Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 7: Deep learning I

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

EE-556 (Fall 2020)

License Information for Mathematics of Data Slides

Outline

- This class
  - Introduction to Deep Learning
  - The Deep Learning Paradigm
  - Challenges in Deep Learning Theory and Applications
  - Introduction to Generalization error bounds
    - Uniform Convergence and Rademacher Complexity
  - Generalization in Deep Learning (Part 1)
- Next class
  - Generalization in Deep Learning (Part 2)

Remark about notation

◦ The Deep Learning literature might use a different notation:

	Our lectures	DL literature
data/sample	$\mathbf{a}$	$\mathbf{x}$
label	b	y
bias	μ	b
weight	$\mathbf{x}, \mathbf{X}$	$\mathbf{w}, \mathbf{W}$

Power of linear classifiers–I

Problem (Recall: Logistic regression)

Given a sample vector $\mathbf{a}_i \in \mathbb{R}^d$ *and a binary class label* $b_i \in \{-1, +1\}$ $(i = 1, \ldots, n)$*, we define the conditional probability of* b_i *given* $\mathbf{a}_i$ *as:*

$$\mathbb{P}(b_i|\mathbf{a}_i, \mathbf{x}) \propto 1/(1 + e^{-b_i\langle\mathbf{x},\mathbf{a}_i\rangle}),$$

where $\mathbf{x} \in \mathbb{R}^d$ *is some weight vector.*

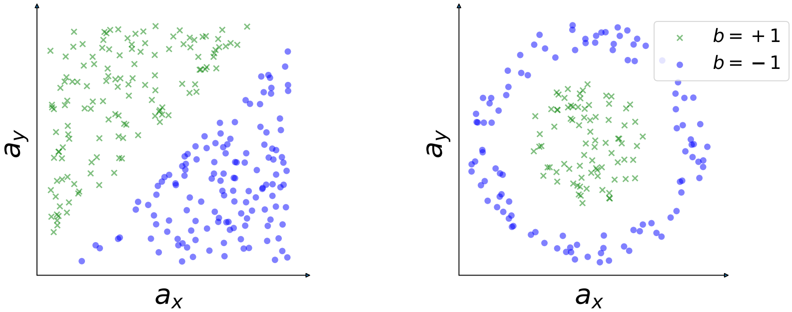


Figure: Linearly separable versus nonlinearly separable dataset

Power of linear classifiers–II

- Lifting dimensions to the rescue
  - Convex optimization objective
  - Might introduce the curse-of-dimensionality
  - Possible to avoid via kernel methods, such as SVMs

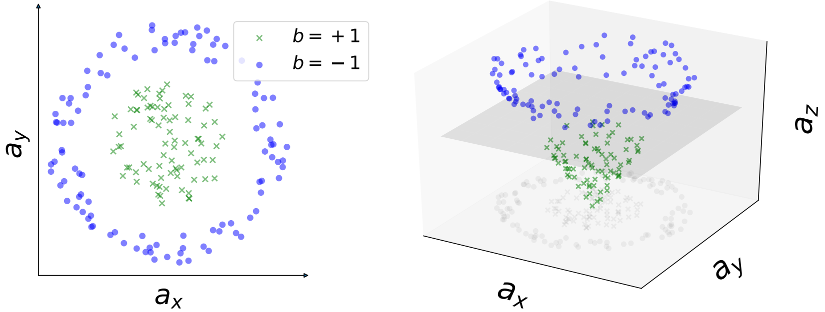


Figure: Non-linearly separable data (left). Linearly separable in $\mathbb{R}^3$ via $\mathbf{a}_z = \sqrt{\mathbf{a}_x^2 + \mathbf{a}_y^2}$ (right).

An important alternative for non-linearly separable data

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

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
- Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \underset{\substack{\uparrow \\ \text{input}}}{\begin{bmatrix} \mathbf{a} \end{bmatrix}}$$

An important alternative for non-linearly separable data

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

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
- Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \underset{\text{weight}}{\begin{bmatrix} \quad \mathbf{X}_1 \quad \end{bmatrix}} \underset{\text{input}}{\begin{bmatrix} \mathbf{a} \end{bmatrix}}$$

An important alternative for non-linearly separable data

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

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
- Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \underset{\text{weight}}{\left[\quad \mathbf{X}_1 \quad\right]} \underset{\text{input}}{\left[\mathbf{a}\right]} + \underset{\text{bias}}{\left[\mu_1\right]}$$

An important alternative for non-linearly separable data

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

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
- Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \underbrace{\overset{\text{activation}}{\overset{\downarrow}{\sigma}} \left(\overset{\text{weight}}{\overset{\downarrow}{\begin{bmatrix} \mathbf{X}_1 \end{bmatrix}}} \overset{\text{input}}{\overset{\downarrow}{\begin{bmatrix} \mathbf{a} \end{bmatrix}}} + \overset{\text{bias}}{\overset{\downarrow}{\begin{bmatrix} \mu_1 \end{bmatrix}}} \right)}_{\text{hidden layer } = \text{ learned features}}$$

An important alternative for non-linearly separable data

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

∘ Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m\times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c\times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
∘ Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \Bigg[\ \mathbf{X}_2\ \Bigg] \underbrace{\overset{\text{activation}}{\overset{\downarrow}{\sigma}} \left(\Bigg[\ \overset{\text{weight}}{\overset{\downarrow}{\mathbf{X}_1}}\ \Bigg] \overset{\text{input}}{\overset{\downarrow}{\Bigg[\mathbf{a}\Bigg]}} + \overset{\text{bias}}{\overset{\downarrow}{\Bigg[\mu_1\Bigg]}} \right)}_{\text{hidden layer = learned features}}$$

An important alternative for non-linearly separable data

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

∘ Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m\times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c\times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
∘ Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \begin{bmatrix} & \mathbf{X}_2 & \end{bmatrix} \underbrace{\overset{\substack{\text{activation}\\ \downarrow}}{\sigma} \left(\overset{\substack{\text{weight}\\ \downarrow}}{\begin{bmatrix} & \mathbf{X}_1 & \end{bmatrix}} \overset{\substack{\text{input}\\ \downarrow}}{\begin{bmatrix} \mathbf{a} \end{bmatrix}} + \overset{\substack{\text{bias}\\ \downarrow}}{\begin{bmatrix} \mu_1 \end{bmatrix}} \right)}_{\text{hidden layer} \,=\, \text{learned features}} + \overset{\substack{\text{bias}\\ \downarrow}}{\begin{bmatrix} \mu_2 \end{bmatrix}}$$

An important alternative for non-linearly separable data

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

- Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
- Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \Big[\ \mathbf{X}_2\ \Big] \underbrace{\overset{\text{activation}}{\overset{\downarrow}{\sigma}} \left(\overset{\text{weight}}{\overset{\downarrow}{\Big[\ \mathbf{X}_1\ \Big]}} \overset{\text{input}}{\overset{\downarrow}{\Big[\mathbf{a}\Big]}} + \overset{\text{bias}}{\overset{\downarrow}{\Big[\mu_1\Big]}} \right)}_{\text{hidden layer = learned features}} + \overset{\text{bias}}{\overset{\downarrow}{\Big[\mu_2\Big]}}, \qquad \mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$$

An important alternative for non-linearly separable data

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

∘ Parameters: $\mathbf{X}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{X}_2 \in \mathbb{R}^{c \times m}$ (weights), $\mu_1 \in \mathbb{R}^m, \mu_2 \in \mathbb{R}^c$ (biases)
∘ Activation function: $\sigma : \mathbb{R} \to \mathbb{R}$

$$h_{\mathbf{x}}(\mathbf{a}) := \left[\; \mathbf{X}_2 \; \right] \underbrace{\overset{\text{activation}}{\overset{\downarrow}{\sigma}} \left(\left[\; \overset{\text{weight}}{\overset{\downarrow}{\mathbf{X}_1}} \; \right] \overset{\text{input}}{\overset{\downarrow}{\left[\mathbf{a} \right]}} + \overset{\text{bias}}{\overset{\downarrow}{\left[\mu_1 \right]}} \right)}_{\text{hidden layer} \,=\, \text{learned features}} + \overset{\text{bias}}{\overset{\downarrow}{\left[\mu_2 \right]}}, \qquad \mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$$

recursively repeat activation + affine transformation to obtain "deeper" networks.

Why neural networks?: An approximation theoretic motivation

Theorem (Universal approximation [3])

Let $\sigma(\cdot)$ be a nonconstant, bounded, and increasing continuous function. Let $I_d = [0,1]^d$. The space of continuous functions on I_d is denoted by $\mathcal{C}(I_d)$.

Given $\epsilon > 0$ and $g \in \mathcal{C}(I_d)$ there exists a 1-hidden-layer network h with m neurons such that h is an ϵ-approximation of g, i.e.,

$$\sup_{\mathbf{a} \in I_d} |g(\mathbf{a}) - h(\mathbf{a})| \leq \epsilon$$

Caveat

The number of neurons m needed to approximate some function g can be arbitrarily large!

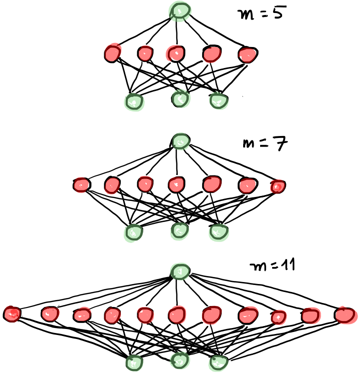


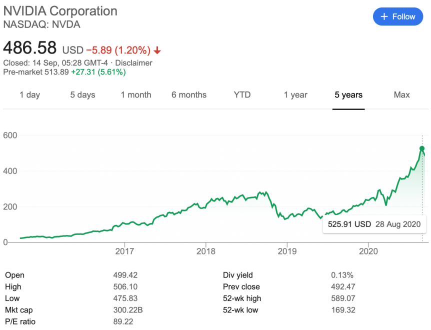
Figure: networks of increasing width

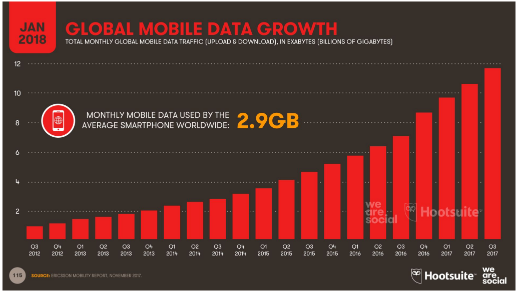
Why were NNs not popular before 2010?

- too big to optimize!
- did not have enough data
- could not find the optimum via algorithms

Why were NNs not popular before 2010?

- too big to optimize!
- did not have enough data
- could not find the optimum via algorithms





Supervised learning: Multi-class classification

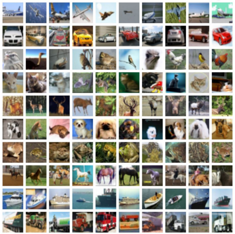


Figure: CIFAR10 dataset: 60000 32x32 color images (3 channels) from 10 classes

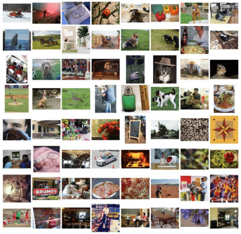


Figure: Imagenet dataset: 14 million color images (varying resolution, 3 channels) from 21K classes

Goal

Image-label pairs $(\mathbf{a}, b) \subseteq \mathbb{R}^d \times \{1, \ldots, c\}$ follow an unknown distibution $\mathbb{P}$. Find $h : \mathbb{R}^d \to \{1, \ldots, c\}$ with minimum *misclassification probability*

$$\min_{h \in \mathcal{H}} \mathbb{P}(h(\mathbf{a}) \neq b)$$

2010-today: Deep Learning becomes popular again

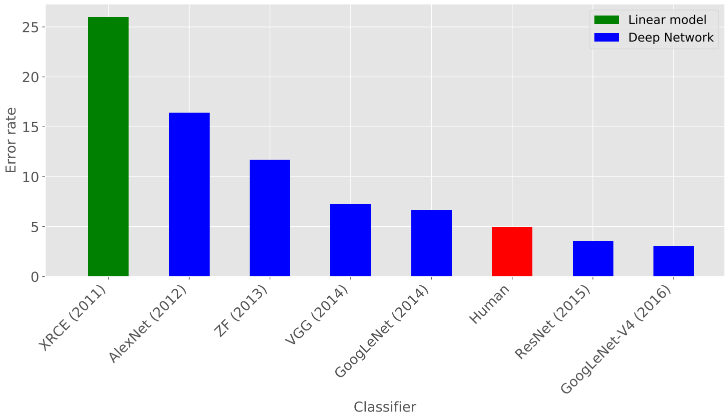


Figure: Error rate on the ImageNet challenge, for different network architectures.

Convolutional architectures in Computer Vision tasks

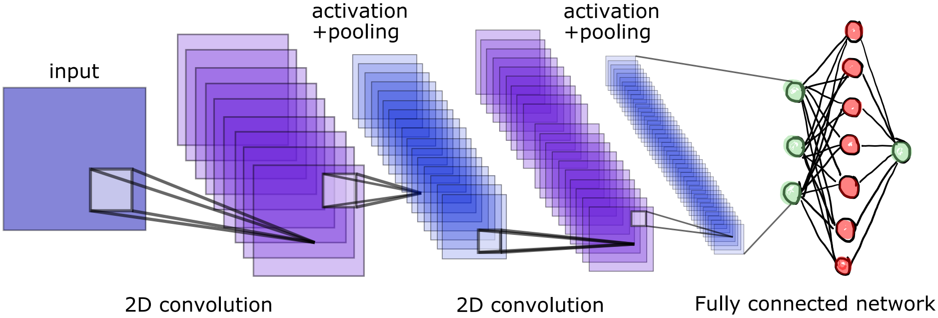
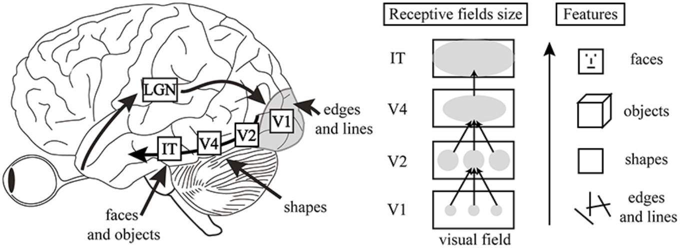
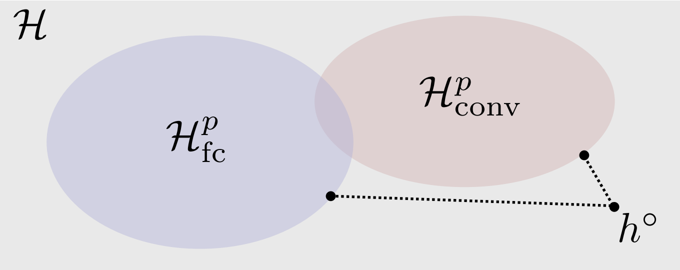


Figure: "Locality" Structure of a 2D deep convolutional neural network.

Inductive Bias: Why convolution works so well in Computer Vision tasks?



h°	true unknown function
$\mathcal{H}$	space of all functions
$\mathcal{H}^p_{\text{fc}}$	fully-connected networks with p parameters
$\mathcal{H}^p_{\text{conv}}$	convolutional networks with p parameters



2010-today: Size of neural networks grows exponentially!

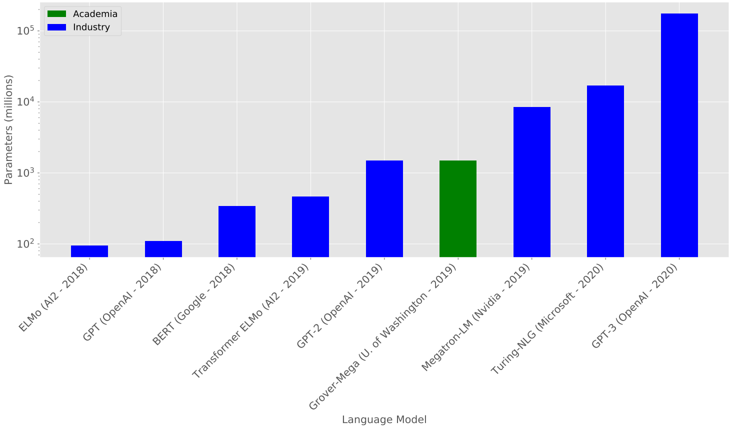


Figure: Number of parameters in Language models based on Deep Learning.

The Landscape of ERM with multilayer networks

Recall: Empirical risk minimization (ERM)

Let $h_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}$ be network and let $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be a sample with $b_i \in \{-1, 1\}$ and $\mathbf{a}_i \in \mathbb{R}^n$. The *empirical risk minimization* (ERM) is defined as

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

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

Some frequently used loss functions

- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \log(1 + \exp(-b \cdot h_{\mathbf{x}}(\mathbf{a})))$ (logistic loss)
- $L(h_{\mathbf{x}}(\mathbf{a}), b) = (b - h_{\mathbf{x}}(\mathbf{a}))^2$ (squared error)
- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \max(0, 1 - b \cdot h_{\mathbf{x}}(\mathbf{a}))$ (hinge loss)

The Landscape of ERM with multilayer networks

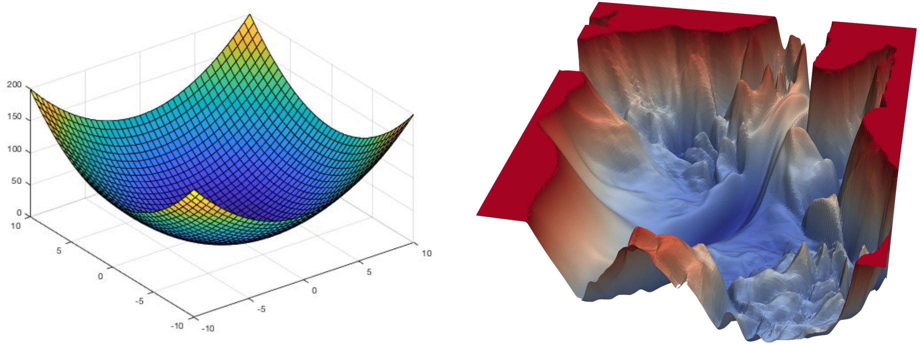
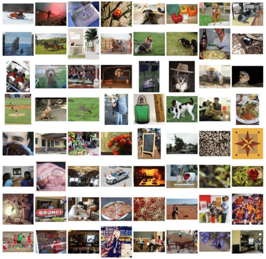


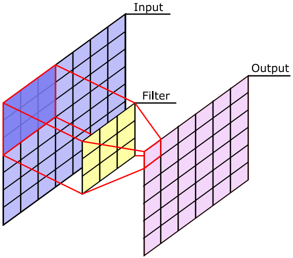
Figure: convex (left) vs non-convex (right) optimization landscape

Conventional wisdom in ML until 2010:
Simple models + simple errors

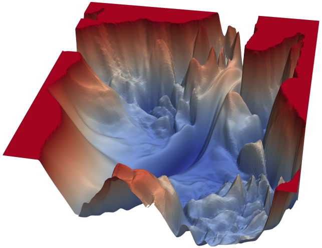
The Deep Learning Paradigm



(a) Massive datasets



(b) Inductive bias from large and complex architectures



(c) ERM using stochastic non-convex first-order optimization algorithms (SGD)

Figure: Most common components in a Deep Learning Pipeline

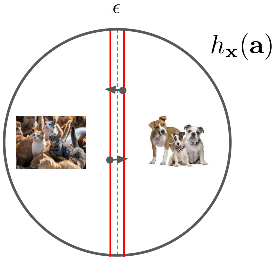
Challenges in DL/ML applications: Robustness (I)

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

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

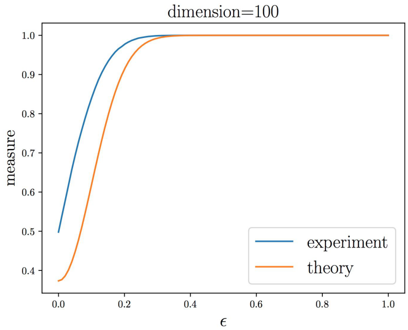
Figure: Natural or human-crafted modifications that trick neural networks used in computer vision tasks

Challenges in DL/ML applications: Robustness (II)



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

(a) Linear classifier on data distributed on a sphere



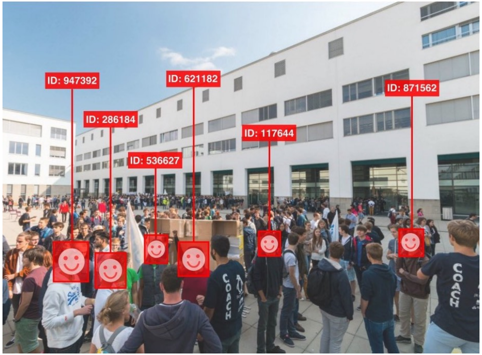
(b) Concentration of measure phenomenon on high dimensions

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

Challenges in DL/ML applications: Robustness (References)

1. Madry, Aleksander and Makelov, Aleksandar and Schmidt, Ludwig and Tsipras, Dimitris and Vladu, Adrian. *Towards Deep Learning Models Resistant to Adversarial Attacks*. ICLR 2018.
2. Raghunathan, A., Steinhardt, J., and Liang, P. S. *Semidefinite relaxations for certifying robustness to adversarial examples*. Neurips 2018.
3. Wong, E. and Kolter, Z. (2018). *Provable defenses against adversarial examples via the convex outer adversarial polytope*. ICML 2018.
4. Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. *Safety verification of deep neural networks*. Computer Aided Verification 2017.
5. Athalye, A., et al. *Synthesizing robust adversarial examples*. International conference on machine learning. PMLR, 2018.
6. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., and Song, D. *Robust physical-world attacks on deep learning visual classification*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1625-1634). 2018.
7. Shafahi A., Ronny Huang, W., Studer, C., Feizi, S. and Goldstein, T. *Are adversarial examples inevitable?*. International Conference on Learning Representations. 2019.

Challenges in DL/ML applications: Surveillance/Privacy/Manipulation



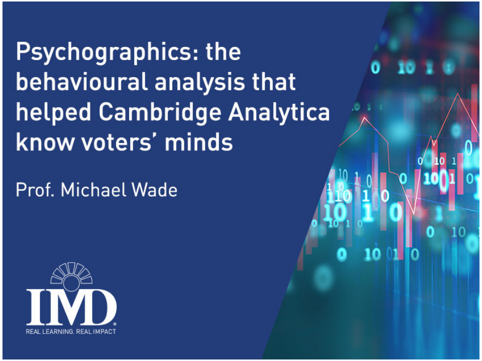
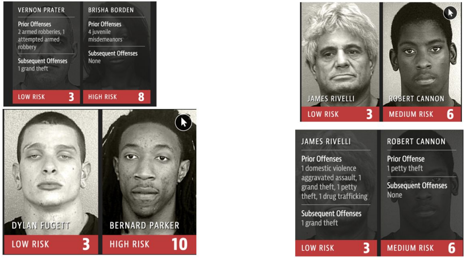


Figure: Political and societal concerns about some DL/ML applications

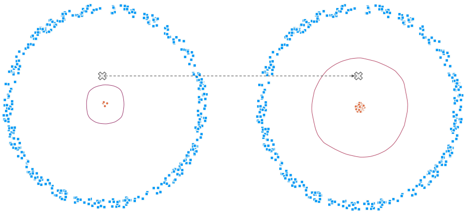
Challenges in DL/ML applications: Surveillance/Privacy/Manipulation (References)

1. Dwork, C., and Roth, A. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 9, 2013.
2. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. *Deep learning with differential privacy*. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 308-318). 2016.
3. Sreenu, G., Saleem Durai, M.A. *Intelligent video surveillance: a review through deep learning techniques for crowd analysis*. J Big Data 6, 48. 2019.
4. O'Neil, C., *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy* Broadway Books, (2016);
5. Wade, M. *Psychographics: the behavioural analysis that helped Cambridge Analytica know voters' minds*. `https://theconversation.com/psychographics-the-behavioural-analysis-that-helped-cambridge-analytica-know-voters-minds-93675` 2018.

Challenges in DL/ML applications: Fairness



(a) Racist classifier



(b) Effect of unbalanced data

Figure: Unfair classifiers due to biased or unbalanced datasets/algorithms

Challenges in DL/ML applications: Fairness (References)

1. Barocas, S. Hardt, M. Narayanan, Arvind. *Fairness in Machine Learning Limitations and Opportunities*. `https://fairmlbook.org/pdf/fairmlbook.pdf` 2020.
2. Hardt, M. *How Big Data Is Unfair*. `https://medium.com/@mrtz/how-big-data-is-unfair-9aa544d739de` 2014.
3. Munoz, C., Smith, M., and Patil, D. *Big Data: A Report on Algorithmic Systems, Opportunity, and Civil Rights*. Executive Office of the President. The White House, 2016.
4. Campolo, A., Sanfilippo, M., Whittaker, M., Crawford, K. *AI Now 2017 Report*. AI Now Institute at New York University, 2017.
5. Friedman, B. and Nissenbaum, H. *Bias in Computer Systems*. ACM Transactions on Information Systems (TOIS) 14, no. 3. 1996: 330–47.
6. Pedreshi, D., Ruggieri, S. and Turini, F. *Discrimination-Aware Data Mining*. Proc. 14th SIGKDD. ACM 2008.
7. Noble, S.U. *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press. 2018.

Challenges in DL/ML applications: Interpretability

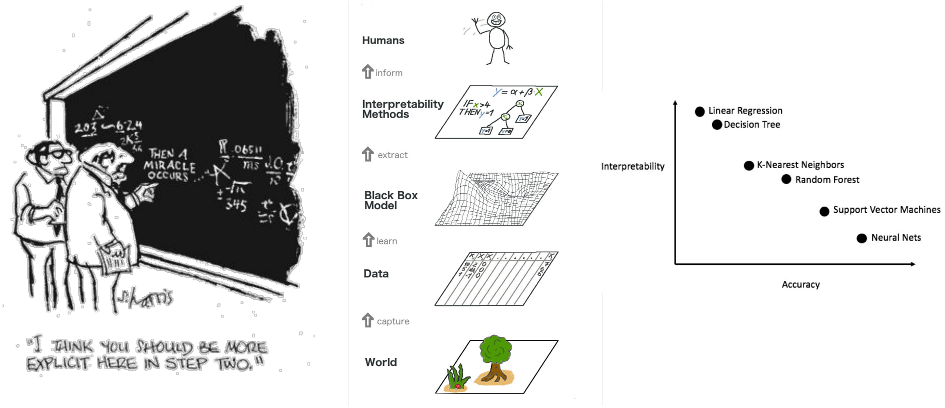
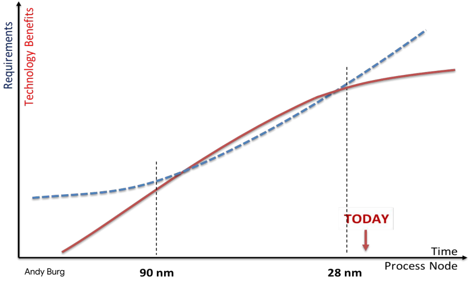


Figure: Performance vs Interpretability trade-offs in DL/ML

Challenges in DL/ML applications: Interpretability (References)

1. Baehrens, David and Schroeter, Timon and Harmeling, Stefan and Kawanabe, Motoaki and Hansen, Katja and Mueller, Klaus-Robert. Simonyan, Karen and Vedaldi, Andrea and Zisserman, Andrew. How to Explain Individual Classification Decisions. JMLR 2010.
2. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv e-prints. arXiv:1312.6034. 2013.
3. Ribeiro, Marco and Singh, Sameer and Guestrin, Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. KDD 2016.
4. Sundararajan, Mukund and Taly, Ankur and Yan, Qiqi. Axiomatic Attribution for Deep Networks. ICML 2017.
5. Shrikumar, Avanti and Greenside, Peyton and Kundaje, Anshul. Learning Important Features Through Propagating Activation Differences. ICML 2017.

Challenges in DL/ML applications: Energy efficiency and cost



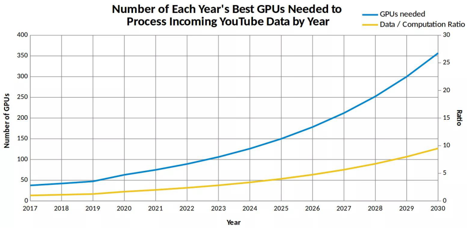
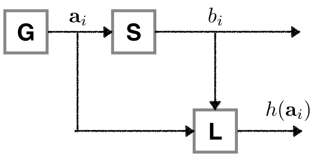


Figure: Efficiency and Scalability concerns in DL/ML

Challenges in DL/ML applications: Energy efficiency and cost (References)

1. García-Marín, E., Rodrigues, C. F., Riley, G., and Grahn, H. *Estimation of energy consumption in machine learning*. Journal of Parallel and Distributed Computing, 134, 75-88. 2019.
2. Strubell, E., Ganesh, A., and McCallum, A. *Energy and policy considerations for deep learning in NLP*. arXiv preprint arXiv:1906.02243. 2019.
3. Goel, A., Tung, C., Lu, Y. H., and Thiruvathukal, G. K. *A Survey of Methods for Low-Power Deep Learning and Computer Vision*. arXiv preprint arXiv:2003.11066. 2020.
4. Conti, F., Rusci, M., and Benini, L. *The Memory Challenge in Ultra-Low Power Deep Learning*. In NANO-CHIPS 2030 (pp. 323-349). Springer, Cham. 2020.

What theoretical challenges in Deep Learning will we study?



Models

Let $\mathcal{X} \subseteq \mathcal{X}^\circ$ be parameter domains, where $\mathcal{X}$ is known. Define

1. $\mathbf{x}^\circ \in \arg\min_{\mathbf{x}\in\mathcal{X}^\circ} R(\mathbf{x})$: true minimum risk model
2. $\mathbf{x}^\natural \in \arg\min_{\mathbf{x}\in\mathcal{X}} R(\mathbf{x})$: assumed minimum risk model
3. $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} R_n(\mathbf{x})$: ERM solution
4. $\mathbf{x}^t$: numerical approximation of $\mathbf{x}^\star$ at time t

Practical performance in Deep Learning

$$\underbrace{R(\mathbf{x}^t) - R(\mathbf{x}^\circ)}_{\bar{\varepsilon}(t,n)} \leq \underbrace{R_n(\mathbf{x}^t) - R_n(\mathbf{x}^\star)}_{\text{optimization error}} + 2\underbrace{\sup_{\mathbf{x}\in\mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})|}_{\text{worst-case generalization error}} + \underbrace{R(\mathbf{x}^\natural) - R(\mathbf{x}^\circ)}_{\text{model error}}$$

where $\bar{\varepsilon}(t,n)$ denotes the total error of the Learning Machine. In Deep Learning applications

1. Optimization error is almost zero, in spite of non-convexity. $\Rightarrow$ lecture 9
2. We expect large generalization error. It does not happen in practice. $\Rightarrow$ lecture 7 (this one) and 8
3. Large architectures + inductive bias might lead to small model error.

Generalization error bounds

The value of $|R(\mathbf{x}) - R_n(\mathbf{x})|$ is called the *generalization error* of the parameter $\mathbf{x}$.

Goal: obtain generalization bounds for multi-layer, fully-connected neural networks

We want to find high-probability upper bounds for the worst case generalization error over a class $\mathcal{X}$:

$$\sup_{\mathbf{x}\in\mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})|$$

Main tool: concentration inequalities!

- Measure of how far is an empirical average from the true mean

Theorem (Hoeffding's Inequality [6])

Let $Y_1, \ldots, Y_n$ be i.i.d. random variables with Y_i taking values in the interval $[a_i, b_i] \subseteq \mathbb{R}$ for all $i = 1, \ldots, n$. Let $S_n := \frac{1}{n}\sum_{i=1}^n Y_i$. It holds that

$$\mathbb{P}\left(|S_n - \mathbb{E}[S_n]| > t\right) \leq 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Warmup: Generalization bound for a singleton

Lemma

For $i = 1, \dots, n$ *let* $(\mathbf{a}_i, b_i) \in \mathbb{R}^p \times \{-1, 1\}$ *be independent random variables and* $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ *be a function parametrized by* $\mathbf{x} \in \mathcal{X}$. *Let* $\mathcal{X} = \{\mathbf{x}_0\}$ *and* $L(h_{\mathbf{x}}(\mathbf{a}), b) = \{\textit{sign}(h_{\mathbf{x}}(\mathbf{a})) \neq b\}$ *be the* 0-1 *loss.*
With probability at least $1 - \delta$, *we have that*

$$\sup_{\mathbf{x} \in \mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})| = |R(\mathbf{x}_0) - R_n(\mathbf{x}_0)| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

Proof.

Note that $\mathbb{E}[\frac{1}{n} \sum_{i=1}^n L(h_{\mathbf{x}_0}(\mathbf{a}_i), b_i)] = R(\mathbf{x}_0)$, the expected risk of the parameter $\mathbf{x}_0$. Moreover $L(h_{\mathbf{x}_0}(\mathbf{a}_i), b_i) \in [0, 1]$. We can use Hoeffding's inequality and obtain

$$\mathbb{P}(|R_n(\mathbf{x}_0) - R(\mathbf{x}_0)| > t) = \mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^n L_i(h_{\mathbf{x}_0}(\mathbf{a}_i), b_i) - R(\mathbf{x}_0)\right| > t\right) \leq 2\exp\left(-2nt^2\right)$$

Setting $\delta := 2\exp\left(-2nt^2\right)$, we have that $t = \sqrt{\frac{\ln 2/\delta}{2n}}$, thus obtaining the result.

□

Generalization bound for finite sets

Lemma

For $i = 1, \dots, n$ let $(\mathbf{a}_i, b_i) \in \mathbb{R}^p \times \{-1, 1\}$ be independent random variables and $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$ be a function parametrized by $\mathbf{x} \in \mathcal{X}$. Let $\mathcal{X}$ be a finite set and $L(h_{\mathbf{x}}(\mathbf{a}), b) = \{\text{sign}(h_{\mathbf{x}}(\mathbf{a})) \neq b\}$ be the 0-1 loss.
With probability at least $1 - \delta$, we have that

$$\sup_{\mathbf{x} \in \mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})| \leq \sqrt{\frac{\ln |\mathcal{X}| + \ln(2/\delta)}{2n}}.$$

Proof.

Let $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{|\mathcal{X}|}\}$. We can use a union bound and the analysis of the singleton case to obtain:

$$\mathbb{P}(\exists j : |R_n(\mathbf{x}_j) - R(\mathbf{x}_j)| > t) \leq \sum_{j=1}^{|\mathcal{X}|} \mathbb{P}(|R_n(\mathbf{x}_j) - R(\mathbf{x}_j)| > t) = 2|\mathcal{X}| \exp\Big(-2nt^2\Big)$$

Setting $\delta := 2|\mathcal{X}| \exp\left(-2nt^2\right)$ we have that $t = \sqrt{\frac{\ln |\mathcal{X}| + \ln \frac{2}{\delta}}{2n}}$, thus obtaining the result. $\square$

Generalization bounds for infinite classes - The Rademacher complexity

However, in most applications in ML/DL we optimize over an infinite parameter space $\mathcal{X}$!

◦ A useful notion of *complexity* to derive generalization bounds for infinite classes of functions:

Definition (Rademacher Complexity [2])

Let $S = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \subseteq \mathbb{R}^p$ and let $\{\sigma_i : i = 1, \ldots, n\}$ be independent Rademacher random variables i.e., taking values uniformly in $\{-1, +1\}$ (coin flip). Let $\mathcal{H}$ be a class of functions of the form $h : \mathbb{R}^p \to \mathbb{R}$. The Rademacher complexity of $\mathcal{H}$ with respect to A is defined as follows:

$$\mathcal{R}_A(\mathcal{H}) := \mathbb{E} \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(\mathbf{a}_i).$$

◦ $\mathcal{R}_A(\mathcal{H})$ measures how well can we fit random signs (± 1) with the output of an element of $\mathcal{H}$ on the set A.

Visualizing Rademacher complexity

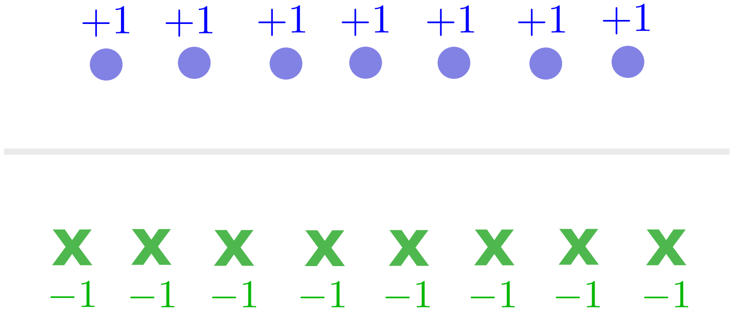
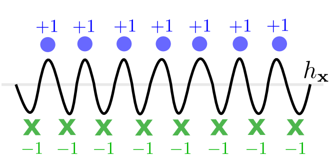
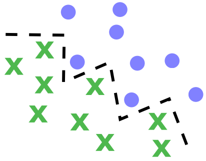


Figure: Rademacher complexity measures correlation with random signs

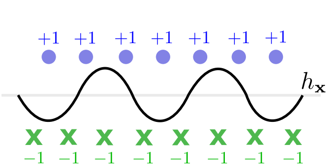
Visualizing Rademacher complexity



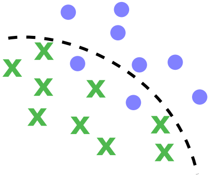
(a) High Rademacher Complexity



(b) Large Generalization error (memorization)



(c) Low Rademacher Complexity



(d) Low Generalization error

Figure: Rademacher complexity and Generalization error

A fundamental theorem about the Rademacher Complexity

Theorem (See Theorem 3.3 and 5.8 in [6])

Suppose that the loss function has the form $L(h_{\mathbf{x}}(\mathbf{a}), b) = \phi(b \cdot h_{\mathbf{x}}(\mathbf{a}))$ *for a* 1*-Lipschitz function* $\phi : \mathbb{R} \to \mathbb{R}$*.*

Let $\mathcal{H}_{\mathcal{X}} := \{h_{\mathbf{x}} : \mathbf{x} \in \mathcal{X}\}$ *be a class of parametric functions* $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$*. For any* $\delta > 0$*, with probability at least* $1 - \delta$ *over the draw of an i.i.d. sample* $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$*, letting* $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$*, the following holds:*

$$\sup_{\mathbf{x} \in \mathcal{X}} |R_n(\mathbf{x}) - R(\mathbf{x})| \le 2\mathbb{E}_A \mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) + \sqrt{\frac{\ln(2/\delta)}{2n}},$$

$$\sup_{\mathbf{x} \in \mathcal{X}} |R_n(\mathbf{x}) - R(\mathbf{x})| \le 2\mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) + 3\sqrt{\frac{\ln(4/\delta)}{2n}}.$$

The assumption is satisfied for common losses

- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \log(1 + \exp(-b \cdot h_{\mathbf{x}}(\mathbf{a}))) \Rightarrow \phi(z) := \log(1 + \exp(z))$ (logistic loss)
- $L(h_{\mathbf{x}}(\mathbf{a}), b) = \max(0, 1 - b \cdot h_{\mathbf{x}}(\mathbf{a})) \Rightarrow \phi(z) := \max(0, 1 - z)$ (hinge loss)

Computing the Rademacher complexity for Linear functions

Theorem

Let $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^p : \|\mathbf{x}\|_2 \leq \lambda\}$ and let $\mathcal{H}_{\mathcal{X}}$ be the class of functions of the form $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}, h_{\mathbf{x}}(\mathbf{a}) = \langle \mathbf{x}, \mathbf{a} \rangle$, for some $\mathbf{x} \in \mathcal{X}\}$. Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \subseteq \mathbb{R}^p$ such that $\max_{i=1,\ldots,n} \|\mathbf{a}_i\| \leq M$. It holds that $\mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) \leq \lambda M/\sqrt{n}$.

Proof.

$$
\begin{aligned}
\mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) &= \mathbb{E} \sup_{\|\mathbf{x}\|_2 \leq \lambda} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \langle \mathbf{x}, \mathbf{a} \rangle \\
&= \mathbb{E} \sup_{\|\mathbf{x}\|_2 \leq \lambda} \frac{1}{n} \left\langle \mathbf{x}, \sum_{i=1}^{n} \sigma_i \mathbf{a} \right\rangle \\
&\leq \frac{1}{n} \lambda \mathbb{E} \left\| \sum_{i=1}^{n} \sigma_i \mathbf{a}_i \right\|_2 \qquad \text{(C-S)}
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow \mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) &\leq \frac{1}{n} \lambda \left(\mathbb{E} \sum_{i=1}^{n} \|\sigma_i \mathbf{a}_i\|_2^2 \right)^{1/2} \qquad \text{(Jensen)} \\
&\leq \frac{1}{n} \lambda \left(\sum_{i=1}^{n} \|\mathbf{a}_i\|_2^2 \right)^{1/2} \\
&\leq \lambda M/\sqrt{n}
\end{aligned}
$$

□

Rademacher complexity estimates of fully connected Neural Networks

Notation

For a matrix $\mathbf{X} \in \mathbb{R}^{n,m}$, $\|\mathbf{X}\|$ denotes its spectral norm. Let $\mathbf{X}_{:,k}$ be the k-th column of $\mathbf{X}$. We define

$$\|\mathbf{X}\|_{2,1} = \|(\|\mathbf{X}_{:,1}\|_2, \ldots, \|\mathbf{X}_{:,m}\|_2)\|_1. \tag{2}$$

Theorem (Spectral bound [1])

For positive integers $p_0, p_1, \ldots, p_d = 1$*, and positive reals* $\lambda_1, \ldots, \lambda_d$ $\nu_1, \ldots, \nu_d$ *define the set*

$$\mathcal{X} := \{(\mathbf{X}_1, \ldots, \mathbf{X}_d) : \mathbf{X}_i \in \mathbb{R}^{p_i \times p_{i-1}}, \|\mathbf{X}_i\| \leq \lambda_i, \|\mathbf{X}_i^T\|_{2,1} \leq \nu_i\}.$$

Let $H_{\mathcal{X}}$ *be the class of neural networks* $h_{\mathbf{x}} : \mathbb{R}^p \to \mathbb{R}$, $h_{\mathbf{x}} = \mathbf{X}_d \circ \sigma \circ \ldots \circ \sigma \circ \mathbf{X}_1$ *where* $\mathbf{x} = (\mathbf{X}_1, \ldots, \mathbf{X}_d) \in \mathcal{X}$*. Suppose that* σ *is 1-Lipschitz. Let* $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \subseteq \mathbb{R}^p$, $M := \max_{i=1,\ldots,n} \|\mathbf{a}_i\|$ *and* $W := \max\{p_i : i = 0, \ldots, d\}$.

The Rademacher complexity of $\mathcal{H}_{\mathcal{X}}$ *with respect to* A *is bounded as*

$$\mathcal{R}_A(\mathcal{H}_{\mathcal{X}}) = \mathcal{O}\left(\frac{\log(W)M}{\sqrt{n}} \prod_{i=1}^{d} \lambda_i \left(\sum_{j=1}^{d} \frac{\nu_j^{2/3}}{\lambda_j^{2/3}}\right)^{3/2}\right) \tag{3}$$

How well do complexity measures correlate with generalization?

name	definition	correlation[1]
Frobenius distance to initialization [7]	$\sum_{i=1}^{d} \|\mathbf{X}_i - \mathbf{X}_i^0\|_F^2$	-0.263
Spectral complexity[2] [1]	$\prod_{i=1}^{d} \|\mathbf{X}_i\| \left(\sum_{i=1}^{d} \frac{\|\mathbf{X}_i\|_{2,1}^{3/2}}{\|\mathbf{X}_i\|^{3/2}} \right)^{2/3}$	$\mathbf{-0.537}$
Parameter Frobenius norm	$\sum_{i=1}^{d} \|\mathbf{X}_i\|_F^2$	0.073
Fisher-Rao [5]	$\frac{(d+1)^2}{n} \sum_{i=1}^{n} \langle \mathbf{x}, \nabla_{\mathbf{x}} \ell(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \rangle$	0.078
Path-norm [8]	$\sum_{(i_0,\ldots,i_d)} \prod_{j=1}^{d} \left(\mathbf{X}_{i_j, i_{j-1}}\right)^2$	$\mathbf{0.373}$

Table: Complexity measures compared in the empirical study [4], and their correlation with generalization

Complexity measures are still far from explaining generalization in Deep Learning!

[1] Kendall's rank correlation coefficient.

[2] The definition in [4] differs slightly.

Wrap up!

- Deep learning recitation on Friday!

References I

[1] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky.
Spectrally-normalized margin bounds for neural networks.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6240–6249. Curran Associates, Inc., 2017.

[2] Peter L Bartlett and Shahar Mendelson.
Rademacher and gaussian complexities: Risk bounds and structural results.
Journal of Machine Learning Research, 3(Nov):463–482, 2002.

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

[4] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio.
Fantastic generalization measures and where to find them.
In *International Conference on Learning Representations*, 2020.

[5] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes.
Fisher-rao metric, geometry, and complexity of neural networks.
volume 89 of *Proceedings of Machine Learning Research*, pages 888–896. PMLR, 16–18 Apr 2019.

References II

[6] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar.
Foundations of Machine Learning.
The MIT Press, 2nd edition, 2018.

[7] Vaishnavh Nagarajan and J. Zico Kolter.
Generalization in Deep Networks: The Role of Distance from Initialization.
arXiv e-prints, page arXiv:1901.01672, January 2019.

[8] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro.
Norm-based capacity control in neural networks.
In *Conference on Learning Theory*, pages 1376–1401, 2015.