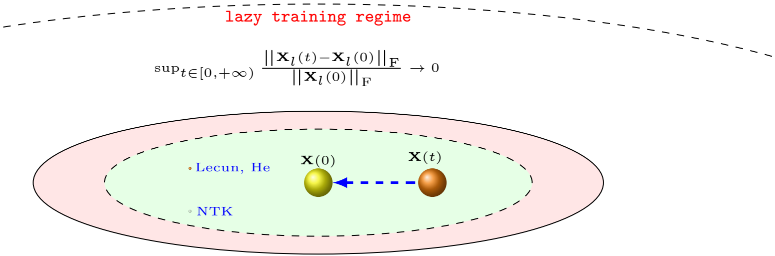


Figure: Training dynamics of two-layer ReLU NNs under different initializations [44, 20, 57].

Robustness in deep learning: initialization in deep ReLU NNs

- 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).

Table: Some commonly used initializations in neural networks.

Initialization name	β_1^2	β^2	β_L^2	α
LeCun [52]	$\frac{1}{p}$	$\frac{1}{m}$	$\frac{1}{m}$	1
He [35]	$\frac{2}{p}$	$\frac{2}{m}$	$\frac{2}{m}$	1
NTK [2]	$\frac{2}{m}$	$\frac{2}{m}$	1	1
Xavier [32]	$\frac{2}{m+p}$	$\frac{1}{m}$	$\frac{2}{m+1}$	1
Mean-field [62]	1	1	1	m
E et al. [26]	1	1	β_c^2	1

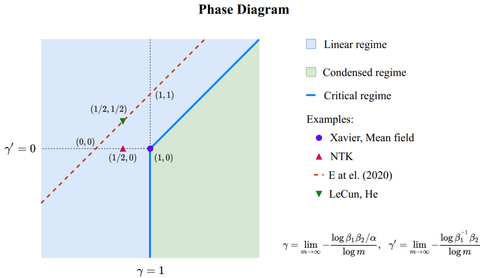


Figure: Phase diagram of two-layer ReLU NNs at infinite-width limit in [56].

Robustness in deep learning: main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

Assumption	Initialization	Our bound for $\mathscr{P}(f,\epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\\|\boldsymbol{x}\\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)(\frac{\sqrt{2}}{2})^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

◦ Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**

Robustness in deep learning: main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

Assumption	Initialization	Our bound for $\mathscr{P}(f, \epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\|\boldsymbol{x}\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)\left(\frac{\sqrt{2}}{2}\right)^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

- Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**
  - width **helps** robustness in the over-parameterized regime

Robustness in deep learning: main results (Lazy-training regime)

Theorem: perturbation stability $\lesssim \mathrm{Func}(m, L, \beta)$

Assumption	Initialization	Our bound for $\mathscr{P}(f,\epsilon)/\epsilon$	Trend of width m [1]	Trend of depth L [1]
$\\|\boldsymbol{x}\\|_2 = 1$	Lecun initialization	$\left(\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}\right)(\frac{\sqrt{2}}{2})^{L-2}$	$\nearrow \searrow$	$\searrow$
	He initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + \sqrt{\frac{1}{p}}$	$\nearrow \searrow$	$\nearrow$
	NTK initialization	$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$	$\nearrow \searrow$	$\nearrow$

[1] The larger perturbation stability means worse average robustness.

- Takeaway messages: **the good (width), the bad (depth), the ugly (initialization)**
  - width **helps** robustness in the over-parameterized regime
  - depth **helps** robustness in Lecun initialization but **hurts** robustness in He/NTK initialization

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 [81] can be reduced to $\sqrt{mL}$ as the $\frac{m}{(\log m)^6} \geq L^{12}$ condition is required).

Metrics	[83]	[81]	[41]
$\mathscr{P}(h,\epsilon)/\epsilon$	$\sqrt{L^3 m}e^{-m/L^3}+1$	$L^2 m^{1/3}\sqrt{\log m}+\sqrt{mL}$	$2^{\frac{3L-5}{2}}\sqrt{m}$

Remarks:
- Consider the over-parameterized regime under NTK initialization [83].
- 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 [78].
- The experimental results and the results of non-Lazy training are in the appendix.

The good, the bad and the ugly in deep learning

	good	bad	ugly
neural networks	performance	analysis	over-parameterization
generalization	benign overfitting	catastrophic overfitting	model complexity
robustness	width	depth	initialization

$\mu-$transfer: Zero-shot hyperparameter transfer with μP

- Zero-shot hyperparameter transfer: can we leverage parameters tuned on smaller models for larger ones?
- Standard parameterizations (using He/LeCun initialization) do not allow for efficient hyperparameter transfer.
- Design principles of maximal update parameterization (μP):
  - Scale the initialization variance and the learning rate in a specific manner
  - Ensure effective updates (i.e., $\Delta\mathbf{X}_l h^{(l-1)}(\mathbf{a}) \sim \Theta(1)$) hold at any layer l and each time step.

Remark: ◦ With this μP-recipe, we can transfer the hyperparameters across different scales.

Implementation of μP: abc-parameterizations

- Initialize the weights of each layer as $\mathbf{X}_l = m^{-a_l}\mathbf{W}_l$, where $\mathbf{W}_l$ is the actual trainable parameter.
- Initialize each entry of $\mathbf{W}_l$ such that $[\mathbf{W}_l]_{ij} \sim \mathcal{N}(0, m^{-2b_l})$.
- Set the SGD learning rate to ηm^{-c}, where η is a width-independent constant.
- The Maximal Update Parametrization, for an L-hidden-layer MLP, is defined by the following:

$$a_l = \begin{cases} -\frac{1}{2} & \text{for } l = 1, \\ 0 & \text{for } 2 \leq l \leq L, \\ \frac{1}{2} & \text{for } l = L+1, \end{cases} \quad b_l = \frac{1}{2} \quad \forall l, \quad c = 0$$

$\mu-$transfer: transferability and empirical results

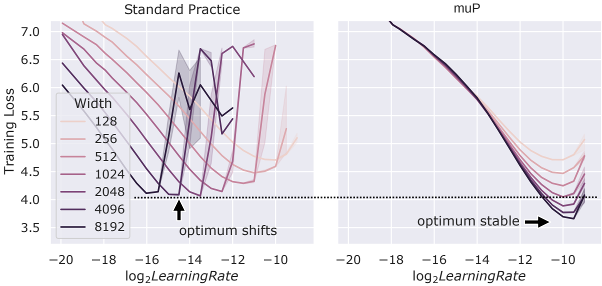


Figure: Loss of Transformers trained with Adam against learning rate for different network widths [82].

Remarks:
- Under μP, the optimal learning rate remains largely unchanged when scaling network width.
- This is not the case with standard parameterizations (i.e., He or LeCun).
- Theoretical foundations for $\mu-$transfer relate to the edge of stability [65].
- μP can be extended to more complex network architectures and algorithms, such as state-space models (SSM) [77] and sharpness-aware minimization (SAM) [34].

Wrap up!

◦ Homework 2 continues on Friday!

⋆Theoretical guarantees for score-matching

A sample complexity bound for score-matching [84]

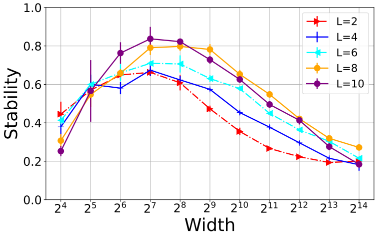
Suppose we train a DNN with SGD to estimate the score. For any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$, with probability at least $1 - 2\delta - 2L\exp(-\Omega(m))$ over the randomness of initialization and noise, it holds that

$$\frac{1}{T-t_0}\int_{t_0}^{T} \|\nabla_{\mathbf{a}_t} \log \mathrm{p}_{0\to t}(\mathbf{a}_t|\mathbf{a}_0) - h_{\mathbf{x}}(\mathbf{a},t)\|^2 \,\mathrm{d}t \lesssim \frac{1}{n\varepsilon^2}\left(\frac{p(T-\log(t_0))}{T-t_0}\right)\log\frac{\mathcal{N}_c(\frac{1}{n},\mathcal{S})}{\delta} + \frac{1}{n} + p\varepsilon^2,$$

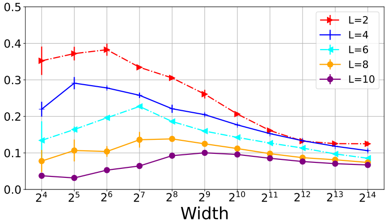
where the $\mathcal{N}_c(\frac{1}{n},\mathcal{S})$ is the covering number [63] of the function space $\mathcal{S}$.

Remarks:
- By choosing $\varepsilon^2 = \frac{1}{\sqrt{n}}$, we can obtain the best bound, which becomes $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$.
- When $t_0 = 0$, neither the setting nor the result are meaningful.
- When $T \gg t_0 \approx 1$, the bound simplifies to $\frac{p+C_d^2}{\sqrt{n}}\log\frac{\mathcal{N}_c(\frac{1}{n},\mathcal{S})}{\delta}$, which independent of time steps T.

*Robustness in deep learning: lazy training experiment for FCN



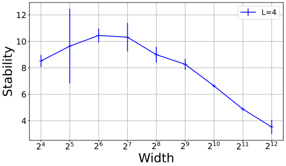
(a) He initialization



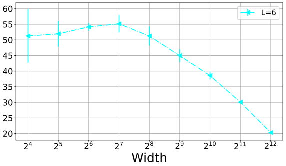
(b) LeCun initialization

Figure: Relationship between the *perturbation stability* and depth of FCN under different depths of $L = 2, 4, 6, 8$ and 10 in [83].

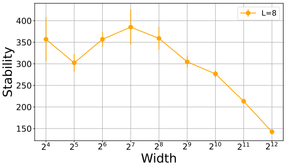
*Robustness in deep learning: lazy training experiment for CNN



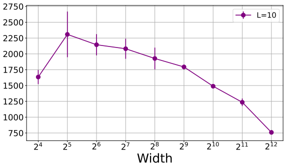
(a) $L = 4$



(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 [83].

⋆Robustness in deep learning: Visualization for non-lazy training regime

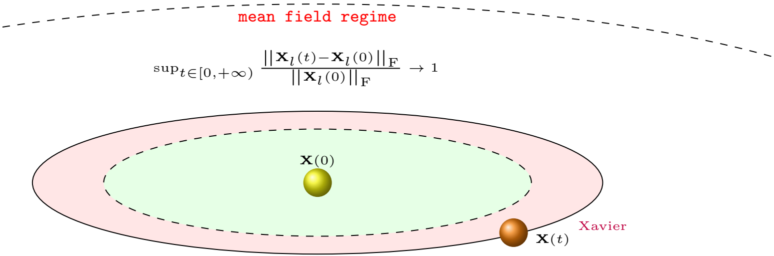


Figure: Training dynamics of two-layer ReLU NNs under different initializations [44, 20, 57].

⋆Robustness in deep learning: Visualization for non-lazy training regime

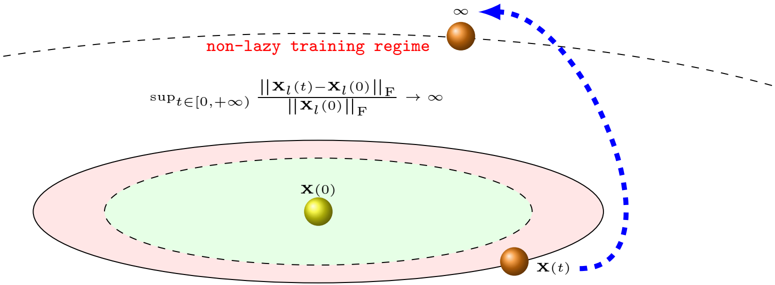


Figure: Training dynamics of two-layer ReLU NNs under different initializations [44, 20, 57].

*Robustness in deep learning: main results (Non-lazy training regime)

A sufficient condition for DNNs

For large enough m and $m \gg p$, w.h.p, DNNs fall into non-lazy training regime if $\alpha \gg (m^{3/2} \sum_{i=1}^{L} \beta_i)^L$.

Remarks: ○ $L = 2$, $\alpha = 1$, $\beta_1 = \beta_2 = \beta \sim \frac{1}{m^c}$ with $c > 1.5$

*Robustness in deep learning: main results (Non-lazy training regime)

A sufficient condition for DNNs

For large enough m and $m \gg p$, w.h.p, DNNs fall into non-lazy training regime if $\alpha \gg (m^{3/2} \sum_{i=1}^{L} \beta_i)^L$.

Remarks: ◦ $L = 2$, $\alpha = 1$, $\beta_1 = \beta_2 = \beta \sim \frac{1}{m^c}$ with $c > 1.5$

Theorem (non-lazy training regime for two-layer NNs)

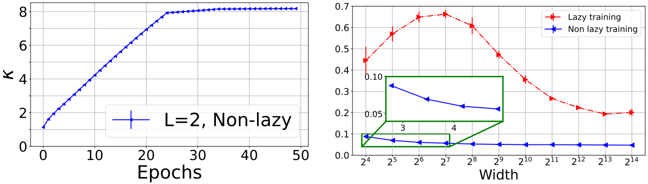
Under this setting with $m \gg n^2$ and standard assumptions, then

$$\textit{perturbation stability} \leq \widetilde{\mathcal{O}}\left(\frac{n}{m^{c+1.5}}\right), \; whp.$$

Remarks: ◦ Width helps robustness in the over-parameterized regime in both lazy/non-lazy training regime

⋆Robustness in deep learning: non-lazy training experiment

$$\text{lazy training ratio } \kappa := \frac{\sum_{l=1}^{L} \|\boldsymbol{X}_l(t) - \boldsymbol{X}_l(0)\|_{\mathrm{F}}}{\sum_{l=1}^{L} \|\boldsymbol{X}_l(0)\|_{\mathrm{F}}}$$



⋆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: $J(\mu) := \underset{s\sim\mathcal{D}}{\mathbb{E}} \boldsymbol{s} V^{\mu}(s) = \underset{s\sim\mathcal{D}}{\mathbb{E}} \boldsymbol{s} Q^{\mu}(s, \mu(s))$

∘ An action robust formulation:

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

∘ See [47] 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: [73]

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

  - ▷ 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 [73, 55]

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

  - ▷ noise injection in the parameter space [66]

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

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

◦ An action robust formulation:

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

◦ See [47] 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 26.)

[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 page 52.)

[3] Brian D.O. Anderson.
Reverse-time diffusion equation models.
Stochastic Processes and their Applications, 1982.
(Cited on page 41.)

[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 6.)

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 50.)

[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 31.)

[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 46 and 56.)

[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 18.)

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 45.)

[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 26.)

[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 46.)

[12] Sebastien Bubeck and Mark Sellke.
A universal law of robustness via isoperimetry.
In *Advances in Neural Information Processing Systems*, 2021.
(Cited on page 46.)

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 16.)

[14] Djalil Chafai.
Entropies, convexity, and functional inequalities.
Journal of Mathematics of Kyoto University, 44(2):325–363, 2004.
(Cited on page 27.)

[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 43.)

[16] Sitan Chen, Aravind Gollakota, Adam Klivans, and Raghu Meka.
Hardness of noise-free learning for two-hidden-layer neural networks.
In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
(Cited on page 43.)

References V

[17] Sitan Chen, Jerry Li, and Yuanzhi Li.
Learning (very) simple generative models is hard.
arXiv preprint arXiv:2205.16003, 2022.
(Cited on page 43.)

[18] 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 27.)

[19] Lenaic Chizat, Edouard Oyallon, and Francis Bach.
On lazy training in differentiable programming.
Advances in Neural Information Processing Systems, 32, 2019.
(Cited on page 50.)

[20] Lenaic Chizat, Edouard Oyallon, and Francis Bach.
On lazy training in differentiable programming.
arXiv preprint arXiv:1812.07956, 2019.
(Cited on pages 51, 65, and 66.)

References VI

[21] 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 27.)

[22] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis.
The complexity of constrained min-max optimization.
arXiv preprint arXiv:2009.09623, 2020.
(Cited on pages 9 and 10.)

[23] Elvis Dohmatob and Alberto Bietti.
On the (non-)robustness of two-layer neural networks in different learning regimes, 2022.
(Cited on page 46.)

[24] 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 27.)

References VII

[25] 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 27.)

[26] 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 page 52.)

[27] Bradley Efron.
Tweedie's formula and selection bias.
Journal of the American Statistical Association, 106(496):1602–1614, 2011.
(Cited on page 33.)

[28] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al.
Scaling rectified flow transformers for high-resolution image synthesis.
In *Forty-first International Conference on Machine Learning*, 2024.
(Cited on page 32.)

References VIII

[29] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar.
On the convergence theory of gradient-based model-agnostic meta-learning algorithms.
CoRR, abs/1908.10400, 2019.
(Cited on page 17.)

[30] 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 48.)

[31] 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 45.)

[32] 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 page 52.)

References IX

[33] 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 16.)

[34] Moritz Haas, Jin Xu, Volkan Cevher, and Leena Chennuru Vankadara.
$\boldsymbol{\mu}\mathbf{P}^{\mathbf{2}}$: Effective sharpness aware minimization requires layerwise perturbation scaling.
2024.
(Cited on page 60.)

[35] 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 page 52.)

[36] 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 44.)

References X

[37] Jonathan Ho and Tim Salimans.
Classifier-free diffusion guidance.
arXiv preprint arXiv:2207.12598, 2022.
(Cited on page 44.)

[38] 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 26.)

[39] 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 8 and 21.)

[40] 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 18 and 19.)

References XI

[41] 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 56.)

[42] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári.
Learning with a strong adversary.
arXiv preprint arXiv:1511.03034, 2015.
(Cited on page 45.)

[43] Aapo Hyvärinen.
Estimation of Non-Normalized Statistical Models by Score Matching.
Journal of Machine Learning Research, 6(24):695–709, 2005.
(Cited on page 28.)

[44] 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 pages 50, 51, 65, and 66.)

References XII

[45] 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 48.)

[46] 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 26.)

[47] 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 13, 70, and 74.)

[48] 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 31.)

References XIII

[49] Galina M Korpelevich.
The extragradient method for finding saddle points and other problems.
Matecon, 12:747–756, 1976.
(Cited on page 16.)

[50] 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 26.)

[51] 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 48.)

[52] 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 page 52.)

References XIV

[53] 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 50.)

[54] Joseph Lehec.
The langevin monte carlo algorithm in the non-smooth log-concave case.
arXiv preprint arXiv:2101.10695, 2021.
(Cited on page 27.)

[55] 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 73.)

[56] 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 50 and 52.)

References XV

[57] 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, 22(71):1–47, 2021.
(Cited on pages 51, 65, and 66.)

[58] Siwei Lyu.
Interpretation and generalization of score matching.
arXiv preprint arXiv:1205.2629, 2012.
(Cited on page 31.)

[59] 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 26.)

[60] 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 26.)

References XVI

[61] 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 16.)

[62] 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 page 52.)

[63] M. Mohri, A. Rostamizadeh, A. Talwalkar, and F. Bach.
Foundations of Machine Learning.
MIT Press, 2018.
(Cited on page 62.)

[64] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil.
A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach.
In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1497–1507. PMLR, 26–28 Aug 2020.
(Cited on page 17.)

References XVII

[65] Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto.
Why do learning rates transfer? reconciling optimization and scaling limits for deep learning.
In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
(Cited on page 60.)

[66] 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 73.)

[67] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang.
Semidefinite relaxations for certifying robustness to adversarial examples.
In *NeurIPS*, 2018.
(Cited on page 48.)

[68] 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 44.)

References XVIII

[69] R. Tyrrell Rockafellar.
Convex Analysis.
Princeton Univ. Press, Princeton, NJ, 1970.
(Cited on page 16.)

[70] R. Tyrrell Rockafellar.
Monotone operators and the proximal point algorithm.
SIAM Journal on Control and Optimization, 14(5):877–898, 1976.
(Cited on page 17.)

[71] 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 27.)

[72] 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 page 44.)

References XIX

[73] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller.
Deterministic policy gradient algorithms.
In *ICML*, 2014.
(Cited on pages 72 and 73.)

[74] 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 28.)

[75] 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 30.)

[76] 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 40 and 41.)

References XX

[77] Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher.
On feature learning in structured state space models.
2024.
(Cited on page 60.)

[78] 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 56.)

[79] Pascal Vincent.
A connection between score matching and denoising autoencoders.
Neural computation, 23(7):1661–1674, 2011.
(Cited on page 30.)

[80] Andre Wibisono.
Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem.
page 35.
(Cited on page 26.)

References XXI

[81] 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 56.)

[82] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao.
Tuning large neural networks via zero-shot hyperparameter transfer.
In *Advances in Neural Information Processing Systems*, pages 17084–17097, 2021.
(Cited on page 60.)

[83] Zhenyu Zhu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher.
Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization).
In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36094–36107. Curran Associates, Inc., 2022.
(Cited on pages 47, 56, 63, and 64.)

[84] Zhenyu Zhu, Francesco Locatello, and Volkan Cevher.
Sample complexity bounds for score-matching: Causal discovery and generative modeling.
In *Advances in Neural Information Processing Systems*, volume 36, 2023.
(Cited on pages 43 and 62.)

References XXII

[85] 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 16.)