Introduction

Mathematical Analysis of Machine Learning Algorithms
(Chapter 1)

Machine Learning Analysis

The goal of mathematical analysis of machine learning algorithms:

- Study the statistical behavior of learning algorithms.
- Study computationally efficient algorithms.

This course mainly focuses on the analysis of two common learning models:

- Supervised learning
- Online learning and sequential prediction

Supervised Learning

In supervised learning, we

- Observe an input random variable (feature vector) $X \in \mathbb{R}^d$ that represents the known information,
- Interested in output variable (label) Y that represents the unknown information which we want to predict.

The goal is to predict Y based on X.

Example 1 (Image Classification)

Predict whether an image (represented as input vector X) contains a cat or a dog (label Y).

Theoretical Questions

A mathematical theory for supervised learning answers the following basic questions, where we take linear model as example.

Theoretical Question for Supervised Learning

- (Generalization bound) Suppose that we learn a d-dimensional linear classifier with n training data by minimizing the training error. Assume that the training error is 10%. What is the classifier's test error on the test data? The test error in this setting is also referred to as the *generalization error*.
- (Oracle inequality) Can we learn a linear classifier that has a test error nearly as small as the optimal linear classifier?
- (Computational efficiency) Is there a computationally efficient procedure to find a linear classifier with small test error?

Online Learning

In online learning, we are interested in the sequential prediction problem, where we repeat the following process sequentially

- Train a statistical model using historic data
- Apply it to the data observed in the next time step.
- Observes the true outcome after prediction

Example 2 (Stock Prediction)

The opening prices of a certain stock at each trading day are $p_1, p_2, \ldots$. At the beginning of each day t, we observe $p_1, \ldots, p_t$, and want to predict p_{t+1} on day $t+1$, so that we use this prediction to trade the stock.

Theoretical Questions

A mathematical theory for online learning needs to answer the following basic questions, where we again take linear model as an example.

Theoretical Question for Online Learning (Regret Analysis)

In the online sequential prediction setting. Given a time step t, can we construct an online learning algorithm that predicts nearly as well as the optimal linear classifier up to time step t?

Model for Supervised Learning

In supervised learning, we have:

- Input random variable (feature vector) $X \in \mathbb{R}^d$ that represents the known information
- Output variable (label) Y that represents the unknown information which we want to predict.
- Prediction function $f(w, \cdot) : \mathbb{R}^d \to \mathbb{R}^k$, with model parameter w
- Loss function $L(f(w, X), Y)$.

The goal is to minimize loss $L(f(w, X), Y)$ on test data.

Prediction Model Class

In practice, the set of prediction rules are derived by parametrized functions $f(w, \cdot)$:

- $w \in \Omega$ is the model parameter that can be learned on the training data.

The prediction quality is measured by a loss function $L(f(x), y)$: the smaller the loss, the better the prediction accuracy.

Example 3

For k-class classification problem, where $Y \in \{1, \ldots, k\}$, we predict Y using the following prediction rule given function $f(w, x) = [f_1(w, x), \ldots, f_k(w, x)] \in \mathbb{R}^k$:

$$q(x) = \arg \max_{\ell \in \{1,\ldots,k\}} f_\ell(w, x).$$

Loss can be the classification error:

$$L(f(x), y) = \mathbb{1}(q(x) \neq y).$$

Learning Algorithm

The supervised learning approach is to estimate $\hat{w} \in \Omega$ based on observed (labeled) historical data $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$.

Definition 4

A supervised learning algorithm $\mathcal{A}$ takes a set of training data $\mathcal{D}_n$ as input, and outputs a function $f(\hat{w}, \cdot)$, where $\hat{w} = \mathcal{A}(\mathcal{D}_n) \in \Omega$.

The most common algorithm, which we will focus in this course, is *empirical risk minimization* (ERM):

$$\hat{w} = \arg\min_{w \in \Omega} \sum_{i=1}^{n} L(f(w, X_i), Y_i). \tag{1}$$

Training versus Test

In the standard theoretical model for analyzing supervised learning, we assume that the training data $\{(X_i, Y_i) : i = 1, \ldots, n\}$ are iid (independent and identically distributed) according to an unknown underlying distribution $\mathcal{D}$.

$$\text{training-loss}(\hat{w}) = \frac{1}{n} \sum_{i=1}^{n} L(f(\hat{w}, X_i), Y_i).$$

Moreover, we assume that the test data (X, Y) (future unseen data) are also taken from the same distribution $\mathcal{D}$.

We are interested in the generalization error of $\hat{f}$ on the test data:

$$\text{test-loss}(\hat{w}) = \mathbb{E}_{(X,Y)\sim\mathcal{D}} L(f(\hat{w}, X), Y).$$

Generalization Analysis

Since we only observe the training error of $\hat{f} = f(\hat{w}, \cdot)$, a major goal is to estimate the generalization error of $\hat{f}$ based on its training error, referred to as *generalization bound*.

Given $\epsilon \geq 0$, we want to determine $\delta_n(\epsilon)$ so that:

$$\Pr\left(\mathbb{E}_{(X,Y)\sim\mathcal{D}} L(f(\hat{w}, X), Y) \geq \frac{1}{n}\sum_{i=1}^{n} L(f(\hat{w}, X_i), Y_i) + \epsilon\right) \leq \delta_n(\epsilon),$$

where the probability is with respect to the randomness over the training data $\mathcal{D}_n$.

In general, $\delta_n(\epsilon) \to 0$ as $n \to \infty$.

Alternative Form

In the literature, the above result is often stated in the following alternative form.
We want to determine a function $\epsilon_n(\delta)$ of δ, with the following type of theorem statement.

With probability at least $1 - \delta$ (over training data $\mathcal{D}_n$):

$$\mathbb{E}_{(X,Y)\sim\mathcal{D}} L(f(\hat{w}, X), Y) \leq \frac{1}{n}\sum_{i=1}^{n} L(f(\hat{w}, X_i), Y_i) + \epsilon_n(\delta). \tag{2}$$

We hope $\epsilon_n(\delta) \to 0$ as $n \to \infty$.

Example 5

Let N be the number of parameters in Ω. We can take

$$\epsilon_n(\delta) = O\left(\sqrt{\frac{\ln(N/\delta)}{n}}\right)$$

under suitable conditions.

Oracle Inequality

Another type of inequalities, often referred to as *oracle inequality*, is to show the following type of results.

With probability at least $1 - \delta$ (over training data $\mathcal{D}_n$):

$$\mathbb{E}_{(X,Y)\sim\mathcal{D}} L(f(\hat{w}, X), Y) \leq \inf_{w\in\Omega} \mathbb{E}_{(X,Y)\sim\mathcal{D}} L(f(w, X), Y) + \epsilon_n(\delta). \tag{3}$$

This shows that the test error achieved by the learning algorithm is nearly as small as that of the optimal test error achieved by $f(w, x)$ with $w \in \Omega$.

Example 6

Let N be the number of parameters in Ω. We can take

$$\epsilon_n(\delta) = O\left(\frac{\ln(N/\delta)}{n}\right)$$

under suitable conditions.

Online Learning

Definition 7

An online learning algorithm $\mathcal{A}$ does the following:

- Learn a model parameter $\hat{w}_t$ at time t based on previously observed data $(X_1, Y_1), \ldots, (X_t, Y_t)$: $\hat{w}_t = \mathcal{A}(\{(X_1, Y_1), \ldots, (X_t, Y_t)\})$.
- Observe the next input vector X_{t+1}, and make prediction $f(\hat{w}_t, X_{t+1})$.
- After the prediction, we observe Y_{t+1}, and then compute the loss $L(f(\hat{w}_t, X_{t+1}), Y_{t+1})$.

The goal of online learning is to minimize the aggregated loss

$$\sum_{t=0}^{T-1} L(f(\hat{w}_t, X_{t+1}), Y_{t+1}).$$

Regret Analysis

In the mathematical analysis of online algorithm, we are interested in the following inequality, referred to as *regret bound*.

$$\mathrm{REG}_T = \sum_{t=0}^{T-1} L(f(\hat{w}_t, X_{t+1}), Y_{t+1}) - \inf_{w \in \Omega} \sum_{t=0}^{T-1} L(f(w, X_{t+1}), Y_{t+1}). \quad (4)$$

The regret REG_T, is the extra loss of the online learning algorithm compared to that of the optimal model at time T in retrospect. It is analogous to oracle inequality.

Example 8

Let N be the number of parameters in Ω. We can take

$$\mathrm{REG}_T(\delta) = O\left(\sqrt{T \ln N}\right)$$

under suitable conditions.

Example

Consider the stock price prediction problem, where the opening prices of a certain stock at each trading day are $p_1, p_2, \ldots$.

At the beginning of each day t, we observe $p_1, \ldots, p_t$, and want to predict p_{t+1} on day $t+1$, so that we use this prediction to trade the stock.

The input X_{t+1} is a d-dimensional real valued vector in R^d that represents the observed historical information of the stock on day t. The output $Y_{t+1} = \ln(p_{t+1}/p_t)$ will be observed on day $t+1$.

We consider linear model with $f(w, x) = w^\top x$, with $\Omega = \mathbb{R}^d$. The quality is measured by the least squares error

$$L(f(w, X_{t+1}), Y_{t+1}) = (f(w, X_{t+1}) - Y_{t+1})^2$$

Example (cont)

The learning algorithm can be empirical risk minimization, where

$$\hat{w}_t = \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{t} (w^\top X_i - Y_i)^2.$$

In regret analysis, we compare the prediction error

$$\sum_{t=0}^{T-1} (\hat{w}_t^\top X_{t+1} - Y_{t+1})^2$$

to the optimal prediction

$$\inf_{w \in \mathbb{R}^d} \sum_{t=0}^{T-1} (w^\top X_{t+1} - Y_{t+1})^2,$$

and want to show it is small.

Computation

In the ERM method, the model parameter $\hat{w}$ is the solution of an optimization problem.

- If the optimization problem is convex, then the solution can be efficiently computed.
- If the optimization problem is nonconvex, then its solution may not be obtained easily.

Theoretically, we study two different types of complexity:

- Statistical complexity
- Computational complexity

For statistical complexity, we may ignore the complexity of computation, and try to derive generalization bounds or regret bounds even though the computational complexity of the underlying learning algorithm (such as ERM) may be high.

Computational Complexity

We are interested in computationally efficient learning algorithms with good generalization performance or good regret bounds.

- For nonconvex models, computational complexity analysis can be rather complexity, with problem specific approaches.
- One general approach is to use convex approximation (also referred to convex relaxation) to solve the nonconvex problem approximately. The theoretical question is whether such approximation have good performance.

Convex Relaxation: Sparse Learning Example

Example 9

In sparse learning, we want to find linear models with sparse weights

$$Y_i = X_i^\top w_* + \text{noise}, \qquad \text{subject to } \|w_*\|_0 \leq s,$$

where $\lambda > 0$ is a tuning parameter, and $\|w_*\|_0$ is the number of nonzero elements of the true parameter w_*. This is a nonconvex optimization problem. In practice, we can use the convex formulation with L_1 regularization as a proxy to the nonconvex L_0 regularization:

$$\hat{w} = \arg \min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} (X_i^\top w - Y_i)^2 + \lambda \|w\|_1,$$

where $\|w\|_1 = \sum_j |w_j|$. A theoretical problem we are interested in is to establish conditions under which one can obtain $\hat{w}$ that is close to w_*.

Basics of Generalization Analysis

The goal of machine learning is to find a function $f(\hat{w}, x)$ that predicts well on unseen data (test data).

However, we only observe the prediction accuracy of $f(\hat{w}, x)$ on the training data. In order to achieve high prediction accuracy, we need to balance the following two aspects of learning:

- The prediction function should fit the training data well; that is, achieving small training error. This requires a more expressive model, with a larger parameter space Ω.
- Performance of the prediction function on the test data should match that on the training data. The difference is smaller for a less expressive model with a smaller parameter space Ω.

Training versus Test

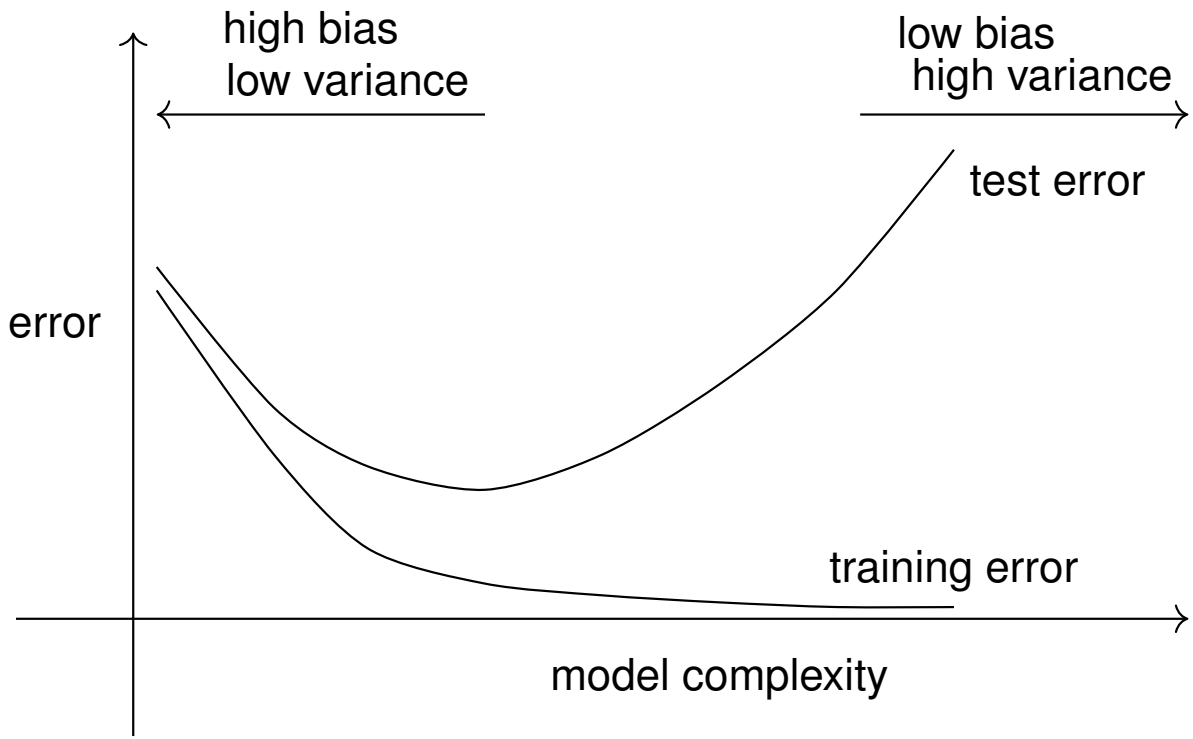

Figure: Training and test errors versus model complexity

We often have a tuning parameter in the learning algorithm that characterizes model complexity.

Example of Overfitting

Let X be a one-dimensional feature uniformly distributed in $[-1, 1]$, with true class label $Y = 1$ when $X \geq 0$ and $Y = -1$ when $X < 0$.

Given training data (X_i, Y_i) $(i = 1, \ldots, n)$, and assume X_i are all different. The following learning algorithm can fit data perfectly:

$$\hat{f}(X) = \begin{cases} Y_i & \text{if } X = X_i \text{ for some } i \\ 1 & \text{otherwise} \end{cases}$$

However, this prediction rule makes no meaningful prediction when X is not in the training data.

Example (cont)

Let $\mathbb{1}(x \in A)$ be the set indicator function that takes value 1 if $x \in A$, and 0 if $x \notin A$.
Assume that we pick the function class

$$\{f(w, x) : f(w, x) = 2\mathbb{1}(x \geq w) - 1\}$$

parametrized by a parameter $w \in R$.

Consider the ERM algorithm to find a classifier $f(\hat{w}, x)$ that minimizes the training error.

Using techniques of this course, it can be shown that both training and test error of this classifier converge to zero when $n \to \infty$. This model balances the training error and generalization performance.

Other Sequential Decision Problems

We consider the multi-armed bandit problem with K arms. The environment generates a sequence of reward vectors for time steps $t \geq 1$ as $r_t = [r_t(1), \ldots, r_t(K)]$. Each $r_t(a)$ is associated with an *arm* $a \in \{1, \ldots, K\}$. At each time step $t = 1, 2, \ldots, T$,

- The player pulls one of the arms $a_t \in \{1, \ldots, K\}$.
- The environment returns the reward $r_t(a_t)$, but does not reveal information on any other arm $a \neq a_t$.

The goal is to maximize the cumulative reward

$$\sum_{t=1}^{T} [r_t(a_t)].$$

In bandits, only the reward corresponding to the pulled arm is observed. This setting is referred to as *incomplete information*. In comparison, online learning assumes that rewards for all arms (independent of whether the arm is pulled) are observed.

Summary (Chapter 1)

- Supervised Learning
- Online Learning
- Training versus Test
- Generalization and Model Complexity
- Other Sequential decision problems: bandits (and more generally, reinforcement learning)