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

Hongyang R. Zhang

January 12, 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:

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

- 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

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

- 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 gap

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

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

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

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

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 2)

Consider the case of a basic one-layer network:

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

- 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:

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

- $Z = \{\alpha = (a_i, w_i, b_i)\}_{i=1}^m$ 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.

- Let $W = [w_1, w_2, \ldots, w_m]$, and $b = [b_1, b_2, \ldots, b_m]$. We may write $f_{a,W,b}$ equation (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?*

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*

$$P_{\theta_i} := (x_1, y_1, x_2, y_2, \ldots, x_{t-1}, y_{t-1}, x_t),$$

ended with a query example x_t. The goal is to predict the correct output y_t corresponding to the query 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.*

1.4 Transfer learning and minimax estimation (Lecture 2)

An important learning paradigm that has emerged in the past few years is transfer learning—transferring the knowledge from one task to help solve another task. How could we develop a more rigorous statistical modeling of transfer learning? A better understanding of this question has applications in language modeling, computer vision, robotics, to name a few.

1.4.1 Transfer learning setup

Perhaps the simplest modeling framework is to examine transfer learning in linear regression tasks. For example, we may consider the case of two linear regression tasks, one called the source task and the other called the target task.

Suppose we have n_1 samples from the source task. We have n_2 samples from the target task. How could we use the samples from the source task to help estimate the target task? Concretely, let the samples of the source task be denoted by $(x_1^{(1)}, y_1^{(1)}), (x_2^{(1)}, y_2^{(1)}), \ldots, (x_{n_1}^{(1)}, y_{n_1}^{(1)})$, where every $x_i^{(1)}$ is a p-dimensional vector and $y_i^{(1)}$ is a real-valued outcome. Similarly, we denote the samples of the target task as $(x_1^{(2)}, y_1^{(2)}), (x_2^{(2)}, y_2^{(2)}), \ldots, (x_{n_2}^{(2)}, y_{n_2}^{(2)})$.

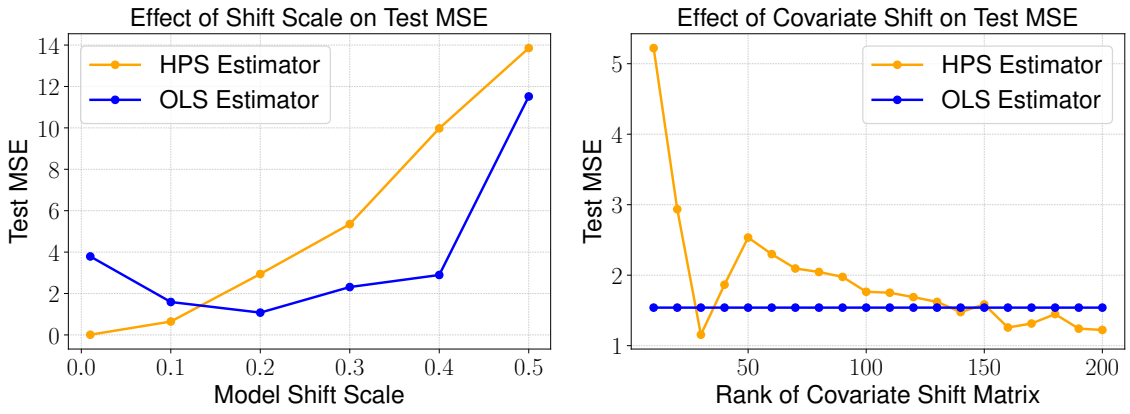
Now we can ask a few more concrete questions:

- How does the difference between $\beta^{(1)}$ and $\beta^{(2)}$ affect transfer learning performance?
- How does the difference between the feature vectors of source task and target task affect transfer learning?

More generally, we may say that the source task and the target task involve a distribution shift between their them. In the area of domain adaptation (Kouw and Loog, 2018):

- Covariate shift refers to scenarios where both tasks follow the same model conditioned on the features (i.e., $\beta^{(1)} = \beta^{(2)}$), but they have different feature distributions.
- Model shift refers to scenarios where the two tasks follow different models conditioned on the same features, that is $\beta^{(1)} \neq \beta^{(2)}$.

We may now ask, how does covariate shift and model shift affect transfer learning performance?



(a) Model shift

(b) Covariate shift

Figure 1: Illustrating the effects of model shift and covariate shift in transfer learning linear regression.

1.4.2 Transfer learning estimators

Typically, there are two strategies for transfer learning, one called hard transfer, where we hard-code the shared component across tasks, the other called soft transfer, where we use separate components for task, and encourage the separate components to be close to each other (Ruder, 2017; Dhifallah and Lu, 2021).

In the context of linear regression, we can define an hard parameter sharing estimator as follows:

$$\hat{L}^{\mathrm{HPS}}(\beta) = \frac{1}{n_1+n_2}\left(\sum_{i=1}^{n_1}\left({x_i^{(1)}}^\top \beta - y_i^{(1)}\right)^2 + \sum_{j=1}^{n_2}\left({x_j^{(2)}}^\top \beta - y_j^{(2)}\right)^2\right) \tag{8}$$

We may also elect to use a soft parameter sharing estimator instead:

$$\hat{L}^{\mathrm{SPS}}(\beta, z) = \frac{1}{n_1+n_2}\left(\sum_{i=1}^{n_1}\left({x_i^{(1)}}^\top (\beta+z) - y_i^{(1)}\right)^2 + \sum_{j=1}^{n_2}\left({x_j^{(2)}}^\top \beta - y_j^{(2)}\right)^2\right) + \lambda \|z\|^2 \tag{9}$$

Essentially, by adjusting λ, we can adjust the magnitude of z, which then determines how far (and how close) the source and target task models are.

A natural baseline is when we do not use the source task data at all. That is, perform least squares regression using target task data alone.

Example 1.5 (Illustration of model and covariate shifts in linear regression)**.** *We shall assume that the source task follows a linear relation specified by an unknown parameter* $\beta^{(1)} \in \mathbb{R}^p$*:*

$$y_i^{(1)} = {x_i^{(1)}}^\top \beta^{(1)} + \epsilon_i^{(1)}, \text{ for all } i = 1, 2, \ldots, n_1, \tag{10}$$

where $\epsilon_i^{(1)}$ *is a white noise with mean zero and variance* σ_1^2*.*

We further assume that the target task follow another linear relation specified by an unknown parameter $\beta^{(2)} \in \mathbb{R}^p$*:*

$$y_i^{(2)} = {x_i^{(2)}}^\top \beta^{(2)} + \epsilon_i^{(2)}, \text{ for all } i = 1, 2, \ldots, n_2, \tag{11}$$

where $\epsilon_i^{(2)}$ *is a white noise with mean zero and variance* σ_2^2.

In Figure 1, we illustrate the effects incurred from model shifts and covariate shifts in transfer learning linear regression, comparing between HPS and OLS. In particular, we capture model shift as the distance error between $\beta^{(1)}$ *and* $\beta^{(2)}$*, and we capture covariate shift as the difference in the population covariance matrix between task one and task two. To generate the condition matrix, we use a rank-r matrix whose trace is equal to p, and set its nonzero eigenvalues as* p/r.

1.4.3 Optimality of the estimator

The above estimation algorithms are based on the best practices of practitioner (see the surveys above). Suppose we analyze their performances. However, how can we know that there are no better estimators out there? How could we understand the fundamental limits of estimation and optimization procedures? These are often called *minimax lower bounds* on the performance of estimators, and it usually falls into the area of information theory (Duchi, 2019). In particular, we will touch on the framework of minimax lower bounds for transfer learning (though the scope of this is much broader than we'll cover in our lectures).

2 Uniform convergence and generalization

Recall that we have introduced the empirical risk and the expected risk of a hypothesis (denoted by $L(h)$ and $\hat{L}(h)$) for some h in a hypothesis class $\mathcal{H}$. Suppose we minimize the empirical risk to get $\hat{h}_{\text{ERM}}$. Two questions:

- Generalization gap: how does the expected and empirical risks compare for ERM, i.e., $L(\hat{h}_{\text{ERM}}) - \hat{L}(\hat{h}_{\text{ERM}})$? This is called the generalization gap.
- Excess risk: how well does ERM do with respect to the best possible hypothesis in the hypothesis class, i.e., $L(\hat{h}_{\text{ERM}}) - \min_{h \in \mathcal{H}} L(h)$? This is also called the excess risk.

A particularly fruitful framework for analyzing learning algorithms is the probably approximately correct (PAC) framework (Valiant, 1984):

> A learning algorithm A PAC learns a hypothesis class $\mathcal{H}$ if
>
> a) For any distribution $\mathbb{P}^\star$ supported over $\mathcal{X} \times \mathcal{Y}$, and any $\epsilon > 0$, $\delta > 0$
>
> b) Upon taking n I.I.D. samples from $\mathbb{P}^\star$, A produces an output $\hat{h} \in \mathcal{H}$ such that with probability at least $1 - \delta$ (over the randomness of the samples)
>
> $$L(\hat{h}) - \hat{L}(\hat{h}) \leq \epsilon$$
>
> c) Further, A runs in time polynomial in $n, d, \epsilon^{-1}, \delta^{-1}$ (where d is the dimension of the input)

Remark: Notice that the running time complexity places a bound on the sample complexity as well. We will assume that the empirical risk minimizer can be computed efficiently. For instance, think of a large neural network whose training loss can be efficiently reduced to reach zero using stochastic gradient descent.

2.1 Learning a realizable, finite hypothesis class (Lecture 2)

The ERM framework is very general – we now give a concrete example to illustrate some basic results.

Assumptions (realizable, finite hypothesis): i) The size of the hypothesis space, $\mathcal{H}$, is finite; ii) There exists a hypothesis $h^\star \in \mathcal{H}$ such that $h^\star$ achieves perfect performance, i.e.,

$$L(h^\star) = \mathop{\mathbb{E}}_{(x,y) \in \mathbb{P}^\star} [\ell(h^\star(x), y)] = 0.$$

Under these assumptions, we shall prove the following property of ERM:

Under the above assumptions, with probability $1 - \delta$,

$$L(\hat{h}_{\text{ERM}}) \leq \frac{\log(|\mathcal{H}|) + \log(\delta^{-1})}{n} \tag{12}$$

In particular, to reduce the expected risk below ϵ, we want $n \geq \frac{\log(|\mathcal{H}|)+\log(\delta^{-1})}{\epsilon}$.
Remarks:

- The excess risk only grows logarithmically with the size of the hypothesis class, so we can afford to use an exponentially large hypothesis space.
- The result is independent of $\mathbb{P}^\star$. This is nice because typically we don't know the true distribution.

Proof. We'd like to upper bound the probability of the bad event that $L(\hat{h}_{\text{ERM}}) > \epsilon$: Let $B \subseteq \mathcal{H}$ be the set of bad hypotheses: $\{B \in \mathcal{H} : L(h) > \epsilon\}$

We can rewrite our goal as upper bounding the probability of selecting a bad hypothesis $\Pr[L(\hat{h}_{\text{ERM}}) > \epsilon] = \Pr[\hat{h}_{\text{ERM}} \in B]$. Recall the empirical risk of ERM is always zero because at least $\hat{L}(h^\star) = 0$. Hence for any "bad" hypothesis in B, they must have zero empirical risk

$$\Pr[\hat{h}_{\text{ERM}} \in B] \leq \Pr[\exists h \in B : \hat{L}(h) = 0]$$

Now we shall deal with the above in two steps. First, bound $\Pr[\hat{L}(h) = 0]$ for a fixed $h \in B$. Notice that on a random example from $\mathcal{P}^\star$, the accuracy of h should be $1 - L(h)$. Since the training data is drawn IID from $\mathcal{P}^\star$, and $L(h) \geq \epsilon$ for any $h \in B$, we get that

$$\Pr[\hat{L}(h) = 0] \leq (1 - L(h))^n \leq (1 - \epsilon)^n \leq \exp^{-\epsilon n},$$

where we use the fact that $1 - x \leq \exp(-x)$.

Second, we want the above to hold simultaneously for all $h \in B$. We can apply the union bound to bound the probability of all bad events:

$$\begin{aligned}
\Pr[\exists h \in B : \hat{L}(h) = 0] &\leq \sum_{h \in B} \Pr[\hat{L}(h) = 0] \\
&\leq |B| \exp(-\epsilon n) \\
&\leq |\mathcal{H}| \exp(-\epsilon n)
\end{aligned}$$

By setting the above at most δ, we conclude that ϵ must be at least $\frac{\log(|\mathcal{H}|)+\log(\delta^{-1})}{n}$. This concludes the proof for learning finite, realizable hypothesis spaces.

Takeaway: The proof of this result is elementary but illustrates an important pattern that will recur in more complex scenarios. We are interested in the expected risk, but only have access to empirical risk to choose the ERM:

- Step 1 (convergence): for a fixed h, show that $\hat{L}(h)$ is close to $L(h)$ with high probability.
- Step 2 (uniform convergence): show that the above holds simultaneously for all hypotheses $h \in \mathcal{H}$.

However, the assumptions are restrictive. There exists a perfect hypothesis (realizability). What happens when the problem is not realizable? To answer this, we introduce the tools of concentration estimates.

Second, the hypothesis class is finite. What happens when the number of hypotheses is infinite? To answer this, we need better ways of measuring the "size" of a set – leading to Rademacher complexity, VC dimension, and PAC-Bayes bounds (to name a few).

Next lecture: In the next lecture, we will look further into uniform convergence bounds, ultimately leading to the *Rademacher complexity* framework.

P.S.

- Have feedback or want to ask a question for the instructor? Leave a note here: https://forms.gle/SCjXUW6dkM8cDi4o9.
- Want to see the latex source code? You can find the tex files here: https://github.com/hongyangsg/cs7140-advanced-ml.

References

Dhifallah, O. and Lu, Y. M. (2021). "Phase transitions in transfer learning for high-dimensional perceptrons". In: *Entropy* 23.4, p. 400 (page 7).

Duchi, J. (2019). "Lecture notes for statistics 311/electrical engineering 377". In: *URL: http://web.stanford.edu/class/stats311/lecture-notes.pdf* (page 8).

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).

Kouw, W. M. and Loog, M. (2018). "An introduction to domain adaptation and transfer learning". In: *arXiv preprint arXiv:1812.11806* (page 6).

Ruder, S. (2017). "An Overview of Multi-Task Learning in Deep Neural Networks". In: *arXiv preprint arXiv:1706.05098* (page 7).

Valiant, L. G. (1984). "A theory of the learnable". In: *Communications of the ACM* 27.11, pp. 1134–1142 (page 9).

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