CS7140 Lecture Notes on Statistical/Theoretical Machine Learning and Its Applications: Spring 2026

Hongyang R. Zhang

January 8, 2026

Contents

1 Overview

Background. Machine learning has been increasingly used in technology platforms and products, affecting our daily lives. Machine learning involves a collection of models, algorithms, and engineering frameworks:

- Regression and classification: least squares estimation, logistic regression, ℓ_1/ℓ_2-regularization, bias-variance tradeoff, cross-validation.
- Neural networks and deep learning: convolutional neural networks, backpropagation, foundation models, language modeling.
- Unsupervised learning: dimension reduction such as principal component analysis, clustering, contrastive learning.
- Reinforcement learning and sequential decision-making: robotics, reinforcement learning from human feedback.
- Generative AI: large language models, diffusion models, multi-modal data.
- Causal machine learning: study the cause-and-effect to estimate the counterfactual.
- Machine learning libraries: numpy, sklearn, pytorch, tensorflow, huggingface...

1.1 What is this course about? (Lecture 1)

This course aims to uncover the common mathematical and statistical principles underlying the diverse array of machine learning models and algorithms. This class primarily focuses on the theoretical analysis of learning algorithms and models. Many of the techniques introduced in this course—which involve a beautiful blend of probability, linear algebra, and optimization—are separate fields in their respective discipline with independent interests outside of machine learning. For example, we will study the supremum of a complex random variable corresponding to the outcome of a learning algorithm applied to train a neural network model. We will show how to design estimation algorithms when working under distribution shifts between the training and test datasets.

From a practical point of view, studying the underlying working mechanisms of a learning algorithm can deepen our understanding of how machine learning models work. For example, suppose we want to design a neural network classifier to predict the sentiment of a document. We train a regression model using word frequencies as features and achieve 100% training accuracy on 1000 training documents and 85% test accuracy on 1000 test documents. How can we reduce the gap between training and test accuracy? Further, what happens if the word frequencies between the training corpus and the test corpus are different? It is possible to answer these questions from an engineering perspective; instead, this course will mostly focus on the mathematical analysis underlying these procedures, although we will provide computational examples from time to time to help you understand the theoretical concepts.

There is a clear gap between theoretical analysis and an algorithm's practical performance. For instance, theoretical analysis is usually conducted under standard technical assumptions that are often made to simplify the analysis. The goal, instead, is to *build a deeper understanding of the underlying key concepts through mathematical modeling and theoretical analysis*. We will see if we succeed in accomplishing this objective by the end of this semester.

The course materials are divided into three parts: *fundamental concepts of statistical learning theory* (January), *generalization and optimization of neural networks and deep learning* (February), and *statistical modeling of emerging learning paradigms* (March).[1]

1.2 Supervised prediction (Lecture 1)

Central questions: *Does minimizing training error lead to low test error? How does the generalization ability depend on the model architecture and the training algorithm?* It turns out that answering these questions is highly non-trivial as it also depends on the underlying data distribution.

To formally study these questions, let us first describe the mathematical setup:

- Let $\mathcal{X}$ denote the feature space. Let $\mathcal{Y}$ denote the space of all possible outcomes. Binary classification example: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{+1, -1\}$
- Consider the problem of predicting an output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$.
- Let $\mathcal{H}$ be a set of hypotheses. Linear model example:

$$\mathcal{H} = \Big\{ x \to \beta^\top x + \eta : \forall \beta \in \mathbb{R}^d, \eta \in \mathbb{R} \Big\}$$

- Let $\ell : (\mathcal{X}, \mathcal{Y}) \times \mathcal{H} \to \mathbb{R}$ be a loss function. For example, the mean squared error (MSE) applied to linear models is

$$\ell((x, y), \beta) = \Big(\beta^\top x + \eta - y \Big)^2, \forall \beta \in \mathbb{R}^d, \forall \eta \in \mathbb{R}$$

- Given n training data samples, denoted by $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the training loss (or empirical risk) of a hypothesis $h \in \mathcal{H}$ is defined as

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i), \forall h \in \mathcal{H} \tag{1}$$

We make a critical assumption about the data-generating process. We assume that every x_i, y_i pair is drawn independently and identically from an unknown distribution $\mathbb{P}^\star$, supported on $\mathcal{X} \times \mathcal{Y}$.

The test loss (or expected risk) of a hypothesis $h \in \mathcal{H}$ is then given by

$$L(h) = \mathop{\mathbb{E}}_{(x,y) \sim \mathbb{P}^\star} \left[\ell(h(x), y) \right]. \tag{2}$$

Example 1.1 (Linear regression)**.** *To make the above setup more concrete, perhaps the best example would be linear regression. There are many standard texts on this topic; see, e.g., Wainwright, 2019. In a standard parametric regression setup, we have n samples* $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$*, where every* $x_i \in \mathbb{R}^p$ *is a p-dimensional feature vector drawn from some unknown distribution* $\mathcal{D}$*, and* $y_i \in \mathbb{R}$*, for every* $i = 1, 2, \ldots, n$*. In addition, suppose that there exists an unknown* $\beta \in \mathbb{R}^p$ *such that*

$$y_i = x_i^\top \beta + \varepsilon_i, \text{ for every } i = 1, 2, \ldots, n, \tag{3}$$

[1] April will be dedicated to course project presentations.

where $x_i^\top \beta = \sum_{j=1}^p x_{i,j}\beta_j$, and $\varepsilon_i \sim \mathcal{N}(0,\sigma^2)$ is a noise random variable with mean zero and variance σ^2.

Given n samples, the goal of this problem is to learn a linear model parameterized by $\hat{\beta}$ that achieves the lowest mean-squared error (MSE) on an unseen sample.

Remark 1.2. *We have assumed the training and test distributions are the same. While this assumption does not hold exactly in practice, morally, the training and test distributions must be related.*

Formulating what it means to be related and not related, and addressing the discrepancy between training and test data, are studied in the area of domain adaptation *or* transfer learning.

The independence assumption, which also does not hold exactly in practice, ensures that more training data gives us more information.

1.2.1 Empirical risk minimization

Consider minimizing the training loss

$$\hat{h}_{\mathrm{ERM}} \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \hat{L}(h). \tag{4}$$

What can say that the relationship between $\hat{L}(\hat{h}_{\mathrm{ERM}})$ and $L(\hat{h}_{\mathrm{ERM}})$? A key challenge is that the randomness of $\hat{h}_{\mathrm{ERM}}$ now depends on $\hat{L}$. Thus, $\hat{L}(\hat{h}_{\mathrm{ERM}})$ involves a correlation between the training data samples and the minimizing hypothesis. A central aspect we will tackle in the first part of the course is developing the machinery to address this challenge.

Example 1.3 (Pretraining and fine-tuning). *An emerging learning paradigm that has emerged over the past few years follows a two-stage procedure involving pretraining on a large amount of unlabeled data, followed by fine-tuning on a small amount of labeled data.*

The pretraining stage usually follows some masked prediction procedure on unlabeled data. The supervised fine-tuning (SFT) procedure can be formulated with the above ERM setup.

- *Suppose we have some model like a neural network, f_{W_0}, parameterized by some initialization W_0.*
- *There is a small amount of training dataset, S, from which we compute the training loss $\hat{L}(f_{W_0})$.*
- *SFT corresponds to minimizing $\hat{L}(f_{W_0})$, usually via a stochastic gradient optimization algorithm.*
- *An important consideration in SFT is overfitting, since the model is pretrained on a large amount of unlabeled data. The size of the model is usually much larger than the size of the training dataset S.*

1.2.2 Uniform convergence and generalization

In the first part of this course, we will show various statements of the following flavor:

$$\boxed{\text{With probability at least } 1-\delta \text{, the gap between test loss and training loss of any hypothesis is upper bounded by some small } \epsilon \text{, that is, } L(h)-\hat{L}(h)\le\epsilon \text{, where the } \epsilon \text{ is generally a function that depends on } \delta \text{ and other aspects of the learning algorithm/model}}$$

More rigorously, we would like to show statements of the following:

$$\Pr\left[\underbrace{L(h)-\hat{L}(h)}_{\text{Generalization gap}} > \epsilon\right] \le 1-\delta, \tag{5}$$

where the randomness is on the training data samples drawn from $\mathbb{P}^\star$. This statement essentially quantifies the generalization gap between the training and test losses of the machine learning model.

Equipped with such a statement, we will then apply the statement to the empirical risk minimizer $\hat{h}_{\text{ERM}}$, since the result essentially holds for any $h \in \mathcal{H}$, which also subsumes $\hat{h}_{\text{ERM}}$ as a special case.

1.3 Multi-layer neural networks and generative models (Lecture 1)

Consider the case of a basic one-layer network:

$$x \to \sum_{i=1}^{m} a_i \sigma(w_i^\top x + b_i), \text{ where } . \tag{6}$$

- σ is the nonlinear activation function. Typical choices of σ: ReLU $x \to \max(0, x)$, sigmoid $x \to \frac{1}{1+\exp(-x)}$. Key property: Lipschitz-continuity: A function $f : \mathbb{R} \to \mathbb{R}$ is said to be C-Lipschitz-continuous if the following is true:

$$|f(x) - f(y)| \le C \cdot |x - y|.$$

- $Z = \{\alpha = (a_i, w_i, b_i)\}_{i=1}^m$ are trainable parameters of the network. By varying them, we could define the function class $\mathcal{H}$ as

$$\mathcal{H} = \{f_\alpha : \forall \alpha \in Z\}.$$

- $\mathcal{H}$ essentially represents a set of one-hidden-layer neural networks with m neurons.
- We can define the weight matrix $W = [w_1, w_2, \ldots, w_m]$, and the bias vector $b = [b_1, b_2, \ldots, b_m]$. Thus, we may write map (6) as $x \to a^\top \sigma(Wx + b)$.

By extending the above setup, we may write a deep network as

$$f_\alpha(x) = \sigma_L\left(W_L \sigma_{L-1}\left(\cdots \sigma_2\left(W_2 \sigma_1\left(W_1 x + b_1\right) + b_2\right) \cdots\right)\right) \tag{7}$$

The depth of the network is given by L. The width is given by $\max(m_1, m_2, \ldots, m_l)$, i.e., the layer with the most neurons in the layer.

Motivating questions: *How could we analyze the training and test losses of a deep network? How well does a deep network generalize, and how does it depend on its depth and width? How does this ability to learn and to generalize rely on the data distributions, and what is the role of optimization algorithms used to train the network?*

Language models: A language model specifies a conditional probability distribution $\Pr_\theta(\cdot \mid P)$, given a prompt sequence P, produces the next-token according to underlying probability masses.

Example 1.4 (In-context learning)**.** *To illustrate the concept of a language model, let us consider a few-shot meta-learning problem (Garg et al., 2022). In this problem, each prompt* P_{θ_i} *involves a sequence of examples or demonstrations* $(x_1, y_1, x_2, y_2, \ldots, x_{t-1}, y_{t-1}, x_t)$*, ended with a query example* x_t*. The goal is to predict the correct output* y_t *corresponding to* x_t*.*

To make this more concrete, suppose that $y_j = \theta_i^\top x_j$*, for every* $j = 1, 2, \ldots, t-1$*. The desired output* $y_t = \theta_i^\top x_j$*.*

- *At training time, the model sees a sequence of prompt-answer pairs* (P_{θ_i}, y_t)*.*
- *At test time, the model sees a new prompt* P_θ *parameterized by some unknown* θ*. The model is asked to first "solve" the linear regression from the in-context examples given in* P_θ*, and then use the "learned" regression model to output the correct answer corresponding to the query.*

Next lecture: In the next lecture, we will wrap up this overview by providing a setup for rigorously modeling several emerging learning paradigms. We will then dive deeper into the uniform convergence framework.

References

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). "What can transformers learn in-context? a case study of simple function classes". In: *Advances in neural information processing systems* 35, pp. 30583–30598 (page 6).

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press (page 3).