Neural Networks: The Good, The Bad, The Ugly

Fanghui Liu (EPFL), Johan A.K. Suykens (KU Leuven), Volkan Cevher (EPFL)

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

ICASSP 2023, Rhodes Island, Greece
Acknowledgements

- LIONS group members (current & alumni): [https://lions.epfl.ch](https://lions.epfl.ch)

- EE-556 (Mathematics of Data): Course material

- Many talented faculty collaborators
  - Panayotis Mertikopoulos, Georgios Piliouras, Kfir Levy, Francis Bach, Joel Tropp, Madeleine Udell, Stephen Becker, Suvrit Sra, Mark Schmidt, Larry Carin, Michael Kapralov, Martin Jaggi, David Carlson, Adrian Weller, Adish Singla, Lorenzo Rosasco, Alessandro Rudi, Stefanie Jegelka, Panos Patrinos, Andreas Krause, Niao He, Bernhard Schoelkopf, Olivier Fercoq...

- Many talented collaborators
  - Francesco Locatello, Chris Russell, Matthaeus Kleindessner, Puya Latafat, Andreas Loukas, Yu-Guan Hsieh
Let’s start with what is really on everybody’s mind: GPT-4

- Previous GPTs: text $\Rightarrow$ text.
- GPT-4: allows text + image $\Rightarrow$ text.
A deep learning optimization problem in supervised learning

**Definition (Optimization formulation)**

The “deep-learning” problem with a neural network $h_x(a)$ is given by

$$x^* \in \arg \min_{x \in \mathcal{X}} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_x(a_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and $L$ is a loss function.

- A single hidden layer neural network with params $x := [X_1, X_2, \mu_1, \mu_2]$

$$h_x(a) := \sigma \left( X_2 \right) \left( \sigma \left( \left[ X_1 \right] a + \left[ \mu_1 \right] \right) + \left[ \mu_2 \right] \right)$$

hidden layer $=$ learned features
A deep learning optimization problem in supervised learning

Definition (Optimization formulation)

The “deep-learning” problem with a neural network $h_x(a)$ is given by

$$x^* \in \arg\min_{x \in \mathcal{X}} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_x(a_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints and $L$ is a loss function.

Some frequently used architectures

- Transformers with self-attention
- Recurrent neural networks
- Convolutional neural networks
- Multi layer perceptron...
Robustness issues in deep learning: Invisibility [81]
Robustness issues in deep learning: Acceleration

Hackers can trick a Tesla into accelerating by 50 miles per hour

A two inch piece of tape fooled the Tesla's cameras and made the car quickly and mistakenly speed up.

By Patrick Howell O'Neil

February 19, 2020

Robustness issues in deep learning: Injections

[Image of text with text boxes and arrows, showing examples of prompt injection attacks on a model called GPT-4.]

---

2 https://www.robustintelligence.com/blog-posts/prompt-injection-attack-on-gpt-4
Today: “Basic” robust machine learning

\[
\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \Phi(x, y)
\]

- A seemingly simple optimization formulation
- Critical in machine learning with many applications
  - Adversarial examples and training
  - Generative adversarial networks
  - Robust reinforcement learning
Warm up: Flexibility of the template

\[ \Phi^* = \min_{x \in X} \max_{y \in Y} \Phi(x, y) \quad (\text{argmin, argmax} \rightarrow x^*, y^*) \]
Warm up: Flexibility of the template

\[ \Phi^* = \min_{x \in X} \max_{y \in Y} \Phi(x, y) \quad (\text{argmin, argmax } \rightarrow x^*, y^*) \]

\[ f^* = \min_{x : x \in X} f(x) \quad (\text{argmin } \rightarrow x^*) \]
Warm up: Flexibility of the template

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

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

- (eula) In the sequel,
  - the set $\mathcal{X}$ is convex
  - all convergence characterizations are with feasible iterates $x^k \in \mathcal{X}$
  - $L$-smooth means $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \forall x, y \in \mathcal{X}$
  - $\nabla$ may refer to the generalized subdifferential
Warm up: Flexibility of the template

\[ \Phi^* = \min_{x \in X} \max_{y \in Y} \Phi(x, y) \quad (\text{argmin, argmax } \rightarrow x^*, y^*) \]

\[ f^* = \min_{x : x \in X} f(x) \quad (\text{argmin } \rightarrow x^*) \]

- (eula) In the sequel,
  - the set \( X \) is convex
  - all convergence characterizations are with feasible iterates \( x^k \in X \)
  - \( L \)-smooth means \( \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in X \)
  - \( \nabla \) may refer to the generalized subdifferential
Towards adversarial training for robustness

Adversarial Training

Let \( h_\mathbf{x} : \mathbb{R}^n \to \mathbb{R} \) be a model with parameters \( \mathbf{x} \) and let \( \{(a_i, b_i)\}_{i=1}^n \), with the data \( a_i \in \mathbb{R}^p \) and the labels \( b_i \). The problem of adversarial training is the following adversarial optimization problem

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

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

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

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

◦ Goal: Find \( x^* \) such that \( \nabla f(x^*) = 0 \).

Recall (Classification of critical points)

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

- \( \lambda_i > 0 \) for all \( i \) \( \Rightarrow \) \( \bar{x} \) is a local minimum
- \( \lambda_i < 0 \) for all \( i \) \( \Rightarrow \) \( \bar{x} \) is a local maximum
- \( \lambda_i > 0, \lambda_j < 0 \) for some \( i, j \) and \( \lambda_i \neq 0 \) for all \( i \) \( \Rightarrow \) \( \bar{x} \) is a saddle point
- Other cases \( \Rightarrow \) inconclusive
Solving the outer problem

Adversarial Training

Let \( h_x : \mathbb{R}^n \rightarrow \mathbb{R} \) be a model with parameters \( x \) and let \( \{(a_i, b_i)\}_{i=1}^n \), with \( a_i \in \mathbb{R}^p \) and \( b_i \) be the corresponding labels. The adversarial training optimization problem is given by

\[
\min_x \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x) := \frac{1}{n} \sum_{i=1}^n \left[ \max_{\delta : \|\delta\| \leq \epsilon} L(h_x (a_i + \delta), b_i) \right] \right\}.
\]

Note that \( L \) is not continuously differentiable due to ReLU, max-pooling, etc.
Solving the outer problem: Gradient computation

**Adversarial Training**

Let \( h_x : \mathbb{R}^p \to \mathbb{R} \) be a model with parameters \( x \) and let \( \{(a_i, b_i)\}_{i=1}^n \), with \( a_i \in \mathbb{R}^p \) and \( b_i \) be the corresponding labels. The adversarial training optimization problem is given by

\[
\min_x \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x) := \frac{1}{n} \sum_{i=1}^n \left[ \max_{\delta : \|\delta\| \leq \epsilon} L(h_x(a_i + \delta), b_i) \right] =: f_i(x) \right\}.
\]

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

**Question**

How can we compute the following stochastic gradient (i.e., \( \mathbb{E}_i \nabla_x f_i(x) = \nabla_x f_i(x) \) for \( i \sim \text{Uniform}\{1, \ldots, n\} \)):

\[
\nabla_x f_i(x) := \nabla_x \left( \max_{\delta : \|\delta\| \leq \epsilon} L(h_x(a_i + \delta), b_i) \right) ?
\]

- **Challenge:** It involves differentiating with respect to a maximization.
Basic questions on solution concepts

○ Consider the finite sum setting

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

○ Goal: Find \( x^* \) such that \( \nabla f(x^*) = 0 \).
Basic questions on solution concepts

◦ Consider the finite sum setting

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

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

1. Does SGD converge with probability 1? [10, 75, 55, 62]

2. Does SGD avoid non-minimum points with probability 1? [51, 29, 62]

3. How fast does SGD converge to local minimizers? [29, 30, 62]

4. Can SGD converge to global minimizers? [41, 43, 32, 84, 35, 70, 53, 22, 90, 46, 76]
Q1: Does SGD converge?

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

1. GD converges from any initialization with constant step-size and full gradients

2. With probability 1, (P)SGD does not converge with constant step-size $\alpha$ [10, 75]

3. With probability 1, SGD converges with vanishing step-size if $x^k$ is bounded with probability 1 [55, 10]

**Boundedness is not required (Theorem 1 of [62])**

Assume Lipschitzness, sublevel regularity, $\mathbb{E}\|g\|^q \leq \sigma^q$ and $\sum_k \alpha_k^{1+q/2} < \infty$ ($q \geq 2$). Then, $x^k$ converges with probability 1.
Q2: Does SGD avoid saddle points?

- SGD avoids strict saddles ($\lambda_{\min}(\nabla^2 f(\bar{x})) < 0$)
  1. GD avoids strict saddles from almost all initializations [51]
  2. With probability $1 - \zeta$, PSGD with constant $\alpha$ escapes strict saddles after $\Omega\left(\log(1/\zeta)/\alpha^2\right)$ iterations [29]
    - However, SGD does not converge with constant $\alpha$
    - We cannot take $\zeta = 0$

---

**SGD avoids traps almost surely (Theorem 3 of [62])**

Assume bounded uniformly exciting noise and $\alpha_k = O\left(\frac{1}{k^{1/\kappa}}\right)$ for $\kappa \in (0, 1]$. Then, SGD avoids strict saddles from any initial condition with probability 1.
Q3: How fast does SGD converge to local minimizers?

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

1. SGD with constant \( \alpha \) can obtain objective value \( \epsilon \)-close to a Hurwicz minimizer in \( \mathcal{O}(1/\epsilon^2) \)-iterations [29, 30]
   - However, SGD does not converge with constant \( \alpha \)
   - Need averaging which is problematic in non-convex optimization

Using a vanishing step-size helps! (Theorem 4 of [62])
Using \( \alpha_k = \mathcal{O}\left(\frac{1}{k}\right) \), SGD enjoys a \( \mathcal{O}\left(\frac{1}{k}\right) \) convergence rate in objective value.
Using $1/k$ step-size decrease helps in practice

○ ResNet training at different cool-down cut-offs
## Basic results on adaptive algorithms

<table>
<thead>
<tr>
<th></th>
<th>GD/SGD</th>
<th>Accelerated GD/SGD</th>
<th>AdaGrad</th>
<th>AcceleGrad/UniXgrad</th>
<th>Adam/AMSGrad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex, stochastic</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^4$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^{5,6}$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^7$</td>
</tr>
<tr>
<td>Convex, deterministic, $L$-smooth</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{k^2}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^5$</td>
<td>$\mathcal{O}\left(\frac{1}{k^2}\right)^{5,6}$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^8$</td>
</tr>
<tr>
<td>Nonconvex, stochastic, $L$-smooth</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^9$</td>
<td>?</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^{10}$</td>
</tr>
<tr>
<td>Nonconvex, deterministic, $L$-smooth</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^9$</td>
<td>?</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^8$</td>
</tr>
</tbody>
</table>

4 Duchi, Hazan, Singer, Adaptive subgradient methods for online learning and stochastic optimization, JMLR, 2011.
5 Levy, Yurtsever, Cevher, Online adaptive methods, universality and acceleration, NeurIPS 2018.
7 Reddi, Kale, Kumar, On the convergence of adam and beyond, ICLR, 2018.
8 Alacaoglu, Malitsky, Mertikopoulos, Cevher, A new regret analysis for Adam-type algorithms, ICML 2020.
13 Chen, Liu, Sun, Hong, On the convergence of a class of adam-type algorithms for non-convex optimization, ICLR 2018.
Danskin’s Theorem (1966): How do we compute the gradient?

**Theorem ([18])**

Let $S$ be compact set, $\Phi : \mathbb{R}^p \times S$ be continuous such that $\Phi(\cdot, y)$ is differentiable for all $y \in S$, and $\nabla_x \Phi(x, y)$ be continuous on $\mathbb{R}^p \times S$. Define

$$f(x) := \max_{y \in S} \Phi(x, y), \quad S^*(x) := \arg \max_{y \in S} \Phi(x, y).$$

Let $\gamma \in \mathbb{R}^p$, and $\|\gamma\|_2 = 1$. The directional derivative $D_\gamma f(\bar{x})$ of $f$ in the direction $\gamma$ at $\bar{x}$ is given by

$$D_\gamma f(\bar{x}) = \max_{y \in S^*(\bar{x})} \langle \gamma, \nabla_x \Phi(\bar{x}, y) \rangle.$$

An immediate consequence

If $\delta^* \in \arg \max_{\delta : \|\delta\| \leq \epsilon} L(h_x (a_i + \delta), b_i)$ is unique, then we have

$$\nabla_x f_i(x) = \nabla_x L(h_x (a_i + \delta^*), b_i).$$
Optimized perturbations are typically not unique!

Figure: (left) Pairwise $\ell_2$-distances between “optimized” perturbations with different initializations are bounded away from zero. (right) The losses of multiple perturbations on the same sample concentrate around a value much larger than the clean loss.
Theoretical foundations

\[ \nabla_x \Phi(x, \delta^*) \quad \nabla_x f(x) \quad \text{descent direction} \quad [58] \]

Published as a conference paper at ICLR 2018

Towards Deep Learning Models Resistant to Adversarial Attacks

AleksanderMadry, Aleksandar Makelov, Ludwig Schmidt, DimitrisTsipras, AdrianVladu
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu

\[ x^k + D(f, x^k) \]
Theoretical foundations

unique $\delta^*$

\[
\nabla_x \Phi(x, \delta^*)
\]

non-unique $\delta^*$

\[
\nabla_x f(x)
\]

descent direction [58]

Published as a conference paper at ICLR 2018

TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{madry,amakelov,ludwigs,tsipras,avladu}@mit.edu
A counterexample

\[ f(x) := \max_{\delta \in [-1,1]} x\delta = |x|. \]

- We have \( S := [-1, 1] \) and \( \Phi(x, \delta) = x\delta. \)
- At \( x = 0 \), we have \( S^*(0) = [-1, 1] \).
- We can choose \( \delta = 1 \in S^*(0) \): \( \Phi(x, 1) = x. \)
A counterexample

\[ f(x) := \max_{\delta \in [-1,1]} x\delta = |x|. \]

- We have \( S := [-1, 1] \) and \( \Phi(x, \delta) = x\delta \).
- At \( x = 0 \), we have \( S^*(0) = [-1, 1] \).
- We can choose \( \delta = 1 \in S^*(0) \): \( \Phi(x, 1) = x \).
  - \( -\nabla_x \Phi(0, 1) = -1 \neq 0 \).
  - Is \(-1\) a descent direction at \( x = 0 \)?
Our understanding [Latorre, Krawczuk, Dadi, Pethick, Cevher, ICLR (2023)]

- The corollary in [58] is false (it is subtle!).
- We constructed a counter example & proposed an alternative way (DDi) of computing “the gradient”:

\[
\begin{align*}
\text{unique } \delta^* & \quad \text{non-unique } \delta^* \\
\nabla_x \Phi(x, \delta^*) & \quad \nabla_x f(x) \quad \text{could be ascent direction!}
\end{align*}
\]

**Figure:** Left and middle pane: comparison DDi and PGD ([58]) on a synthetic problem. Right pane: DDi vs PGD on CIFAR10.
Comparison with the state-of-the-art

**Figure:** (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN,ReLU,momentum).
Comparison with the state-of-the-art

**Figure:** (left) PGD vs DDi on CIFAR10, in a setting covered by theory. (right) An ablation testing the effect of adding back the elements not covered by theory (BN, ReLU, momentum).

DDi + Graduate Student Descent may improve things?
Learning without concentration

- We can minimize $W_1 (\hat{\mu}_n, h_x \# p_\Omega)$ with respect to $x$.
- Figure: Empirical distribution (blue), $\hat{\mu}_n = \sum_{i=1}^n \delta_i$

A plug-in empirical estimator

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

$$W_1 (\mu^\natural, h_x \# p_\Omega) \leq W_1 (\mu^\natural, \hat{\mu}_n) + W_1 (\hat{\mu}_n, h_x \# p_\Omega),$$

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

Theorem (Slow convergence of empirical measures in 1-Wasserstein [78, 23])

Let $\mu^\natural$ be a measure defined on $\mathbb{R}^P$ and let $\hat{\mu}_n$ be its empirical measure. Then the $\hat{\mu}_n$ converges, in the worst case, at the following rate,

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

Remarks:

- Using an empirical estimator in high-dimensions is terrible in the worst case.
- However, it does not directly say that $W_1 (\mu^\natural, h_x \# p_\Omega)$ will be large.
- So we can still proceed and hope our parameterization interpolates harmlessly.
Duality of 1-Wasserstein

- How do we get a sub-gradient of $W_1(\hat{\mu}_n, h_x\# p_\Omega)$ with respect to $x$?

**Theorem (Kantorovich-Rubinstein duality)**

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

**Remark:**
- $d$ is the “dual” variable. In the literature, it is commonly referred to as the “discriminator.”

**Inner product is an expectation**

$$\langle d, \mu \rangle = \int dd\mu = \int d(a)d\mu(a) = E_{a \sim \mu}[d(a)]. \tag{4}$$

**Kantorovich-Rubinstein duality applied to our objective**

$$W_1(\hat{\mu}_n, h_x\# \omega) = \sup \left\{ E_{a \sim \hat{\mu}_n}[d(a)] - E_{a \sim h_x\# \omega}[d(a)] : d \text{ is 1-Lipschitz} \right\} \tag{5}$$
Another minimax example: Generative adversarial networks (GANs)

- Ingredients:
  - fixed noise distribution \( p_{\Omega} \) (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 [2]

Define a parameterized function \( d_y(a) \), where \( y \in \mathcal{Y} \) such that \( d_y(a) \) is 1-Lipschitz. In this case, the Wasserstein GAN training problem is given by

\[
\min_{x \in \mathcal{X}} \left( \max_{y \in \mathcal{Y}} \mathbb{E}_{a \sim \hat{\mu}_n} [d_y(a)] - \mathbb{E}_{\omega \sim p_{\Omega}} [d_y(h_x(\omega))] \right). \tag{6}
\]

This problem is already captured by the template \( \min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \Phi(x, y) \). Note that the original problem is a direct non-smooth minimization problem and the Rubinstein-Kantarovic duality results in the minimax template.

Remarks:
- Cannot solve in a manner similar to adversarial training a la Danskin. Need a direct approach.
- Scalability, mode collapse, catastrophic forgetting. Heuristics galore!
- Enforce Lipschitz constraint weight clipping, gradient penalty, spectral normalization [2, 34, 63].
Abstract minmax formulation

Minimax formulation

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

where

- \(\Phi\) is differentiable and nonconvex in \(x\) and nonconcave in \(y\),
- 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?
Solving the minimax problem: Solution concepts

- Consider the unconstrained setting:
  \[ \Phi^* = \min_x \max_y \Phi(x, y) \]

- **Goal**: Find an LNE point \((x^*, y^*)\).

**Definition (Local Nash Equilibrium)**

A pure strategy \((x^*, y^*)\) is called a local Nash equilibrium if

\[ \Phi(x^*, y) \leq \Phi(x^*, y^*) \leq \Phi(x, y^*) \]  

(LNE)

for all \(x\) and \(y\) within some neighborhood of \(x^*\) and \(y^*\), i.e., \(\|x - x^*\| \leq \epsilon\) and \(\|y - y^*\| \leq \epsilon\) for some \(\epsilon > 0\).

**Necessary conditions**

Through a Taylor expansion around \(x^*\) and \(y^*\) one can show that a LNE implies

\[ \nabla_x \Phi(x, y), -\nabla_y \Phi(x, y) = 0; \]

\[ \nabla_{xx} \Phi(x, y), -\nabla_{yy} \Phi(x, y) \succeq 0. \]
Abstract minmax formulation

Minimax formulation

\[
\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \Phi(x, y),
\] (8)

where
- \( \Phi \) is differentiable and nonconvex in \( x \) and nonconcave in \( y \),
- 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 algorithms converge?

A buffet of negative results [19]

“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.”

Basic algorithms for minimax

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

![Diagram](image)

**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) \[74, 33\]
  - Extra-gradient (EG) \[48\]
  - Optimistic GDA (OGDA) \[88, 59\]
  - Reflected-Forward-Backward-Splitting (RFBS) \[14\]

- EG and OGDA are approximations of the PPM
  - \( z^{k+1} = z^k - \alpha V(z^k) \).
  - \( z^{k+1} = z^k - \alpha V(z^{k+1}) \).
  - \( z^{k+1} = z^k - \alpha V(z^k - \alpha V(z^{k-1})) \).
  - \( z^{k+1} = z^k - \alpha [2V(z^k) - V(z^{k-1})] \).
  - \( z^{k+1} = z^k - \alpha (2z^k - z^{k-1}) \).
Where do the algorithms converge?

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

- Given $V(z)$, define stochastic estimates of $V(z, \zeta) = V(z) + U(z, \zeta)$, where
  - $U(z, \zeta)$ is a bias term,
  - We often have unbiasedness: $EU(z, \zeta) = 0$,
  - The bias term can have bounded moments,
  - We often have bounded variance: $P(\|U(z, \zeta)\| \geq t) \leq 2 \exp(-\frac{t^2}{2\sigma^2})$ for $\sigma > 0$.

- An abstract template for generalized Robbins-Monro schemes, dubbed as $\mathcal{A}$:
  \[ z^{k+1} = z^k - \alpha_k V(z^k, \zeta^k). \]

The dessert section in the buffet of negative results: [39]

1. Bounded trajectories of $\mathcal{A}$ always converge to an internally chain-transitive (ICT) set.
2. Trajectories of $\mathcal{A}$ may converge with arbitrarily high probability to spurious attractors that contain no critical point of $\Phi$. 
Minimax is more difficult than just optimization [39]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [11].
  - For optimization, \( \{ \text{attracting ICT} \} \equiv \{ \text{solutions} \} \)
  - For minimax, \( \{ \text{attracting ICT} \} \equiv \{ \text{solutions} \} \cup \{ \text{spurious sets} \} \)

- “Almost” bilinear ≠ bilinear:
  \[
  \Phi(x, y) = xy + \epsilon \phi(x), \quad \phi(x) = \frac{1}{2} x^2 - \frac{1}{4} x^4
  \]

- The “forsaken” solutions:
  \[
  \Phi(y, x) = y(x-0.5)+\phi(y)-\phi(x), \quad \phi(u) = \frac{1}{4} u^2 - \frac{1}{2} u^4 + \frac{1}{6} u^6
  \]
Minimax is more difficult than just optimization [39]

- Internally chain-transitive (ICT) sets characterize the convergence of dynamical systems [11].
  - For optimization, \( \{ \text{attracting ICT} \} \equiv \{ \text{solutions} \} \)
  - For minimax, \( \{ \text{attracting ICT} \} \equiv \{ \text{solutions} \} \cup \{ \text{spurious sets} \} \)

- “Almost” bilinear \( \neq \) bilinear:
  \[
  \Phi(x, y) = xy + \epsilon \phi(x), \quad \phi(x) = \frac{1}{2} x^2 - \frac{1}{4} x^4
  \]

- The “forsaken” solutions:
  \[
  \Phi(y, x) = y(x-0.5)+\phi(y)-\phi(x), \quad \phi(u) = \frac{1}{4} u^2 - \frac{1}{2} u^4 + \frac{1}{6} u^6
  \]
When do the algorithms converge?

Assumption (weak Minty variational inequality)

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

$$\langle V(z), z - z^* \rangle \geq \rho \|V(z)\|^2, \quad \text{for all } z \in \mathbb{R}^n. \quad (9)$$

- A variant EG+ converges when $\rho > -\frac{1}{8L}$
  - Diakonikolas, Daskalakis, Jordan, AISTATS 2021.
- It still cannot handle the examples of [39].

- Complete picture under weak MVI (ICLR’22 and ’23)
  - Pethick, Lalafat, Patrinos, Fercoq, and Cevher.
  - constrained and regularized settings with $\rho > -\frac{1}{2L}$
  - matching lower bounds
  - stochastic variants handling the examples of [39]
  - adaptive variants handling the examples of [39]

Figure: The operator $V(z)$ is allowed to point away from the solution by some amount when $\rho$ is negative.
Solving stochastic weak MVIs without increasing batch size

\[ \tilde{z}^k = z^k - \gamma V(z^k) \quad \text{(EG+)} \]

\[ z^{k+1} = z^k - \alpha \gamma V(\tilde{z}^k) \]

- Extragradient+
  - the smaller \( \alpha \in (0, 1) \), the better \[20]\n  - \( \rho > -\frac{1}{2L} \) \[72]\n
- Stochastic extragradient+
  - converges for affine \( V \), \( \rho > (1 - \alpha_k) \gamma / 2 \) \[71]\n  - may not converge for monotone setting

- Bias corrected stochastic extragradient+
  - a.s. convergence with \( \rho > -\frac{1}{2L} \alpha_k \to 0 \)
  - alternation allows even bigger step-sizes

- constrained and regularized settings w/ prox
Solving stochastic weak MVIs without increasing batch size

\[ \bar{z}^k = z^k - \gamma V(z^k) \quad \text{(EG+)} \]
\[ z^{k+1} = z^k - \alpha \gamma V(\bar{z}^k) \]

\[ \bar{z}^k = z^k - \beta_k \gamma V(z^k, \zeta^k) \quad \text{(SEG)} \]
\[ z^{k+1} = z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k) \]

- Extragradient+
  - the smaller \( \alpha \in (0, 1) \), the better [20]
  - \( \rho > -\frac{1}{2L} \) [72]

- Stochastic extragradient
  - \( \beta_k > \alpha_k \): two time scale
  - \( \beta_k \propto 1/k \) and \( \alpha_k \propto 1/k \) for \( \rho = 0 \) [40]
Solving stochastic weak MVIs without increasing batch size

\[
\begin{align*}
\tilde{z}^k &= z^k - \gamma V(z^k) \\
z^{k+1} &= z^k - \alpha \gamma V(\tilde{z}^k)
\end{align*}
\]

(EG+)

- Extragradient+
  - the smaller \( \alpha \in (0, 1) \), the better \([20]\)
  - \( \rho > -\frac{1}{2L} \) \([72]\)

\[
\begin{align*}
\bar{z}^k &= z^k - \beta_k \gamma V(z^k, \zeta^k) \\
z^{k+1} &= z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\end{align*}
\]

(SEG)

- Stochastic extragradient
  - \( \beta_k > \alpha_k \): two time scale
  - \( \beta_k \propto 1/k \) and \( \alpha_k \propto 1/k \) for \( \rho = 0 \) \([40]\)

\[
\begin{align*}
\bar{z}^k &= z^k - \gamma V(z^k, \zeta^k) \\
z^{k+1} &= z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\end{align*}
\]

(SEG+)

- Stochastic extragradient+
  - converges for affine \( V \), \( \rho > (1 - \alpha_k)\gamma/2 \) \([71]\)
  - may not converge for monotone setting
Solving stochastic weak MVIs without increasing batch size

\[
\tilde{z}^k = z^k - \gamma V(z^k) \quad \text{(EG+)}
\]

\[
z^{k+1} = z^k - \alpha \gamma V(\tilde{z}^k)
\]

\[
\bar{z}^k = z^k - \beta_k \gamma V(z^k, \zeta^k) \quad \text{(SEG)}
\]

\[
z^{k+1} = z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\]

\[
\bar{z}^k = z^k - \gamma V(z^k, \zeta^k) \quad \text{(SEG+)}
\]

\[
z^{k+1} = z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\]

\[H(z, \zeta) \overset{\text{def}}{=} z - \gamma V(z, \zeta)\]

\[
\tilde{z}^k = H(z^k, \zeta^k) + (1 - \alpha_k) \left( \bar{z}^{k-1} - H(z^{k-1}, \zeta^k) \right)
\]

\[
z^{k+1} = z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\]

- Extragradient+
  - the smaller \( \alpha \in (0, 1) \), the better \[20\]
  - \( \rho > -\frac{1}{2L} \) \[72\]

- Stochastic extragradient
  - \( \beta_k > \alpha_k \): two time scale
  - \( \beta_k \propto 1/k \) and \( \alpha_k \propto 1/k \) for \( \rho = 0 \) \[40\]

- Stochastic extragradient+
  - converges for affine \( V \), \( \rho > (1 - \alpha_k)\gamma/2 \) \[71\]
  - may not converge for monotone setting

- Bias corrected stochastic extragradient+
  - a.s. convergence with \( \rho > -\frac{1}{2L} \) \( \omega/\alpha_k \to 0 \)
  - alternation allows even bigger step-sizes \[71\]
Solving stochastic weak MVIs without increasing batch size

\[
\tilde{z}^k = z^k - \gamma V(z^k) \quad \text{(EG+)}
\]
\[
z^{k+1} = z^k - \alpha \gamma V(\tilde{z}^k)
\]

\[
\bar{z}^k = z^k - \beta_k \gamma V(z^k, \zeta^k) \quad \text{(SEG)}
\]
\[
z^{k+1} = z^k - \alpha_k \gamma V(\bar{z}^k, \bar{\zeta}^k)
\]

\[
\tilde{z}^k = z^k - \gamma V(z^k, \zeta^k) \quad \text{(SEG+)}
\]
\[
z^{k+1} = z^k - \alpha_k \gamma V(z^k, \bar{\zeta}^k)
\]

- Extragradient+
  - the smaller $\alpha \in (0, 1)$, the better [20]
  - $\rho > -\frac{1}{2L}$ [72]

- Stochastic extragradient
  - $\beta_k > \alpha_k$: two time scale
  - $\beta_k \propto 1/k$ and $\alpha_k \propto 1/k$ for $\rho = 0$ [40]

- Stochastic extragradient+
  - converges for affine $V$, $\rho > (1 - \alpha_k)\gamma/2$ [71]
  - may not converge for monotone setting

- Bias corrected stochastic extragradient+
  - a.s. convergence with $\rho > -\frac{1}{2L} \omega/\alpha_k \rightarrow 0$
  - alternation allows even bigger step-sizes
  - constrained and regularized settings w/ prox
GANs with SEG+

Figure: A performance comparison of GAN training by Adam, EG with stochastic gradients, and SEG+.
An alternative proposal: From pure to mixed Nash equilibrium (NE)

- Rethinking minimax problem as pure strategy game formulation
  \[
  \min_{x \in X} \max_{y \in Y} \Phi(x, y)
  \]

- A corresponding mixed strategy formulation
  \[
  \min_{p \in \mathcal{M}(X)} \max_{q \in \mathcal{M}(Y)} \mathbb{E}_{x \sim p} \mathbb{E}_{y \sim q} \left[ \Phi(x, y) \right]
  \]

  \[\mathcal{M}(Z) := \{ \text{all randomized strategies on } Z \}\]
GAN training as infinite dimensional matrix games

- A different way of looking at GAN objective
  - \langle p \rangle_h := \int h \, dp \text{ for a measure } p \text{ and function } h \tag{Riesz representation}
  - the linear operator $G$ and its adjoint $G^\dagger$:
    \[
    (Gq)(x) := \mathbb{E}_{y \sim q} [\Phi(x, y)] \\
    (G^\dagger p)(y) := \mathbb{E}_{x \sim p} [\Phi(x, y)],
    \]

- Mixed NE formulation $\simeq$ finite two-player games
  \[
  \min_{p \in \mathcal{M}(\mathcal{X})} \max_{q \in \mathcal{M}(\mathcal{Y})} \mathbb{E}_{x \sim p} \mathbb{E}_{y \sim q} [\Phi(x, y)]
  \]
  $\Uparrow$
  \[
  \min_{p \in \mathcal{M}(\mathcal{X})} \max_{q \in \mathcal{M}(\mathcal{Y})} \langle p \rangle Gq
  \]

- If $\mathcal{X}$ and $\mathcal{Y}$ are finite $\Rightarrow$ mirror descent
- There is a way to solve this *infinite* dimensional problem: Mirror descent + Langevin dynamics \[38\]

\[\text{Tutorial at ICASSP 2023} \mid \text{fanghui.liu,volkan.cevher}@epfl.ch\]
Escaping traps with the mixed-NE concept

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

---

Take home messages

- Even the simplified view of robust & adversarial ML is challenging
- \min - \max\text{-type has spurious attractors with no equivalent concept in } \min\text{-type}
- Not all step-size schedules are considered in our work: Possible to "converge" under some settings
- Other successful attempts consider "mixed Nash" concepts
- Promising new direction: Higher-order adaptive methods

Take home messages

- Even the simplified view of robust & adversarial ML is challenging
- min-max-type has spurious attractors with no equivalent concept in min-type
- Not all step-size schedules are considered in our work: Possible to “converge” under some settings
- Other successful attempts\(^1\) consider “mixed Nash” concepts\(^2\)

- Promising new direction: Higher-order adaptive methods\(^3\)

---


The mystery in deep learning

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*  
Massachusetts Institute of Technology  
chiyuan@mit.edu

Samy Bengio  
Google Brain  
bengio@google.com

Moritz Hardt  
Google Brain  
mrtz@google.com

Benjamin Recht†  
University of California, Berkeley  
brecht@berkeley.edu

Oriol Vinyals  
Google DeepMind  
vinyals@google.com

ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks alm.

A gap between theory and practice

- In practice, simple algorithms like SGD can train neural networks to zero error and achieve low test error.
- This happens even for large and complex neural network architectures.
- Complexity measures like the Rademacher complexity suggest the opposite behaviour (overfitting)
Q4: Can SGD converge to global minimizers?

- A few phenomena about neural networks [85]:
  - Deep neural networks can fit random labels
  - First-order methods can find global minimizers

Figure: DNN Training curves on CIFAR10, from [85]
Q4: Can SGD converge to global minimizers?

- A few phenomena about neural networks [85]:
  - Deep neural networks can fit random labels
  - First-order methods can find global minimizers

- **Overparametrization** can explain these mysteries!

**Overparametrization**

Number of parameters $\gg$ number of training data.
GD finds global minimizers of overparametrized networks

\[ h_x(a) := X_2 \sigma \left( X_1 a + \mu_1 \right) + \mu_2 \]

**Theorem (Linear convergence of Gradient Descent [22])**

- \( f(a; X_1, X_2) \): 1-hidden-layer network with width \( m \), hidden layer weights \( X_1 \), output layer weights \( X_2 \) and ReLU activation.
- \( m = \Omega(\frac{n^6}{\delta^2}) \) where \( n \) = number of samples.
- \( X_1^0 \) is initialized with a normal distribution, \( X_2^0 \sim \text{Unif}[-1, 1]^m \).
- Stepsize \( \eta = O(n^{-2}) \).

With probability at least \( 1 - \delta \), for the empirical risk \( R_n \) will converge to zero with a geometric rate of \( (1 - \eta) \).
Overparametrization is an active area of research

<table>
<thead>
<tr>
<th>Reference</th>
<th>Number of parameters</th>
<th>Depth $d$</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[41]</td>
<td>$\tilde{\Omega}(n)$</td>
<td>1, 2</td>
<td>Existence of zero error</td>
</tr>
<tr>
<td>[84, 70]</td>
<td>$\tilde{\Omega}(n)$</td>
<td>Any $d$</td>
<td>Existence of zero error</td>
</tr>
<tr>
<td>[53]</td>
<td>$\tilde{\Omega}(\text{poly}(n))$</td>
<td>1</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[22]</td>
<td>$\tilde{\Omega}(n^6)$</td>
<td>1</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[1, 89]</td>
<td>$\tilde{\Omega}(\text{poly}(n, d))$</td>
<td>Any $d$</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[21]</td>
<td>$\tilde{\Omega}(n^8 2^O(d))$</td>
<td>Any $d$</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[90]</td>
<td>$\tilde{\Omega}(n^8 d^{12})$</td>
<td>Any $d$</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[46]</td>
<td>$\tilde{\Omega}(n)$ (Training last layer)</td>
<td>Any $d$</td>
<td>(S)GD global convergence</td>
</tr>
<tr>
<td>[77]</td>
<td>$\tilde{\Omega}(n^{\frac{3}{2}})$ (Training all layers)</td>
<td>1</td>
<td>(S)GD global convergence</td>
</tr>
</tbody>
</table>

Table: Summary of results on overparametrization. Minimum number of parameters required as a function of data size $n$ and depth $d$. The result is classified either as *Existence* i.e., there exists a neural network achieving zero error on the data, or *(S)GD global convergence* i.e., (S)GD converges to zero training error, a much stronger condition.
It is time for the short break!
The role of over-parameterization in machine learning

- Robustness and generalization
- The role of over-parameterization
- Motivation: initialization

Deep learning theory

Good, bad, ugly
Over-parameterization: more parameters than training data

- MLP: $< 1$ million parameters
- ResNet-152: 60.3 million parameters
- Transformer: 340 million parameters
- GPT-2: 1.5 billion parameters
- GPT-3, Chat-GPT: 175 billion parameters
- GPT-4

Timeline:
- AlexNet: before 2012
- 2012: MLP
- 2015: ResNet-152
- 2017: Transformer
- 2019: GPT-2
- 2020: GPT-3, Chat-GPT
- 2022: GPT-4

Contact: {fanghui.liu, volkan.cevher}@epfl.ch
Over-parameterization: more parameters than training data

**Figure:** Larger models make increasingly efficient use of in-context information: source from Open AI.
Recall DNNs: the good in fitting ...

- A gap between theory and practice:
  - DNNs can fit random labels
  - SGD: zero training error and low test error

Figure: DNN Training curves on CIFAR10, from [85]
Recall DNNs: the bad in **robustness**...

(a) Invisibility [81]

(b) Stop sign classified as 45 mph sign [26]
Recall DNNs: the bad in robustness...

(a) Invisibility [81]  
(b) Stop sign classified as 45 mph sign [26]

the ugly in over-parameterization?
A toy example: curve fitting
A toy example: curve fitting

(a) under-fitting
A toy example: curve fitting

(a) under-fitting

(b) sweet spot

(c) overfitting

(d) benign overfitting

Figure: Test performance on curve fitting: source from Open AI.

Tutorial at ICASSP 2023 | {fanghui.liu,volkan.cevher}@epfl.ch
A toy example: curve fitting

(a) under-fitting  
(b) sweet spot  
(c) overfitting
A toy example: curve fitting

Figure: Test performance on curve fitting: source from Open AI.
Benign overfitting and double descent

- A bit more on **benign overfitting** [5, 15, 27]:
  - model is very complex
  - perfectly fit noisy data and generalize well
Benign overfitting and double descent

- A bit more on benign overfitting [5, 15, 27]:
  - model is very complex
  - perfectly fit noisy data and generalize well

![Diagram](image)

**Figure:** classical learning theory vs. double descent: source from [8].
Machine learning algorithms

- Linear non-separable
- Linear separable

Feature mapping: complex in low dimensions → simple in higher dimensions

Separating hyperplane
Feature mapping: from kernel methods to neural networks

- Kernel Methods
  - data-independent

- Neural Networks
  - data-dependent

Neural tangent kernel (NTK) [44]

feature mapping
Feature mapping: from kernel methods to neural networks

Kernel Methods
- data-independent

Neural Networks
- data-dependent

Neural tangent kernel (NTK) [44]

feature mapping

\[ k(a, a') = \langle \phi(a), \phi(a') \rangle_H \]
Function space: from kernel methods to neural networks

- Kernel Methods
  - reproducing kernel Hilbert space (RKHS)
- Neural Networks
  - Neural tangent kernel (NTK)
  - e.g., Hölder space, Besov space

Curse of dimensionality [3, 83, 13]

efficiently approximate non-smooth functions?
NN architecture

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

\[ h^{(l)}(\mathbf{a}) = \sigma \left( \mathbf{X}_l \right) \underbrace{\left[ \begin{array}{c} \mathbf{h}^{(l-1)}(\mathbf{a}) \end{array} \right]}_{\text{input features}} \underbrace{\mathbf{X}_l}_{\text{weight}} \underbrace{\mathbf{X}_l}_{\text{input features}}, \]

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

- Elements of NN architectures we will discuss in the sequel:
  - 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: \( \mathbf{X}_1 \sim \mathcal{N}(0, \beta^2_1), \mathbf{X}_L \sim \mathcal{N}(0, \beta^2_L), \mathbf{X}_l \sim \mathcal{N}(0, \beta^2) \) for \( l = 2, 3, \cdots, L - 1 \) (weights).
  - Activation function ReLU: \( \sigma(\cdot) = \max(\cdot, 0) : \mathbb{R} \rightarrow \mathbb{R} \).
  - Without loss of generality, we will avoid the bias variables in the sequel.
### Table: Some commonly used initializations in neural networks.

<table>
<thead>
<tr>
<th>Initialization name</th>
<th>$\beta_1^2$</th>
<th>$\beta_2^2$</th>
<th>$\beta_L^2$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeCun [50]</td>
<td>$\frac{1}{p}$</td>
<td>$\frac{1}{m}$</td>
<td>$\frac{1}{m}$</td>
<td>1</td>
</tr>
<tr>
<td>He [37]</td>
<td>$\frac{2}{p}$</td>
<td>$\frac{2}{m}$</td>
<td>$\frac{2}{m}$</td>
<td>1</td>
</tr>
<tr>
<td>NTK [1]</td>
<td>$\frac{2}{m}$</td>
<td>$\frac{2}{m}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Xavier [31]</td>
<td>$\frac{2}{m+p}$</td>
<td>$\frac{1}{m}$</td>
<td>$\frac{2}{m+1}$</td>
<td>1</td>
</tr>
<tr>
<td>Mean-field [61]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$m$</td>
</tr>
<tr>
<td>E et al. [25]</td>
<td>1</td>
<td>1</td>
<td>$\beta_C^2$</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

Define an $L$-layer fully-connected ReLU NN via ($L$-Layer NN). After training time $t$, as $m \to \infty$, if the following condition holds

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

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

**Remarks:**

- In this regime, training $h$ and $h_0$ is equivalent if taking Taylor expansion.
- Which conditions allow for lazy training to occur?
Lazy training: a consequence of overparametrization or scaling?

Theorem (Lazy training for two-layer ReLU networks [16], modified version)

Two layer networks $h(a, \{x, v\}) : a \mapsto \alpha(m) \sum_{j=1}^{m} v_j \text{ReLU}(x_j^\top a)$ with Gaussian initialization $v_i, x_i \sim \mathcal{N}(0, \beta^2)$ will fall within the lazy regime as long as

$$\lim_{m \to \infty} m \beta = \infty.$$

Remarks:
- The loss changes a lot but the neural network output changes little.
- Other conditions for deep neural networks can be found here [16, 7].
Lazy training regime: visualization

\[ \mathcal{F}_{NN,m} = \left\{ h_m(a; \{x, v\}) = \sum_{i=1}^{m} v_i \max (\langle x_i, a \rangle, 0) : v_i \in \mathbb{R}, x_i \in \mathbb{R}^d \right\} \]

**Figure:** Training dynamics of two-layer ReLU NNs under different initializations [44, 17, 57].
Lazy training regime: visualization

\[ F_{\text{NN}, m} = \left\{ h_m(\mathbf{a}; \{ \mathbf{x}, \mathbf{v} \}) = \sum_{i=1}^{m} v_i \max (\langle \mathbf{x}_i, \mathbf{a} \rangle, 0) : v_i \in \mathbb{R}, \mathbf{x}_i \in \mathbb{R}^d \right\} \]

lazy training ratio \( \kappa := \frac{\sum_{l=1}^{L} \| \mathbf{X}_l(t) - \mathbf{X}_l(0) \|_F}{\sum_{l=1}^{L} \| \mathbf{X}_l(0) \|_F} \)
Non-lazy training regime: visualization

mean field regime

\[
\sup_{t \in [0, +\infty)} \frac{\|X_l(t) - X_l(0)\|_F}{\|X_l(0)\|_F} \to 1
\]

Figure: Training dynamics of two-layer ReLU NNs under different initializations [44, 17, 57].
Non-lazy training regime: visualization

Figure: Training dynamics of two-layer ReLU NNs under different initializations [44, 17, 57].
Our understanding [Zhu, Liu, Chrysos, Cevher, NeurIPS (2022)]

Helps! [12]

Hurts! [80, 42]
Our understanding [Zhu, Liu, Chrysos, Cevher, NeurIPS (2022)]

Definition (Lipschitz constant with respect to the input)

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

Remarks:
- Lipschitz constant can be used to describe the worst-case robustness.
- Lipschitz constant theoretically correlates with the generalization ability of NN classifiers [4].
Robustness in deep learning: metrics

- Conflicting messages that can change due to
  - initialization (e.g., lazy training, non-lazy training)
  - architecture (e.g., width, depth)

**Definition (perturbation stability [87])**

The perturbation stability of a ReLU DNN $h_x(a)$ is

$$
\mathcal{P}(h, \epsilon) = \mathbb{E}_{a, \hat{a}, x} \left\| \nabla_a h_x(a)^\top (a - \hat{a}) \right\|_2, \quad \forall a \sim D_A, \quad \hat{a} \sim \text{Unif}(B(\epsilon, a)),
$$

where $\epsilon$ is the perturbation radius.
Robustness in deep learning: metrics

- Conflicting messages that can change due to
  - initialization (e.g., lazy training, non-lazy training)
  - architecture (e.g., width, depth)

**Definition (perturbation stability [87]: lazy training regime)**

The perturbation stability of a ReLU DNN $h_x(a)$ is

$$
\mathcal{P}(h, \epsilon) = \mathbb{E}_{a, \hat{a}, x(0)} \left\| \nabla_{a} h_{x}(a)^{\top}(a - \hat{a}) \right\|_{2}, \quad \forall a \sim D_{A}, \; \hat{a} \sim \text{Unif}(B(\epsilon, a)),
$$

where $\epsilon$ is the perturbation radius.
Robustness in deep learning: metrics

- Conflicting messages that can change due to
  - initialization (e.g., lazy training, non-lazy training)
  - architecture (e.g., width, depth)

**Definition (perturbation stability [87]: non-lazy training regime)**

The perturbation stability of a ReLU DNN $h_x(a)$ is

$$
\mathcal{P}(h, \epsilon) = \mathbb{E}_{a, \hat{a}} \left\| \nabla a h_x(a)^\top (a - \hat{a}) \right\|_2, \quad \forall a \sim \mathcal{D}_A, \ \hat{a} \sim \text{Unif}(B(\epsilon, a))
$$

where $\epsilon$ is the perturbation radius.
Main results (Lazy-training regime)

**Theorem** [87]: \( \cdot \lesssim \text{Func}(m, L, \beta) \)

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Initialization</th>
<th>Our bound for ( \mathcal{P}(f, \epsilon)/\epsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( |a|_2 = 1 )</td>
<td>Lecun initialization</td>
<td>( \sqrt{\frac{L^3 m}{p} e^{-m/L^3}} + \sqrt{\frac{1}{p}} \left( \frac{\sqrt{2}}{2} L - 2 \right) )</td>
</tr>
<tr>
<td></td>
<td>He initialization</td>
<td>( \sqrt{\frac{L^3 m}{p} e^{-m/L^3}} + \sqrt{\frac{1}{p}} )</td>
</tr>
<tr>
<td></td>
<td>NTK initialization</td>
<td>( \sqrt{\frac{L^3 m}{p} e^{-m/L^3}} + 1 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trend of width ( m )</th>
<th>Trend of depth ( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \uparrow )</td>
<td>( \downarrow )</td>
</tr>
<tr>
<td>( \downarrow )</td>
<td>( \uparrow )</td>
</tr>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
</tr>
</tbody>
</table>


**Remarks:**
- width helps robustness in the over-parameterized regime
- depth helps robustness in Lecun initialization but hurts robustness in He/NTK initialization
Experiments: lazy training experiment for FCNN

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Ours (NTK initialization)</th>
<th>[80]</th>
<th>[42]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}(f, \epsilon) / \epsilon$</td>
<td>$\sqrt{\frac{L^3 m}{p}} e^{-m/L^3} + 1$</td>
<td>$L^2 m^{1/3} \sqrt{\log m + \sqrt{mL}}$</td>
<td>$2 \frac{3L - 5}{2} \sqrt{L}$</td>
</tr>
</tbody>
</table>

(a) LeCun initialization  
(b) He initialization  
(c) NTK initialization
Experiments: lazy training experiment for CNN

![Graphs showing stability against width for different depths of L = 4, 6, 8, and 10.](image)

(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 [87].
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:
- $L = 2$, $\alpha = 1$, $\beta_1 = \beta_2 = \beta \sim \frac{1}{mc}$ with $c > 1.5$
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

$$\text{perturbation stability} \leq \tilde{O}\left(\frac{n}{m^{c+1.5}}\right), \text{ whp.}$$

Remarks:
- width helps robustness in the over-parameterized regime in both lazy/non-lazy training regime
### Experiment: Non-lazy training regime

Lazy training ratio \( \kappa \) :=

\[
\frac{\sum_{l=1}^{L} \| X_l(t) - X_l(0) \|_F}{\sum_{l=1}^{L} \| X_l(0) \|_F}
\]

![Graph showing lazy training ratio \( \kappa \) as a function of epochs for different widths, with a blue line labeled L=2, Non-lazy.]
Why robust generalization is difficult?

Figure: Robust classifiers exist if the perturbation is less than the separation: source from [82].

<table>
<thead>
<tr>
<th></th>
<th>perturbation $\epsilon$</th>
<th>Train-Train</th>
<th>Test-Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.1</td>
<td>0.737</td>
<td>0.812</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>0.031</td>
<td>0.212</td>
<td>0.220</td>
</tr>
<tr>
<td>SVHN</td>
<td>0.031</td>
<td>0.094</td>
<td>0.110</td>
</tr>
<tr>
<td>ResImageNet</td>
<td>0.005</td>
<td>0.180</td>
<td>0.224</td>
</tr>
</tbody>
</table>

Table: Separation of real data under typical perturbation radii. [82]

Theorem (Curse of dimensionality [52])

For a ReLU DNN with $m$ parameter, for any $\epsilon$-separated set $A$, $B \subset [0,1]^P$, it requires $m = \Omega(\epsilon^{-P})$ to classify $A$ and $B$. 
Recall empirical risk minimization...

- Goal of ML: find a “good” estimator $h$ approximating the lowest expected risk

$$\inf_{h \in H} R(h), \quad R(h) := \mathbb{E}_{(a, b) \sim \rho} L(h(a), b),$$

given training data $\{(a_i, b_i)\}_{i=1}^n$

$$h^* = \arg \min_{h \in H} R_n(h) := \frac{1}{n} \sum_{i=1}^n L(h_x(a_i), b_i)$$
Recall empirical risk minimization...

- Goal of ML: find a “good” estimator \( h \) approximating the lowest expected risk

\[
\inf_{h \in \mathcal{H}} R(h), \quad R(h) := \mathbb{E}_{(a, b) \sim \rho} L(h(a), b),
\]

given training data \( \{(a_i, b_i)\}_{i=1}^n \)

\[
h^* = \arg\min_{h \in \mathcal{H}} R_n(h) := \frac{1}{n} \sum_{i=1}^n L(h(x_i), b_i)
\]

- generalization error:

\[
R(h^*) - R_n(h^*) = O(n^{-\alpha}), \quad \text{for some } \alpha > 0, \text{whp.}
\]
Recall empirical risk minimization...

- Goal of ML: find a “good” estimator $h$ approximating the lowest expected risk

$$\inf_{h \in \mathcal{H}} R(h), \quad R(h) := \mathbb{E}_{(a,b) \sim \rho} L(h(a), b),$$

given training data $\{(a_i, b_i)\}_{i=1}^n$

$$h^* = \arg\min_{h \in \mathcal{H}} R_n(h) := \frac{1}{n} \sum_{i=1}^n L(h_x(a_i), b_i)$$

- generalization error:

$$R(h^*) - R_n(h^*) = \mathcal{O}(n^{-\alpha}), \quad \text{for some } \alpha > 0, \text{ whp.}$$

- uniform convergence: $\sup_{h \in \mathcal{H}} |R(h) - R_n(h)|$
Recall empirical risk minimization...

- Goal of ML: find a “good” estimator $h$ approximating the lowest expected risk

$$\inf_{h \in \mathcal{H}} R(h), \quad R(h) := \mathbb{E}_{(a, b) \sim \rho} L(h(a), b),$$

given training data $\{(a_i, b_i)\}_{i=1}^n$

$$h^* = \arg \min_{h \in \mathcal{H}} R_n(h) := \frac{1}{n} \sum_{i=1}^n L(h_x(a_i), b_i)$$

- generalization error:

$$R(h^*) - R_n(h^*) = O(n^{-\alpha}), \quad \text{for some } \alpha > 0, \text{whp.}$$

- uniform convergence: $\sup_{h \in \mathcal{H}} |R(h) - R_n(h)|$

$$R(h^*) \leq \frac{1}{n} \sum_{i=1}^n L(h_x^*(a_i), b_i) + O \left( \sqrt{\frac{c^*}{n}} \right), \text{whp.}$$

uniform laws of large numbers + capacity control
Rademacher complexity

Definition (Empirical Rademacher Complexity [6])

Let \( \mathcal{H} \) be a class of functions of the form \( h : \mathbb{R}^p \to \mathbb{R} \). The empirical Rademacher complexity of \( \mathcal{H} \) with respect to \( A \) is defined as:

\[
\mathcal{R}_A(\mathcal{H}) := \mathbb{E}_v \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \langle v_i, h(a_i) \rangle, \quad \Pr(v_i = 1) = \Pr(v_i = -1) = 1/2.
\]

Remark: \( \mathcal{R}_A(\mathcal{H}) \) measures how well we fit random \((\pm 1)\) with the output of an element of \( \mathcal{H} \) on the set \( A \).
Visualizing Rademacher complexity

(a) High Rademacher Complexity

(b) Large Generalization error (memorization)

(c) Low Rademacher Complexity

(d) Low Generalization error

Figure: Rademacher complexity and Generalization error
Visualizing Rademacher complexity

(a) High Rademacher Complexity

(b) Large Generalization error (memorization)

(c) Low Rademacher Complexity

(d) Low Generalization error

Figure: Rademacher complexity and Generalization error

\[
\sup_{h \in \mathcal{H}} |R(h) - R_n(h)| \lesssim \mathcal{R}_A(\mathcal{H}) + O\left(\frac{1}{\sqrt{n}}\right), \text{whp.}
\]
Why uniform convergence fails in deep learning?

\[ R(h^*) \leq \frac{1}{n} \sum_{i=1}^{n} L(h^*_x(a_i), b_i) + O\left(\sqrt{\frac{c^*}{n}}\right), \text{whp.} \]

Figure: DNN Training curves on CIFAR10: source from [85]
Why uniform convergence fails in deep learning?

\[ R(h^*) \leq \frac{1}{n} \sum_{i=1}^{n} L(h_x^*(a_i), b_i) + \mathcal{O} \left( \sqrt{\frac{c^*}{n}} \right), \text{whp.} \]

\[ \text{Figure: Interpolation still generalizes well under noisy data on MNIST: source from [9].} \]
Why uniform convergence fails in deep learning?

\[ R(h^*) \leq \frac{1}{n} \sum_{i=1}^{n} L(h^*_x(a_i), b_i) + \mathcal{O}\left(\sqrt{\frac{c^*}{n}}\right), \text{whp.} \]

- Observation: Generalization bounds vs. #training data [64, 86]

**Figure:** Interpolation still generalizes well under noisy data on MNIST: source from [9].
When does uniform convergence work?

\[ \{ \mathbf{x} : \| \mathbf{x} \|_2 \leq B \} \quad \{ \mathbf{x} : \| \mathbf{x} \|_2 \leq B, R_n(\mathbf{x}) = 0 \} \]

**Figure**: Uniform convergence of interpolators: source from [86].

**Definition (One-side uniform convergence [86])**

\[
\sup_{\| \mathbf{x} \| \leq B, R_n(h_{\mathbf{x}}) = 0} \left\{ R(h_{\mathbf{x}}) - R_n(h_{\mathbf{x}}) \right\}
\]
Results for benign overfitting

**Theorem (Simplified version of Corollary 1 in [47])**

*Under standard Gaussian data, noise setting, for over-parameterized least squares, we have*

\[
\sup_{\|x\| \leq B, R_n(h_x) = 0} R(h_x) \lesssim \frac{B^2 \text{Tr}(\Sigma)}{n}, \text{whp.}
\]

**Remarks:**
- Via covariance splitting \( \Sigma = \Sigma_1 \oplus \Sigma_2 \), we can improve this result if
  - \( \Sigma_1 \) is low rank
  - \( \Sigma_2 \) has fast eigenvalue decay [47]
  - the target function has small norm
- Beyond linear regression [5]: NNs in non-lazy training regimes [27, 49]
Beyond benign overfitting

○ Under the settings below, we will have benign overfitting: $\mathbb{R}(h^*_X) \to \sigma^2$
  ▶ early-stopped DNNs
  ▶ kernel ridge regression
  ▶ k-NN ($k \sim \log n$)
  ▶ Nadaraya-Watson kernel smoothing

**Figure**: As $n \to \infty$ and fixed $p$, interpolating methods can exhibit three types of overfitting: source from [60].

- Benign
- Tempered
- Catastrophic
Beyond benign overfitting

Figure: As $n \to \infty$ and fixed $p$, interpolating methods can exhibit three types of overfitting: source from [60].

- Under the settings below, we will have tempered overfitting: $R(h^*_x) \to c\sigma^2$
  - interpolating DNNs
  - Laplace kernel regression
  - ReLU NTKs
  - k-NN (constant $k$)
  - simplicial interpolation
Beyond benign overfitting

Figure: As $n \to \infty$ and fixed $p$, interpolating methods can exhibit three types of overfitting: source from [60].

- Under the settings below, we will have catastrophic overfitting: $R(h^*_x) \to \infty$
  - Gaussian kernel regression
  - critically-parameterized regression
How well do complexity measures correlate with generalization?

<table>
<thead>
<tr>
<th>name</th>
<th>definition</th>
<th>correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frobenius distance to initialization [65]</td>
<td>$\sum_{i=1}^{L} |X_i - X^0_i|^2_F$</td>
<td>-0.263</td>
</tr>
<tr>
<td>Spectral complexity [4]</td>
<td>$\prod_{i=1}^{L} |X_i| \left(\sum_{i=1}^{L} |X_i|^{3/2}/|X_i|^{3/2}\right)^{2/3}$</td>
<td>-0.537</td>
</tr>
<tr>
<td>Parameter Frobenius norm</td>
<td>$\sum_{i=1}^{L} |X_i|^2_F$</td>
<td>0.073</td>
</tr>
<tr>
<td>Path-norm [68]</td>
<td>$\sum_{(i_0,...,i_L)} \prod_{j=1}^{L} \left(X_{i_j, i_{j-1}}\right)^2$</td>
<td>0.373</td>
</tr>
</tbody>
</table>

Table: Complexity measures compared in the empirical study [45], and their correlation with generalization

Complexity measures are still far from explaining generalization in Deep Learning!
A more recent evaluation of many complexity measures is available [24].
Double descent

- A failure of conventional wisdom

![Diagram](image)

**Figure:** The classical U-shaped risk curve vs. double-descent risk curve: source from [8].

- classical large-sample limit setting: $n \to \infty$ under fixed $p$
- modern high dimensional setting: $n, m, p$ are comparably large
Double descent curve in practice (I)

- Typical examples:
  - linear/nonlinear regression [36]
  - random features, random forest, and shallow neural networks [8]

Figure: Experiments on MNIST: source from [8].
Double descent curve in practice (II)

Figure: Left: Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. Right: Test error, shown for varying train epochs: source from [66].
Double descent curve in practice (III)

**Figure:** Left: The double descent phenomenon, where the number of parameters is used as the model complexity. Middle: The norm of the learned model is peaked around $n \approx p$. Right: The test error against the norm of the learnt model. The color bar indicate the number of parameters and the arrows indicates the direction of increasing model size. Their relationship are closer to the convention wisdom than to a double descent. source: [69]. This is the same setting as in Section 5.2 of [67].
From neural networks to random features model [73]

- 1-hidden-layer neural network with \( m \) neurons (fully-connected architecture)

  ▶ Let \( X_1 \in \mathbb{R}^{m \times p} \), \( a \in \mathbb{R}^p \), \( X_2 \in \mathbb{R}^m \), and \( \mu_2 \in \mathbb{R} \)

  \[
  h_{x}(a) := \begin{bmatrix} X_2 \end{bmatrix} \sigma \left( \begin{bmatrix} X_1 \end{bmatrix} a + \begin{bmatrix} \mu_1 \end{bmatrix} \right) + \begin{bmatrix} \mu_2 \end{bmatrix}, \quad x := [X_1, X_2, \mu_1, \mu_2]
  \]

  ▶ \( X_1 \): Gaussian initialization and then fixed

  ▶ \( X_2 \): to be learned
Our understanding on double descent [Liu, Suykens, Cevher, NeurIPS (2022)]

- High dimensional setting: #training data $n$, #neurons $m$, input dimension $p$ are comparably large.

![Graphs](image.png)

Figure: Test MSE, Bias, and Variance of RF regression as a function of the ratio $m/n$ on MNIST data set (digit 3 vs. 7) for $p = 784$ and $n = 600$ across the Gaussian kernel. Source: [54].

Remarks:
- interplay between excess risk and optimization
- monotonic decreasing bias and unimodal variance $\Rightarrow$ double descent
- converge to $O(1)$ order
- constant step-size SGD vs. min norm solution
Conclusions: Good, bad, ugly

<table>
<thead>
<tr>
<th>good</th>
<th>bad</th>
<th>ugly</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel methods</td>
<td>analysis</td>
<td>performance</td>
</tr>
<tr>
<td>neural networks</td>
<td>performance</td>
<td>analysis</td>
</tr>
<tr>
<td>robustness</td>
<td>width</td>
<td>depth</td>
</tr>
<tr>
<td>generalization</td>
<td>benign overfitting</td>
<td>catastrophic overfitting</td>
</tr>
</tbody>
</table>

\[
s^\infty_{t \in [0, +\infty)} \frac{||X_l(t) - X_l(0)||_F}{||X_l(0)||_F} \rightarrow ???
\]

lazy training regime

mean field regime

non-lazy training regime

Lecun, He - NTK

Xavier
Conclusions: Function spaces vs models

Understanding from a function space perspective!

- RKHS
  - bivariate form
- hyper-RKHS
  - hyper-kernel methods^2
- Barron space
  - two-layer NNs^3
- Besov space
  - deep NNs^4

kernel methods^1
scalability
random features
NTK
over-parameterization

Research Statement
Understanding generalization in machine learning algorithms: a function approximation perspective

Achieving this goal requires to study what regularizer \( \Omega(f) \) can be defined and controlled on the functions defined by models, and what function space \( F \) is suitable for learning.

The commonly used function space in learning theory is the reproducing kernel Hilbert space (RKHS) [Aro50], which provides the ability to approximate functions by nonparametric functional representations. The point-wise convergence property makes RKHS an appealing choice in machine learning problems with nice theoretical guarantees in an approximation theory view. My major research interests starts with kernel learning algorithms, kernel approximation for scalability, and theoretically understanding machine learning algorithms in under- and over-parameterized regimes.

1 Current achievements
My research endeavour has led to several scientific contributions at the flagship conferences and journals in machine learning. Here I center around the work in recent years on learning in hyper-RKHS [LSH + 21, JMLR21], kernel approximation via random features, double descent [LSC22, NeurIPS22], deep neural function approximation [LVC22, NeurIPS22].
Thanks for your attention!

Q & A
References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song.
A convergence theory for deep learning via over-parameterization.
(Cited on pages 60 and 80.)

Wasserstein generative adversarial networks.
(Cited on page 36.)

Breaking the curse of dimensionality with convex neural networks.
(Cited on page 78.)

(Cited on pages 87, 88, and 114.)
References II

Benign overfitting in linear regression.  
(Cited on pages 73, 74, and 110.)

Rademacher and gaussian complexities: Risk bounds and structural results.  
(Cited on page 103.)

Deep learning: a statistical viewpoint.  
(Cited on page 82.)

Reconciling modern machine-learning practice and the classical bias–variance trade-off.  
(Cited on pages 73, 74, 115, and 116.)
References III

To understand deep learning we need to understand kernel learning.  
(Cited on pages 107 and 108.)

Dynamics of stochastic approximation algorithms.  
(Cited on pages 18, 19, and 20.)

Asymptotic pseudotrajectories and chain recurrent flows, with applications.  
(Cited on pages 42 and 43.)

A universal law of robustness via isoperimetry.  
(Cited on pages 87 and 88.)
References IV


[17] Lenaic Chizat, Edouard Oyallon, and Francis Bach.  
On lazy training in differentiable programming.  
(Cited on pages 83, 85, and 86.)

The theory of max-min, with applications.  
(Cited on page 25.)

The complexity of constrained min-max optimization.  
(Cited on page 39.)

Efficient methods for structured nonconvex-nonconcave min-max optimization.  
(Cited on pages 45, 46, 47, 48, and 49.)
Gradient descent finds global minima of deep neural networks.  
(Cited on page 60.)

[22] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh.  
Gradient descent provably optimizes over-parameterized neural networks.  
(Cited on pages 18, 19, 59, and 60.)

The speed of mean glivenko-cantelli convergence.  
(Cited on page 34.)

In search of robust measures of generalization.  
(Cited on page 114.)
References VII

A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics.
(Cited on page 80.)

Robust physical-world attacks on deep learning visual classification.
(Cited on pages 66 and 67.)

[27] Spencer Frei, Niladri S Chatterji, and Peter Bartlett.
Benign overfitting without linearity: Neural network classifiers trained by gradient descent for noisy linear data.
(Cited on pages 73, 74, and 110.)

[28] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points—online stochastic gradient for tensor decomposition.
(Cited on pages 18 and 19.)
References VIII

[29] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points — Online stochastic gradient for tensor decomposition.
(Cited on pages 18, 19, 21, and 22.)

Stochastic first- and zeroth-order methods for nonconvex stochastic programming.
(Cited on pages 18, 19, and 22.)

[31] Xavier Glorot and Yoshua Bengio.
Understanding the difficulty of training deep feedforward neural networks.
(Cited on page 80.)

Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions.
(Cited on pages 18 and 19.)
[33] Osman Güler.
On the convergence of the proximal point algorithm for convex minimization.
(Cited on page 40.)

[34] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville.
Improved training of wasserstein gans.
(Cited on page 36.)

Identity matters in deep learning.
(Cited on pages 18 and 19.)

Surprises in high-dimensional ridgeless least squares interpolation.
(Cited on page 116.)
References

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 
(Cited on page 80.)

[38] Ya-Ping Hsieh, Chen Liu, and Volkan Cevher. 
Finding mixed Nash equilibria of generative adversarial networks. 
(Cited on page 52.)

The limits of min-max optimization algorithms: Convergence to spurious non-critical sets. 
(Cited on pages 41, 42, 43, and 44.)

[40] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. 
Explore aggressively, update conservatively: Stochastic extragradient methods with variable stepsize scaling. 
(Cited on pages 45, 46, 47, 48, and 49.)


References XII

Fantastic generalization measures and where to find them. 
(Cited on page 114.)

Gradient descent finds global minima for generalizable deep neural networks of practical sizes. 
(Cited on pages 18, 19, and 60.)

Uniform convergence of interpolators: Gaussian width, norm bounds and benign overfitting. 
(Cited on page 110.)

The extragradient method for finding saddle points and other problems. 
(Cited on page 40.)
Benign overfitting for two-layer relu networks.  
(Cited on page 110.)

Efficient backprop.  
(Cited on page 80.)

First-order methods almost always avoid strict saddle points.  
(Cited on pages 18, 19, and 21.)

Why robust generalization in deep learning is difficult: Perspective of expressive power.  
(Cited on page 98.)
[53] Yuanzhi Li and Yingyu Liang. 
Learning overparameterized neural networks via stochastic gradient descent on structured data. 
(Cited on pages 18, 19, and 60.)

On the double descent of random features models trained with sgd. 
(Cited on page 120.)

[55] Lennart Ljung. 
Analysis of recursive stochastic algorithms. 
(Cited on pages 18, 19, and 20.)

Phase diagram for two-layer relu neural networks at infinite-width limit. 
(Cited on pages 80 and 81.)
Phase diagram for two-layer relu neural networks at infinite-width limit.
(Cited on pages 83, 85, and 86.)

[58] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu.
Towards deep learning models resistant to adversarial attacks.
(Cited on pages 27, 28, and 31.)

[59] Yura Malitsky and Matthew K Tam.
A forward-backward splitting method for monotone inclusions without cocoercivity.
(Cited on page 40.)

Benign, tempered, or catastrophic: A taxonomy of overfitting.
(Cited on pages 111, 112, and 113.)
A mean field view of the landscape of two-layers neural networks.
Proceedings of the National Academy of Sciences (PNAS), 2018.
(Cited on page 80.)

On the almost sure convergence of stochastic gradient descent in non-convex problems.
(Cited on pages 18, 19, 20, 21, and 22.)

[63] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
(Cited on page 36.)

[64] Vaishnavh Nagarajan and J Zico Kolter.
Uniform convergence may be unable to explain generalization in deep learning.
(Cited on page 108.)
References XVII


[70] Quynh Nguyen and Matthias Hein.  
Optimization landscape and expressivity of deep cnns.  
(Cited on pages 18, 19, and 60.)

Solving stochastic weak minty variational inequalities without increasing batch size.  
(Cited on pages 45, 46, 47, 48, and 49.)

Escaping limit cycles: Global convergence for constrained nonconvex-nonconcave minimax problems.  
(Cited on pages 45, 46, 47, 48, and 49.)

[73] Ali Rahimi and Benjamin Recht.  
Random features for large-scale kernel machines.  
(Cited on page 119.)
*Convex Analysis.*  
(Cited on page 40.)

Stochastic approximations with constant step size and differential inclusions.  
(Cited on pages 18, 19, and 20.)

Subquadratic overparameterization for shallow neural networks.  
(Cited on pages 18 and 19.)

Subquadratic overparameterization for shallow neural networks.  
(Cited on page 60.)
References XX

Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance.
(Cited on page 34.)

[79] Max Welling and Yee W Teh.
Bayesian learning via stochastic gradient langevin dynamics.
(Cited on pages 18 and 19.)

[80] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu.
Do wider neural networks really help adversarial robustness?
(Cited on pages 87, 88, and 93.)

[81] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein.
Making an invisibility cloak: Real world adversarial attacks on object detectors.
(Cited on pages 6, 66, and 67.)
References XXI


[86] Lijia Zhou, Danica J Sutherland, and Nati Srebro.  
On uniform convergence and low-norm interpolation learning.  
(Cited on pages 108 and 109.)

[87] Zhenyu Zhu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher.  
Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization).  
(Cited on pages 89, 90, 91, 92, and 94.)

Online convex programming and generalized infinitesimal gradient ascent.  
(Cited on page 40.)

[89] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu.  
Gradient descent optimizes over-parameterized deep relu networks.  
(Cited on page 60.)