Rethinking classic learning theory in deep neural networks

Presentation by Hikaru Ibayashi

Notes by Sai Anuroop Kesanapalli

CSCI 699, Spring 2023

Instructors: Prof. Shaddin Dughmi and Prof. Vatsal Sharan

University of Southern California

February 15, 2023

Contents

§1 Overview

We will be going through the following two papers in these notes:

- **Understanding deep learning requires rethinking generalization** [1]: Suggests an experiment that makes us *rethink* classic learning theory
- **Uniform convergence may be unable to explain generalization in deep learning** [2]: Proposes a learning task where classic learning theory *provably fails*

These two papers are representative of the line of thought that the current classic learning theory is not sufficient to explain many mysterious phenomena of deep neural networks, especially generalization.

§2 Understanding deep learning requires rethinking generalization

We first introduce some notation for understanding relevant concepts presented in this paper.

§2.1 Notation for classification tasks

Notation 2.1

Consider the following.

- $x \in \mathcal{X}$: input
- $y \in \mathcal{Y}$: label
- $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$: training set
- $\mathcal{H}$: hypothesis class
- $\mathcal{A}$: learning algorithm, a function from $\mathcal{S}$ to $\mathcal{H}$
- $\mathcal{L}_{\mathcal{S}}(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(x_i), y_i)$: train loss
- $\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(h(x), y)]$: test loss
- $\Delta h = \mathcal{L}_{\mathcal{D}}(h) - \mathcal{L}_{\mathcal{S}}(h)$: generalization error

The question we ask here is: Can we give a bound to Δh?

Before answering this question, let us first revisit a couple of popular complexity measures, namely *VC Dimension* and *Rademacher Complexity*.

§2.2 Complexity measures and generalization error bound

What are complexity measures? These are measures of how complex a hypothesis class is. Intuitively, the less complex the measure is, the more generalizing a hypothesis class is.

[1] Zhang et al., ICLR 2017: https://openreview.net/pdf?id=Sy8gdB9xx

[2] Nagarajan et al., NeurIPS 2019: https://proceedings.neurips.cc/paper/2019/file/05e97c207235d63ceb1db43c60db7bbb-Paper.pdf

Definition 2.2 (VC Dimension) — Denoted by $VC(\mathcal{H})$, it is the maximal size of a set $\mathcal{C} \subset \mathcal{X}$ that can be shattered by the hypothesis class $\mathcal{H}$. It roughly corresponds to the number of parameters.

Bound: $\Delta h \leq \frac{1}{\delta}\sqrt{\frac{2VC(\mathcal{H})}{m}}$ with probability $1-\delta$ [3]

Definition 2.3 (Rademacher Complexity) — It tells how wrong a hypothesis could be and is defined as follows.
$RC(l \circ \mathcal{H}, \mathcal{S}) := \frac{1}{m}\mathbb{E}_{\sigma \sim \{\pm 1\}^m}\Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i l(h(x_i), y_i)\Big]$

Bound: $\Delta h \leq 2\mathbb{E}_{\mathcal{S} \sim \mathcal{D}^m}\Big[RC(l \circ \mathcal{H}, \mathcal{S})\Big]$ [4]

This paper suggests that when it comes to deep neural network (DNN) models, we need to *rethink* these complexity measures.

Discussion 2.4

Here are some relevant questions asked on the topic of this subsection:

- **Is the VC dimension too high?**
  It can be infinite in some cases.
- **Do we have enough training samples m?**
  Classic learning theory implicitly expects that m is sufficiently large.
- **Is m << number of model parameters?**
  Yes, that is the common scenario in modern deep learning. In this sense, we could say deep learning is beyond the scope of what classic learning theory naturally expects.

We are now ready to go through the randomization test that this paper had conducted.

§2.3 Randomization test

First, the paper reports zero training error and strong generalization achieved by various models (Inception, AlexNet and MLP) on CIFAR-10 dataset (let us name these models as being *correct*). However, they then randomly shuffle the labels of the dataset and then observe that the models yet again achieve zero training error but now have no generalization at all (let us call these models as being *haywire*). This result is surprising as the models seem to fit both the real and the randomized dataset perfectly. The paper then uses this experiment to highlight the potential drawbacks of the aforementioned two complexity measures, as discussed next.

[3] Theorem 6.11,
[4] Theorem 26.3 in,
Understanding Machine Learning: From Theory to Algorithms:
https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

Discussion 2.5

Here are some relevant questions asked on the topic of this subsection:

- **Is the top-5 test accuracy of AlexNet on ImageNet with random labels really better than chance? It says 0.56 accuracy.**
  That is a non-significant difference out of randomness. Since the experiments are randomization tests, one could expect test accuracies to be ~ 0.5.

§2.4 Failure of classic complexity measures

Firstly, VC dimension cannot distinguish between the *correct* and *haywire* models as they both have the same value of $VC(\mathcal{H})$. To shed more light, let us compute the bound on generalization error using VC dimension, for the case of AlexNet ($\sim 62,000,000$ parameters) trained on CIFAR-10 dataset ($50,000$ training samples), and letting $\delta = 0.01$:

$$\Delta h \leq \frac{1}{0.01}\sqrt{\frac{2 \times 62,000,000}{50,000}} = 4979.96$$

Now coming to Rademacher complexity, recall that it measures how wrong h could be. Randomization test earlier showed us that h could go horribly wrong, as in the case of *haywire* models. To bring more perspective, consider the following result:

$$RC(l \circ \mathcal{H}, \mathcal{S}) = \frac{1}{m}\mathbb{E}_{\sigma \sim \{\pm 1\}^m}\Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i l(h(x_i), y_i)\Big] = 1$$

$$\Delta h \leq 2\mathbb{E}_{\mathcal{S} \sim \mathcal{D}^m}\Big[RC(l \circ \mathcal{H}, \mathcal{S})\Big] = 2 \cdot 1$$

$$\text{Since } -1 \leq \Delta h \leq 1, \text{ this bound is } \textit{vacuous}!$$

Finally, this paper also comments on the effectiveness of regularization on generalization as presented next.

§2.5 Role of regularization

In practice, it is common to use regularization for achieving generalization. Some *explicit* techniques are data augmentation, weight decay and dropout. However, this paper suggests that these are neither *necessary* nor *sufficient*. Whereas, there are some tricks that are a commonplace but have *implicit* regularization effect. Some of these are early stopping and batch normalization. Further, this paper suggests that there could some *unidentified implicit* regularization effects of the training algorithm $\mathcal{A}$ itself. In retrospect, the last suggestion invoked many research attempts that tried to identify those *unidentified implicit* regularization effects of the training algorithm.

To summarize, this was a breakthrough paper which cast light on the fact that the models can fit both the real dataset and randomized ones, and that the current complexity measures are incapable of distinguishing these models. It also opened avenues of research to identify the biases that the training algorithms themselves carry. We next discuss one such response to this paper.

Discussion 2.6

Here are some relevant questions asked on the topic of this subsection:

- **What does one mean by saying that the regularization techniques are not *sufficient*?**
  Let us clarify what the necessary and sufficient conditions mean in this context.

  1. Necessary regularizer:
     A model is generalizing $\implies$ Regularizer $\mathcal{R}$ is used
  2. Sufficient regularizer:
     Regularizer $\mathcal{R}$ is used $\implies$ A model is generalizing

  The necessary condition of the presented regularizer is clearly falsified in this paper because the model is generalizing even if we remove any of the regularizers. But as is asked, it is unclear to see if the sufficient condition is falsified in this paper. In their experiments, they do not provide such an experiment where a model is not generalizing even with one of the regularizers. However, we imagine it is difficult to falsify the sufficient condition. For example, even if we use weight decay, it would be easy to construct a bad model by adjusting other configurations, such as the initial weights, number of epochs and so on. Therefore, it is reasonable to think that none of the presented regularizers meets sufficient condition, although we do not feel that the claim in the paper is well-defined.

- **Are training algorithms for deep learning implicitly biased?**
  Yes. For example, it is now known that *stochastic gradient descent* has a bias towards minimum norm solutions and in moderate/ annealing regimes of learning rates, it converges along the direction of large eigenvalues of the data matrix whereas *gradient descent* converges along the directions of small eigenvalues.

- **If the data is simple, does the model return a simple hypothesis, and if the data is essentially noise, does the model have preference towards complex hypotheses?**

- **Are the *unidentified implicit regularization* effects stated with respect to optimizers such as SGD/ Adam or is this a more general statement?**
  In retrospect, we think this is a more general statement. However, in the paper (refer Section 5) these effects are illustrated with a simple linear model trained using SGD. Further, recent line of works tried to identify more such effects in these optimizers.

- **When training on random labels, is the model generalizing to something other than the true distribution?**
  The answer is No. In their randomization test, the training data and labels are completely independent. Thus there is zero information to learn from the training dataset. We also encourage the readers to refer to papers on *behavioral memorization* that one of the questioners had pointed out during this discussion.

§3 Uniform convergence may be unable to explain generalization in deep learning

§3.1 Key claim

Let us first define *uniform convergence bound*.

Definition 3.1 — The *uniform convergence bound* with respect to loss $\mathcal{L}$ is the smallest value $\epsilon_{unif}(m, \delta)$ such that:
$\mathbb{P}_{\mathcal{S}\sim\mathcal{D}^m}\Big[\sup_{h\in\mathcal{H}} |\mathcal{L}_{\mathcal{D}}(h) - \hat{\mathcal{L}}_{\mathcal{S}}(h)| \leq \epsilon_{unif}(m, \delta)\Big] \geq 1 - \delta$

Recall that the previous paper suggested to us that, "the entire hypothesis class $\mathcal{H}$ is too big, so we should think about an algorithm-dependent $\mathcal{H}_{\mathcal{A}}$, otherwise, we will only get vacuous bounds." Now, this paper suggests that, "even if we consider the tightest possible algorithm-dependent $\mathcal{H}_{\mathcal{A}}$, uniform convergence bound still remains vacuous!" It further states that, "there exists a learning task where $\mathcal{A}$ can find a generalizing h, but the uniform convergence bound is vacuous."

Let us next formally define what a *tightest algorithm-dependent uniform convergence bound is*.

§3.2 Tightest algorithm-dependent uniform convergence bound

Definition 3.2 — The *tightest algorithm-dependent uniform convergence bound* with respect to loss $\mathcal{L}$ is the smallest value $\epsilon_{\text{unif-alg}}(m, \delta)$ for which there exists a set of sample sets $\mathcal{S}_\delta$ such that: $\mathbb{P}_{\mathcal{S}\sim\mathcal{D}^m}[\mathcal{S} \in \mathcal{S}_\delta] \geq 1 - \delta$ and if we define the space of hypotheses explored by $\mathcal{A}$ on $\mathcal{S}_\delta$ as $\mathcal{H}_\delta := \bigcup_{\mathcal{S}\in\mathcal{S}_\delta}\{h_{\mathcal{S}}\} \subseteq \mathcal{H}$, the following holds:
$\sup_{\mathcal{S}\in\mathcal{S}_\delta} \sup_{h\in\mathcal{H}_\delta} |\mathcal{L}_{\mathcal{D}}(h) - \hat{\mathcal{L}}_{\mathcal{S}}(h)| \leq \epsilon_{\text{unif-alg}}(m, \delta)$.

In the above definition, $\sup_{\mathcal{S}\in\mathcal{S}_\delta} \sup_{h\in\mathcal{H}_\delta} |\mathcal{L}_{\mathcal{D}}(h) - \hat{\mathcal{L}}_{\mathcal{S}}(h)| \leq \epsilon_{\text{unif-alg}}(m, \delta)$ is the largest possible generalization error in $\mathcal{H}_\delta$.

Discussion 3.3

Here are some relevant questions asked on the topic of this subsection:

- **In the above definition, are we fixing a distribution?**
  Yes.
- **Is there any assumption that the set $\mathcal{S}_\delta$ or $\mathcal{H}$ is *closed*?**
  No. The figures in the slides are just for illustration.
- **How can one define $\mathcal{S}_\delta$ in the above manner? Can there not be multiple such $\mathcal{S}_\delta$'s?**
  This is an existence definition.

We next provide a failure mode of *uniform convergence bound* as suggested in the paper.

§3.3 A setup where UC bound provably fails

Notation 3.4

Consider the following setup.

- $x \in \mathcal{X} : x = (x_1, x_2)$ where $x_1 \in \mathbb{R}^K$, $x_2 \in \mathbb{R}^D$, K is a small constant and D is large
- $y \in \mathcal{Y} = \{-1, +1\}$
- Given a fixed vector u such that $\|u\|_2 = \frac{1}{\sqrt{m}}$, $x_1 = 2 \cdot y \cdot u$, $x_2 \sim \mathcal{N}\left(0, \frac{32}{D} I\right)$
- $h \in \mathcal{H} : w = (w_1, w_2) \in \mathbb{R}^{K+D}$, $h_w(x) = w_1 x_1 + w_2 x_2$
- $\mathcal{A}$: gradient descent
- $$\mathcal{L}^{(\gamma)}(y', y) = \begin{cases} 1, & \text{if } yy' \le 0 \\ 1 - \frac{yy'}{\gamma}, & \text{if } yy' \in (0, \gamma) \\ 0, & \text{if } yy' \ge \gamma \end{cases}$$

Discussion 3.5

Here are some relevant questions asked on the topic of this subsection:

- **In this setup, if *uniform convergence bound* fails, does there exist some other concept which succeeds?**
  We do not know yet. However, there are some papers which propose *stability* as a more fundamental concept (of which *uniform convergence bound* is a special case) that could explain generalization even when *uniform convergence bound* fails in doing so.
- **One would ideally want w_2 to be 0. What is the ideal hypothesis in this setting?**
  Yes, the ideal hypothesis is $w_1 = u$ and $w_2 = 0$.
- **Is this setup specific to *gradient descent*? In other words, does the main theorem hold even if we use an oracle optimizer that can always find a solution with zero train error? Related to that, is this a convex problem?**
  We are not sure and plan to investigate further. This is not a convex problem as $\mathcal{L}^{(\gamma)}(y', y)$ (*ramp* loss) used is non-convex.
- **Did the authors of the paper concoct this loss function to drive their point through this setup?**
  We suspect so, but are not fully sure. Again, we plan to investigate a little further. This could be designed for making the problem solvable by gradient descent.
- **What is the difference between this work and other works which**

propose that *empirical risk minimizers* do not generalize well but some other concepts such as *stability* explain better?
This work highlights failure modes of *uniform convergence bound* based approaches. However, they do not propose any alternatives to it.

- **Is the statement *uniform convergence* implies generalizability in the supervised learning setup always true?**
For classification and regression settings, yes! But the statement is not true in general.

- **What does one exactly mean when they say that *uniform convergence* fails?**
This means that the bounds given by *uniform convergence* are vacuous. It would be more clear once you see the main theorem.

- **Is the loss function $\mathcal{L}^{(\gamma)}(y', y)$ a continuous approximation (surrogate) for $0-1$?**
Yes, it is robust variation of $0-1$ loss, often called the *ramp* loss.

We next state the main theorem of this paper.

§3.4 Main theorem

Theorem 3.6

For any $\epsilon, \delta > 0$, $\delta \leq 1/4$, when $D = \Omega\Big(\max\Big(m \ln \frac{m}{\delta}, m \ln \frac{1}{\epsilon}\Big)\Big)$, $\gamma \in [0,1]$, the $\mathcal{L}^{(\gamma)}$ loss satisfies $\epsilon_{\text{gen}}(m,\delta) \leq \epsilon$, while $\epsilon_{\text{unif-alg}}(m,\delta) \geq 1-\epsilon$. Furthermore, for all $\gamma \geq 0$, for the $\mathcal{L}^{(\gamma)}$ loss, $\epsilon_{\text{unif-alg}}(m,\delta) \geq 1 - \epsilon_{\text{gen}}(m,\delta)$.

In words, it says that

1. Gradient descent can find a generalizing h
2. But the *tightest algorithm-dependent uniform convergence bound* is vacuous

We now provide a sketch of the proof of the above theorem along with the visual illustration (Figure 1).

Proof. Consider the following.

- $\epsilon_{\text{unif-alg}}$ always comes with some $\mathcal{S}_\delta$
- But for any $\mathcal{S}_\delta$, we can find the following $\mathcal{S}_*$
  1. $\mathcal{S}_* \in \mathcal{S}_\delta$
  2. $\mathcal{S}'_* \in \mathcal{S}_\delta$, where $\mathcal{S}'_* = \{((x_1, -x_2), y) | ((x_1, x_2), y) \in \mathcal{S}_*\}$
  3. $h_{\mathcal{S}_*}$ has generalization error less than ϵ
  4. $h_{\mathcal{S}_*}$ completely misclassifies $\mathcal{S}'_*$

□

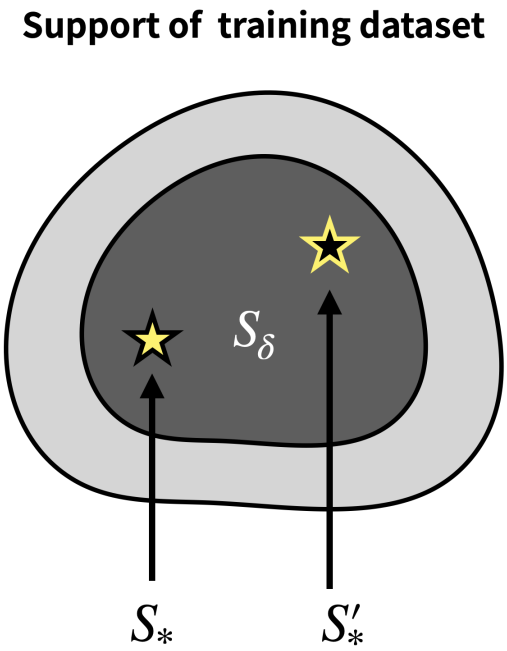

Figure 1: Illustration of the proof outline.

Discussion 3.7

Here are some relevant questions asked on the topic of this subsection:

- **How does one prove that *stochastic gradient descent* finds a solution that generalizes well?**
  Out of all models that exactly fit the data, it is known that SGD will often converge to the solution with minimum norm. However, this notion of minimum norm is not predictive of generalization performance. So while this minimum-norm intuition may provide some guidance, it is only a part of the larger generalization puzzle!
- **How does *stochastic gradient descent* behave when the optimization has a closed-form solution?**
  Even if the optimization has a closed-form solution and if the model is over-parameterized, of the many possible solutions, again, SGD has a preference towards minimum-norm solutions.
- **How does *stochastic gradient descent* behave in a convex setting?**
  Can think along similar lines as in the above answer.

We next provide an experiment suggested in the paper to illustrate that *uniform convergence bound* fails.

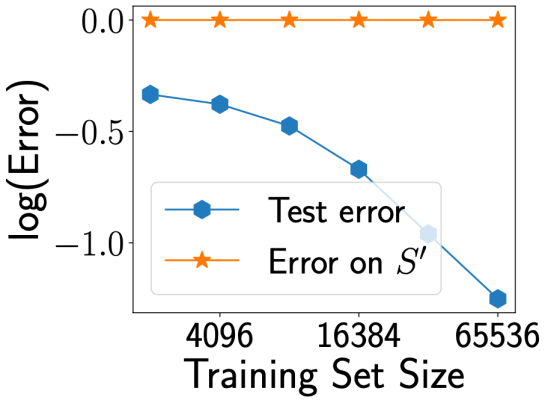

Figure 2: An experimental result from the paper that shows *uniform convergence* fails. Even though the test error (generalization error) decreases as the training size gets larger, the performance on S' stays constant. Since the uniform convergence bound is at least as loose as the performance on S', it is vacuous.

§3.5 An experiment where UC bound fails

Notation 3.8

Consider the following experiment.

- $\mathcal{H}$: two-layer ReLU networks (100,000 hidden units)
- $\mathcal{A}$: stochastic gradient descent
- $x \in \mathcal{X}$: a 1000-dimensional hypersphere with radius 1 and 1.1
- $y \in \mathcal{Y}$: $\{-1, +1\}$
- Generate $\mathcal{S}'$ by flipping the radii
- Notice that $\mathcal{S} \sim \mathcal{D}$ and $\mathcal{S}' \sim \mathcal{D}$

It is noted here that $h_{\mathcal{S}}$ generalizes well but performs poorly on $\mathcal{S}'$ (Figure 2).

Discussion 3.9

Here are some relevant questions asked on the topic of this subsection:

- **Can one always cook-up a dataset $\mathcal{S}'$ which always exactly challenges the hypothesis h returned by $\mathcal{A}$?**
  Yes.

Finally, we summarize the *deep learning conjecture* presented in the paper.

§3.6 Deep learning conjecture

- Over-parameterized deep networks mainly behave like a very simple (such as a linear) model and roughly fit the training data
- Plenty of parameters are unused but some of them learn *unnecessary knowledge* from training data
- Such *unnecessary knowledge* does not affect generalization performance, for example, even if one has knowledge that "the earth is flat", one can have normal conversations in 99% of their daily life and yet few people think that they are strange!
- However, one can always find a dataset where such *unnecessary knowledge* seriously affects the performance, which establishes a loose *uniform convergence bound*.

Discussion 3.10

Here are some relevant questions asked on the topic of this subsection:

- **Can one interpret *unnecessary knowledge* as *lack of knowledge* in the above context?**
  Yes and this could be better answered in the talk on *Memorization and Learning*.
- **Is the suggestion made in this paper to consider simple models a good one?**
  We believe so! It opened avenues for research into understanding how overparameterized deep neural networks behave like simpler models and how they *look* shallow to *gradient descent* family of optimizers.

THE END