CS 839: Theoretical Foundations of Deep Learning		**Spring 2022**
Lecture 2 Challenges in Theoretical Analysis of Deep Learning		
Instructor: Yingyu Liang	*Date: Jan 27th, 2022*	*Scriber: Yingyu Liang*

1 Elements of Statistical Learning Theory Framework

Let's review some basics of some traditional framework for analyzing supervised machine learning methods.

Let $\mathcal{X}$ denote the input space, and $\mathcal{Y}$ the label space. Typically, $\mathcal{X} \subseteq \mathbb{R}^d$, and $\mathcal{Y} = \{-1, +1\}$ for classification and $\mathcal{Y} = \mathbb{R}$ for regression. The learning algorithm is given a training data set $S = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, and a hypothesis/model class $\mathcal{H}$ with functions $h : \mathcal{X} \mapsto \mathcal{Y}$. Its goal is to find a function $h \in \mathcal{H}$ using the training data such that h can have good prediction performance on future test data.

Some connection between the training and test data is thus needed; the typical assumption is that they are all i.i.d. samples from some unknown ground-truth data distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

Some performance measure is also needed. Let $\ell(\widehat{y}, y) : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ be some loss function measuring the difference between the prediction $\widehat{y}$ and the true label y. The prediction performance of a function $h \in \mathcal{H}$ over the data distribution is called the risk (or expected/population risk, etc.):

$$R(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)] \tag{1}$$

and that over the training data set is called the empirical risk (or traing loss, etc.):

$$\widehat{R}_S(h) := \mathbb{E}_{(x,y)\sim S}[\ell(h(x), y)] = \frac{1}{|S|} \sum_{(x,y)\in S} \ell(h(x), y). \tag{2}$$

For binary classification, typical loss functions include

- 0-1 error $\ell(\widehat{y}, y) = \mathbb{I}[\widehat{y} \neq y]$,
- binary cross-entropy loss (or log loss) $\ell(\widehat{y}, y) = -y \log \widehat{y} - (1 - y) \log(1 - \widehat{y})$ (assuming $y \in \{0, 1\}$ and $\widehat{y}$ is the predicted probability for label 1),
- hinge-loss $\ell(\widehat{y}, y) = \max\{0, 1 - y\widehat{y}\}$ (assuming $y \in \{-1, +1\}$, and $\widehat{y} \in \mathbb{R}$ is a scoring function rather than the actual predicted label),
- logistic loss $\ell(\widehat{y}, y) = \log(1 + \exp(-y\widehat{y}))$ (assuming $y \in \{-1, +1\}$, $\widehat{y} \in \mathbb{R}$, and $1/(1 + \exp(-\widehat{y}))$ is the predicted probability for label 1. Equivalent to the cross-entropy loss up to scaling).

For regression, the typical loss function is $\ell(\widehat{y}, y) = (y - \widehat{y})^2$.

2 Risk Decomposition and Typical Analysis Paradigms

In key quantity in the analysis of machine learning is the risk: the goal is to get a function with low risk (i.e., good generalization performance). The analysis is typically divided into different aspects: approximation/representation, statistics/generalization, and optimization. Then one can thoroughly investigate each aspect using the corresponding tools. To motivate this, consider the following decomposition of the risk from [2].

Let's introduce some intermediate notions:

- g^*: the Bayesian optimal predictor (classifier or regression function) on $\mathcal{D}$
- h_{opt}: the optimal hypothesis for the risk (i.e., $h_{\text{opt}} \in \mathcal{H}$ and $R(h_{\text{opt}}) = \inf_{h \in \mathcal{H}} R(h)$)
- $\widehat{h}_{\text{opt}}$: the optimal hypothesis for the empirical risk (i.e., $\widehat{h}_{\text{opt}} \in \mathcal{H}$ and $\widehat{R}_S(\widehat{h}_{\text{opt}}) = \inf_{h \in \mathcal{H}} \widehat{R}_S(h)$)
- $\widehat{h}$: the hypothesis output by the learning algorithm

Then we can decompose the risk of $\widehat{h}$ as follows:

$$
\begin{aligned}
& R(\widehat{h}) - R(g^*) & \\
&= R(h_{\text{opt}}) - R(g^*) && \text{approximation} \\
&+ R(\widehat{h}_{\text{opt}}) - R(h_{\text{opt}}) && \text{estimation} \\
&+ R(\widehat{h}) - R(\widehat{h}_{\text{opt}}) && \text{optimization}
\end{aligned}
$$

$R(g^*)$ corresponds to the inherit noise in the data, and $R(\widehat{h}) - R(g^*)$ is called the excess risk. The first line corresponds to approximation: g^* is the best among all measurable functions while h_{opt} is the best among $\mathcal{H}$, so the difference reflects the approximation/representation power of the hypothesis class $\mathcal{H}$. The second line corresponds to estimation: $\widehat{h}_{\text{opt}}$ is the best using the sample while h_{opt} is the best using the distribution, so the difference reflects the effect of the sampling. The third line corresponds to optimization: $\widehat{h}$ is the hypothesis obtained by the learning algorithm while $\widehat{h}_{\text{opt}}$ can be viewed as the hypothesis obtained by an algorithm that returns the optimal solution for the optimization, so the difference reflects the effect of the optimization algorithm used for learning.

One can then ask the corresponding questions about each aspect separately. For example, study the approximation power of the hypothesis class: whether there exist $h \in \mathcal{H}$ with small risk? Or study only the statistical aspect: assuming there exist low risk (even 0 risk) functions in the hypothesis class, and an optimization oracle that can return the optimal solution for whatever optimization objective (e.g., Empirical Risk Minimization), how many samples are needed to learn a low risk function? Or study only the optimization aspect: on the optimization problem in training, does the algorithm get a near-optimal solution w.r.t. the optimization objective?

Typical Analysis Paradigms. Note that while the decomposition motivates these aspects, the different parts are still not exactly those quantities analyzed. For example, the last line is about the risk R, but when analyzing optimization alone, one would like to analyze the objective on the training data (the empirical risk $\widehat{R}_S$). That is, one would like to analyze $\widehat{R}_S(\widehat{h}) - \widehat{R}_S(\widehat{h}_{\text{opt}})$. We thus further decompose the risk. For simplicity, assume $R(g^*) = 0$ and $R(h_{\text{opt}}) = R(g^*)$. Then the risk decomposition above simplifies to

$$\begin{aligned} R(\widehat{h}) = R(\widehat{h}_{\text{opt}}) \\ + R(\widehat{h}) - R(\widehat{h}_{\text{opt}}). \end{aligned}$$

In fact, we can consider any function $h_r \in \mathcal{H}$ as a reference:

$$\begin{aligned} R(\widehat{h}) = R(h_r) \\ + R(\widehat{h}) - R(h_r). \end{aligned}$$

Introducing the intermediate quantities $\widehat{R}_S(\widehat{h})$ and $\widehat{R}_S(\widehat{h}_r)$, we have

$$\begin{aligned} & R(\widehat{h}) - R(h_r) \\ = {} & R(\widehat{h}) - \widehat{R}_S(\widehat{h}) && \text{generalization gap} \\ + {} & \widehat{R}_S(\widehat{h}) - \widehat{R}_S(h_r) && \text{optimization} \\ + {} & \widehat{R}_S(h_r) - R(h_r) && \text{concentration} \end{aligned}$$

Now consider the special case where $h_r = h_{\text{opt}}$. By bounding the gap between the risk and the empirical risk uniformly, and noting $\widehat{R}_S(h_{\text{opt}}) \geq \widehat{R}_S(\widehat{h}_{\text{opt}})$, we arrive at:

$$\begin{aligned} R(\widehat{h}) - R(h_{\text{opt}}) &\leq \widehat{R}_S(\widehat{h}) - \widehat{R}_S(h_{\text{opt}}) + 2\sup_{h\in\mathcal{H}} |\widehat{R}_S(h) - R(h)| \\ &\leq \widehat{R}_S(\widehat{h}) - \widehat{R}_S(\widehat{h}_{\text{opt}}) + 2\sup_{h\in\mathcal{H}} |\widehat{R}_S(h) - R(h)|. \end{aligned}$$

The first part $\widehat{R}_S(\widehat{h}) - \widehat{R}_S(\widehat{h}_{\text{opt}})$ is exactly the quantity typically studied in optimization, and the second part is exactly the quantity bounded by uniform convergence bounds (e.g., VC-dim theory). This motivates the typical paradigm in traditional machine learning analysis: separate the optimization and generalization; study the optimization on the training data usually using convex optimization tools; study the generalization gap usually using uniform convergence bounds from concentration inequalities.

Example. Consider binary classification by linear functions. $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{-1, +1\}$. For the data distribution, the label is given by a ground-truth linear classifier $y = \text{sign}(\langle w^*, x\rangle)$ with parameter w^*, and the input distribution can be any distribution on $\mathcal{X}$. The hypothesis class is linear classifiers: $\mathcal{H} = \{h_w(x) = \text{sign}(g_w(x)) : g_w(x) = \langle w, x\rangle, w \in \mathbb{R}^d\}$. The training uses hinge loss on g_w: $\ell(g_w(x), y) = \max\{0, 1 - yg_w(x)\}$ and returns $\widehat{h}$.

Consider the risk decomposition above, but apply it on 0-1 loss (instead of on hinge loss used in training, for reasons explained below). Note that $R(h_{\text{opt}}) = \widehat{R}_S(\widehat{h}_{\text{opt}}) = 0$, so

$$R(\widehat{h}) \leq \widehat{R}_S(\widehat{h}) + 2\sup_{h\in\mathcal{H}} |\widehat{R}_S(h) - R(h)|.$$

The first term $R_S(\widehat{h})$ is on 0-1 loss, which is upper-bounded by hinge loss used in training, so we only need to upper bound the average hinge loss on the training data (i.e., the training objective). This is convex, so with convex optimization tools, we can claim that the optimization can make $R_S(\widehat{h})$ to be (near) optimal which is 0. The second term can be bounded using VC-dim theory. Since the VC-dim of linear classifiers in $\mathbb{R}^d$ is $d+1$, we have $\sup_{h\in\mathcal{H}} |\widehat{R}_S(h) - R(h)| = \widetilde{O}(d/n)$ where n is the size of the training data set.[1] Therefore, we conclude $R(\widehat{h}) = \widetilde{O}(d/n)$.

(We use 0-1 loss in the risk decomposition so VC-dim bound can be applied on the second term. But 0-1 loss is not easy to optimize; rather, typically some convex surrogate upper bound like hinge loss is used in training so that convex optimization can be applied.)

3 New Challenges in Analyzing Deep Learning

Let's consider apply the above analysis paradigm to deep learning: separate the optimization and generalization and analyze them one by one. There are two new key challenges.

For the optimization, the challenge is clear: the optimization is non-convex in general. Even training a three-node network can be NP-complete in the worst case [1]. In contrast, the practical networks can be of hundreds of layers with millions of nodes but can be trained to small training losses with relatively simple algorithm (in particular, stochastic gradient descent).

We can still try to analyze the generalization part alone, assuming the optimization can be done to optimal. However, a surprising new challenge comes out. We can indeed use for example VC-dim or Rademacher complexity to get a uniform convergence bound. But it turns out that these bounds are vacuous: they are too loose to explain the practice superior generalization performance, because the practical networks are of high complexity and have very large VC-dim or Rademacher complexity. One may wonder maybe VC-dim and Rademacher complexity are not the right complexity notion; maybe one can look for another notion and then apply the uniform convergence bound.

However, [3] clearly demonstrated that practical networks can indeed be of high complexity. In fact, they are overparameterized: they have the capacity to fit any labellings on the training inputs. The following experiment was performed: replace the original labels in the training data with completely random labels; train the practical neural network on these random labeled data (with practical learning methods including regularization etc.). The key observation is that the networks can still get 0 training loss (though may take longer time to reach 0 loss than on original labeled data). The observation has several surprising implications:

- Practical neural networks are overparameterized. They indeed have high complexity, as high as can fit any labels on the training data.
- Even optimization on random labels remains easy. Previously it was conjectured by some that the optimization is easy because the ground-truth labeling function has

[1] This is for the realizable case when $R(h_{\text{opt}}) = 0$. For the non-realizable case when $R(h_{\text{opt}}) \neq 0$, the uniform convergence bound is $\widetilde{O}(\sqrt{d/n})$.

special structure. But this observation says even without structure the optimization can still be easy.

- Optimization automatically adapts to the structure of the data. With random labels, the network fits the training labels by memorization (i.e., no generalization). With practical labels with structure, it learns the underlying structure rather than pure memorization (i.e., good generalization).

It also means that the traditional analysis approach of decoupling optimization and generalization cannot work for deep learning. This is because for the training data (with the original labels), there can exist different 0 training loss solutions, and some have low risk but some have high risk.[2] Then decoupling optimization and generalization analysis cannot explain why the practical learning returns a solution of the first kind (with low risk) but not the second kind (with high risk). The analysis of optimization and generalization is thus interwove together.

A conjecture is that the optimization has some implicit regularization effect that restricts the learning dynamics to a subset of the whole hypothesis class. This algorithm-dependent effective subset is not of high capacity so can lead to good generalization. Then many fundamental questions arise: What is this subset? Why can the algorithm regularize the learning to this subset? Why can the algorithm find a 0 training loss solution in the subset? Why can this subset be so small to guarantee generalization? New perspectives for the analysis of deep learning are thus needed.

References

[1] Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[2] Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 161–168, 2007.

[3] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[2]Though bad-risk 0-training-loss solutions on the original labeled data are not directly observed in the experiments in [3], one can think of the following thought experiment: sample training data S from the data distribution $\mathcal{D}$, sample training data S' from the data distirbution $\mathcal{D}'$ which has the same input marginal distribution as $\mathcal{D}$ but flips the labels, then train on $S \cup S'$. One can get a network with 0 training loss on $S \cup S'$, so 0 training loss on S. But this network should not have low risk on both $\mathcal{D}$ and $\mathcal{D}'$, because $\mathcal{D}$ and $\mathcal{D}'$ are symmetric for the learning algorithm, and they have opposite labels. So this network has 0 training loss but high risk.