Theoretical Deep Learning

Lecture 1: Supervised learning

July 25, 2021

Lecturer: Lei Wu *Scribe: Lei Wu*

The task of machine learning is to discover useful information from the given finite data. Roughly speaking, there are three categories of learning problems.

- Supervised learning. We are given $S = \{(x_i, y_i)\}$, where y_i is the label of x_i. Our task is to learn the mapping from x to y using these samples.
- Unsupervised learning. We have only $S = \{x_i\}$ without any labels. The task is to discover some useful information about the underlying distribution of x. The definition of "useful information" varies with applications.
- Semi-supervised learning. We have two set of data: S_1 and S_2, where $S_1 = \{x_i\}$ are unlabeled, and $S_2 = \{(x_j, y_j)\}$ are labeled. The task of semi-supervised learning is either learning the input-output map with the help of unlabeled data or discovering the relevant information of the input distribution with the help of those labels.

This course will focus on the supervised learning setting. The unsupervised learning and semi-supervised learning are also important but more challenging.

1 Setup of supervised learning

Assume that the label is generated by

$$y_i = f^*(x_i) + \varepsilon_i,$$

where f^* is the target function and ε_i is the "measurement noise". Our task is to learn (i.e., approximate) f^*. There are two major categories of supervised learning problems:

- Regression problem: labels take continuous values.
- Classification problem: labels take the discrete values.

Let f be a hypothesis. The objective of supervised learning is to minimize the *population risk*

$$\mathcal{R}(f) := \mathbb{E}_{x,y}[\ell(f(x), y)].$$

However, in practice, we can only deal with the *empirical risk*:

$$\hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is the loss function.

- For regression problem, $\mathcal{Y} = \mathbb{R}$ and the popular loss function is $\ell(y_1, y_2) = (y_1 - y_2)^2$.

- For classification problem, $\mathcal{Y} = \{0, 1, \ldots, K\}$. The ideal loss function is the $0-1$ loss: $\ell(y_1, y_2) = 1(y_1 \neq y_2)$, i.e., 0 if $y_1 = y_2$ and 1 otherwise. Note that this loss is not differentiable and hence, one often use other surrogate loss functions.

The standard approach of supervised learning to minimize the regularized empirical risk

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{\mathcal{R}}_n(f) + \lambda \|f\|. \tag{1.1}$$

Here, $\mathcal{F}$ denotes the hypothesis space, which is the set of functions that can be represented by our model. $\|f\|$ measures the complexity of a candidate f, which usually encode our prior knowledge about the target function f^*. λ is the tradeoff hyperparameter. A classical example is the LASSO regression:

$$\hat{\beta}_n = \operatorname*{argmin}_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 + \lambda \|\beta\|_{\ell_1}.$$

In this case,

- $f(x) = \beta^T x$ and the hypothesis space is $\mathcal{F} = \{\beta^T x : \beta \in \mathbb{R}^d\}$.
- The prior knowledge is that the ground truth β^* is sparse. The learning procedure encodes this knowledge via penalizing the ℓ_1 norm of β.

However, our ultimate goal is not minimizing the fitting error at the n points. We would like to minimize the population risk. Notice that for a fixed f (independent of the training samples S)

$$\mathbb{E}_S[\hat{\mathcal{R}}_n(f)] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_S[\ell(f(x_i), y_i)] = \frac{1}{n} \cdot n\mathcal{R}(f) = \mathcal{R}(f), \tag{1.2}$$

which means that empirical risk is an unbiased estimator of the population risk. This explains why we want to minimize the empirical risk.

Unfortunately, (1.2) does not hold for the estimator $\hat{f}_n$, since it depends on the samples $(x_1, y_1), \ldots, (x_n, y_n)$. Only minimizing the empirical risk may yield a solution which generalizes badly on unseen data. This phenomenon is called *overfitting*. The reason why we need to add the regularization term is to prevent from selecting a overfitted solution. See Figure 1 for an illustration.

2 Generalization analysis via a priori estimates

There are two nature questions emerging for supervised learning.

- For the estimator 1.1, what kind of functions can be learned efficiently?
- How do we evaluate the performance of $\hat{f}_n$?

For the second question, the performance is characterized by the generalization error:

$$\|\hat{f}_n - f^*\|_{L^2(\mathbb{P}_x)}. \tag{2.1}$$

Here, we focus on the regression problem for simplicity. We need to understand how small (2.1) is.

Let $\tilde{f} = \operatorname{argmin}_{f \in \mathcal{F}} \mathbb{E}[(f(x) - f^*(x))^2]$. Then, the generalization error can be decomposed as follows:

$$\|\hat{f}_n - f^*\|_{L^2(\mathbb{P}_x)} \leq \underbrace{\|\hat{f}_n - \tilde{f}\|_{L^2(\mathbb{P}_x)}}_{\text{Estimation error}} + \underbrace{\|\tilde{f} - f^*\|_{L^2(\mathbb{P}_x)}}_{\text{Approximation error}}.$$

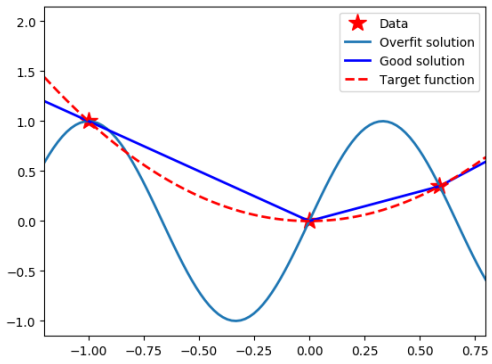

Figure 1: Illustration of overfitting.

- The approximation error is caused by the choice of our model. For instance, the target function is nonlinear but the model is linear, this will introduce an inevitable approximation error. More often, our model is parameterized by m parameters, e.g., $f(x;a) = \sum_{j=1}^{m} a_j \psi_j(x)$ where $\{\psi_j\}$ is a set of nonlinear basis functions. Then, the approximation error will decay with m. It is important to understand how fast the approximation error rate decays.

- The estimation error is because of the only accessibility of finite samples. This term decrease with the number of samples.

Consider a specific model: piecewise polynomials. It is well-known that the generalization error goes as follows

$$\|\hat{f}_n - f^*\|_{L^2(\mathbb{P}_x)} \lesssim \frac{\|f^*\|_{H^s}}{n^{s/d}}.$$

where H^s is the Sobolev space, s denotes the smoothness, and d is the input dimension. This means that for a given accuracy ε, we need $O(\varepsilon^{-d/s})$ samples. Unfortunately, this rate suffers from the *curse of dimensionality* (CoD), i.e., number of samples needed depends on the input dimension exponentially.

Generally, we would like to seek a similar estimate (bound) of the generalization error for neural networks:

$$\|\hat{f}_n - f^*\|_{L^2(\mathbb{P}_x)} \leq e(1/m, 1/n, \|f^*\|_*, d). \tag{2.2}$$

This estimate is a priori since the RHS only depends on the target function f^* not the solution $\hat{f}_n$ [1].

- Note that the $\|\cdot\|_*$ norm measures the complexity of target function. Obviously, simpler functions should be easier to estimate and approximate. The specific form of $\|\cdot\|_*$ depends on the choice of model $\mathcal{F}$ and the regularization.

  - Piecewise linear functions v.s. polynomials. The former works very well for less smooth functions and the latter works well for smooth functions.
  - For the linear regression, ridge works well for target with small ℓ_2 norm; lasso works well for target with small ℓ_1 norm.

[1] The bound (2.2) is also called *oracle inequality* in statistics.

- The dependence of d is crucial in the high-dimensional case. For a given model $\mathcal{F}$ and the regularization, we need to identify an appropriate function norm $\|\cdot\|_*$ such that the estimate 2.2 depends on d only polynomially. In other words, we seek to identify the target functions such that the estimator 1.1 can learn them efficiently.

Throughout this course, we will attempt to use the above framework to quantitatively answer the following questions:

- What kind of functions can be learned efficiently using neural networks?
- Why do neural networks outperform those traditional methods? Why do deep networks perform better than shallow networks?