Five Lectures on the Computational Complexity of Deep Learning [DRAFT]

Theodor Misiakiewicz[1]

August 2, 2025

[1]Department of Statistics and Data Science, Yale University

Abstract

These lecture notes are based on lectures given by T. M. at the Princeton Machine Learning Theory Summer School (2025). We thank Boris Hanin for organizing this wonderful summer school.

These notes are currently at the stage of a draft. Please do not distribute.

Contents

Chapter 1

Introduction

In these lectures, we explore foundational concepts and recent developments on the computational complexity of Deep learning—and more broadly, machine learning. My primary goal is to introduce systematic approaches for understanding the computational complexity in statistical learning tasks, with a particular emphasis on techniques for establishing computational lower bounds. Rather than a comprehensive treatment, these lectures are structured as a series of vignettes, highlighting key intuitions, central ideas, and formal frameworks essential for deriving these lower bounds. Throughout, I will provide references for students interested in delving deeper into specific topics.

One might ask: Why emphasize lower bounds? Isn't that too gloomy an outlook on life? Perhaps. However, lower bounds serve a crucial role in theoretical machine learning. They clarify inherent difficulties in learning tasks—irrespective of the algorithm—, provide 'yardsticks' against which we compare our learning algorithms, identify fundamental tradeoffs between resources (e.g., sample vs. computation), and formalize "no free lunch" by quantifying what assumptions are necessary for learning. Given the intricacy of deep learning methods, focusing on lower bounds enables us to abstract intricate implementation details and concentrate on some essential aspects of learning problems. This, in turn, provides crucial insights into the underlying principles that determine effective learning and algorithm design.

By pursuing this perspective, we will touch upon several foundational and contemporary topics in machine learning theory, including:

- Statistical-to-computational gaps and the inherent trade-offs between runtime efficiency and sample complexity.

- Proper versus improper learning, generalization-computation trade-offs, and how overparametrization can affect tractability.
- Adaptive ('feature learning') versus non-adaptive ('fixed feature' or kernel) methods.
- Key frameworks for capturing learning hardness: reductions, junk flow, Statistical Query algorithms, and Low-degree polynomials.

Outline. The classical approach to computational hardness relies on reductions from problems presumed hard under standard complexity assumptions (such as $\mathtt{P} \neq \mathtt{NP}$ or cryptographic hardness). This will be the subject of **Chapter 1**. However, such reductions are often difficult—or even impossible—to apply in statistical settings that emphasize average-case rather than worst-case hardness—that is, hardness with high probability over random problem instances. Over the past few decades, alternative frameworks have been developed to prove lower bounds for restricted classes of algorithms. These approaches are typically easier to implement, yield sharper predictions, and apply more broadly across many problems. In the rest of the lectures, we will present lower bounds on kernel methods (**Chapter 2**), noisy gradient descent (**Chapter 3**), Statistical Query (SQ) algorithms (**Chapter 4**), and low-degree polynomial algorithms (**Chapter 5**).

1.1 Supervised learning

We focus on the standard supervised learning problem where we are given a collection of n data points $\{(y_i, \boldsymbol{x}_i)\}_{i \le n} \sim_{iid} \mathcal{D}$ with $\boldsymbol{x}_i \in \mathcal{X}$ the vector of *covariates*, $y_i \in \mathcal{Y}$ the *response* or *label*, and $\mathcal{D}$ an unknown probability distribution on $\mathcal{Y} \times \mathcal{X}$.

Throughout, we will denote $\mathbb{P}_{\mathcal{X}}$ the marginal distribution of the covariate vector $\boldsymbol{x}$. We will further denote $\mathcal{P}(\mathcal{Z})$ the space of probability distributions over the (measurable) space $\mathcal{Z}$, and $L^2(\mathcal{D})$ the space of squared-integrable functions with respect to $\mathcal{D} \in \mathcal{P}(\mathcal{Z})$, with inner-product

$$\langle f, g \rangle_{L^2(\mathcal{D})} = \mathbb{E}_{\boldsymbol{z} \sim \mathcal{D}}[f(\boldsymbol{z}) g(\boldsymbol{z})], \qquad \text{for all } f, g \in L^2(\mathcal{D}), \tag{1.1.1}$$

and norm $\|f\|_{L^2(\mathcal{D})} = \langle f, f \rangle_{L^2(\mathcal{D})}^{1/2}$. For notational simplicity, we will write $\langle \cdot, \cdot \rangle_{\mathcal{D}}$—or simply $\langle \cdot, \cdot \rangle_{L^2}$ when clear from context—instead of $\langle \cdot, \cdot \rangle_{L^2(\mathcal{D})}$.

Given these n training data points, the goal is to learn a *model* $\hat{f} : \mathcal{X} \to \mathcal{Y}$ (also known as a *predictor*) such that, given a new covariate vector $\boldsymbol{x}_{\text{new}}$, it predicts the response y_{new} via $\hat{f}(\boldsymbol{x}_{\text{new}})$. The performance of the model is measured by a *test error* (also known as *population risk*)

$$\mathcal{R}(\hat{f}; \mathcal{D}) = \mathbb{E}_{(y_{\text{new}}, \boldsymbol{x}_{\text{new}}) \sim \mathcal{D}} \left[\ell\big(y_{\text{new}}, \hat{f}(\boldsymbol{x}_{\text{new}})\big) \right], \tag{1.1.2}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a loss function.

Example 1.1.1. *For concreteness, we will often assume that* $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$, *and that the data* $(y, \boldsymbol{x}) \sim \mathcal{D}$ *is generated via*

$$y = f_*(\boldsymbol{x}) + \varepsilon, \tag{1.1.3}$$

where $f_* \in L^2(\mathbb{P}_{\mathcal{X}})$ *is the* target *or* regression function *and* ε *is an independent label noise* $\mathbb{E}[\varepsilon] = 0$, $\mathbb{E}[\varepsilon^2] = \sigma_\varepsilon^2$, *so that* $f_*(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}[y|\boldsymbol{x}]$. *In this case, we will focus on the squared loss* $\ell(y, \hat{y}) = (y - \hat{y})^2$, *and refer to this setting as the* regression setting.

In the classification setting, *the label* y *takes discrete values. Typically, we will take* $\mathcal{Y} = \{-1, +1\}$ *and focus on the* 0-1 *loss* $\ell(y, \hat{y}) = \mathbb{1}[y \neq \hat{y}]$, *so that* $\mathcal{R}(\hat{f}; \mathcal{D}) = \mathbb{P}_{(y,\boldsymbol{x}) \sim \mathcal{D}}[y \neq \hat{f}(\boldsymbol{x})]$—*the probability of correctly predicting the label. When* $\hat{f}$ *is real-valued, we will simply identify* $\hat{y} = \operatorname{sign}(\hat{f}(\boldsymbol{x}))$.

Sometimes, it will be convenient to substract from the test error the *Bayes risk*—that is, the minimum error achieved by any predictor f. This is known as the *excess test error* or *excess risk*:

$$\mathcal{R}_{\text{exc}}(\hat{f}; \mathcal{D}) = \mathcal{R}(\hat{f}; \mathcal{D}) - \inf_{f : \mathcal{X} \to \mathcal{Y}} \mathcal{R}(f; \mathcal{D}). \tag{1.1.4}$$

In the case of squared loss, this is simply given by

$$\mathcal{R}_{\text{exc}}(\hat{f}; \mathcal{D}) = \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathcal{X}}} \left[\big(f_*(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\big)^2 \right], \tag{1.1.5}$$

where $f_*(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}]$ is the regression function.

Typically, the model $\hat{f}$ is chosen restricted to a parametric model class

$$\hat{f} \in \mathcal{F} = \{\boldsymbol{x} \mapsto f(\boldsymbol{x}; \boldsymbol{\theta}) : \boldsymbol{\theta} \in \mathbb{R}^p\},$$

where $\boldsymbol{\theta} \in \mathbb{R}^p$ is a vector of parameters. For example,

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i=1}^{M} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle + b_i), \tag{1.1.6}$$

with $\boldsymbol{\theta} = (a_1, \dots, a_M, b_1, \dots, b_M, \boldsymbol{w}_1, \dots, \boldsymbol{w}_M) \in \mathbb{R}^{M(d+2)}$. Then, $\mathcal{F}$ is the class of two-layer neural networks with M neurons and activation $\sigma : \mathbb{R} \to \mathbb{R}$. With a slight abuse of notation, we will write $\mathcal{R}(\boldsymbol{\theta}; \mathcal{D}) := \mathcal{R}(f(\cdot; \boldsymbol{\theta}), \mathcal{D})$.

Empirical Risk Minimization (ERM). Our goal is to learn a model $f(\cdot, \boldsymbol{\theta})$ that minimizes the population risk (1.1.2). The main approach is to fit $\hat{\boldsymbol{\theta}} \in \mathbb{R}^p$ by minimizing the empirical risk over the n training data points:

$$\hat{\boldsymbol{\theta}} := \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \hat{\mathcal{R}}_n(\boldsymbol{\theta}) + r(\boldsymbol{\theta}), \qquad \hat{\mathcal{R}}_n(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})), \tag{1.1.7}$$

where we included a possible regularizer $r : \mathbb{R}^p \to \mathbb{R}_{\geq 0}$—e.g., $r(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|_2^2$.

Modern machine learning attempt to construct approximate minimizers of this empirical risk via gradient-type algorithms (first order methods). The simplest is gradient descent (GD), which, starting from a (random) initialization $\boldsymbol{\theta}_0 \in \mathbb{R}^p$, iteratively update the parameters as

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta_k \nabla_{\boldsymbol{\theta}} (\hat{\mathcal{R}}_n(\boldsymbol{\theta}^k) + r(\boldsymbol{\theta}^k)). \tag{1.1.8}$$

In practice, stochastic gradient descent (SGD) with small batch is preferred: at each step, b data points $\{(y_i, \boldsymbol{x}_i)\}_{i \in \mathcal{I}_k}$ with $\mathcal{I}_k \subset [n], |\mathcal{I}_k| = b$, are chosen uniformly at random from the training data set and

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta_k \frac{1}{b} \sum_{i \in \mathcal{I}_k} \nabla_{\boldsymbol{\theta}} \ell(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}^k)) - \nabla_{\boldsymbol{\theta}} r(\boldsymbol{\theta}). \tag{1.1.9}$$

Performance of predictors. There are three factors that can limit the performance (that is, the test error) of a trained model $f(\cdot, \hat{\boldsymbol{\theta}})$:

- **Expressivity:** The function class $f(\cdot, \boldsymbol{\theta})$ might not be expressive enough to model the true input-output relationship $(y, \boldsymbol{x}) \sim \mathcal{D}$. However, in modern practice, we typically use highly overparameterized neural networks that are extremely expressive. As a result, we will largely set aside this issue (see Section 1.3 for a discussion on the role of overparameterization).

- **Sample size:** The training data may not contain enough information to accurately learn the target function $f_*(\boldsymbol{x})$ and generalize beyond the training set to new unseen examples.

- **Runtime:** Even when sufficient data is available, learning a good model may be computationally infeasible. For instance, the empirical risk minimization problem (1.1.7) is generally highly non-convex, and there is no guarantee that gradient descent (1.1.8) will converge to a global minimizer—or to a sufficiently good solution.

These lectures focus on the third factor: the computational complexity of supervised learning. There are several reasons for emphasizing this limitation, and we list three below:

1. The statistical aspects of learning are comparatively better understood, with mature theories such as uniform convergence (e.g., [SSBD14, BFT17, SH20]) and minimax theory (e.g., [Wai19, Chapter 15]). See [BLLT20, BMR21] on recent work on the phenomenon of *benign overfitting*.

2. Being computationally efficient is a more stringent requirement than being statistically efficient, simply because runtime grows at least linearly in the number of samples. In fact, many problems display so-called *Statistical-Computational gaps*, that is, they exhibit regimes where learning is information-theoretically possible, yet no known polynomial-time algorithm succeeds. See [Wei25] for a nice overview of this phenomenon through the lens of the low-degree polynomial framework.

3. In many respects, the success of deep learning presents a *computational mystery* rather than a *statistical mystery*. Furthermore, classical statistical learning theory—based on i.i.d. samples and function approximation—fails to capture many aspects of modern practice, where data is often curated, mixed with synthetic data, or presented in a curriculum-like sequence.

In the remainder of these lectures, our focus will be on developing techniques to establish lower bounds on the computational complexity of the supervised learning problem. At this stage, we have not precisely defined what we mean by computational complexity, and in fact, the notion will vary depending on the context. Broadly speaking, you can think of it as the runtime required for an algorithm to succeed—measured in terms of the number of elementary operations it performs—and how this runtime scales with various problem parameters (e.g., input dimension, accuracy, etc.). Each lower bound framework we study will rely on a different proxy for computational

complexity (sample size, number of gradient steps, number of queries, or degree of the polynomial). In each case, we will clearly state what notion of complexity is being bounded and under what assumptions.

1.1.1 Hypothesis classes and running examples

Before proceeding, let us make one final—but essential—remark. To meaningfully define a learning lower bound against broad classes of algorithms, we must consider a family of learning problems. That is, we consider a hypothesis class $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Y} \times \mathcal{X})$, where each $\mathcal{D} \in \mathcal{H}$ is a possible data-generating distribution, and require that our algorithms succeed on all—or most of—these distributions. Otherwise, the learning problem is not well defined: if $\mathcal{H} = \{\mathcal{D}_0\}$ is a singleton, then a trivial algorithm that simply outputs $\mathcal{D}_0$—without observing any data—solves the problem in zero samples and constant time. Hence, nontrivial lower bounds require some form of uncertainty about the underlying distribution to be learned.

In some cases, with input marginal $\mathbb{P}_{\mathcal{X}}$ fixed and shared by all $\mathcal{D} \in \mathcal{H}$, the algorithm will only access the data through the regression function $h = \mathbb{E}_{\mathcal{D}}[y|\boldsymbol{x}]$. In this case, we can identify $\mathcal{H}$ with its target functions, and with a slight abuse of notations, directly consider $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{Y}\}$ to be a set of functions.

Throughout these lectures, we will have two running examples of hypothesis classes to illustrate and compare each lower bounds. Namely, we consider learning *Gaussian single-index models* and *sparse functions on the hypercube*.

Learning Gaussian single-index models

Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$. Given a target function $f_* : \mathbb{R} \to \mathbb{R}$ such that $\mathbb{E}[f_*(G)^2] < \infty$, $G \sim \mathcal{N}(0,1)$, we consider the hypothesis class

$$\mathcal{H}^{(d)}_{f_*,\mathsf{SI}} := \{\mathcal{D}_{f_*,\boldsymbol{w}} : \boldsymbol{w} \in \mathbb{S}^{d-1}\}, \tag{1.1.10}$$

(here, $\mathbb{S}^{d-1} = \{\boldsymbol{w} \in \mathbb{R}^d : \|\boldsymbol{w}\|_2 = 1\}$ denotes the unit sphere in d dimensions) where $(y, \boldsymbol{x}) \sim \mathcal{D}_{f_*,\boldsymbol{w}}$ is generated as[1]

$$y = f_*(\langle \boldsymbol{w}_*, \boldsymbol{x}\rangle) + \varepsilon, \quad \text{with } \boldsymbol{x} \sim \mathcal{N}(0, \mathbf{I}_d), \text{ and } \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{1.1.11}$$

[1]There is no added difficulty when considering a general link function $y|\boldsymbol{x} \sim \rho(\cdot|\langle \boldsymbol{w}_*, \boldsymbol{x}\rangle)$ where $\rho \in \mathcal{P}(\mathbb{R} \times \mathbb{R})$ with $(Y, G) \sim \rho$ and $G \sim \mathcal{N}(0,1)$.

In other words, the response y only depends on $\boldsymbol{x}$ through its one-dimensional projection $\langle \boldsymbol{w}_*, \boldsymbol{x} \rangle$. For simplicity, we will keep the dependency on f_* and simply write $\mathcal{H}_{\mathsf{SI}}^{(d)}$ and $\mathcal{D}_{\boldsymbol{w}}$ when clear from context.

The distribution $\mathcal{D}_{\boldsymbol{w}}$ is a simple example of a *generalized linear model* [McC84, BKM+19]. It is also often called a *single-index model*—the label y only depends on one 'index' $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ of the input data. We will refer to $\mathcal{H}_{\mathsf{SI}}^{(d)}$ as an hypothesis class of *Gaussian single-index models*.

When fixing f_* and σ_ε, this naturally induces a sequence of hypothesis classes indexed by d. We will be interested in understanding the complexity of learning $\mathcal{H}_{\mathsf{SI}}^{(d)}$ as $d \to \infty$. Information-theoretically, $n = \Theta_d(d)$ is necessary and sufficient to learn this family. However, we will see that in order to have a polynomial-time algorithm (polynomial in d) that solves this problem, one needs $n = \Theta_d(d^{k_*/2})$ samples, where k_* is the *generative exponent* of f_* [DPVLB24]. Thus, these models display a Statistical-Computational gap where, if $d \lesssim n \ll d^{k_*/2}$, learning is information theoretically possible, but computationally hard.

For convenience, we will denote $\gamma_d = \mathcal{N}(0, \mathbf{I}_d)$.

Remark 1.1.2 (Unknown link function)**.** *Here, learning $\mathcal{H}_{\mathsf{SI}}^{(d)}$ amounts to recovering the vector $\boldsymbol{w}_* \in \mathbb{S}^{d-1}$. This corresponds to the Bayesian setting where the only unknown is the support direction. If instead f_* is only partially known, that is, we only know that $f_* \in \mathcal{L} \subseteq \{f : \mathbb{R} \to \mathbb{R}\}$, e.g., the class of 1-Lipschitz functions, then we could consider instead*

$$\mathcal{H} = \bigcup_{f_* \in \mathcal{L}} \mathcal{H}_{f_*,\mathsf{SI}}^{(d)} = \left\{ \mathcal{D}_{f_*,\boldsymbol{w}} \ : \ \boldsymbol{w} \in \mathbb{S}^{d-1}, f_* \in \mathcal{L} \right\}.$$

However, (1) lower bounds will be driven by the most difficult subset $\mathcal{H}_{f_*,\mathsf{SI}}^{(d)}$, and (2) in high-dimensions, the complexity of learning these distributions is dominated by recovering the support direction. Thus, we will not explore further this direction and only consider settings where f_* is fixed, known.

Learning sparse functions on the hypercube

Let $\mathcal{X} = \{-1, +1\}^d$ be the discrete hypercube in d dimensions, and $\mathcal{Y} = \mathbb{R}$. Fix an integer $P \in \mathbb{N}$. Given a target function $f_* : \mathbb{R}^P \to \mathbb{R}$, we consider the hypothesis class

$$\mathcal{H}_{\mathsf{Sp}}^{(d)} := \{\mathcal{D}_S : S \subseteq [d], |S| = P\}, \tag{1.1.12}$$

where S is a size-P ordered subset of $[d]$, and $(y, \boldsymbol{x}) \sim \mathcal{D}_{S_*}$ is generated as[2]

$$y = f_*(\boldsymbol{x}_{S_*}) + \varepsilon, \quad \text{with } \boldsymbol{x} \sim \mathrm{Unif}(\{\pm 1\}^d), \text{ and } \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{1.1.13}$$

where we denote $\boldsymbol{x}_{S_*} = (x_i)_{i \in S_*} \in \{\pm 1\}^P$. In other words, the response y only depends on P 'active' coordinates $x_i, i \in S_*$ and is independent of the rest of the input coordinates. We will refer to $\mathcal{H}_{\mathsf{Sp}}^{(d)}$ as a class of *sparse functions on the hypercube*. Note that learning $\mathcal{H}$ amounts to recovering the support $S_* \subseteq [d], |S_*| = P$ (see Remark 1.1.2).

Again, when fixing f_* and σ_ε, this induces a sequence of hypothesis classes indexed by d, and we will be interested in understanding the complexity of learning $\mathcal{H}_{\mathsf{SI}}^{(d)}$ as $d \to \infty$. In this case, $n = \Theta_d(\log(d))$ is necessary and sufficient to learn this family in polynomial time (there is no Statistical-Computational gap). Indeed consider the following simple algorithm:

1. For each ordered subset $S \subseteq [d], |S| = P$, compute $\hat{T}_S = \hat{\mathbb{E}}_n[y f_*(\boldsymbol{x}_S)]$, the empirical average over n samples.

2. Output $\hat{S} = \arg\max_S \hat{T}_S$.

By union bound, if $n = O(\log(d))$, then $\hat{S} = S_*$ with high probability[3], and the algorithm runs in time $O(nd^P)$.

Instead, we will see that no known algorithm exists that solves this problem with less than $\Theta_d(d^{k_*})$ runtime, where k_* is the *SQ-Leap* of f_* [JMS24]. For example, if $f_*(\boldsymbol{x}_{S_*}) = \prod_{i \in S_*} x_i$ is the parity function on S_*, then $k_* = P$ (this is the runtime of the above algorithm, which is optimal here).

If $P := P(d)$ is allowed to grow with d, then the brute force algorithm that enumerates all possible supports will have superpolynomial runtime. In that case, $n = \Theta_d(P \log(d))$ is information-theoretic optimal[4], but only (sequences of) functions that have SQ-Leap uniformly bounded by a constant will be learnable in polynomial time. We will discuss simple examples with $P := P(d)$, but otherwise, will keep P (and f_*) fixed.

For convenience, we will denote $\nu_d = \mathrm{Unif}(\{\pm 1\}^d)$.

[2]Again, there is no added difficulty when considering a general link function $y|\boldsymbol{x} \sim \rho(\cdot|\boldsymbol{x}_S)$ where $\rho \in \mathcal{P}(\mathbb{R} \times \{\pm 1\}^P)$ with $(Y, X) \sim \rho$ and $X \sim \mathrm{Unif}(\{\pm 1\}^P)$. Furthermore, the results I will present generalize straightforwardly to $\boldsymbol{x}$ drawn from a general product distribution on $\mathbb{R}^d$.

[3]Note that if f_* has internal symmetries, that is $f_*(\boldsymbol{x}_S) = f_*(\boldsymbol{x}_{S'})$ for $S \neq S'$, we can only recover S up to these symmetries.

[4]This requires additional assumptions as f_* now depends on d.

1.2 Hardness of training neural networks

We will start by presenting a landmark result by Blum and Rivest (1988) [BR88] which showed that training neural networks is NP-complete, that is, under the assumption that P $\neq$ NP:

> *No polynomial-time algorithm can (unconditionally) minimize the Empirical Risk* (1.1.7) *over neural networks.*

To understand the statement of this result, let's recall informally some basic definitions from computational complexity theory (see textbooks for formal definitions, e.g., [AB09]).

Definition 1.2.1 (Informal, complexity classes)**.**

- P: *Class of problems that can be solved in polynomial time.*
- NP: *Class of problem whose solution can be verified in polynomial time.*
- Polynomial-time reduction *We say that a problem* A *is as hard as a problem* B *if there exists a polynomial reduction (mapping) from instances of* B *to instances of* A*. Hence, if we can solve* A *in polynomial time, we can solve* B *in polynomial time by first mapping it to* A*. Conversely, if we expect* B *to be hard to solve, then we expect* A *to be hard too.*
- NP-complete: *A problem* A *in NP is NP-complete if any problem in NP is reducible to* A *in polynomial time. This means that* A *is as hard as any problem in NP.*
- P $\neq$ NP: *If P $\neq$ NP, then NP-complete problems cannot be solved in polynomial time.*

Blum and Rivest (1988) considered a 3-neural network $f(\cdot, \boldsymbol{\theta})$, as depicted in Figure 1.1, with two hidden neurons and an output neuron, and activation function $\sigma(x) = \text{sign}(x)$. Given n data points $\{(y_i, \boldsymbol{x}_i)\}_{i \leq n} \in \{\pm 1\} \times \{0, 1\}^d$, consider the empirical risk minimization problem

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2. \tag{1.2.1}$$

If one can solve this minimization problem (within $< 4/n$ of the true minimizer), one can answer the following question:

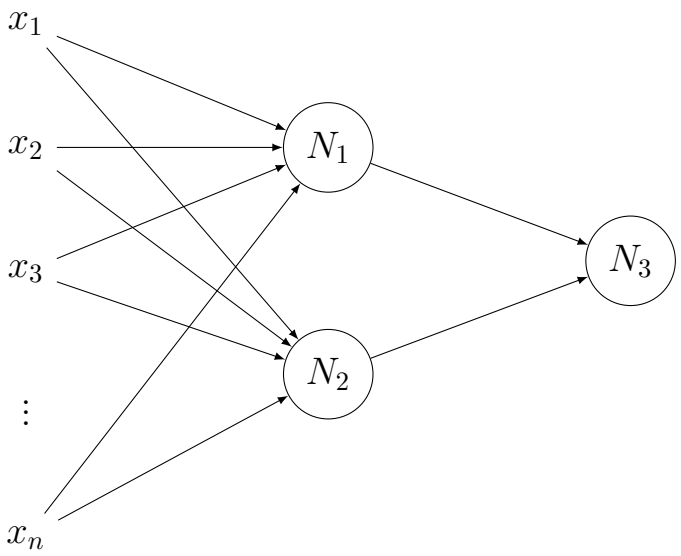

Figure 1.1: Three-nodes neural network with input $\boldsymbol{x} = (x_1, \ldots, x_n)$, two hidden nodes, and one output node, with the output of each nodes being $\varphi_i(\boldsymbol{z}) = \text{sign}(\langle \boldsymbol{w}_i, \boldsymbol{z} \rangle + b_i)$.

Q^*: *Given a set of $O(d)$ examples $(y_i, \boldsymbol{x}_i)$ with $\boldsymbol{x}_i \in \{0,1\}^d$ and $y_i \in \{\pm 1\}$, does there exist a 3-neuron network that interpolates this data (achieves 0 training error)?*

Theorem 1.2.2 ([BR88])**.** *Training 3-neuron networks is NP-complete.*

This result implies that if $\mathsf{P} \neq \mathsf{NP}$, no algorithm can minimize the empirical risk (and solve Q^*) in $\mathsf{poly}(d)$ runtime. The idea of the proof is that we can reduce a NP-complete problem to Q^*, thus making Q^* NP-complete. Specifically, Blum and Rivest considered the *Set-Splitting* problem.

Problem 1.2.3 (Set-Splitting)**.** *Given a finite set S and a collection of subsets $\{C_i | C_i \subset S\}$, does there exists subsets $S_1, S_2 \subset S$ with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$ such that $C_i \not\subset S_1$ and $C_i \not\subset S_2$ for all i?*

Theorem 1.2.4 (Lovasz [GJ02])**.** *Set-Splitting is NP-complete.*

The reduction from Set-Splitting to Q^* is quite short, albeit not very illuminating, and we will skip it. I encourage the interested student to check the original proof in [BR88].

Blum and Rivest's result was inspired by earlier work by Judd [Jud87] who showed that deciding whether a neural network can correctly predict

two-thirds of the training samples is NP-complete (thus, even approximate training is hard). However, Judd did not specify which networks are hard to learn and Blum and Rivest showed that even a simple 3-node network is already hard to train. There has been several follow-up works, including recent results that showed that hardness persists even in low dimensions ($d = 2$) or if the linear threshold activation is replaced by a ReLu [FH23].

1.3 Proper versus improper learning

Is Blum and Rivest's hardness result—that no polynomial-time algorithms can solve the Empirical Risk Minimization (ERM) problem—compelling? Indeed, their result holds under

- **Worst-case over training data:** The algorithm must solve ERM for *any* collection of training samples $\{(y_i, \boldsymbol{x}_i)\}_{i \le n}$ (at least, restricted to data that is binary valued $(y, \boldsymbol{x}) \in \{\pm 1\} \times \{0, 1\}^d$).
- **Fixed architecture:** It is NP-complete to decide whether there exists a three-node network that perfectly fits the data.

However, in machine learning, our goal differs in two crucial ways:

1. We are typically concerned with data drawn from an underlying distribution, and our algorithms only need to succeed on *typical* datasets—i.e., with high probability over the training samples. This shifts the focus from *worst-case hardness* to *average-case hardness*.
2. Our objective is not merely to decide whether a fixed architecture can fit well the training data, but rather to construct a model that generalizes well to new, unseen data.

In learning theory, the distinction in the second point corresponds to the difference between *proper learning* (where the learned model must belong to a fixed class) and *improper learning* (where any predictive model is allowed, regardless of whether it lies in the original hypothesis class). Let's informally recall their definitions (see [SSBD14] for details):

Definition 1.3.1 (Informal, proper versus improper learning)**.** *For simplicity, consider the realizable setting and an hypothesis class* $\mathcal{H} \subset \{h : \mathcal{X} \mapsto \{-1, +1\}\}$ *(e.g.,* $\mathcal{H}$ *is the set of all* 3*-nodes networks from Figure 1.1). Consider the* 0-1 *loss* $\ell : \{\pm 1\} \times \{\pm 1\} \to \{0, 1\}$ *given by* $\ell(y, \hat{y}) = \mathbb{1}[y \neq \hat{y}]$.

(1) (Proper learning.) *Given data n data points $(y_i, \boldsymbol{x}_i) \sim \mathcal{D}$, where $y_i = h(\boldsymbol{x}_i)$ for some $h \in \mathcal{H}$, the algorithm outputs $\hat{h} \in \mathcal{H}$ such that*

$$\mathcal{R}(\hat{h}; \mathcal{D}) = \mathbb{P}[\hat{h}(\boldsymbol{x}) \neq h(\boldsymbol{x})] \leq \varepsilon. \tag{1.3.1}$$

(2) (Improper learning.) *The algorithm is allowed to output $\hat{h} \notin \mathcal{H}$ (e.g., from a much broader class of models) such that*

$$\mathcal{R}(\hat{h}; \mathcal{D}) = \mathbb{P}[\hat{h}(\boldsymbol{x}) \neq h(\boldsymbol{x})] \leq \varepsilon. \tag{1.3.2}$$

In other words, improper learners are allowed to use a much richer, expressive model class even if the data comes from a simpler class. The following is a classical result in learning theory:

> *Hardness of proper learning does not imply hardness of improper learning.*

Below, we provide a standard counterexample: learning Disjunctive Normal Forms (DNFs).

Definition 1.3.2 (k-term Disjunctive Normal Form (DNF))**.** *Consider $X_1, X_2, \ldots$ boolean variables $X_j \in \{0, 1\}$. $\mathcal{H}$ is the class of* k-term Disjunctive Normal Form *(DNF) if it contains all boolean formulas*[5] *that can written as*

$$h(X_1, X_2, \ldots) = \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{m_k} L_{ij} \right), \tag{1.3.3}$$

where $L_{ij} = X_{s_{ij}}$ or $\neg X_{s_{ij}}$ for some $s_{ij} \geq 1$. Here $\vee$ denotes 'or', $\wedge$ denotes 'and', $\neg$ denotes 'not', and $\bigwedge_{j=1}^{m_k} L_{ij} = L_{i1} \wedge L_{i2} \wedge \cdots \wedge L_{im_k}$, $\bigvee_{i=1}^{k} M_i = M_1 \vee M_2 \vee \cdots \vee M_k$.

Pitt and Valliant (1988) showed that one can reduce the k-NM-Coloring problem, which is NP-complete (2-NM-Coloring corresponds exactly to the Set-Splitting Problem 1.2.3), to the k-term DNF problem.

[5] Here we take all possible input sizes $\{0, 1\}^*$. To restrict to a given input size d, we can simply consider DNFs over at most d boolean variables. Note that h in (1.3.3) depends on at most $m_1 + \ldots + m_k$ variables.

Theorem 1.3.3 ([PV88]). *Unless*[6] $\texttt{RP} = \texttt{NP}$, *there is no polynomial time algorithm that can learn*[7] *properly* k*-term DNFs for* $k \geq 2$.

In contrast, an earlier work by Valiant (1984) showed the following:

Theorem 1.3.4 ([Val84]). *There is a polynomial time algorithm that learns improperly* k*-term DNFs for any* $k \geq 1$.

In this case, the algorithm is allowed to output a function form the larger model class of *k-Conjunctive Normal Forms* (k-CNFs).

Definition 1.3.5 (k-Conjunctive Normal Form (k-CNF)). *Consider* $X_1, X_2, \ldots$ *boolean variables* $X_j \in \{0, 1\}$. $\mathcal{H}$ *is the class of* k-Conjunctive Normal Form *(CNF) if it contains all boolean formulas that can written as*

$$h(X_1, X_2, \ldots) = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{k} L_{ij} \right). \tag{1.3.4}$$

Lemma 1.3.6. *Every* k*-term DNF can be written as a* k*-CNF.*

We leave the proof of this lemma as an exercise. As a starting point, show that for $k = 3$, a 3-DNF can always be rewritten as

$$M_1 \vee M_2 \vee M_3 = \bigwedge_{L_1 \in M_1, L_2 \in M_2, L_3 \in M_3} (L_1 \vee L_2 \vee L_3).$$

Lemma 1.3.7. *There exists a polynomial time algorithm that properly learns* k*-CNFs.*

Using that k-term DNFs are contained in k-CNFs, this lemma directly implies that k-term DNFs are improperly learnable (Theorem 1.3.4).

Proof. Consider the set of k-CNFs over d variables. We construct an algorithm that always outputs a k-CNFs that perfectly fits the n training data points. Denote $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_+})$ the subset of the data that are positively labeled. We construct a sequence a 3-CNFs as follows:

[6] RP is the class of problems that can be solved in polynomial time by a randomized algorithm. We have P $\subseteq$ RP $\subseteq$ NP. One can show that hardness of solving the ERM problem is equivalent to hardness of proper learning, see [SSBD14, Exercise 8.7.4]: if $\mathcal{H}$ is properly learnable by a polynomial time algorithm, then the ERM problem over $\mathcal{H}$ is in RP.

[7] Here, hardness holds in the *distribution-free* setting where the algorithm is required to succeed for all input distributions over $\{0, 1\}^d$. Later work proved hardness for fixed distributions (e.g., uniform distribution [Kha93]) for polynomial-sized DNFs, under a cryptographic hardness assumption. See next section.

- Start with h_0 the conjunction of all possible k-sized clauses $M_j = L_{j1} \vee L_{j2} \vee \ldots \vee L_{jk}$ over the d variables. There are $\binom{2d}{k}$ of them. In particular, $h_0(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \{0,1\}^d$ (indeed, for all $\boldsymbol{x}$, there will be a clause that is not satisfied).

- For each $i \leq n_+$, construct h_i be removing from h_{i-1} all the $L_{j1} \vee L_{j2} \vee \ldots \vee L_{jk} = 0$, that is, the clauses that do not agree with this data point. After this step $h_i(\boldsymbol{x}_j) = 1$ for all $j \leq i$ and $h_i(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \notin \{1, \ldots, i\}$.

- Output the k-CNF $\hat{h} = h_{n_+}$. By construction, $h_{n_+}(\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$ in the training sets that are positively labels, and $h_{n_+}(\boldsymbol{x}) = 0$ for all other $\boldsymbol{x}$. Thus, h_{n_+} correctly predicts the labels of all n training data points.

This algorithm runs in $O(nd^k)$ time and returns a model with zero training error. Let's bound the test error of $\hat{h}$ using a standard uniform convergence bound. Consider h_* the true target 3-CNF:

$$\begin{aligned}
\mathcal{R}(\hat{h}; \mathcal{D}_{h_*}) &= \mathbb{P}(\hat{h}(\boldsymbol{x}) \neq h_*(\boldsymbol{x})) \\
&= \mathbb{P}(\hat{h}(\boldsymbol{x}) \neq h_*(\boldsymbol{x})) - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{h}(\boldsymbol{x}_i) \neq h_*(\boldsymbol{x}_i)] \\
&\leq \sup_{h \in \mathcal{H}^{(d)}_{k\text{-CNF}}} \left| \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{h}(\boldsymbol{x}_i) \neq h_*(\boldsymbol{x}_i)] - \mathbb{P}(\hat{h}(\boldsymbol{x}) \neq h_*(\boldsymbol{x})) \right|.
\end{aligned}$$

where we used on the second line that the train error is zero, and we denoted $\mathcal{H}^{(d)}_{k\text{-CNF}}$ the set of all 3-CNFs over d variables. By Hoeffding's inequality and union bound over $\mathcal{H}^{(d)}_{k\text{-CNF}}$,

$$\begin{aligned}
&\mathbb{P}\left(\sup_{h \in \mathcal{H}^{(d)}_{k\text{-CNF}}} \left| \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{h}(\boldsymbol{x}_i) \neq h_*(\boldsymbol{x}_i)] - \mathbb{P}(\hat{h}(\boldsymbol{x}) \neq h_*(\boldsymbol{x})) \right| \geq t \right) \\
&\leq 2 \left| \mathcal{H}^{(d)}_{k\text{-CNF}} \right| \exp\left\{-2nt^2\right\} \leq \exp\left\{(2d)^k - 2nt^2\right\},
\end{aligned}$$

where we used that $\log_2 \left| \mathcal{H}^{(d)}_{k\text{-CNF}} \right| \leq \binom{2d}{k}$. We conclude that there exists an algorithm that, given n samples from any 3-CNF and input distribution, runs in time $O(nd^k)$ and outputs a model $\hat{h}$ such that

$$\mathcal{R}(\hat{h}; \mathcal{D}_{h_*}) \leq \varepsilon, \quad \text{with probability at least } 1 - \delta,$$

as soon as $n \geq \frac{1}{2\varepsilon^2}\left\{(2d)^k + \log(1/\delta)\right\}$. Hence, 3-CNFs are properly learnable in polynomial time. □

Summary: The example of k-term DNFs illustrates an important lesson: hardness results for proper learning should be interpreted with caution. In many cases, efficient learning is still possible by moving to a richer hypothesis class and allowing for improper learning. That is, rather than requiring the learned model to belong to the same class as the true function, we permit it to belong to a larger, more expressive class—one that is easier to learn from a computational standpoint.

This aligns closely with modern deep learning practice: we train highly-overparametrized neural networks—that is, models that are far more expressive than necessary to fit the training data—, and this overparametrization appears to dramatically simplify the ERM problem, to the point that simple gradient algorithms can reliably find global minimizers. This phenomenon is sometimes referred to as *tractability via overparametrization*, and we refer to [BMR21] for a discussion.

Remark 1.3.8 (Computation-Statistical trade-off)**.** *While enlarging the function class can make computation easier, it comes with a trade-off: generalization may become more challenging. A richer hypothesis class typically requires more data to learn effectively. For instance, as seen in the proof of Lemma 1.3.7, the required sample size n scales with* $\log(|\mathcal{H}|)$*, the logarithm of the size of the hypothesis class.*

> There is a fundamental *computational-statistical trade-off* between proper versus improper learning.

1.4 Hardness of improper learning

If hardness of proper learning can be circumvented through improper learning, can we directly show hardness for improper learning? In particular, given the topic of these lectures, can we show hardness of improper learning of neural networks? This question has been the subject of extensive investigation. A number of results now demonstrate the hardness of improper learning under various assumptions.

In this section, we focus on a particularly influential result by Klivans and Sherstov (2006) [KS09], which illustrates the typical approach used to

establish such hardness results. Klivans and Sherstov showed that, under a certain *cryptographic assumption*, no polynomial-time algorithm can improperly learn 2-hidden layer neural networks with d^{ε} neurons[8], for any $\varepsilon > 0$.

Remark 1.4.1. *What do we mean by a cryptographic assumption? In cryptography, the goal is to design protocols that keep information secure, even in the presence of partial access to that information. However, it is currently unknown whether any widely-used cryptographic protocol can be proven secure under general complexity-theoretic assumptions such as* $\mathtt{P} \neq \mathtt{NP}$*. As a result, modern cryptographic systems are often based on more specific assumptions—typically, the conjectured hardness of well-studied computational problems. The credibility of such assumptions grows with the widespread use of the cryptographic protocols that rely on them.*

In the case of Klivans and Sherstov, the hardness result is based on the assumed intractability of the *Shortest Vector Problem* (SVP) in lattices, a central problem in lattice-based cryptography. This assumption has been extensively studied and underpins many post-quantum cryptographic protocols.

Problem 1.4.2 (Shortest Vector Problem (SVP))**.** *Consider a lattice in n dimensions generated by n basis vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$:*

$$\mathbb{L}_n = \left\{ a_1 \boldsymbol{v}_1 + a_2 \boldsymbol{v}_2 + \ldots + a_n \boldsymbol{v}_n : a_1, a_2, \ldots, a_n \in \mathbb{Z} \right\}. \tag{1.4.1}$$

In the $f(n)$-SVP problem, the goal is to approximate the length of the shortest nonzero vector in $\mathbb{L}_n$ within a factor $f(n)$.

Assuming hardness of the SVP problem, Klivans and Sherstov proved that no polynomial-time algorithm can learn improperly the following hypothesis class of *intersection of halfspaces*.

Definition 1.4.3 (Intersection of k halfspaces)**.** *Consider $\boldsymbol{x} \in \{\pm 1\}^d$. A function $h : \{\pm 1\}^d \to \{0, 1\}$ is called an* intersection of k halfspaces *if there exist k weights $\boldsymbol{w}_j \in \{\pm 1\}^d$ such that*

$$h(\boldsymbol{x}) = \prod_{i=1}^{k} \mathbb{1}[\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle > 0]. \tag{1.4.2}$$

We denote $\mathcal{H}^{(d)}_{k-\text{hlf}}$ the set of all intersections of k halfspaces.

[8] This result was later strengthened by [DLSS14], who showed hardness for learning networks with merely $\omega_d(1)$ neurons, under a different (average-case) assumption.

Theorem 1.4.4 ([KS09]). *Assuming $d^{3/2}$-SVP is hard, then for every $\varepsilon > 0$, the hypothesis class $\mathcal{H}^{(d)}_{k-\mathrm{hlf}}$ cannot be learned efficiently, even improperly.*

The function $h(\boldsymbol{x})$ in Equation (1.4.2) can be written as a 2-hidden layer neural network with ReLu activation, and $2k$ neurons on the first layer and 1 output neuron. For each halfspace with vector $\boldsymbol{w}_j$, we add on layer 1 two neurons $(\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle)_+$ and $(\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle - 1)_+$ (where $(x)_+ = x\mathbb{1}[x \geq 0]$ is the ReLu activation), such that

$$(\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle)_+ - (\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle - 1)_+ = \begin{cases} 1 & \text{if } \langle \boldsymbol{w}_j, \boldsymbol{x}\rangle > 0; \\ 0 & \text{otherwise.} \end{cases}$$

Then the output neuron on layer 2 computes

$$\left(\sum_{j=1}^{k} [(\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle)_+ - (\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle - 1)_+] + 1 - k\right)_+ = \begin{cases} 1 & \text{if } h(\boldsymbol{x}) = 1; \\ 0 & \text{if } h(\boldsymbol{x}) = 0. \end{cases}$$

Thus, Theorem 1.4.4 implies that the class of two-hidden-layer neural networks with d^{ε}-neurons cannot be learned efficiently, even improperly. Note that such networks can be learned statistically efficiently: if one could find a global minimizer of the ERM problem (1.1.7) over the class of neural networks with d^{ε}-neurons, then standard generalization bounds (e.g., see [BFT17]) show that one can achieve small test error with $n = \mathrm{poly}(d)$ samples.

However, Theorem 1.4.4 implies a much stronger result than hardness of the ERM problem over d^{ε}-sized neural networks (which is already implied by Theorem 1.2.2): it rules out any polynomial-time learner, even one that uses a larger, more expressive hypothesis class than the ground-truth network. That is, even if we use a much larger, overparameterized neural network—so that the ERM problem becomes amenable to gradient-based methods—the learned model will still fail to generalize. Overparametrization—which is known to improve optimization in some settings—does not help here!

Proof sketch (Theorem 1.4.4). We now outline the high-level strategy behind the proof. The proof illustrates a general approach for establishing hardness of improper learning by reduction from cryptographic assumptions.

Consider a public-key encryption scheme. Such a system involves two functions: a public encryption function encr and a private decryption function decr. Given a message $\mathsf{m} \in \{0,1\}^p$, the public key is used to generate an encrypted version $\mathsf{x} = \mathsf{encr}(\mathsf{m}) \in \{0,1\}^q$. Then the encrypted message is

decrypted using the private key $\mathsf{decr}(\mathsf{x}) = \mathsf{m}$. Encryption is computationally efficient. On the other hand, decryption should be easy only with access to the private key. In other words, it should be computationally infeasible to recover m from x without the key.

Now suppose that the decryption function decr belongs to a hypothesis class $\mathcal{H}$. If this class could be learned efficiently (even improperly), then an adversary could generate a training set of input-output pairs:

$$\boldsymbol{x} = \mathsf{encr}(\mathsf{m}), \qquad y = \mathsf{m} = \mathsf{decr}(\boldsymbol{x}).$$

(Indeed, the encryption function encr is public and can be computed efficiently.) We can then use a learning algorithm to recover a function that approximates decr on new inputs. This would break the cryptosystem, contradicting the assumed hardness of decryption.

Klivans and Sherstov instantiate this general idea using Regev's cryptosystem (a public-private key pair)—an encryption scheme whose security relies on the assumed hardness of SVP. The decryption function in Regev's scheme can be expressed as the intersection of d^{ε} halfspaces. Therefore, learning intersections of d^{ε} halfspaces would break Regev's cryptosystem, which implies that such learning is computationally hard under the SVP assumption. □

Summary. How compelling is the hardness result in Theorem 1.4.4? Compared to the hardness result in Section 1.2, which concerned proper learning, Theorem 1.4.4 is much stronger: it establishes hardness for improper learning and rules out any polynomial-time learning algorithms (under a cryptographic hardness assumption).

However, the result has an important limitation: it is a worst-case hardness result over input distributions. That is, it shows that there exists an input distribution for which improper learning is computationally hard—but this distribution is not necessarily natural and may not resemble real-world data. In fact, from the proof, we see that the hard distribution is constructed by encrypting messages using a cryptographic scheme, which is not very natural! Does the hardness of improperly learning intersections of halfspaces persist under natural input distributions? Vempala [Vem97] showed that intersections of k halfspaces are learnable under standard Gaussian input $\boldsymbol{x} \sim \mathcal{N}(0, \mathbf{I}_d)$, with both sample and computational complexity bounded by $\mathrm{poly}(d, k) + k^{O(k)}$. In particular, this result implies that the problem remains tractable when the number of halfspaces satisfies $k = O_d(\log d / \log\log d)$, while Daniely et al. [DLSS14] showed that the problem is distribution-free

hard as soon as $k = \omega_d(1)$.

In recent decades, there have been many hardness results on improper learning in both distribution-free and distribution-specific settings. For example, a recent result by Li et al. [LZZ24] establishes the hardness of improper learning for one-hidden-layer neural networks with $\omega(\sqrt{d \log d})$ hidden units and Gaussian input distribution, under hardness assumption for the *Continuous Learning with Errors* (CLWE) problem. We refer the reader to [DV21, LZZ24] and the references therein for an overview of recent developments.

1.5 Restricted models of computation

In this first chapter, we have reviewed classical hardness results based on reductions from well-known problems that are conjectured to be computationally intractable (based on $\mathtt{P} \neq \mathtt{NP}$ or cryptographic hardness assumption). These reductions allow us to derive lower bounds under broadly accepted complexity-theoretic assumptions, lending theoretical weight to the intractability of the learning tasks.

However, this general reduction-based approach has several limitations: 1) Reductions are often non-robust and technically delicate, making them hard to extend to many interesting settings; 2) The resulting results are usually qualitative (e.g., showing that a problem is NP-hard) rather than quantitative; 3) They provide only limited insights on why algorithms used in practice succeed or fail on natural problems.

An alternative line of research has emerged over the past few decades: proving computational lower bounds against restricted models of computation—that is, against specific families of algorithms. While these approaches fall short of ruling out all polynomial-time algorithms (which would anyway require resolving $\mathtt{P} \neq \mathtt{NP}$), they can exclude large and popular classes of algorithms. In some cases, these restricted models are even conjectured to capture the limits of all efficient algorithms for large classes of problems. These restricted-model approaches are often much easier to implement, apply broadly to many settings, and yield sharper lower bounds.

In the remainder of these lectures, we will study four important families of algorithms (or model of computation). For each, we will present techniques to establish lower bounds and discuss their implications:

Chapter 2: Kernel methods. Kernel methods represent one of the sim-

plest class of learning algorithms. Despite their simplicity, they remain foundational to machine learning and are still among the most widely used methods in practice. From a theoretical perspective, kernel methods have seen a resurgence of interest due to the recent observation that neural networks trained in the so-called *lazy regime* behave as kernel methods. For an overview of this connection, see [BMR21, MM24].

Understanding the limitations of fixed feature, kernel methods will help clarify the advantages of more complex learning models, such as neural networks trained in the non-linear, 'feature learning' regime.

Chapter 3: Noisy gradient descent. This chapter studies a simplified proxy for stochastic gradient descent (SGD), in which the algorithm performs population gradient descent with added Gaussian noise. Though this is not an actual implementation of SGD, it captures many key features of its dynamics and has been shown to provide accurate predictions in several settings. As SGD remains the dominant algorithm in deep learning, understanding its limitations is of central interest. We will show how to prove lower bounds on noisy population gradient descent using a 'junk flow' argument developed by Abbe and Sandon [AS20].

Chapter 4: Statistical Query algorithms. Introduced by Kearns in the 1990s, the SQ model abstracts algorithms that access the data only through statistical expectations (e.g., empirical means), rather than individual examples. Many robust and noise-tolerant algorithms fall within the SQ framework. Lower bounds in this model imply hardness for a broad range of practical algorithms and have played a central role in understanding the limits of learning in high-noise or agnostic settings.

Chapter 5: Low-Degree Polynomial algorithms. This model captures algorithms whose outputs can be expressed as low-degree polynomials of the input data. It has been particularly influential in average-case complexity and high-dimensional statistics. The "low-degree method" is conjectured to characterize the power of all efficient algorithms in a wide range of problems, including planted clique or tensor PCA. We will show how to derive sharp lower bounds for learning Gaussian single-index models using this framework.

Chapter 2

Kernel methods

In this chapter, we study *kernel methods*, also known as *kernel machines* or *RKHS methods*. These algorithms form a rich yet tractable class of learning methods that allows the construction of non-linear predictors—often in infinite-dimensional spaces—while maintaining computational efficiency thanks to the celebrated *kernel trick*.

Formally, kernel methods arise from a specific choice of models and regularization in the ERM problem (1.1.7). The idea is to first map the input data $\boldsymbol{x} \in \mathbb{R}^d$ into a rich, higher-dimensional feature space via a feature map $\Phi(\boldsymbol{x}) \in \mathbb{R}^p$ (possibly with $p = \infty$), and then fit a linear predictor of the form $f(\boldsymbol{x}; \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}) \rangle$ subject to a *RKHS-norm* regularization penalty (here, $r(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|_2^2$). Although the model is linear in the parameters $\boldsymbol{\theta}$, it is non-linear in the input data $\boldsymbol{x}$—a key property that generalizes standard linear regression in statistics (which correspond to $\Phi(\boldsymbol{x}) = \boldsymbol{x}$). Intuitively, this embedding can transform non-linearly separable data into a space where linear classification becomes possible.

We begin our exploration of restricted algorithmic classes with kernel methods because of their foundational role. Despite their apparent simplicity, they remain one of the most versatile and widely used technique in statistics and machine learning. Moreover, they are deeply connected to modern deep learning: in a certain optimization regime—the so-called *lazy* or *linearized regime* [JGH18, COB19]—neural networks behave like kernel machines. We will present a simple, general *dimension lower bound* on learning with kernel methods in the regression setting (with squared test loss).

2.1 Feature space and RKHS

Consider a general input space $\mathcal{X}$. Below, we construct the classes $\mathcal{F}$ of models $f : \mathcal{X} \to \mathbb{R}$ associated to kernel methods.

There are several equivalent ways to define this function space; here, we adopt a particularly intuitive one: we define it through a *featurization map* and an associated Hilbert *feature space*, and interpret our model class as the set of all linear functionals on this space.

Definition 2.1.1 (Feature space, featurization map, and linear models)**.**

(1) (Feature space.) *A* feature space *is a Hilbert space* $(\mathfrak{F}, \langle \cdot, \cdot \rangle_{\mathfrak{F}})$ *with inner product* $\langle \cdot, \cdot \rangle_{\mathfrak{F}}$ *and associated norm* $\|\Phi\|_{\mathfrak{F}} = \langle \Phi, \Phi \rangle_{\mathfrak{F}}^{1/2}$.

(2) (Featurization map.) *A* featurization map *is a function* $\Phi : \mathcal{X} \to \mathfrak{F}$ *that embeds the input data* $\boldsymbol{x} \in \mathcal{X}$ *into the feature space* $\Phi(\boldsymbol{x}) \in \mathfrak{F}$.

(3) (Linear model class.) *Given* $(\mathfrak{F}, \Phi)$*, the associated class of models is the set of all linear functionals with respect to this embedding*

$$\mathcal{F} = \big\{ \boldsymbol{x} \mapsto f(\boldsymbol{x}; \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}) \rangle_{\mathfrak{F}} : \boldsymbol{\theta} \in \mathfrak{F} \big\}. \tag{2.1.1}$$

This defines a function space $\mathcal{F}$ consisting of functions $f : \mathcal{X} \to \mathbb{R}$ that are linear in the feature space. The inner product and norm on $\mathcal{F}$ are inherited from the feature space $\mathfrak{F}$ via:

$$\langle f(\cdot; \boldsymbol{\theta}), f(\cdot; \boldsymbol{\theta}') \rangle_{\mathcal{F}} = \langle \boldsymbol{\theta}, \boldsymbol{\theta}' \rangle_{\mathfrak{F}}, \qquad \|f(\cdot; \boldsymbol{\theta})\|_{\mathcal{F}} = \|\boldsymbol{\theta}\|_{\mathfrak{F}}. \tag{2.1.2}$$

This identification endows $\mathcal{F}$ with a Hilbert space structure, assuming the map $\boldsymbol{\theta} \mapsto f(\cdot; \boldsymbol{\theta})$ is injective.

Remark 2.1.2. *In general, multiple parameter vectors* $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathfrak{F}$ *may define the same function: that is,* $f(\cdot; \boldsymbol{\theta}) = f(\cdot; \boldsymbol{\theta}')$ *as functions on* $\mathcal{X}$*, even if* $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$*. To define a proper Hilbert space structure on* $\mathcal{F}$*, we must quotient out this ambiguity. That is, we define an equivalence relation on* $\mathfrak{F}$ *by* $\boldsymbol{\theta} \sim \boldsymbol{\theta}'$ *if* $f(\boldsymbol{x}; \boldsymbol{\theta}) = f(\boldsymbol{x}; \boldsymbol{\theta}')$ *for all* $\boldsymbol{x} \in \mathcal{X}$*, and set*

$$\tilde{\mathfrak{F}} := \mathfrak{F} / \ker(\Phi)$$

where $\ker(\Phi) = \{\boldsymbol{\theta} \in \mathfrak{F} : \langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}) \rangle_{\mathfrak{F}} = 0 \ \ \forall \boldsymbol{x} \in \mathcal{X}\}$ *is the linear subspace of parameter vectors that correspond to the zero function. The inner product on* $\tilde{\mathfrak{F}}$ *is well-defined. The map from* $\tilde{\mathfrak{F}}$ *to* $\mathcal{F}$ *is now injective and induces a valid Hilbert space structure (Prove it!).*

We next present three examples of feature space, featurization map, and associated linear model classes.

Example 2.1.3 (Standard linear regression)**.** *Let $\mathcal{X} = \mathbb{R}^d$. We define the feature space $\mathfrak{F} = \mathbb{R}^d$ equipped with the standard Euclidean inner product:*

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = u_1 v_1 + \ldots + u_d v_d.$$

We take the identity map as the feature map: $\Phi(\boldsymbol{x}) = \boldsymbol{x}$. Then, the class of linear models associated to $(\mathfrak{F}, \Phi)$ becomes:

$$\mathcal{F} = \left\{ f(\boldsymbol{x}; \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle : \boldsymbol{\theta} \in \mathbb{R}^d \right\},$$

which corresponds to standard linear regression. The associated function norm is simply $\|f(\cdot; \boldsymbol{\theta})\|_{\mathcal{F}} = \|\boldsymbol{\theta}\|_2$.

Example 2.1.4 (Polynomial regression model)**.** *For simplicity, let $\mathcal{X} = \mathbb{R}$. Choose the feature space $\mathfrak{F} = \mathbb{R}^{k+1}$ with the standard Euclidean inner product. Define the feature map*

$$\Phi(x) = (1, x, x^2, \ldots, x^k).$$

Then the linear model class associated to $(\mathfrak{F}, \Phi)$ consists of all univariate polynomials of degree at most k:

$$f(x; \boldsymbol{\theta}) = \theta_1 + \theta_2 x + \theta_3 x^2 + \ldots + \theta_{k+1} x^k.$$

As before, the function norm is given by $\|f(\cdot; \boldsymbol{\theta})\|_{\mathcal{F}} = \|\boldsymbol{\theta}\|_2$.

Example 2.1.5 (Infinite-width two-layer neural networks)**.** *Let (Ω, μ) be a probability space and define the feature space as the Hilbert space of square-integrable functions:*

$$\mathfrak{F} = L^2(\Omega, \mu) = \left\{ a : \Omega \to \mathbb{R} : \|a\|_{L^2}^2 = \int_\Omega a(\boldsymbol{w})^2 \mu(\mathrm{d}\boldsymbol{w}) < \infty \right\},$$

with inner product

$$\langle a, b \rangle_{L^2} = \int_\Omega a(\boldsymbol{w}) b(\boldsymbol{w}) \mu(\mathrm{d}\boldsymbol{w}).$$

Let $\varphi : \mathcal{X} \times \Omega \to \mathbb{R}$ be a mapping such that $\varphi(\boldsymbol{x}; \cdot) \in L^2(\Omega, \mu)$ for all $\boldsymbol{x} \in \mathcal{X}$. For example, $\varphi(\boldsymbol{x}; \boldsymbol{w}) = \sigma(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)$, where σ is a non-linear activation function (e.g., ReLU). Define the featurization map $\Phi : \mathcal{X} \to L^2(\Omega, \mu)$ to be

$$\Phi(\boldsymbol{x}) := \varphi(\boldsymbol{x}; \cdot), \quad \Phi(\boldsymbol{x})(\boldsymbol{w}) = \varphi(\boldsymbol{x}; \boldsymbol{w}).$$

Then the model class is:

$$\mathcal{F} = \left\{ \boldsymbol{x} \mapsto f(\boldsymbol{x}; a) = \langle a, \Phi(\boldsymbol{x}) \rangle_{L^2} = \int_\Omega a(\boldsymbol{w}) \varphi(\boldsymbol{x}; \boldsymbol{w}) \mu(\mathrm{d}\boldsymbol{w}) : a \in L^2(\Omega, \mu) \right\},$$

with norm $\|f(\cdot; a)\|_{\mathcal{F}} = \|a\|_{L^2}$*. This corresponds to the class of infinite-width two-layer neural network where the second layer weights have bounded* L^2 *norm with respect to the distribution* μ *of the first layer weights* $\boldsymbol{w}$[1]*.*

The function space $(\mathcal{F}, \langle \cdot, \cdot \rangle_{\mathcal{F}})$ introduced in Equation (2.1.1) has an important structure: it is a *Reproducing Kernel Hilbert Space* (RKHS). While we initially defined this function space via a feature map into a Hilbert space, RKHS theory provides an alternative—but equivalent—construction via a kernel that does not require explicitly defining the feature map. This kernel-based viewpoint will be important in the next section, and we briefly introduce it here. For a comprehensive treatment, see [BTA11].

Definition 2.1.6 (Reproducing Kernel Hilbert space (RKHS))**.** *A Hilbert space* $(\mathfrak{H}, \langle \cdot, \cdot \rangle_{\mathfrak{H}})$ *consisting of real-valued functions* $f : \mathcal{X} \to \mathbb{R}$ *is a* Reproducing Kernel Hilbert space *(RKHS) if there exists a* reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, that is, a symmetric, positive semi-definite (PSD) kernel*[2] *such that* $K(\boldsymbol{x}, \cdot) \in \mathfrak{H}$ *for all* $\boldsymbol{x} \in \mathcal{X}$ *and*

$$\langle f, K(\boldsymbol{x}, \cdot) \rangle_{\mathfrak{H}} = f(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \mathcal{X}, \ \ \forall f \in \mathfrak{H}. \tag{2.1.3}$$

Every RKHS defines a unique reproducing kernel, and conversely, the Moore–Aronszajn theorem [Aro50] states that any symmetric, positive definite kernel defines a unique RKHS.

The function space $\mathcal{F} = \{ f(\cdot; \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \Phi(\cdot) \rangle_{\mathfrak{F}} : \boldsymbol{\theta} \in \mathfrak{F} \}$ with induced inner product by $\mathfrak{F}$ (see Remark 2.1.2) is indeed an RKHS. Its reproducing kernel is

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{F}}, \tag{2.1.4}$$

which is clearly symmetric and PSD, and satisfies the reproducing property (2.1.3). The kernels associated with the above three examples are:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle, \qquad \text{(Example 2.1.3)}$$

[1] This will have important consequences on learning: many natural functions—with low-dimensional structure—will not lie in $\mathcal{F}$ and will not be learned efficiently by kernel methods. See [Bac17, CMM21].

[2] A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is PSD if for all integer $n \in \mathbb{N}$, points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$ and scalars $a_1, \ldots, a_n \in \mathbb{R}$,

$$\sum_{i,j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

$$K(x, x') = 1 + \sum_{s=1}^{k} (xx')^s, \qquad \text{(Example 2.1.4)}$$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \int_{\Omega} \varphi(\boldsymbol{x}; \boldsymbol{w}) \varphi(\boldsymbol{x}'; \boldsymbol{w}) \mu(\mathrm{d}\boldsymbol{w}). \qquad \text{(Example 2.1.5)}$$

Feature Maps vs. Kernels. The feature map and kernel perspectives are mathematically equivalent ways of defining the same function space. Given any RKHS $(\mathfrak{H}, \langle \cdot, \cdot \rangle_{\mathfrak{H}})$ with reproducing kernel K, one can always choose the feature space to be the RKHS itself $\mathfrak{F} = \mathfrak{H}$ and define the canonical feature map:

$$\Phi : \mathcal{X} \to \mathfrak{H}, \qquad \Phi(\boldsymbol{x}) := K(\boldsymbol{x}, \cdot).$$

This construction satisfies:

$$f(\boldsymbol{x}) = \langle f, K(\boldsymbol{x}, \cdot) \rangle_{\mathfrak{H}} = \langle f, \Phi(\boldsymbol{\theta}) \rangle_{\mathfrak{F}},$$

by the reproducing property (2.1.3). Note that a given RKHS admits infinitely many equivalent featurizations $(\mathfrak{F}, \Phi)$ yielding the same kernel[3].

Why work with kernels? The construction via feature map provides a concrete, geometric intuition for kernel methods: they correspond to linear models in an (often infinite-dimensional) Hilbert space of features, where inner products and norms are naturally inherited from the underlying feature space. However, in practice, we often prefer the definition through kernels: the kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ can be defined and evaluated directly using simple formulas, while the featurization map $\Phi(\boldsymbol{x})$ is often infinite dimensional and not explicitly computable. For example, the Radial Basis Function (RBF) kernel,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}\right),$$

is straightforward to compute, but the corresponding feature map lives in an infinite-dimensional space and is difficult to construct explicitly.

Moreover, as we see in the next section, all computations in kernel methods, such during training or prediction, can be expressed purely in terms of the kernel thanks to the so-called *kernel trick*.

[3]While all such maps are equivalent in terms of learning, some featurization might provide better finite-dimensional approximations, e.g., such as random features [BBV06, RR07].

Kernel diagonalization. Before turning back to kernel methods, let us introduce one last characterization of the RKHS—through the eigendecomposition of the kernel. This gives a more transparent definition of RKHS as subsets of $L^2(\mathbb{P}_{\mathcal{X}})$, and plays a crucial role in the generalization performance of kernel methods. Given a kernel function K such that $\mathbb{E}_{\boldsymbol{x}}[K(\boldsymbol{x},\boldsymbol{x})] < \infty$, we can define the associated 'integral' operator $\mathbb{K} : L^2(\mathbb{P}_{\mathcal{X}}) \to L^2(\mathbb{P}_{\mathcal{X}})$ as

$$(\mathbb{K}f)(\boldsymbol{x}) = \int_{\mathcal{X}} K(\boldsymbol{x},\boldsymbol{x}')f(\boldsymbol{x}')\mathbb{P}_{\mathcal{X}}(\mathrm{d}\boldsymbol{x}'). \tag{2.1.5}$$

Note that $\mathbb{K}$ is a compact, self-adjoint linear operator on $L^2(\mathbb{P}_{\mathcal{X}})$ with

$$\mathrm{Tr}(\mathbb{K}) = \mathbb{E}_{\boldsymbol{x}}\left[\|K(\boldsymbol{x},\cdot)\|_{\mathfrak{H}}^2\right] = \mathbb{E}_{\boldsymbol{x}}[K(\boldsymbol{x},\boldsymbol{x})] < \infty.$$

We say that the operator $\mathbb{K}$ is 'trace-class' when it has bounded trace.

By the spectral theorem for compact self-adjoint linear operators, the integral operator admits the following diagonalization:

$$\mathbb{K} = \sum_{j\geq 1} \lambda_j \psi_j \psi_j^*,$$

where $\lambda_1 \geq \lambda_2 \geq \cdots$ are the (positive) eigenvalues in non-increasing order, and $\{\psi_j\}_{j\geq 1}$ is the set of orthonormal eigenfunctions. That is,

$$\mathbb{K}\psi_j = \lambda_j \psi_j, \qquad \langle \psi_j, \psi_{j'} \rangle_{L^2} = \delta_{jj'}.$$

Denote $\mathcal{V} = \mathrm{span}\{\psi_j : j \geq 1\}$ the subspace of $L^2(\mathbb{P}_{\mathcal{X}})$ spanned by these eigenfunctions. In particular, $\mathcal{V} = L^2(\mathbb{P}_{\mathcal{X}})$ if and only if $\{\psi_j\}_{j\geq 1}$ forms a complete basis of $L^2(\mathbb{P}_{\mathcal{X}})$.

In fact, the image of the operator is exactly $\mathrm{Im}(\mathbb{K}) = \mathfrak{H}$ the RKHS with reproducing kernel K: you can check that indeed $\langle \mathbb{K}f, K(\boldsymbol{x},\cdot)\rangle_{\mathfrak{H}} = \mathbb{K}f(\boldsymbol{x})$ by definition. In particular, consider $f \in \mathcal{V}$ and its expansion in the eigenfunction basis:

$$f = \sum_{j\geq 1} \beta_j \psi_j, \qquad \beta_j := \langle f, \psi_j \rangle_{L^2}, \qquad \text{with } \|f\|_{L^2}^2 = \sum_{j\geq 1} \beta_j^2 < \infty.$$

Then, the RKHS norm of f is given by

$$\|f\|_{\mathfrak{H}}^2 = \sum_{j=1}^{\infty} \frac{\beta_j^2}{\lambda_j}.$$

Thus, the RKHS $\mathfrak{H}$ is simply the subset of all functions in $\mathcal{V}$ with $\|f\|_{\mathfrak{H}} < \infty$, that is,

$$\mathfrak{H} := \left\{ f \in \mathcal{V} \ : \ \|f\|_{\mathfrak{H}}^2 = \sum_{j \geq 1} \lambda_j^{-1} \langle f, \psi_j \rangle_{L^2}^2 < \infty \right\}. \tag{2.1.6}$$

The RKHS $\mathfrak{H}$ is a dense subset of $\mathcal{V}$—and of $L^2(\mathbb{P}_{\mathcal{X}})$ if $\{\psi_j\}_{j \geq 1}$ forms a complete basis. In this case, we can approximate any $f \in L^2(\mathbb{P}_{\mathcal{X}})$ by a sequence of functions in $\mathfrak{H}$: the RKHS model class satisfies a 'universal approximation' property. But as we will see, it is not always statistically efficient.

Finally, we can use the above characterization (2.1.6) to define another canonical feature map. Let the feature space $\mathfrak{F}$ be $(\ell_2, \langle \cdot, \cdot \rangle_{\ell_2})$ the Hilbert space of square summable sequences, with

$$\langle a, b \rangle_{\ell_2} = \sum_{j=1}^{\infty} a_j b_j.$$

Then we can define the featurization map as $\Phi(\boldsymbol{x}) = \{\sqrt{\lambda_j}\psi_j\}_{j=1}^{\infty}$. Thus, the set of linear models is simply (equality in L^2)

$$f(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}) \rangle_{\ell_2} = \sum_{j=1}^{\infty} \theta_j \sqrt{\lambda_j} \psi_j(\boldsymbol{x}), \qquad \|\boldsymbol{\theta}\|_{\ell_2}^2 = \sum_{j=1}^{\infty} \theta_j^2 < \infty.$$

We can see $\mathbb{K}$ acting on $\mathcal{V}$ as acting on the ℓ_2 space, with $\mathbb{K}(a_j)_{j=1}^{\infty} = (\lambda_j a_j)_{j=1}^{\infty}$.

2.2 Kernel methods and the kernel trick

A kernel method is simply the solution to the Empirical Risk Minimization problem (1.1.7) over a Reproducing Kernel Hilbert Space $(\mathfrak{H}, \langle \cdot, \cdot \rangle_{\mathfrak{H}})$, with a regularization penalty proportional to the RKHS norm. That is, given training data $(y_i, \boldsymbol{x}_i) \in \mathbb{R} \times \mathcal{X}$, $i = 1, \ldots, n$, the learned predictor $\hat{f}$ solves

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathfrak{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)) + \lambda \|f\|_{\mathfrak{H}}^2 \right\}. \tag{2.2.1}$$

Equivalently, if the RKHS $\mathfrak{H}$ is defined via an embedding $(\mathfrak{F}, \Phi)$, then the solution of the ERM problem is given by $\hat{f} = \langle \hat{\boldsymbol{\theta}}, \Phi(\cdot) \rangle_{\mathfrak{F}}$ with

$$\hat{\boldsymbol{\theta}} = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathfrak{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}}) + \lambda \|\boldsymbol{\theta}\|_{\mathfrak{F}}^2 \right\}. \tag{2.2.2}$$

This formulation appears computationally intractable: it is an optimization over an infinite-dimensional space. So is the method practical? Can it be implemented efficiently? Surprisingly, *yes*—and this is due to the following *representer theorem*, which shows that the solution always lies in a finite-dimensional subspace of $\mathfrak{F}$, namely the span of the embedded training data.

Theorem 2.2.1 (Representer theorem)**.** *Let ℓ be any loss function (e.g., not necessarily convex). Then the solution $\hat{\boldsymbol{\theta}}$ of the ERM problem (2.2.2) satisfies*

$$\hat{\boldsymbol{\theta}} \in \operatorname{span}\{\Phi(\boldsymbol{x}_1), \dots, \Phi(\boldsymbol{x}_n)\}. \tag{2.2.3}$$

In other words, there exist coefficients $\hat{a}_1, \dots \hat{a}_n \in \mathbb{R}$ such that

$$\hat{\boldsymbol{\theta}} = \sum_{i=1}^{n} \hat{a}_i \Phi(\boldsymbol{x}_i). \tag{2.2.4}$$

Proof. The proof is simple and relies only on the Hilbert space structure of $\mathfrak{F}$. Let $V_{\|} = \operatorname{span}\{\Phi(\boldsymbol{x}_i) : i \in [n]\}$, and let $V_{\perp}$ be its orthogonal complement in $\mathfrak{F}$, so that

$$\mathfrak{F} = V_{\|} \oplus V_{\perp}.$$

Then any $\boldsymbol{\theta} \in \mathfrak{F}$ can be decomposed uniquely as $\boldsymbol{\theta} = \boldsymbol{\theta}_{\|} + \boldsymbol{\theta}_{\perp}$ with $\boldsymbol{\theta}_{\|} \in V_{\|}$ and $\boldsymbol{\theta}_{\perp} \in V_{\perp}$. Because $\Phi(\boldsymbol{x}_i) \in V_{\|}$, we have

$$\langle \boldsymbol{\theta}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}} = \langle \boldsymbol{\theta}_{\|}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}} + \langle \boldsymbol{\theta}_{\perp}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}} = \langle \boldsymbol{\theta}_{\|}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}}.$$

Thus, $\boldsymbol{\theta}_{\perp}$ has no effect on the loss, but it contributes positively to the regularization term $\|\boldsymbol{\theta}\|_{\mathfrak{F}}^2 = \|\boldsymbol{\theta}_{\|}\|_{\mathfrak{F}}^2 + \|\boldsymbol{\theta}_{\perp}\|_{\mathfrak{F}}^2$. Hence, the optimal solution

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \boldsymbol{\theta}_{\|}, \Phi(\boldsymbol{x}_i) \rangle_{\mathfrak{F}}) + \lambda \|\boldsymbol{\theta}_{\|}\|_{\mathcal{F}}^2 + \lambda \|\boldsymbol{\theta}_{\perp}\|_{\mathcal{F}}^2 \right\}$$

must satisfy $\boldsymbol{\theta}_{\perp} = 0$, i.e., $\hat{\boldsymbol{\theta}} \in V_{\perp}$. □

Thanks to the representer theorem, we can now reduce the infinite-dimensional optimization problem (2.2.2) to a finite-dimensional one over the coefficients $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathbb{R}^n$, where $\boldsymbol{\theta} = \sum_{i=1}^{n} a_i \Phi(\boldsymbol{x}_i)$. Define:

$$\boldsymbol{k}_n(\boldsymbol{x}) := (K(\boldsymbol{x}, \boldsymbol{x}_i))_{i \in [n]} \in \mathbb{R}^n, \qquad \boldsymbol{K}_n := (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{ij \in [n]} \in \mathbb{R}^{n \times n},$$

where $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathfrak{F}}$ is the reproducing kernel. Then we can rewrite:

$$\langle \Phi(\boldsymbol{x}), \boldsymbol{\theta} \rangle_{\mathfrak{F}} = \sum_{i=1}^{n} a_i K(\boldsymbol{x}, \boldsymbol{x}_i) = \boldsymbol{a}^\mathsf{T} \boldsymbol{k}_n(\boldsymbol{x}),$$

$$\|\boldsymbol{\theta}\|_{\mathfrak{F}}^2 = \sum_{i,j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{a}^\mathsf{T} \boldsymbol{K}_n \boldsymbol{a}.$$

Substituting into the ERM objective, we obtain a finite-dimensional optimization problem:

$$\hat{\boldsymbol{a}} = \operatorname*{arg\,min}_{\boldsymbol{a} \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \boldsymbol{a}^\mathsf{T} \boldsymbol{k}_n(\boldsymbol{x}_i)) + \lambda \boldsymbol{a}^\mathsf{T} \boldsymbol{K}_n \boldsymbol{a} \right\}. \tag{2.2.5}$$

In particular, if ℓ is convex in its second argument, this is an n-dimensional convex optimization problem and can be solved efficiently! Once we have computed $\hat{\boldsymbol{a}}$, the predictor is given by

$$\hat{f}(\boldsymbol{x}) = \langle \hat{\boldsymbol{\theta}}, \Phi(\boldsymbol{x}) \rangle_{\mathfrak{F}} = \sum_{i=1}^{n} \hat{a}_i K(\boldsymbol{x}, \boldsymbol{x}_i) = \hat{\boldsymbol{a}}^\mathsf{T} \boldsymbol{k}_n(\boldsymbol{x}). \tag{2.2.6}$$

Thus to train kernel methods and make new predictions, we never need to compute the feature map $\Phi(\boldsymbol{x})$ explicitly—only the inner products $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathfrak{F}}$. This observation is known as the *kernel trick*: it allows to construct expressive, non-linear classifiers in infinite-dimensional feature space without ever having to explicitly represent the features. This is the reason kernel methods became so popular!

For concreteness, we present two popular examples of kernel methods below.

Example 2.2.2 (Kernel Ridge Regression (KRR))**.** *Let the loss function be the squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. Then* kernel ridge regression *(KRR) corresponds to solving the ERM problem*

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathfrak{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \big(y_i - f(\boldsymbol{x}_i)\big)^2 + \lambda \|f\|_{\mathfrak{H}}^2 \right\}.$$

By the representer problem, the solution admits an explicit formula $\hat{f} = \hat{\boldsymbol{a}}^\mathsf{T} \boldsymbol{k}_n(\cdot)$ where

$$\hat{\boldsymbol{a}} = \operatorname*{arg\,min}_{\boldsymbol{a} \in \mathbb{R}^n} \left\{ \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{K}_n \boldsymbol{a}\|_2^2 + \lambda \boldsymbol{a}^\mathsf{T} \boldsymbol{K}_n \boldsymbol{a} \right\} = (\boldsymbol{K}_n + n\lambda)^{-1} \boldsymbol{y},$$

where $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathbb{R}^n$*, so that* $\hat{f} = \boldsymbol{y}^\mathsf{T}(\boldsymbol{K}_n + n\lambda)^{-1}\boldsymbol{k}_n(\cdot)$*. Thanks to this analytical expression, one can derive very precise theory on the performance of KRR, see for example [CDV07, DW18, MM24, MS24].*

Example 2.2.3 (Support Vector Machines (SVM))**.** *SVMs are designed for binary classification. Suppose* $y_i \in \{-1, +1\}$ *and let the loss function be the hinge loss* $\ell(y, \hat{y}) = \max(0, 1 - y\hat{y})$*. The SVM classifier minimizes the regularized empirical risk*

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathfrak{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right) + \lambda \|f\|_{\mathfrak{H}}^2 \right\},$$

and we predict new labels via the sign $\hat{y} = \operatorname{sign}(\hat{f}(\boldsymbol{x}))$*.*

It is standard to solve SVMs via their dual formulation. By the representer theorem, we can write the dual variables $\boldsymbol{\alpha} \in \mathbb{R}^n$ *which satisfy*

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

$$\text{subject to} \quad 0 \le \alpha_i \le \frac{1}{2n\lambda} \ \ \forall i, \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

The predictor is given by $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$*. Most of the coefficients* α_i *are zero—only the* support vectors *(i.e., data points close to the decision boundary) have non-zero weights.*

2.3 Dimension lower bounds

In this section, we present a simple and general lower bound on the performance of kernel methods. As we saw in the previous section, kernel methods trained on n data points involve solving an n-dimensional convex problem, and the computational complexity of these methods is closely tied to the number of training samples. Thus, our result will be a lower bound directly on the statistical complexity of kernel methods.

For concreteness, we focus on the regression setting and quantify performance using the test error under squared loss. Our lower bound will apply to any predictor obtained by a kernel method, regardless of the choice of

RKHS, training loss ℓ, or regularization parameter $\lambda > 0$ in the ERM problem (2.2.1). For related results in the classification setting (where the test loss is a classification loss), we refer the reader to [KMS20].

The argument relies on a geometric observation: kernel methods construct predictors from a fixed, finite-dimensional subspace—one that depends only on the input data, and *not* on the target function. Therefore, their ability to approximate a class of target functions is fundamentally limited by the expressive power of that subspace. Our lower bound will be the consequence of a general *dimension lower bound* that states that any linear subspace of functions that aims to approximate a collection of functions $\mathcal{H}$ must have dimension at least proportional to some complexity measure of $\mathcal{H}$.

Such dimension lower bounds have appeared in various forms in the literature—for instance, in approximation theory [Bar93]—and have recently gained renewed attention in the context of learning theory [KMS20, DM20, HSSVG21, AZL20, AAM22]. Below we follow the presentation and proof by Hsu [Hsu21].

Theorem 2.3.1 ([Hsu21, AAM22])**.** *Let $\mathcal{R}$ be a Hilbert space with inner-product $\langle \cdot, \cdot \rangle_{\mathcal{R}}$, and $\mathcal{H} = \{h_1, \dots, h_k\} \subset \mathcal{R}$ be a set of k elements in $\mathcal{R}$ with $\|h_i\|_{\mathcal{R}} = 1, \forall i \in [k]$. Let $\mathcal{T} \subset \mathcal{R}$ be a finite-dimensional linear subspace of $\mathcal{R}$, with dimension m. Define the average squared approximation error:*

$$\varepsilon := \frac{1}{k} \sum_{i=1}^{k} \inf_{f \in \mathcal{T}} \|f - h_i\|_{\mathcal{R}}^2.$$

Then

$$m \geq \frac{k}{\|\boldsymbol{G}\|_{\mathrm{op}}}(1 - \varepsilon),$$

where $\boldsymbol{G} = (G_{ij})_{ij \in [k]} = (\langle h_i, h_j \rangle_{\mathcal{R}})_{ij \in [k]} \in \mathbb{R}^{k \times k}$ is the Gram matrix of $\mathcal{H}$.

Proof. Let $\phi_1, \phi_2, \dots, \phi_m$ be an orthonormal basis of $\mathcal{T}$ (with $\langle \phi_i, \phi_j \rangle_{\mathcal{R}} = \delta_{ij}$) and let $\Pi_{\mathcal{T}}$ be the orthonal projection onto $\mathcal{T}$ in $(\mathcal{R}, \langle \cdot, \cdot \rangle)$. We can write for every $h \in \mathcal{R}$:

$$\Pi_{\mathcal{T}} h = \sum_{j=1}^{m} \phi_j \langle \phi_j, h \rangle_{\mathcal{R}}.$$

By definition,

$$\begin{aligned}\varepsilon = \frac{1}{k}\sum_{i=1}^{k} \inf_{f\in\mathcal{T}} \|f - h_i\|_{\mathcal{R}}^2 &= \frac{1}{k}\sum_{i=1}^{k} 1 - \|\Pi_{\mathcal{T}} h_i\|_{\mathcal{R}}^2 \\ &= 1 - \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{m} \langle \phi_j, h_i\rangle_{\mathcal{R}}^2,\end{aligned} \tag{2.3.1}$$

where we used that $\|h_i\|_{\mathcal{R}} = 1$ in the second equality. For any $f \in \mathcal{R}$, we have

$$\begin{aligned}\sum_{i=1}^{k} \langle f, h_i\rangle_{\mathcal{R}}^2 &= \left\langle f, \sum_{i=1}^{k} \langle f, h_i\rangle_{\mathcal{R}} h_i \right\rangle_{\mathcal{R}} \\ &\le \|f\|_{\mathcal{R}} \left(\sum_{i,j=1}^{k} \langle f, h_i\rangle_{\mathcal{R}} \langle f, h_j\rangle_{\mathcal{R}} \langle h_i, h_j\rangle \right)^{1/2} \\ &\le \|f\|_{\mathcal{R}} (\boldsymbol{b}^\top \boldsymbol{G}\boldsymbol{b})^{1/2} \le \|\boldsymbol{g}\|_{\mathcal{R}} \|\boldsymbol{G}\|_{\mathrm{op}}^{1/2} \|\boldsymbol{b}\|_2,\end{aligned}$$

where we introduced the vector $\boldsymbol{b} := (\langle f, h_j\rangle)_{j=1}^k$. Note that $\|\boldsymbol{b}\|_2^2$ corresponds exactly to the left hand-side of the above inequality. Squaring both sides and dividing by $\|\boldsymbol{b}\|_2^2$, we obtain

$$\sum_{i=1}^{k} \langle f, h_i\rangle_{\mathcal{R}}^2 \le \|f\|_{\mathcal{R}}^2 \|\boldsymbol{G}\|_{\mathrm{op}}.$$

Using this upper bound in Equation (2.3.1), we deduce that

$$1 - \varepsilon = \frac{1}{k}\sum_{j=1}^{m}\sum_{i=1}^{k} \langle \phi_j, h_i\rangle_{\mathcal{R}}^2 \le \frac{m}{k}\|\boldsymbol{G}\|_{\mathrm{op}}.$$

Rearranging the terms yields the desired bound. □

Theorem 2.3.1 gives a general lower bound on function approximation in Hilbert spaces. Given a function class $\mathcal{H} = \{h_1, \ldots, h_k\}$ of unit-norm elements in a Hilbert space $(\mathcal{R}, \langle\cdot,\cdot\rangle_{\mathcal{R}})$, we wish to approximate each function h_i using a model $f \in \mathcal{T}$, where $\mathcal{T}$ is a finite-dimensional linear subspace of $\mathcal{R}$. For each $h_i \in \mathcal{H}$, the best possible approximation error in squared norm is given by

$$\inf_{f\in\mathcal{T}} \|f - h_i\|_{\mathcal{R}}^2.$$

Then, Theorem 2.3.1 states that in order to achieve ε average squared approximation error over the class $\mathcal{H}$, the subspace $\mathcal{T}$ must have dimension at least

$$\dim(\mathcal{T}) \geq \frac{|\mathcal{H}|}{\|\boldsymbol{G}\|_{\mathrm{op}}}(1-\varepsilon).$$

If the functions $\{h_i\}$ are orthonormal, such that $\|\boldsymbol{G}\|_{\mathrm{op}} = \|\mathbf{I}_k\|_{\mathrm{op}} = 1$, the bound becomes

$$\dim(\mathcal{T}) \geq (1-\varepsilon)|\mathcal{H}|.$$

Thus to achieve small approximation error (say, $\varepsilon < 0.1$), the subspace must have dimension nearly equal to the size of the function class.

When the functions h_i are not orthogonal, the bound weakens, and is controlled by the spectral norm of the Gram matrix. A useful upper bound is given by

$$\begin{aligned} \frac{1}{k}\|\boldsymbol{G}\|_{\mathrm{op}} \leq \frac{1}{k}\|\boldsymbol{G}\|_{1,\infty} &= \max_{h\in\mathcal{H}} \frac{1}{k}\sum_{h'\in\mathcal{H}} |\langle h, h'\rangle_{\mathcal{R}}| \\ &= \sup_{h\in\mathcal{H}} \mathbb{E}_{h'\sim\mathrm{Unif}(\mathcal{H})}[|\langle h, h'\rangle_{\mathcal{R}}|]. \end{aligned} \tag{2.3.2}$$

This provides a natural control in terms of pairwise correlations. If all functions are nearly orthogonal, then the Gram matrix remains well-conditioned, and the lower bound remains large. In some cases, it may be beneficial to select a subset of $\mathcal{H}$ with low pairwise correlations to tighten the bound.

We emphasize here that this dimension lower bound is extremely general. It does not rely on the structure of any particular learning problem, statistical assumption, or specific algorithm. It simply applies to any predictor constrained to lie in a fixed, finite-dimensional subspace of a Hilbert space. In particular, it applies well beyond supervised learning—for example, to settings involving non-adaptive membership queries [Hsu21].

Lower bound for kernel methods. Let us now explain how to derive a lower bound for kernel methods from Theorem 2.3.1. Consider an input space $\mathcal{X}$ with distribution $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$, and n covariate vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$. Let $\mathcal{R} := L^2(\mathbb{P}_{\mathcal{X}})$. Consider an RKHS $\mathfrak{H} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$ with reproducing kernel K, and denote $p = \dim(\mathfrak{H}) \in \mathbb{N} \cup \{\infty\}$ its dimension[4]. By the representer theorem, any kernel method predictor $\hat{f}$ when the n data points have input

[4]E.g., $p \leq d$ in Example 2.1.3, $p \leq k+1$ in Example 2.1.4, and typically $p = \infty$ in Example 2.1.5.

vectors $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ lie in the linear subspace

$$\hat{f} \in \operatorname{span}\{K(\boldsymbol{x}_i, \cdot) : i \in [n]\} =: \mathcal{T}.$$

Conditional on the covariate vectors $\boldsymbol{x}_i$, this is a fixed linear subspace with

$$\dim(\mathcal{T}) \leq \min(n, p).$$

Consider a set of target functions $\mathcal{H} = \{h_1, \ldots, h_k\} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$. Let $\hat{f}^{(1)}, \ldots, \hat{f}^{(k)}$ be the predictors obtained by the kernel method when the labels come from each of these target functions. Then the excess test error (1.1.5) is given by

$$\mathcal{R}_{\text{exc}}(\hat{f}^{(i)}; h_i) = \|\hat{f}^{(i)} - h_i\|_{L^2}^2 \geq \inf_{f \in \mathcal{T}} \|f - h_i\|_{L^2}^2.$$

Thus, Theorem 2.3.1 directly implies the following result:

Corollary 2.3.2. *For any function space $L^2(\mathbb{P}_{\mathcal{X}})$, any set of unit-norm target functions $\mathcal{H} = \{h_1, \ldots, h_k\} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$, and any kernel method (2.2.1), if the average excess test error with squared loss over $\mathcal{H}$ is less than ε, then we must have*

$$\min(n, p) \geq \frac{k}{\|\boldsymbol{G}\|_{\text{op}}}(1 - \varepsilon), \tag{2.3.3}$$

where $p = \dim(\mathfrak{H})$ is the dimension of the RKHS and $\boldsymbol{G} = (G_{ij})_{ij \in [k]} = (\langle h_i, h_j \rangle_{\mathcal{R}})_{ij \in [k]}$ is the Gram matrix of $\mathcal{H}$.

Note that this result does not depend on any assumption over the $\boldsymbol{x}_i$'s other than they are chosen independently of the label: it holds for any 'non-adaptive' set of covariate vectors—that is, chosen independently of the labels—and not only for i.i.d. samples.

We further present a tightening of this bound from [AAM22] that will be useful in our examples. Some projections of the target functions are harder to fit for kernel methods [GMMM21, MMM22]. Consider some linear subspace $\Omega \subseteq L^2(\mathbb{P}_{\mathcal{X}})$ and the orthogonal decomposition $L^2(\mathbb{P}_{\mathcal{X}}) = \Omega \oplus \Omega^{\perp}$. Denote P_{Ω} and $\mathsf{P}_{\Omega^{\perp}} = \mathbf{I} - \mathsf{P}_{\Omega}$ the orthogonal projections onto Ω and $\Omega^{\perp}$ respectively. We can distinguish the error incurred on each of these two orthogonal subspaces and lower bound the excess test error as

$$\mathcal{R}_{\text{exc}}(\hat{f}^{(i)}; h_i) \geq \|\mathsf{P}_{\Omega}(\hat{f}^{(i)} - h_i)\|_{L^2}^2 \geq \inf_{f \in \mathcal{T}} \|\mathsf{P}_{\Omega}(f - h_i)\|_{L^2}^2.$$

Thus, in Theorem (2.3.1), we can choose $\mathcal{R}$ to be the subspace Ω instead of $L^2(\mathcal{X}, \mathbb{P}_{\boldsymbol{x}})$ and $\mathcal{H} = \{\mathsf{P}_{\Omega} h_1, \ldots, \mathsf{P}_{\Omega} h_k\}$. If we assume $\|\mathsf{P}_{\Omega} h_i\|_{L^2}^2 = \alpha$ for all

$i \in [k]$, then if the average excess test error with squared loss over $\mathcal{H}$ is less than ε, then we must have

$$\min(n, p) \geq \frac{k}{\|\boldsymbol{G}_\Omega\|_{\text{op}}}(\alpha - \varepsilon), \tag{2.3.4}$$

where $\boldsymbol{G}_\Omega = (\langle h_i, \mathsf{P}_\Omega h_j \rangle_{\mathcal{R}})_{ij \in [k]}$.

2.4 Examples

We now return to the two hypothesis classes introduced in Section 1.1.1 in Chapter 1—namely, *Gaussian single-index models* (Equation (1.1.10)) and *sparse functions on the hypercube* (Equation (1.1.12)). We apply the dimension lower bound in Corollary 2.3.2 to lower bound the performance of kernel methods when learning these hypothesis classes.

2.4.1 Gaussian single-index models

Function space on Gaussian data. Recall that we denote $\gamma_d := \mathcal{N}(0, \mathbf{I}_d)$. We start by recalling the definition of the orthogonal basis of multivariate Hermite polynomials in $L^2(\gamma_d)$. Let $\{\mathrm{He}_k\}_{k \geq 0}$ denote the orthogonal basis of (univariate) Hermite polynomials in $L^2(\gamma_1)$, that is, He_k is a degree-k polynomial such that

$$\mathbb{E}_{G \sim \gamma_1}[\mathrm{He}_k(G)\mathrm{He}_s(G)] = k! \delta_{k=s}. \tag{2.4.1}$$

For every $\boldsymbol{k} = (k_1, \ldots, k_d) \in \mathbb{Z}^d_{\geq 0}$, denote $\mathrm{He}_{\boldsymbol{k}}$ the multivariate Hermite polynomial defined as

$$\mathrm{He}_{\boldsymbol{k}}(\boldsymbol{x}) = \prod_{i=1}^{d} \mathrm{He}_{k_i}(x_i). \tag{2.4.2}$$

The set of all multivariate Hermite polynomial $\{\mathrm{He}_{\boldsymbol{k}}\}_{\boldsymbol{k} \in \mathbb{Z}^d_{\geq 0}}$ forms an orthogonal basis of $L^2(\gamma_d)$, with (denoting $(\boldsymbol{k})! := k_1! k_2! \ldots k_d!$)

$$\mathbb{E}_{\boldsymbol{x} \sim \gamma_d}[\mathrm{He}_{\boldsymbol{k}}(\boldsymbol{x})\mathrm{He}_{\boldsymbol{k}'}(\boldsymbol{x})] = (\boldsymbol{k})! \cdot \delta_{\boldsymbol{k}=\boldsymbol{k}'}.$$

Define

$$\Omega_{d,k} = \operatorname{span}\{\mathrm{He}_{\boldsymbol{k}} : \|\boldsymbol{k}\|_1 = k\},$$

the linear subspace of degree-k Hermite polynomials. Denote its dimension

$$B_{d,k} := \dim(\Omega_{d,k}) = \binom{d+k-1}{k}.$$

Thus, the function space $L^2(\gamma_d)$ admits the following orthogonal decomposition

$$L^2(\gamma_d) = \bigoplus_{k=0}^{\infty} \Omega_{d,k}. \tag{2.4.3}$$

Let P_k denote the orthogonal projection onto $\Omega_{d,k}$ in $L^2(\gamma_d)$. We will further introduce $\mathsf{P}_{\le k}$ the projection onto the subspace of polynomials of degree at most k (denoted $\Omega_{d,\le k} = \bigoplus_{s=0}^{k} \Omega_{d,s}$) and $\mathsf{P}_{\ge k} = \mathbf{I} - \mathsf{P}_{\le k-1}$ (the projection onto $\Omega_{d,\ge k} = \bigoplus_{s\ge k} \Omega_{d,s}$), that is the projection onto

$$\Omega_{d,\le k} = \bigoplus_{s=0}^{k} \Omega_{d,s}, \qquad \Omega_{d,\ge k} = \bigoplus_{s=k}^{\infty} \Omega_{d,s},$$

respectively.

We list below some basic properties of (univariate) Hermite polynomials (Prove them!):

Proposition 2.4.1 (Properties of Hermite polynomials)**.** ;

(i) Hermite polynomials satisfy the following three-term recurrence relation: $\mathrm{He}_0(x) = 1$, $\mathrm{He}_1(x) = x$, *and*

$$|\mathrm{He}_{k+1}(x) = x\mathrm{He}_k(x) - k\mathrm{He}_{k-1}(x). \tag{2.4.4}$$

(ii) We have

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{He}_k(x) = k\mathrm{He}_{k-1}(x). \tag{2.4.5}$$

(iii) Let $g : \mathbb{R} \to \mathbb{R}$ be a k-times differentiable function. Then

$$\mathbb{E}_{G\sim\gamma_1}[\mathrm{He}_k(G)g(G)] = \mathbb{E}_{G\sim\gamma_1}[g^{(k)}(G)]. \tag{2.4.6}$$

Using Proposition 2.4.1.(iii), we deduce the following useful identity (Prove it!):

Lemma 2.4.2. *Let $k, k' \in \mathbb{N}$ and $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{S}^{d-1}$. Then*

$$\mathbb{E}_{\boldsymbol{x}\sim\gamma_d}[\mathrm{He}_k(\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\mathrm{He}_{k'}(\langle \boldsymbol{w}', \boldsymbol{x}\rangle)] = k!\langle \boldsymbol{w}, \boldsymbol{w}'\rangle^k \delta_{k=k'}. \tag{2.4.7}$$

Finally, note that we can expand the regression function $h(\boldsymbol{x}) = f_*(\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ into (univariate) Hermite polynomial:

$$\begin{aligned} f_*(\langle \boldsymbol{w}, \boldsymbol{x}\rangle) &= \sum_{k=0}^{\infty} \frac{\mu_k}{k!}\mathrm{He}_k(\langle \boldsymbol{w}, \boldsymbol{x}\rangle), \\ \mu_k &:= \mathbb{E}_{G\sim\gamma_1}[f_*(G)\mathrm{He}_k(G)]. \end{aligned} \tag{2.4.8}$$

In particular, $\mathsf{P}_k h(\boldsymbol{x}) = \frac{\mu_k}{k!}\mathrm{He}_k(\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$, and

$$\|\mathsf{P}_{\geq k} h\|_{L^2}^2 = \|\mathsf{P}_{\geq k} f_*\|_{L^2}^2 = \sum_{\ell=k}^{\infty} \frac{\mu_\ell^2}{\ell!}. \tag{2.4.9}$$

A first simple lower bound. Let us apply Corollary 2.3.2 to derive a lower bound for learning the class $\mathcal{H}_{\mathsf{SI}}^{(d)}$ of Gaussian single-index models (1.1.10) with kernel methods. For any two functions[5] $h, h' \in \mathcal{H}_{\mathsf{SI}}^{(d)}$ with support vectors $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{S}^{d-1}$, we have for any $k \in \mathbb{N}$:

$$\langle h, \mathsf{P}_{\geq k} h'\rangle_{L^2} = \mathbb{E}[\mathsf{P}_{\geq k} f_*(\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\mathsf{P}_{\geq k} f_*(\langle \boldsymbol{w}', \boldsymbol{x}\rangle)] = \sum_{\ell=k}^{\infty} \frac{\mu_\ell^2}{\ell!}\langle \boldsymbol{w}, \boldsymbol{w}'\rangle^\ell,$$

where we used the expansion (2.4.8) and Lemma 2.4.2. Thus, (recalling Equation (2.4.9))

$$|\langle h, \mathsf{P}_{\geq k} h'\rangle_{L^2}| \leq |\langle \boldsymbol{w}, \boldsymbol{w}'\rangle|^k \|\mathsf{P}_{\geq k} f_*\|_{L^2}^2.$$

We can now apply the correlation upper bound (2.3.2) on $\|\boldsymbol{G}\|_{\mathrm{op}}$:

$$\begin{aligned} \sup_{h\in\mathcal{H}} \mathbb{E}_{h'\sim\mathrm{Unif}(\mathcal{H})}[|\langle h, \mathsf{P}_{\geq k} h'\rangle_{L^2}|] &\leq \|\mathsf{P}_{\geq k} f_*\|_{L^2}^2 \mathbb{E}_{\boldsymbol{w}'\sim\mathrm{Unif}(\mathbb{S}^{d-1})}\left[|\langle \boldsymbol{w}, \boldsymbol{w}'\rangle|^k\right] \\ &\leq \frac{C_k}{d^{k/2}}\|\mathsf{P}_{\geq k} f_*\|_{L^2}^2. \end{aligned} \tag{2.4.10}$$

Then the lower bound (2.3.4) immediately implies the following: for every $k \in \mathbb{N}$, the average test error will be at least $(1-\eta)\|\mathsf{P}_{\geq k} f_*\|_{L^2}^2$, unless the number of samples is

$$n \geq c_k d^{k/2}\eta.$$

[5] Recall that, with a slight of abuse of notations, we identify distributions in $\mathcal{H}_{\mathsf{SI}}^{(d)}$ with their target functions $\mathbb{E}[y|\boldsymbol{x}] = f_*(\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$, $\boldsymbol{w} \in \mathbb{S}^{d-1}$.

Note that the test error if we fit exactly the projection of f_* onto degree-$(k-1)$ polynomials is $\|\mathsf{P}_{\geq k} f_*\|_{L^2}^2$. Thus the above bound states that kernel methods will not do better than fitting a degree-$(k-1)$ approximation to the target function unless $n = \Omega_d(d^{k/2})$.

Tight lower bound. In fact, we can tighten this lower bound to $n = \Omega_d(d^k)$. The argument is involved and we defer the proof of this lower bound to Appendix A.2. The reason that the correlation (2.4.10) scales as $d^{-k/2}$ instead of d^{-k} is because $\mathsf{P}_{\geq k} f_*(\langle \boldsymbol{w}, \boldsymbol{x} \rangle)$ has vanishing—but non-zero—components on subspaces of spherical harmonics of degree $\ell < k$. Projecting on the subspace orthogonal to these components allows to bring down the correlation to d^{-k}. While these lower-degree components do not play a role in kernel methods, they will have important algorithmic consequences when considering more powerful methods (e.g., see Section 3.4.1 and [JKMS25]).

We deduce the following lower bound on kernel methods when learning Gaussian single-index models (see Proposition A.2.1 in Appendix A.2): [TBD]

Proposition 2.4.3. *For any kernel method trained over n samples, let $\hat{f}_{\boldsymbol{w}}$ be the solution when target function has support $\boldsymbol{w}$. Then*

$$\begin{aligned} \mathbb{E}_{\boldsymbol{w}\sim\bar{\gamma}_d}\left[\mathcal{R}_{exc}(\hat{f}_{\boldsymbol{w}}; \mathcal{D}_{\boldsymbol{w}})\right] &= \mathbb{E}_{\boldsymbol{w}\sim\bar{\gamma}_d}\left[\|f_*(\langle \boldsymbol{w}, \cdot\rangle) - \hat{f}_{\boldsymbol{w}}\|_{L^2}^2\right] \\ &\geq \left(1 - o_d(1) - c_k \frac{d^k}{n}\right) \|\mathsf{P}_{\geq k} f_*\|_{L^2}^2. \end{aligned} \tag{2.4.11}$$

Matching upper bound. [TBD] This lower bound is tight: matching with kernel. Simply consider featurization map (see remark on the eigendecomposition of the kernel)

$$\Phi(\boldsymbol{x}) = \left(\frac{1}{(\boldsymbol{k})! B_{d,k}} \mathrm{He}_{\boldsymbol{k}}(\boldsymbol{x})\right)_{\boldsymbol{k}\in\mathbb{Z}_{\geq 0}^d}.$$

Then, this will exactly (see [MMM22, MS24]). Spherical data case was done in [GMMM21, Mis22]. [**Write a theorem**] Description of staircase decay.

2.4.2 Sparse functions on the hypercube

[TBD] For convenience, we will denote $\nu_d := \mathrm{Unif}(\{\pm 1\}^d)$.

Definition 2.4.4. *Fourier functions*

$$\chi_S(\boldsymbol{x}) = \prod_{i \in S} x_i$$

$V_{d,k}$ *and* P_k.

Here simple inner-product kernel will match lower bound [Mis22].

2.4.3 Summary

[TBD] this only depends on the degree of the polynomial and nothing else. In fact, with this sample complexity, we can learn any polynomials of that degree.

No adaptivity: does not exploit the fact that low dimensional structure (single index or depend on sparse support).

In contrast we could do the following: first learn the support either vector $\boldsymbol{w}$ or support S, then problem becomes effectively low-dimensional.

Chapter 3

Noisy gradient descent

In this chapter, we study a simplified model of gradient-based optimization algorithms, which we refer to as *noisy gradient descent*. In this model, the algorithm has access to population gradients—i.e., gradients of the population loss—perturbed by additive Gaussian noise. Although it is not a statistical algorithm—operating on finite samples—, noisy gradient descent captures some key aspects of stochastic gradient descent (SGD), the backbone of modern machine learning. Importantly, it falls within the broader class of *statistical query* (SQ) algorithms, which we will explore in the next chapter.

Our goal here is to establish a lower bound on the number of gradient steps required by noisy gradient descent to achieve small test error. To this end, we follow a 'junk flow' argument developed by Abbe and Sandon [AS20], which compares the algorithm's output on real data to its output on corrupted data—where the labels have been (partially) replaced with random noise. This argument illustrates a general strategy for proving lower bounds: if an algorithm cannot distinguish true data from corrupted data, it cannot learn beyond a certain precision.

The presentation and proofs in this chapter will follow [ABA22].

3.1 Noisy gradient descent

We consider a general parametrized model $f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \to \mathbb{R}$, with parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, and only assume that $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})$ exists for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{\theta} \in \mathbb{R}^p$ (or almost everywhere). For example, $f(\cdot;\boldsymbol{\theta})$ may represent a neural network with weights $\boldsymbol{\theta}$, such as the two-layer neural network (1.1.6).

We focus on the squared loss $\ell(y,\hat{y}) = (y-\hat{y})^2$ and assume data points

$(y, \boldsymbol{x}) \sim \mathcal{D} \in \mathcal{P}(\mathbb{R} \times \mathcal{X})$. The population (or test) loss is given by

$$\mathcal{R}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{(y,\boldsymbol{x})\sim\mathcal{D}}\big[\big(y - f(\boldsymbol{x}; \boldsymbol{\theta})\big)^2\big]. \tag{3.1.1}$$

Let $h(\boldsymbol{x}) := \mathbb{E}_{\mathcal{D}}[y|\boldsymbol{x}]$ denote the regression function. Then the excess risk of a predictor $f(\cdot; \boldsymbol{\theta})$ is

$$\mathcal{R}_h(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{\mathcal{X}}}\big[\big(h(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta})\big)^2\big] = \|h - f(\cdot; \boldsymbol{\theta})\|_{L^2}^2.$$

Stochastic Gradient Descent (SGD). A common approach to minimize the test error is to run *online SGD* on the population loss (3.1.1): starting from an initial (random) parameter $\boldsymbol{\theta}^0 \sim \rho_0 \in \mathcal{P}(\mathbb{R}^p)$, we update iteratively the parameter by sampling a new data point $(y_k, \boldsymbol{x}_k) \sim \mathcal{D}$ at each step (so that $\{(y_k, \boldsymbol{x}_k)\}_{k\geq 0}$ are iid):

$$\begin{aligned} \boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k - \eta_k \nabla_{\boldsymbol{\theta}} \ell(y_k, f(\boldsymbol{x}_k; \boldsymbol{\theta}^k)) \\ &= \boldsymbol{\theta}^k + \eta_k (y_k - f(\boldsymbol{x}_k; \boldsymbol{\theta}^k)) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_k; \boldsymbol{\theta}^k), \end{aligned} \tag{3.1.2}$$

where $\eta_k > 0$ is the step size at step k. This is often referred to as *one-pass SGD*, where each training data point is used only once during training. This is a reasonable assumption for large datasets, where each sample is only seen at most a few times.

Note that we can rewrite the update rule (3.1.2) as

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta_k \underbrace{\nabla_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}^k; \mathbb{P})}_{\text{population gradient}} - \eta_k \underbrace{\big\{\nabla_{\boldsymbol{\theta}} \ell(y_k, f(\boldsymbol{x}_k; \boldsymbol{\theta}^k)) - \nabla_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}^k; \mathbb{P})\big\}}_{\text{mean zero independent variable}}.$$

The first term is deterministic conditional on $\boldsymbol{\theta}^k$ while the second term is a mean-zero noise vector that introduces randomness into the trajectory. Directly analyzing this trajectory is difficult. Instead, we consider a stylized model where this noise term is replaced by independent Gaussian noise vectors $\boldsymbol{\xi}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p)$. This defines the noisy gradient descent model.

Gradient clipping. We will further consider clipping the gradient, which is often used in practice to avoid instability from exploding gradients. That is, we replace $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_k; \boldsymbol{\theta}^k)$ in the gradient update by its projection onto an ℓ_2 ball of radius R:

$$\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_k; \boldsymbol{\theta}^k)],$$

where $B(0,R) = \{\boldsymbol{z} : \|\boldsymbol{z}\|_2 \le R\} \subset \mathbb{R}^p$. This will ensure that the magnitude of the update is controlled relative to the noise level τ (see Chapter 4). Define the clipped population gradient:

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{\theta}) &= -\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\Big[\big(y - f(\boldsymbol{x};\boldsymbol{\theta})\big)\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})]\Big] \\ &= -\mathbb{E}_{\boldsymbol{x}}\Big[\big(h(\boldsymbol{x}) - f(\boldsymbol{x};\boldsymbol{\theta})\big)\Pi_{B(0,R)}\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})\Big]. \end{aligned} \tag{3.1.3}$$

Note that the population gradient (and therefore the parameters $\boldsymbol{\theta}^k$) only depend on the data distribution $\mathcal{D}$ through the target function h. We will denote $\boldsymbol{g}_h(\boldsymbol{\theta}) := \boldsymbol{g}_h(\boldsymbol{\theta})$ and $\boldsymbol{\theta}_k^k := \boldsymbol{\theta}^k$ to emphasize this dependency.

Noisy Gradient Descent. Putting everything together, *noisy gradient descent* with respect to the target function h is defined as follows:

- Initialize $\boldsymbol{\theta}^0 \sim \rho_0$.
- For each step k, update:

$$\boldsymbol{\theta}_h^{k+1} = \boldsymbol{\theta}_h^k - \eta_k \boldsymbol{g}_h(\boldsymbol{\theta}^k) + \eta_k \boldsymbol{\xi}^k, \tag{3.1.4}$$

  where $\boldsymbol{\xi}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p)$ are independent Gaussian noise vectors, and $\boldsymbol{g}_h(\boldsymbol{\theta}^k)$ are the clipped population gradient (3.1.3).

By using the randomness in $\boldsymbol{\xi}^k$, this model will allow for simple lower bounds.

3.2 Lower bound via junk flow

In this section, we establish a lower bound on the number of steps required for noisy gradient descent to achieve low test error when learning a class of target functions $\mathcal{H} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$. Consider a prior $\mu_{\mathcal{H}} \in \mathcal{P}(\mathcal{H})$ over these target functions. Our lower bound will hold with high probability over the random choice of target function $h \sim \mu_{\mathcal{H}}$ and the randomness in the optimization trajectory.

The difficulty of learning the function class $\mathcal{H}$ using noisy gradient descent is governed by its *correlation alignment complexity* (see [AS20, ACHM22] for earlier notions). This notion captures how well the target functions align with any fixed direction in the function space.

Definition 3.2.1 (Correlation alignment complexity)**.** *Let $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$ be the input distribution, and $\mu_{\mathcal{H}} \in \mathcal{P}(\mathcal{H})$ be the prior distribution over target functions $h \in \mathcal{H} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$. Let $g \in L^2(\mathbb{P}_{\mathcal{X}})$ be a reference function. The* correlation alignment complexity *of $(\mu_{\mathcal{H}}, g)$ is defined as*

$$\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g) = \left[\sup_{\|\phi\|_{L^2(\mathbb{P}_{\mathcal{X}})} \le 1} \mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\langle \phi, h - g\rangle^2_{L^2(\mathbb{P}_{\mathcal{X}})}\right]\right]^{-1}. \tag{3.2.1}$$

When $g \equiv 0$, we will write $\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}})$ for simplicity.

Intuitively, small alignment means that the target functions in $\mathcal{H}$ (centered by a reference function) are nearly orthogonal to any fixed direction. We will come back to this notion in Chapter 4 when we discuss Statistical Query algorithms, and justify the name 'correlation alignment'.

We now state the main result of this chapter: a lower bound on the performance of noisy gradient descent in terms of this alignment complexity.

Theorem 3.2.2 ([ABA22, Theorem A.4])**.** *Fix an input distribution $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$ and a parametrized model $f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \to \mathbb{R}$, with parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. Let $\boldsymbol{\theta}_h^k$ be the (random) parameter after k steps of noisy gradient descent (3.1.4) with target function $h \in L^2(\mathbb{P}_{\mathcal{X}})$, initial distribution $\rho_0 \in \mathcal{P}(\mathbb{R}^p)$, step sizes $\{\eta_s\}_{s\ge 0}$, clipping radius $R > 0$, and noise level τ.*

Then for any prior $\mu_{\mathcal{H}} \in \mathcal{P}(\mathcal{H})$ over target functions, any offset $g \in L^2(\mathbb{P}_{\mathcal{X}})$, and any $\varepsilon > 0$, we have

$$\mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_h^k}\left[\mathcal{R}_h(\boldsymbol{\theta}_h^k) \le \|h - g\|_{L^2}^2 - \varepsilon\right] \le \frac{R}{2\tau}\sqrt{\frac{k}{\mathsf{A}}} + \frac{1}{\varepsilon\cdot\mathsf{A}}, \tag{3.2.2}$$

where $\mathsf{A} := \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)$ is the correlation alignment complexity from Definition 3.2.1.

This result shows that unless the number of steps k is sufficiently large, noisy gradient descent cannot significantly improve upon the trivial predictor that always outputs g. In particular, if $\mathcal{A}$ is large, then the test error remains close to the trivial test error $\|h - g\|_{L^2}^2$ with high probability over $h \sim \mu_{\mathcal{H}}$ and the randomness in the trajectories, unless

$$k \gtrsim \left(\frac{\tau}{R}\right)^2 \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g). \tag{3.2.3}$$

Here τ/R can be interpreted as the gradient precision.

Thus if the alignment complexity A is large, learning requires either a large number of steps k or a small gradient precision τ/R. We will see later that this style of lower bound is typical when considering SQ algorithms (Chapter 4), where τ corresponds to the precision of the query expectations.

Proof via 'junk flow'. We now outline the proof of Theorem 3.2.2 and defer the technical details to Section 3.3.

The central idea is to compare two noisy gradient descent trajectories: one trained on the true target function h, and one trained on a fixed 'junk' target function g. If the two trajectories remain close in distribution, then the algorithm cannot distinguish whether it is learning h or g. Since the junk flow has no information about the true target function h, its test error remains high. Hence, unless the trajectories diverge, the algorithm cannot achieve low test error.

Specifically, we consider:

- *GD trajectory on true target h:* Let $\boldsymbol{\theta}_h^0, \dots, \boldsymbol{\theta}_h^k$ be the trajectory of noisy gradient descent (3.1.4) with target h. That is, we initialize with $\boldsymbol{\theta}_h^0 \sim \rho_0$ and updated the parameters via

$$\boldsymbol{\theta}_h^{k+1} = \boldsymbol{\theta}_h^k - \eta_k \boldsymbol{g}_h(\boldsymbol{\theta}_h^k) + \eta_k \boldsymbol{\xi}^k, \quad \text{where } \boldsymbol{\xi}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p).$$

- *GD trajectory on junk target g:* Let $\boldsymbol{\theta}_g^0, \dots, \boldsymbol{\theta}_g^k$ be the trajectory of noisy gradient descent (3.1.4) with junk target $g \in L^2(\mathbb{P}_{\mathcal{X}})$, initialized with $\boldsymbol{\theta}_g^0 \sim \rho_0$ and updated via

$$\boldsymbol{\theta}_g^{k+1} = \boldsymbol{\theta}_g^k - \eta_k \boldsymbol{g}_g(\boldsymbol{\theta}_g^k) + \eta_k \tilde{\boldsymbol{\xi}}^k, \quad \text{where } \tilde{\boldsymbol{\xi}}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p).$$

Theorem 3.2.2 is then a consequence of the following two lemmas.

The first lemma shows that the distributions of $\boldsymbol{\theta}_h^k$ and $\boldsymbol{\theta}_g^k$ stays close to each other (in total variation distance, see Definition 3.3.1) with high probability over $h \sim \mu_{\mathcal{H}}$.

Lemma 3.2.3 (Proximity of the two trajectories)**.** *Under the setting of Theorem 3.2.2, we have*

$$\mathbb{E}_{h \sim \mu_{\mathcal{H}}} \left[\mathsf{TV}(\boldsymbol{\theta}_g^k, \boldsymbol{\theta}_h^k) \right] \le \frac{R}{2\tau} \sqrt{\frac{k}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)}}, \tag{3.2.4}$$

Equivalently (see Equation (3.3.3)), there exists a joint coupling of $\boldsymbol{\theta}_h^k$ and $\boldsymbol{\theta}_g^k$ such that

$$\mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_h^k,\boldsymbol{\theta}_g^k}\left[\boldsymbol{\theta}_h^k \neq \boldsymbol{\theta}_g^k\right] \le \frac{R}{2\tau}\sqrt{\frac{k}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)}}.$$

In words, if $k \ll (\tau/R)^2\mathsf{Align}$, then with high-probability over $h \sim \mu_{\mathcal{H}}$ and noise in the algorithm, the outputs $\boldsymbol{\theta}_h^k$ and $\boldsymbol{\theta}_g^k$ of noisy gradient descent are indistinguishable.

The second lemma lower bound the test error on the junk flow trajectory.

Lemma 3.2.4 (Test error of the junk flow)**.** *Under the setting of Theorem 3.2.2, we have for any $\varepsilon > 0$,*

$$\mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_g^k}\left[\mathcal{R}_h(\boldsymbol{\theta}_g^k) \le \|f-g\|_{L^2(\mu_{\mathcal{X}})}^2 - \varepsilon\right] \le \frac{1}{\varepsilon\cdot\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)}.$$

The proofs of Lemma 3.2.3 and Lemma 3.2.4 can be found in Section 3.3. Note that these proofs use very little about the specific form of the update equation (3.1.4)—only the additive Gaussian noise structure—, and we will generalize Lemma 3.2.3 to a broader class of algorithms in the next chapter.

We are now ready to prove Theorem 3.2.2:

Proof of Theorem 3.2.2. For any coupling between $\boldsymbol{\theta}_h^k$ and $\boldsymbol{\theta}_g^k$, and for any $t > 0$, we have

$$\mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_h^k}\left[\mathcal{R}_h(\boldsymbol{\theta}_h^k) \le t\right] \le \mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_h^k,\boldsymbol{\theta}_g^k}\left[\boldsymbol{\theta}_h^k \neq \boldsymbol{\theta}_g^k\right] + \mathbb{P}_{h\sim\mu_{\mathcal{H}},\boldsymbol{\theta}_g^k}\left[\mathcal{R}_h(\boldsymbol{\theta}_g^k) \le t\right].$$

Taking the optimal coupling, the first term is bounded in Lemma 3.2.3, while the second term is bounded in Lemma 3.2.4. □

3.3 Proofs of auxiliary lemmas

We first recall some definitions and standard properties of the total variation (TV) distance and Kullback-Leibler (KL) divergence.

Definition 3.3.1 (Divergences between probability measures)**.** *Let P and Q be two probability measures on a measurable space $(\Omega, \mathcal{F})$.*

(1) (Total Variation distance.) *The* total variation distance *between P and Q is defined as*

$$\mathsf{TV}(P,Q) = \sup_{A\in\mathcal{F}} |P(A) - Q(A)|. \tag{3.3.1}$$

(2) (Kullback–Leibler divergence.) *Further assume that*[1] $P \ll Q$. *The* Kullback–Leibler divergence *between P and Q is defined as*

$$\mathsf{KL}(P||Q) = \int_\Omega \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}(\omega)\right) \mathrm{d}P(\omega), \tag{3.3.2}$$

where $\frac{\mathrm{d}P}{\mathrm{d}Q}$ is the Radon-Nikodym derivative of P relative to Q.

With a slight abuse of notations, we will write $\mathsf{TV}(X,Y)$ and $\mathsf{KL}(X||Y)$ instead of $\mathsf{TV}(P,Q)$ and $\mathsf{KL}(P||Q)$ when $X \sim P$ and $Y \sim Q$, and sometimes $\mathsf{TV}(P(X),Q(Y))$. In particular, we have the following equivalent characterization of the TV distance

$$\mathsf{TV}(X,Y) = \inf_{\text{coupling}(X,Y)} \mathbb{P}(X \neq Y), \tag{3.3.3}$$

where coupling(X,Y) is the set of joint distribution over (X,Y) with marginals P and Q.

We next present some basic properties of TV and KL which will be useful when proving the auxiliary lemmas.

Proposition 3.3.2 (Properties of TV and KL)**.**

(a) (Pinsker's inequality.) *We have*

$$\mathsf{TV}(P,Q) \leq \sqrt{\frac{1}{2}\mathsf{KL}(P||Q)}. \tag{3.3.4}$$

(b) (Data processing inequality.) *Let P and Q be two distributions for a pair of variables (X,Y). We have*

$$\mathsf{TV}(P(X),Q(X)) \leq \mathsf{TV}(P(X,Y),Q(X,Y)). \tag{3.3.5}$$

[1] P is dominated by Q, denoted $P \ll Q$, if and only if for all $A \in \mathcal{F}$, $Q(A) = 0$ implies $P(A) = 0$.

(c) (Chain rule for KL.) *We have*

$$\begin{aligned} &\mathsf{KL}(P(X,Y)||Q(X,Y)) \\ &= \mathsf{KL}(P(X)||Q(X)) + \mathbb{E}_{x\sim P}\left[\mathsf{KL}(P(Y|X=x)||Q(Y|X=x))\right]. \end{aligned} \tag{3.3.6}$$

The above properties and their proofs can be found in any information theory textbook (e.g., [Duc24, Chapter 2]).

Proof of Lemma 3.2.3. Write $\boldsymbol{\theta}_h^{\le k} = (\boldsymbol{\theta}_h^0, \ldots, \boldsymbol{\theta}_h^k)$ and $\boldsymbol{\theta}_g^{\le k} = (\boldsymbol{\theta}_g^0, \ldots, \boldsymbol{\theta}_g^k)$ the two trajectories, and denote $\mathcal{L}(\boldsymbol{\theta}_h^{\le k})$ and $\mathcal{L}(\boldsymbol{\theta}_g^{\le k})$ their laws. We first bound the KL divergence between $\boldsymbol{\theta}_g^{\le k}$ and $\boldsymbol{\theta}_h^{\le k}$:

$$\begin{aligned} &\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{\le k})||\mathcal{L}(\boldsymbol{\theta}_h^{\le k})) \\ &\overset{(a)}{=} \mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^0)||\mathcal{L}(\boldsymbol{\theta}_h^0)) + \sum_{s=0}^{k-1} \mathbb{E}_{\boldsymbol{\theta}_g^{\le s}}\left[\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{s+1}|\boldsymbol{\theta}_g^{\le s})||\mathcal{L}(\boldsymbol{\theta}_h^{s+1}|\boldsymbol{\theta}_h^{\le s} = \boldsymbol{\theta}_g^{\le s}))\right] \\ &\overset{(b)}{=} \sum_{s=0}^{k-1} \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{L}(\boldsymbol{\theta}_g^s)}\left[\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{s+1}|\boldsymbol{\theta}_g^s = \boldsymbol{\theta})||\mathcal{L}(\boldsymbol{\theta}_h^{s+1}|\boldsymbol{\theta}_h^s = \boldsymbol{\theta}))\right], \end{aligned}$$

where we used (a) the KL divergence chain rule (Proposition 3.3.2.(3)), and (b) $\mathcal{L}(\boldsymbol{\theta}_g^0) = \mathcal{L}(\boldsymbol{\theta}_h^0) = \rho_0$ and the Markov property of GD training. By definition of the update rule in noisy GD (3.1.4), we have

$$\begin{aligned} \boldsymbol{\theta}_g^{s+1}\Big|_{\boldsymbol{\theta}_g^s=\boldsymbol{\theta}} &\sim \mathcal{N}(\boldsymbol{\theta} - \eta_k \boldsymbol{g}_g(\boldsymbol{\theta}), \eta_k^2\tau^2\mathbf{I}_p), \\ \boldsymbol{\theta}_h^{s+1}\Big|_{\boldsymbol{\theta}_h^s=\boldsymbol{\theta}} &\sim \mathcal{N}(\boldsymbol{\theta} - \eta_k \boldsymbol{g}_h(\boldsymbol{\theta}), \eta_k^2\tau^2\mathbf{I}_p). \end{aligned}$$

Thus using the formula of KL divergence between two Gaussians[2], we deduce

$$\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{\le k})||\mathcal{L}(\boldsymbol{\theta}_h^{\le k})) = \sum_{s=0}^{k-1} \frac{1}{2\tau^2}\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{L}(\boldsymbol{\theta}_g^s)}\left[\|\boldsymbol{g}_g(\boldsymbol{\theta}) - \boldsymbol{g}_h(\boldsymbol{\theta})\|_2^2\right]. \tag{3.3.7}$$

By definition of the clipped population gradients, we have

$$\boldsymbol{g}_g(\boldsymbol{\theta}) - \boldsymbol{g}_h(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}\left[(h(\boldsymbol{x}) - g(\boldsymbol{x}))\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})]\right],$$

[2] We have $\mathsf{KL}(\mathcal{N}(\boldsymbol{\mu}, \kappa^2\mathbf{I}_p)||\mathcal{N}(\boldsymbol{\mu}', \kappa^2\mathbf{I}_p)) = \frac{1}{2\kappa^2}\|\boldsymbol{\mu} - \boldsymbol{\mu}'\|_2^2$.

so that,

$$\begin{aligned}\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\|\boldsymbol{g}_g(\boldsymbol{\theta})-\boldsymbol{g}_h(\boldsymbol{\theta})\|_2^2\right] &= \sum_{i=1}^{p}\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\langle h-g,\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}}f(\cdot;\boldsymbol{\theta})]_i\rangle_{L^2(\mathcal{X})}^2\right]\\ &\overset{(a)}{\leq} \sum_{i=1}^{p}\frac{\|\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}}f(\cdot;\boldsymbol{\theta})]_i\|_{L^2}^2}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},g)}\\ &\overset{(b)}{\leq} \frac{R^2}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},g)},\end{aligned}$$

where we used (a) the definition of the alignment complexity $\mathcal{C}(\overline{\mu}_{\mathcal{H}},\mathbb{P}_{\boldsymbol{x}})$, and (b) $\|\Pi_{B(0,R)}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta})]\|_2^2\leq R^2$ by definition. Injecting this bound into equation (3.3.7), we obtain the upper bound

$$\begin{aligned}&\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{\leq k})||\mathcal{L}(\boldsymbol{\theta}_h^{\leq k}))\right]\\ &=\sum_{s=0}^{k-1}\frac{1}{2\tau^2}\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{L}(\boldsymbol{\theta}_g^s)}\left[\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\|\boldsymbol{g}_g(\boldsymbol{\theta})-\boldsymbol{g}_h(\boldsymbol{\theta})\|_2^2\right]\right] \qquad (3.3.8)\\ &\leq k\frac{R^2}{2\tau^2}\cdot\frac{1}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},g)}.\end{aligned}$$

Finally, we conclude the proof of this lemma by observing that

$$\begin{aligned}\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\mathsf{TV}(\mathcal{L}(\boldsymbol{\theta}_g^k),\mathcal{L}(\boldsymbol{\theta}_h^k))\right] &\overset{(a)}{\leq} \mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\mathsf{TV}(\mathcal{L}(\boldsymbol{\theta}_g^{\leq k}),\mathcal{L}(\boldsymbol{\theta}_h^{\leq k}))\right]\\ &\overset{(b)}{\leq} \mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\sqrt{\frac{1}{2}\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{\leq k})||\mathcal{L}(\boldsymbol{\theta}_h^{\leq k}))}\right]\\ &\overset{(c)}{\leq} \sqrt{\frac{1}{2}\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_g^{\leq k})||\mathcal{L}(\boldsymbol{\theta}_h^{\leq k}))\right]}\\ &\overset{(d)}{\leq} \frac{R}{2\tau}\sqrt{\frac{k}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},g)}},\end{aligned}$$

where we used (a) the data processing inequality (Proposition 3.3.2.(2)), (b) Pinsker's inequality (Proposition 3.3.2.(1)), (c) Jensen's inequality, and (d) the bound on KL divergence in (3.3.8). □

Proof of Lemma 3.2.4. Define

$$\varphi_h(\boldsymbol{\theta}_g^k):=\langle h-g,f(\cdot;\boldsymbol{\theta}_g^k)-g\rangle_{L^2(\mathcal{X})}.$$

Recall that g is fixed and $\boldsymbol{\theta}_g^k$ is independent of $h \sim \mu_{\mathcal{H}}$. Therefore, by definition of the alignment complexity,

$$\mathbb{E}_{h\sim\mu_{\mathcal{H}}}[\varphi_h(\boldsymbol{\theta}_g^k)^2] \leq \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)^{-1}\|f(\cdot;\boldsymbol{\theta}_g^k) - g\|_{L^2}^2.$$

Denote $\mathcal{E}$ the event $\varphi_h(\boldsymbol{\theta}_g^k)^2 \leq \varepsilon\|f(\cdot;\boldsymbol{\theta}_g^k) - g\|_{L^2}^2$ such that, by Marhov's inequality and the above display

$$\mathbb{P}(\mathcal{E}^c) = \mathbb{P}\left(\varphi_h(\boldsymbol{\theta}_g^k)^2 > \varepsilon\|f(\cdot;\boldsymbol{\theta}_g^k) - g\|_{L^2}^2\right) \leq \frac{1}{\varepsilon \cdot \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)}.$$

Let's consider the excess test error. Under the even $\mathcal{E}$, we have

$$\begin{aligned}
\mathcal{R}_h(\boldsymbol{\theta}_g^k) &= \|h - f(\cdot;\boldsymbol{\theta}_g^k)\|_{L^2}^2 \\
&= \|h - g\|_{L^2}^2 - 2\varphi_h(\boldsymbol{\theta}_g^k) + \|g - f(\cdot;\boldsymbol{\theta}_g^k)\|_{L^2}^2 \\
&= \|h - g\|_{L^2}^2 - 2\sqrt{\varepsilon}\|g - f(\cdot;\boldsymbol{\theta}_g^k)\|_{L^2} + \|g - f(\cdot;\boldsymbol{\theta}_g^k)\|_{L^2}^2 \\
&\geq \|h - g\|_{L^2}^2 - \varepsilon,
\end{aligned}$$

where on the last line, we optimized over $\delta := \|g - f(\cdot;\boldsymbol{\theta}_g^k)\|_{L^2}$. We deduce by contraposition that

$$\mathbb{P}\left[\mathcal{R}_h(\boldsymbol{\theta}_g^k) \leq \|h - g\|_{L^2}^2 - \varepsilon\right] \leq \mathbb{P}(\mathcal{E}^c) \leq \frac{1}{\varepsilon \cdot \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g)},$$

which concludes the proof of the second lemma. □

3.4 Examples

3.4.1 Gaussian single-index models

[TBD]

Definition 3.4.1 (Information exponent)**.**

Proposition 3.4.2. *Computation of the alignment complexity*

Proof. □

Consequence (LB on the test error)
There exists an SGD algorithm that succeeds with this many steps.

3.4.2 Sparse functions on the hypercube

[TBD]

Definition 3.4.3 (Leap complexity)**.**

Proposition 3.4.4. *Computation of the alignment complexity*

Consequence (LB on the test error)

There exists an algorithm that succeeds with this many steps. Evidence that GD on a fully connected neural networks can also succeed with this many steps. (or max with d)

Chapter 4

Statistical Query algorithms

In this chapter, we introduce the *statistical query* (SQ) model, a framework originally proposed by Kearns [Kea98] for designing algorithms that are robust to noise. In the SQ model, learning algorithms only access data through approximate expectations, instead of individual samples directly. Specifically, they are restricted to querying quantities of the form

$$\mathbb{E}_{(y,\boldsymbol{x})\sim\mathcal{D}}[\phi(y,\boldsymbol{x})],$$

for some query function ϕ, and receive an approximate value up to some specified tolerance. A wide variety of algorithms can be implemented in this model, including many gradient-based methods.

The SQ model has become a popular framework in learning theory for two main reasons:

- It enables the design of noise-tolerant algorithms: any algorithm that can be implemented in the SQ model is guaranteed to be robust to small perturbations or statistical noise in the training data.
- It provides a convenient and powerful tool for proving lower bounds in a wide range of settings. These bounds hold for any algorithm relying only on low-precision expectations of the data, under *adversarial* or *worst-case* noise assumption.

The goal of this chapter is to provide a concise introduction to the SQ model. We start with its formal definition and basic properties, then present several techniques for proving lower bounds within this framework. For a more complete survey, we refer the reader to [Rey20] and references therein.

4.1 Basic model and definitions

A helpful way to view the SQ model is as a restricted version of the standard *probably approximately correct* (PAC) model introduced by Valiant [Val84]. In the PAC model, the learner receives n i.i.d. samples $(y_i, \boldsymbol{x}_i) \in \mathcal{Y} \times \mathcal{X}$ drawn from some unknown distribution $\mathbb{P}$, and may use them arbitrarily to output a predictor $\hat{f} : \mathcal{X} \to \mathcal{Y}$.

In contrast, in the SQ model, the algorithm is not given direct access to the samples. Instead, it can only access the distribution through approximate expectations of queries—functions $\phi : \mathcal{Y} \times \mathcal{X} \to [-1, 1]$. Formally, the interaction is mediated by an oracle: SQ algorithms issue a query to an oracle which returns a value in some range.

Definition 4.1.1 ($\mathsf{STAT}(\tau)$ oracle)**.** *Let $\mathcal{D} \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be the data distribution. Given a tolerance parameter $\tau > 0$, the* statistical query oracle $\mathsf{STAT}(\tau)$ *takes as input a query function $\phi : \mathcal{Y} \times \mathcal{X} \to [-1, 1]$ and returns a value in the range:*

$$\Big[\mathbb{E}_{\mathcal{D}}[\phi(y, \boldsymbol{x})] - \tau, \mathbb{E}_{\mathcal{D}}[\phi(y, \boldsymbol{x})] + \tau\Big]. \tag{4.1.1}$$

Importantly, any SQ algorithm can be simulated from i.i.d. samples using empirical averages. By Hoeffding's inequality, if ϕ is bounded in $[-1, 1]$, then

$$\hat{\mathbb{E}}_n[\phi(y, \boldsymbol{x})] = \frac{1}{n}\sum_{i=1}^{n} \phi(y_i, \boldsymbol{x}_i) \in \Big[\mathbb{E}_{\mathcal{D}}[\phi(y, \boldsymbol{x})] - \tau, \mathbb{E}_{\mathcal{D}}[\phi(y, \boldsymbol{x})] + \tau\Big],$$

with probability at least $1 - \delta$, provided that

$$n \geq \frac{\log(2/\delta)}{\tau^2}.$$

Thus, any SQ algorithm that makes q query calls to $\mathsf{STAT}(\tau)$ can be simulated with probability at least $1 - \delta$ using

$$n = O\left(\frac{\log(q/\delta)}{\tau^2}\right)$$

samples, by union bound. This motivates interpreting the tolerance parameter as $\tau \approx 1/\sqrt{n}$.

Many variants of the SQ oracle $\mathsf{STAT}(\tau)$ have been introduced in the literature, including honest SQ [Yan05], multi-sample SQ [BKW03], correlation SQ [BF02], VSTAT [FGR+17], or differentiable learning SQ [JMS24], and we will discuss some of them later in this chapter.

Remark 4.1.2 (Importance of bounded queries)**.** *To ensure that the tolerance τ is meaningful, the scale of the query function ϕ must be controlled. Without such a normalization, one could trivially reduce the error tolerance by scaling up the query function, or simulate multiple queries at once. A common assumption is that queries lie in the unit ball of L^∞, i.e., $\|\phi\|_\infty \leq 1$. More generally, oracles and query normalizations are chosen so that they can be efficiently simulated from samples.*

SQ algorithms. In summary, a SQ algorithm $\mathcal{A}$ with q query calls and tolerance τ (based on oracle $\mathsf{STAT}(\tau)$) proceeds as follows. It takes a data distribution $\mathcal{D} \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ and operates in q rounds:

- At each round $t = 1, \ldots, q$, the algorithm $\mathcal{A}$ issues a query $\phi_t : \mathcal{Y} \times \mathcal{X} \to [-1, 1]$ and receive a response v_t from $\mathsf{STAT}(\tau)$, that is

$$|v_t - \mathbb{E}_{\mathcal{D}}[\phi_t(y, \boldsymbol{x})]| \leq \tau. \tag{4.1.2}$$

  The query ϕ_t is chosen adaptively, that is, it can depend on past responses $v_1, \ldots, v_{t-1}$.

- After q round, the algorithm outputs an answer $\mathcal{A}(\mathcal{D})$ based on all the responses $v_1, \ldots, v_q$. For example, $\mathcal{A}(\mathcal{D})$ is a predictor $\hat{f} : \mathcal{X} \to \mathcal{Y}$.

We saw that any such algorithm can be implemented using $O(\log(q)/\tau^2)$ samples. Thus, if a SQ algorithm can solve a problem with

- At most polynomially many queries;
- Each query has tolerance at most polynomially small;
- We can compute ϕ_t from $v_1, \ldots, v_{t-1}$ and evaluate ϕ_t efficiently;

Then the problem can be solved in polynomial time using i.i.d. samples.

Robustness to label noise. A key feature of the SQ model is that it guarantees robustness to certain types of label noise. For example, consider the binary classification setting $\mathcal{Y} = \{-1, 1\}$, and suppose that labels are corrupted independently with probability $\eta \in [0, 1)$: with probability $1 - \eta$, y is drawn from $\mathcal{D}$, and with probability η, y is drawn uniformly at random. Let $\mathcal{D}_\eta$ denote the corrupted distribution. Then for any query ϕ, we have

$$\mathbb{E}_{\mathcal{D}_\eta}[\phi(y, \boldsymbol{x})] = (1 - \eta)\mathbb{E}_{\mathcal{D}}[\phi(y, \boldsymbol{x})] + \eta \mathbb{E}_{\mathcal{D}}\left[\frac{1}{2}(\phi(1, \boldsymbol{x}) + \phi(-1, \boldsymbol{x}))\right].$$

Thus, we can simulate the query oracle from corrupted data with estimator

$$\frac{1}{n(1-\eta)}\sum_{i=1}^{n}\phi(y_i,\boldsymbol{x}_i)-\frac{\eta}{2}(\phi(1,\boldsymbol{x}_i)+\phi(-1,\boldsymbol{x}_i)),$$

with probability at least $1-\delta$ and sample size

$$n=O\left(\frac{\log(1/\delta)}{(1-\eta)^2\tau^2}\right).$$

Thus, the SQ framework gives us a way to design noise-tolerant algorithm, which was the original motivation by Kearns [Kea98].

SQ lower bounds. One of the most appealing features of the SQ model is that it provides a powerful framework for proving computational lower bounds. In this model, lower bounds are established against a strong notion of success: a SQ algorithm is said to succeed only if it performs well for *every* sequence of valid oracle responses consistent with the allowed tolerance.

Example 4.1.3. *Consider a class of distributions $\mathcal{H}\subseteq\mathcal{P}(\mathcal{Y}\times\mathcal{X})$ and a loss function ℓ. The goal is to design an algorithm $\mathcal{A}$ that, for any distribution $\mathcal{D}\in\mathcal{H}$, outputs a predictor $\mathcal{A}(\mathcal{D}):\mathcal{X}\to\mathcal{Y}$ satisfying*

$$\mathbb{E}_{(y,\boldsymbol{x})\sim\mathcal{D}}\left[\ell(y,\mathcal{A}(\mathcal{D})(\boldsymbol{x}))\right]\leq\varepsilon.$$

An SQ algorithm is said to succeed if it achieves this guarantee for every $\mathcal{D}\in\mathcal{H}$, regardless of the specific (valid) oracle responses it receives. That is, for any sequence of responses $v_1,\ldots,v_q$ satisfying

$$|v_t-\mathbb{E}_{\mathcal{D}}[\phi_t(y,\boldsymbol{x})]|\leq\tau,\qquad\text{for each query }\phi_t,$$

the algorithm must still output a predictor with test error at most ε.

In other words, lower bounds in the SQ model are established against *adversarial* or *worst-case* noise in the expectations, rather than typical statistical noise. This is a priori a much more pessimistic noise model than *statistical noise* from i.i.d. samples. Nonetheless, it makes proving lower bounds against large classes of algorithm easier to prove, and remarkably, it often matches the actual computational and sample complexity of the best-known algorithms under sampling noise.

Remark 4.1.4 (Hardness of noise-tolerant algorithms)**.** *In general, SQ lower bounds are believed to capture the computational complexity of noise-tolerant algorithms. Several (non-SQ) algorithms are known to perform much better than these lower bounds in noiseless settings (see Remark 4.2.8). That said, SQ lower bounds should be interpreted with caution. While they accurately capture the complexity of many noisy learning problems, there are cases where they are off (see Remark 4.2.8). Understanding when SQ lower bounds can accurately capture computational hardness is an active area of research.*

Interpreting SQ lower bounds. Lower bounds in the SQ model typically come in two flavors:

- **Query complexity lower bound:** These show that any SQ algorithm that succeeds at solving a problem must satisfy $q/\tau^2 \geq B$, where B is a problem-dependent quantity. Since $\tau \asymp 1/\sqrt{n}$ under empirical average simulation (which requires $\Omega(n)$ operations to evaluate), this can be interpreted as a lower bound on the runtime of the algorithm. Note that this lower bound rules out a continuous trade-off between number of queries q and tolerance τ.

- **Tolerance upper bound:** These typically show that if $\tau \geq 1/\sqrt{B}$, then any SQ algorithm requires an exponential number of queries. Under the correspondence $\tau = 1/\sqrt{n}$, this implies that if $n \leq B$, then no efficient SQ algorithm can succeed. This is often interpreted as a lower bound on the number of samples for polynomial-time (noise-tolerant) algorithms to exist.

We will show examples for both flavors of lower bounds. Again, the correspondence to runtime or sample size lower bounds should be interpreted with caution—as indications of hardness and not definitive barriers.

A first lower bound. As an illustration of the power of the SQ framework, we present a first, simple SQ lower bound on *detection problems*. More refined and stronger results will be presented in Section 4.3.

[Should I call it decision problem? Many-to-one, composite versus simple, alternatives versus null.]

Consider the following *detection* or *hypothesis testing* problem: given a class of target distributions $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ and a null distribution $\mathcal{D}_0 \notin \mathcal{H}$, the goal is to decide whether the observed data is drawn from $\mathcal{D}_0$ or from

some $\mathcal{D} \in \mathcal{H}$:

$$\mathrm{H}_1 : \quad \mathcal{D} \in \mathcal{H}, \qquad \text{versus} \qquad \mathrm{H}_0 : \quad \mathcal{D} = \mathcal{D}_0. \tag{4.1.3}$$

This detection problem is easier than the corresponding learning problem—any algorithm that learns the class $\mathcal{H}$ (to sufficient precision) can also solve the detection task. Therefore, a lower bound on the detection problem readily implies a lower bound on learning. This reduction from learning to testing is a common strategy for proving lower bounds (see discussion in Chapter 5 on low-degree polynomial algorithms).

In this setting, an SQ algorithm $\mathcal{A}$ takes a source distribution $\mathcal{D}$ and outputs a binary decision $\mathcal{A}(\mathcal{D}) \in \{0, 1\}$. We say that $\mathcal{A}$ succeeds if, for all $\mathcal{D} \in \mathcal{H} \cup \{\mathcal{D}_0\}$ and for any sequence of valid oracle responses, the algorithm outputs:

$$\mathcal{A}(\mathcal{D}) = \begin{cases} 1 & \text{if } \mathcal{D} \in \mathcal{H}; \\ 0 & \text{if } \mathcal{D} = \mathcal{D}_0. \end{cases} \tag{4.1.4}$$

To quantify the hardness of this problem, we define a *SQ-alignment complexity* measure analogous to the alignment complexity used in Chapter 3 for noisy gradient descent (Definition 3.2.1). For every $\mathcal{D} \in \mathcal{H}$, let $L_{\mathcal{D}} := \frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}$ denote the likelihood ratio, which we assume belongs to $L^2(\mathcal{D}_0)$ (see Remark 4.1.8).

Definition 4.1.5 (SQ-alignment complexity)**.** *Let $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be a class of distributions, $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ a null distribution, and $\mu_{\mathcal{H}}$ a prior over $\mathbb{P} \in \mathcal{H}$. The* SQ-alignment complexity *of $(\mu_{\mathcal{H}}, \mathcal{D}_0)$ is defined as*

$$\mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0) := \left[\sup_{\|\phi\|_{\mathcal{D}_0} \le 1} \mathbb{E}_{\mathcal{D} \sim \mu_{\mathcal{H}}} \left[\langle L_{\mathcal{D}} - 1, \phi \rangle_{\mathcal{D}_0}^2 \right] \right]^{-1}, \tag{4.1.5}$$

where $L_{\mathcal{D}} := \frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}$ is the likelihood ratio of $\mathcal{D}$ with respect to $\mathcal{D}_0$.

Theorem 4.1.6 (SQ detection lower bound)**.** *Let $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be a class of target distributions and $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ a null distribution. Suppose there exists a SQ algorithm that solves the detection problem (4.1.3) with q queries and tolerance τ. Then for any prior $\mu_{\mathcal{H}} \in \mathcal{P}(\mathcal{H})$, we must have*

$$q/\tau^2 \ge \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0). \tag{4.1.6}$$

Proof. Fix a prior $\mu_{\mathcal{H}}$ over $\mathcal{H}$. The proof follows a standard SQ lower bound strategy. By definition, the algorithm must succeed for every source distributions $\mathcal{D} \in \mathcal{H} \cup \mathcal{D}_0$ and every sequence of responses compatible with the tolerance τ. Thus it is sufficient to display a deterministic sequence of queries $\phi_1, \ldots, \phi_q$ and responses $v_1, \ldots v_q$ that are compatible between $\mathcal{D}$ and $\mathcal{D}_0$ with positive probability over $\mathcal{D} \sim \mu_{\mathcal{H}}$.

Let $\phi_1, \ldots, \phi_q$ be the sequence of queries made by the algorithm $\mathcal{A}$ when it receives responses $v_t = \mathbb{E}_{\mathcal{D}_0}[\phi_t(y, \boldsymbol{x})]$. This sequence of queries is fixed, independent of $\mathcal{D} \sim \mu_{\mathcal{H}}$, and consistent with source distribution $\mathcal{D}_0$. Let us show that unless the bound (4.1.6) holds, then this query sequence is compatible with source $\mathcal{D} \in \mathcal{H}$ with positive probability over $\mathcal{D} \sim \mu_{\mathcal{H}}$. We have

$$
\begin{aligned}
&\mathbb{P}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left(\exists t \in [q], |\mathbb{E}_{\mathcal{D}}[\phi_t(y,\boldsymbol{x})] - \mathbb{E}_{\mathcal{D}_0}[\phi_t(y,\boldsymbol{x})]| > \tau\right) \\
&\overset{(a)}{\leq} q \sup_{\phi:\mathcal{Y}\times\mathcal{X}\to[-1,1]} \mathbb{P}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left(|\mathbb{E}_{\mathcal{D}}[\phi(y,\boldsymbol{x})] - \mathbb{E}_{\mathcal{D}_0}[\phi(y,\boldsymbol{x})]| > \tau\right) \\
&\overset{(b)}{\leq} q \sup_{\|\phi\|_{\mathcal{D}_0}\leq 1} \frac{\mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left[|\mathbb{E}_{\mathcal{D}}[\phi(y,\boldsymbol{x})] - \mathbb{E}_{\mathcal{D}_0}[\phi(y,\boldsymbol{x})]|^2\right]}{\tau^2} \\
&\overset{(c)}{=} \frac{q}{\tau^2} \sup_{\|\phi\|_{\mathcal{D}_0}\leq 1} \mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left[\mathbb{E}_{\mathcal{D}_0}\left[\left(\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}(y,\boldsymbol{x}) - 1\right)\phi(y,\boldsymbol{x})\right]^2\right] \\
&\overset{(d)}{=} \frac{q}{\tau^2}\frac{1}{\mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}},\mathcal{D}_0)},
\end{aligned}
\tag{4.1.7}
$$

where we used (a) a union bound over $t \in [q]$ and that $\phi_t \in \{\phi : \mathcal{Y}\times\mathcal{X} \to \mathbb{R} : \|\phi\|_\infty \leq 1\}$, (b) that $\{\phi : \mathcal{Y}\times\mathcal{X} \to \mathbb{R} : \|\phi\|_\infty \leq 1\} \subseteq \{\phi : \mathcal{Y}\times\mathcal{X} \to \mathbb{R} : \|\phi\|_{L^2(\mathcal{D}_0)} \leq 1\}$ and Markov's inequality, (c) the change of measure

$$
\mathbb{E}_{\mathcal{D}}\left[\phi(y,\boldsymbol{x})\right] = \mathbb{E}_{\mathcal{D}_0}\left[\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}(y,\boldsymbol{x})\phi(y,\boldsymbol{x})\right].
$$

and (d) the definition of the SQ-alignment complexity (4.1.5).

Hence, if $q/\tau^2 < \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}},\mathcal{D}_0)$, then with positive probability over $\mathcal{D} \sim \mu_{\mathcal{H}}$, the responses $v_1, \ldots, v_q$ are compatible with both $\mathcal{D}$ and $\mathcal{D}_0$. This implies that the algorithm cannot distinguish between the null and the alternative on at least one distribution in $\mathcal{H}$, contradicting the assumed correctness of the SQ algorithm. Therefore, we must have

$$
\frac{q}{\tau^2} \geq \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}},\mathcal{D}_0),
$$

which concludes the proof. □

The SQ-alignment complexity and the lower bound in Theorem 4.1.6 closely parallel the correlation alignment complexity and lower bound established in Theorem 3.2.2 for noisy gradient descent (GD). There, we showed that learning fails unless

$$k\left(\frac{R}{\tau}\right)^2 \gg \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, \mathcal{D}_0),$$

where k denotes the number of gradient steps, R is the clipping threshold on the gradients, and τ^2 is the variance of the injected Gaussian noise. Notably, the quantity τ/R in the noisy GD model plays the same role as the tolerance parameter τ in the SQ model where we 'clipped' the queries to $R = 1$.

In fact, we will show in the next section that the exact same lower bound

$$\frac{q}{\tau^2} \geq \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, \mathcal{D}_0),$$

with SQ-alignment replaced by the correlation alignment complexity, holds for the restricted class of SQ algorithms known as the *correlational statistical query* (CSQ) model (see Theorem 4.2.3).

It is worth emphasizing how much simpler the proof of Theorem 4.1.6 is under the SQ model—with adversarial noise—, compared to noisy GD with additive Gaussian noise. The worst-case assumption makes the analysis significantly cleaner since one can use any deterministic sequence of perturbations, instead of having to track the accumulation of Gaussian noise, as in the junk flow argument. In Section 4.4, we will revisit this connection and show how the junk flow proof can be adapted to rederive the SQ lower bound (4.1.6) under Gaussian additive noise, thereby connecting the two noise models for the alignment complexity lower bound.

Remark 4.1.7 (Randomized statistical dimension)**.** *From [Fel17] **[Tight characterization of detection, briefly describe]***

Remark 4.1.8 (Squared-integrable likelihood ratios)**.** *In our definition of the SQ-alignment complicity, we implicitly assume that each $\mathcal{D} \in \mathcal{H}$ is absolutely continuous with respect to $\mathcal{D}_0$, and that the Radon–Nikodym derivative $L_{\mathcal{D}} = \frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}$ belongs to $L^2(\mathcal{D}_0)$. This square-integrability condition is an important, but often implicit assumption in the literature. Intuitively, it ensures that the target distributions $\mathcal{D} \in \mathcal{H}$ have 'enough noise' with respect to the null $\mathcal{D}_0$.*

This 'noise' assumption is necessary to derive meaningful SQ lower bounds in regression settings: SQ algorithm are allowed to issue arbitrary bounded measurable queries (which might not be efficiently computable) and without such assumptions, arbitrarily complex query functions could be used to learn otherwise hard problems. For example, Vempala and Wilmes [VW19] showed that any finite class $\mathcal{H}$ of noiseless functions can be learned using just $\log|\mathcal{H}|$ queries of constant tolerance (better than the bound (4.1.6)), using non-robust, complex queries. See also related discussions in [Val12, SVWX17, JMS24].

4.2 Correlation Statistical Queries

The lower bound in Theorem 4.1.6 provided a simple illustration of the SQ framework, but more involved arguments are needed to obtain sharper bounds—particularly, upper bounds on the tolerance parameter τ.

Before turning to those in Section 4.3, we examine a simpler and natural subclass of SQ algorithms known as the *correlation statistical query* (CSQ) model, introduced by Bshouty and Feldman [BF02]. The CSQ model is a restricted version of the SQ model in which queries are limited to correlations between the label and a function of the input. This restriction makes the model easier to analyze, while still capturing the behavior of many important algorithms. In particular, the CSQ model often yields lower bounds that are easier to apply and interpret.

Definition of the CSQ model. In the CSQ model, queries are restricted to functions of the form $\phi(y, \boldsymbol{x}) = y \cdot \phi(\boldsymbol{x})$, for some $\phi : \mathcal{X} \to [-1, 1]$. In particular, this implies that the algorithm only interacts with the regression function $h(\boldsymbol{x}) := \mathbb{E}[y|\boldsymbol{x}]$, since $\mathbb{E}[y\phi(\boldsymbol{x})] = \mathbb{E}[h(\boldsymbol{x})\phi(\boldsymbol{x})]$.

Definition 4.2.1 ($\mathsf{CSQ}(\tau)$ oracle)**.** *Let $\mathcal{D} \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be the data distribution. Given a tolerance parameter $\tau > 0$, the* correlation statistical query oracle $\mathsf{CSQ}(\tau)$ *takes as input a query function $\phi : \mathcal{X} \to [-1, 1]$ and returns a value in the range:*

$$\Big[\mathbb{E}_{\mathcal{D}}[y\phi(\boldsymbol{x})] - \tau, \mathbb{E}_{\mathcal{D}}[y\phi(\boldsymbol{x})] + \tau\Big]. \tag{4.2.1}$$

[In order to be implementable by SQ we need bound $\|y\|_\infty$. More generally, we need a tail bound to ensure that we can implement from samples efficiently.]

Several important algorithms naturally operate within the CSQ model. Consider, for instance, gradient descent on the squared loss:

$$\hat{\mathcal{R}}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i;\boldsymbol{\theta}))^2,$$

with model parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. At each step, GD performs the update

$$\begin{aligned}\boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k - \nabla_{\boldsymbol{\theta}}\hat{\mathcal{R}}_n(\boldsymbol{\theta}^k) \\ &= \boldsymbol{\theta}^k + \underbrace{\frac{2}{n}\sum_{i=1}^{n} y_i \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i;\boldsymbol{\theta}^k)}_{\text{depends on labels (CSQ)}} - \underbrace{\frac{2}{n}\sum_{i=1} f(\boldsymbol{x}_i;\boldsymbol{\theta}^k)\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i;\boldsymbol{\theta}^k)}_{\text{does not depend on the labels}}. \end{aligned} \tag{4.2.2}$$

The first term is a vector of p CSQ queries (one per parameter), and the second term depends only on the input distribution. Therefore, gradient descent on squared loss with T steps can be implemented using pT CSQ queries, each with tolerance $\tau \approx 1/\sqrt{n}$. If the loss is not a squared loss (or logistic loss), the queries are not CSQ anymore, but still fall within the general SQ model.

Remark 4.2.2 (Data reuse)**.** *If the same data is reused at each step, gradient descent can implement 'sample extraction' [AKM+21] and vastly outperform SQ lower bounds in noiseless settings. However this requires a very non-standard model $f(\cdot;\boldsymbol{\theta})$. In the online setting (no data reuse), we expect CSQ to correctly capture the performance of gradient algorithms in many settings.*

SQ versus CSQ. In general, the CSQ model is strictly weaker than the SQ model. However, under certain conditions, the two models are equivalent. Specifically, if the labels are binary values $|\mathcal{Y}| = 2$ (binary classification) and the input distribution $\mathbb{P}\boldsymbol{x}$ is fixed, then any SQ query can be simulated by a CSQ query. To see this, let $\mathcal{Y} = \{a, b\}$. Then any function $\phi(y, \boldsymbol{x})$ can be decomposed as

$$\begin{aligned}\phi(y,\boldsymbol{x}) &= \phi(a,\boldsymbol{x})\frac{y-b}{a-b} + \phi(b,\boldsymbol{x})\frac{a-y}{a-b} \\ &= \underbrace{\frac{1}{a-b} y\left[\phi(a,\boldsymbol{x}) - \phi(b,\boldsymbol{x})\right]}_{\text{CSQ query}} + \underbrace{\frac{1}{a-b}\left[a\phi(b,\boldsymbol{x}) - b\phi(a,\boldsymbol{x})\right]}_{\text{does not depend on the labels}}. \end{aligned} \tag{4.2.3}$$

Thus, any SQ query can be expressed as the sum of a CSQ query and a term that depends only on the input distribution. Therefore, any SQ algorithm can be simulated by a CSQ algorithm in this setting. To show SQ lower bounds, it is sufficient to show a lower bound on CSQ algorithms.

However, this equivalence breaks down in more general settings:

- When $|\mathcal{Y}| > 2$, SQ algorithms are strictly more powerful than CSQ algorithms. We illustrate this gap when learning single-index models and sparse functions, where super-polynomial separations can appear.

- Even with binary labels, if the input distribution $\mathbb{P}_{\boldsymbol{x}}$ is not fixed (i.e., in the distribution-free setting), SQ can outperform CSQ. For instance, Feldman [Fel11] shows that linear threshold functions are SQ-learnable but not CSQ-learnable in the distribution-free model.

Alignment complexity. First, let us return to the detection problem (4.1.3) and derive a lower bound for the restricted class of CSQ algorithms. Assume a fixed input distribution $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$, and suppose that both the null distribution $\mathcal{D}_0$ and all target distributions $\mathcal{D} \in \mathcal{H}$ share this marginal on $\mathcal{X}$. In this setting, we can identify each distribution with its corresponding regression function: $h(\boldsymbol{x}) := \mathbb{E}_{\mathcal{D}}[y|\boldsymbol{x}] \in L^2(\mathbb{P}_{\mathcal{X}})$ for $\mathcal{D} \in \mathcal{H}$, and $g(\boldsymbol{x}) := \mathbb{E}_{\mathcal{D}_0}[y|\boldsymbol{x}] \in L^2(\mathbb{P}_{\mathcal{X}})$.

We recall the correlation alignment complexity from Definition 3.2.1: for a prior $\mu_{\mathcal{H}} \in \mathcal{P}(\mathcal{H})$ over regression functions,

$$\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g) = \left[\sup_{\|\phi\|_{\mathbb{P}_{\mathcal{X}}} \le 1} \mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\langle h - g, \phi\rangle^2_{\mathbb{P}_{\mathcal{X}}}\right]\right]^{-1}$$

We now state the CSQ analogue of Theorem 4.1.6.

Theorem 4.2.3 (CSQ detection lower bound)**.** *Fix an input distribution $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$. Let $\mathcal{H} \subseteq L^2(\mathbb{P}_{\mathcal{X}})$ be a class of regression functions, and $g \in L^2(\mathcal{X})$ the regression function of the null distribution. Suppose there exists a CSQ algorithm that solves the detection problem (4.1.3) with q queries and tolerance τ. Then for any prior $\mu_{\mathcal{H}} \in \mathcal{P}_{\mathcal{H}}$, we must have*

$$\frac{q}{\tau^2} \ge \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}, g). \tag{4.2.4}$$

Proof. This result follows from the same argument as in Theorem 4.1.6, with only difference the upper bound in (4.1.7):

$$\begin{aligned}
&\mathbb{P}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left(\exists t\in[q], |\mathbb{E}_{\mathcal{D}}[y\phi_t(\boldsymbol{x})]-\mathbb{E}_{\mathcal{D}_0}[y\phi_t(\boldsymbol{x})]|>\tau\right)\\
&\le \frac{q}{\tau^2}\sup_{\|\phi\|_{\mathbb{P}_{\mathcal{X}}}\le 1}\mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left[|\mathbb{E}_{\mathcal{D}}[y\phi(\boldsymbol{x})]-\mathbb{E}_{\mathcal{D}_0}[y\phi(\boldsymbol{x})]|^2\right]\\
&= \frac{q}{\tau^2}\sup_{\|\phi\|_{\mathbb{P}_{\mathcal{X}}}\le 1}\mathbb{E}_{h\sim\mu_{\mathcal{H}}}\left[\mathbb{E}_{\mathbb{P}_{\mathcal{X}}}\left[(h(\boldsymbol{x})-g(\boldsymbol{x}))\,\phi(\boldsymbol{x})\right]^2\right]\\
&= \frac{q}{\tau^2}\frac{1}{\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},g)}.
\end{aligned}$$

The rest of the proof is exactly the same. □

Statistical Query dimension. We now present a classical method for proving CSQ lower bounds, via the *statistical query dimension* of the target function class. This dimension is usually defined for binary classification, but we present a natural extension to regression settings.

Definition 4.2.4 (Statistical Query dimension)**.** *Let $\mathbb{P}_{\mathcal{X}}\in\mathcal{P}(\mathcal{X})$ be an input distribution and let $\mathcal{H}$ be a class of functions $h:\mathcal{X}\to\mathbb{R}$ with $\|h\|_{\mathbb{P}_{\mathcal{X}}}\le 1$. The* statistical query dimension *(SQ dimension) of $\mathcal{H}$ is the largest integer N such that there exist N functions $\{h_1,\dots,h_N\}\subseteq\mathcal{H}$ satisfying:*

$$\|h_j\|_{\mathbb{P}_{\mathcal{X}}}=1, \qquad \text{and} \qquad |\langle h_i,h_j\rangle_{\mathbb{P}_{\mathcal{X}}}|\le\frac{1}{N}, \quad \text{for all } i\neq j. \tag{4.2.5}$$

Intuitively, if a function class contains many nearly orthogonal functions, then any algorithm accessing only noisy correlation queries must spend significant effort to distinguish between them. Specifically, the next theorem implies that if $\mathcal{H}$ has SQ dimension N, then any CSQ algorithm will need at least $q/\tau^2=\Omega(N)$ query complexity to distinguish them.

In fact, this could already be derived from the correlation alignment complexity: for $\mu_{\mathcal{H}}$ uniform over the set of N functions (4.2.5), one can show that $\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}},\mathbb{P}_{\mathcal{X}})=\Omega(N)$ (thus, CSQ requires $\Omega(N)$ queries to distinguish all $h\in\mathcal{H}$ from the null regression function). However, the following result will give a stronger, quantitative lower bound for CSQ algorithms:

Theorem 4.2.5 ([Szö09, Theorem 5])**.** *Let $\mathbb{P}_{\mathcal{X}}\in\mathcal{P}(\mathcal{X})$ be an input distribution, and let $\mathcal{H}$ be a class of functions containing N functions $h_1,\dots,h_N$ satisfying:*

$$\|h_j\|_{\mathbb{P}_{\mathcal{X}}}=1, \quad \text{and} \quad |\langle h_i,h_j\rangle_{\mathbb{P}_{\mathcal{X}}}|\le\varepsilon, \ \ \forall i\neq j\in[N]. \tag{4.2.6}$$

Suppose a CSQ algorithm $\mathcal{A}$ makes q queries of tolerance τ, such that, for any target function $h \in \mathcal{H}$, it outputs a predictor $\mathcal{A}(h) : \mathcal{X} \to \mathbb{R}$. If

$$q \leq N\frac{\tau^2 - \varepsilon}{2}, \qquad \text{and} \qquad N \geq \frac{2}{\varepsilon}, \tag{4.2.7}$$

then we must have

$$\sup_{h \in \mathcal{H}} \|\mathcal{A}(h) - h\|^2_{\mathbb{P}_\mathcal{X}} \geq 1 - 2\varepsilon. \tag{4.2.8}$$

Proof. Let us first outline the proof strategy. The core idea is to show that if q is too small, CSQ algorithms with at most q query calls cannot distinguish between all N functions h_j. Let us sketch the main steps:

Step 1: Each query with tolerance τ allows to rule out the functions h_j whose correlation is not τ-close to the response. We show that each query eliminates at most $1/(\tau^2 - \varepsilon)$ functions, so that after $q \leq N(\tau^2 - \varepsilon)/2$ queries, we ruled out at most $N/2$ functions. Thus, there exists a subset $\tilde{\mathcal{H}} = \{\tilde{h}_1, \ldots, \tilde{h}_k\}$ of $k \geq N/2$ functions that are compatible with all q responses of the CSQ algorithm.

Step 2: For these k indistinguishable functions, the algorithm must output the same predictor $h_* := \mathcal{A}(\tilde{h}_i)$, for all $\tilde{h}_i \in \tilde{\mathcal{H}}$. We show that this implies that

$$\sup_{h \in \mathcal{H}} \|\mathcal{A}(h) - h\|^2_{\mathbb{P}_\mathcal{X}} \geq \max_{i \in [k]} \|h_* - \tilde{h}_i\|^2_{\mathbb{P}_\mathcal{X}} \geq 1 - 2\varepsilon.$$

Step 1: Let us bound how many hypotheses are eliminated by each query. The noise is adversarial and it is not enough to consider a deterministic sequence of answers. Let's simply fix $v_1 = v_2 \ldots = v_q = 0$ and the associated deterministic sequence of queries $\phi_1, \ldots, \phi_q$.

For each $t \in [q]$, define

$$S_t := \{i \in [N] : |\mathbb{E}_{\mathbb{P}_\mathcal{X}}[h_i(\boldsymbol{x})\phi_t(\boldsymbol{x})]| > \tau\},$$

so that $S_t \subseteq [N]$ are the indices of all the h_j's that we are able to rule out at

step t. Denote $N_t := |S_t|$ and $s_{t,j} = \text{sign}(\langle h_j, \phi_t \rangle_{\mathbb{P}_\mathcal{X}})$. By definition of S_t,

$$\begin{aligned} N_t \tau &\leq \sum_{j \in S_t} s_{j,t} \langle h_j, \phi_t \rangle_{\mathbb{P}_\mathcal{X}} \\ &\leq \|\phi_t\|_{\mathbb{P}_\mathcal{X}} \left\| \sum_{j \in S_t} s_{j,t} h_j \right\|_{\mathbb{P}_\mathcal{X}} \\ &\leq \sqrt{\sum_{j \in S_t} \|h_j\|^2_{\mathbb{P}_\mathcal{X}} + \sum_{i \neq j \in S_t} s_{i,t} s_{j,t} \langle h_j, h_i \rangle_{\mathbb{P}_\mathcal{X}}}, \end{aligned}$$

where we used Cauchy-Schwarz inequality and $\|\phi_t\|_{\mathcal{P}_\mathcal{X}} \leq \|\phi_t\|_\infty \leq 1$. Squaring both sides of the inequality and assumption (4.2.6), we deduce that

$$(N_t \tau)^2 \leq N_t + (N_t)^2 \varepsilon \quad \Longrightarrow \quad N_t \leq \frac{1}{\tau^2 - \varepsilon}.$$

Hence, the total number of eliminated functions after q queries is at most:

$$\sum_{t=1}^{q} N_t \leq \frac{q}{\tau^2 - \varepsilon}.$$

If $q \leq N(\tau^2 - \varepsilon)/2$, this is at most $N/2$, so at least $N/2$ functions remain compatible with all q responses.

Step 2: Let $\tilde{h}_1, \ldots, \tilde{h}_k$ be the indistinguishable functions, with $k \geq N/2$. Since $\mathcal{A}(\tilde{h}_1) = \ldots = \mathcal{A}(\tilde{h}_k) = h_*$, we have

$$\begin{aligned} \sup_{h \in \mathcal{H}} \|\mathcal{A}(h) - h\|^2_{\mathbb{P}_\mathcal{X}} &\geq \max_{i \in [k]} \|h_* - \tilde{h}_i\|^2_{\mathbb{P}_\mathcal{X}} \\ &\geq \min_{h \in L^2(\mathbb{P}_\mathcal{X})} \max_{i \in [k]} \|h - \tilde{h}_i\|^2_{\mathbb{P}_\mathcal{X}} \\ &\geq \min_{h \in L^2(\mathbb{P}_\mathcal{X})} \frac{1}{k} \sum_{i=1}^{k} \|h - \tilde{h}_i\|^2_{\mathbb{P}_\mathcal{X}}. \end{aligned}$$

This last equation is minimized at $\frac{1}{k} \sum_{j=1}^{k} \tilde{h}_j$. Expanding the square and

rearranging the terms, we obtain

$$\begin{aligned}\frac{1}{k}\sum_{i=1}^{k}\left\|\tilde{h}_i-\frac{1}{k}\sum_{j=1}^{k}\tilde{h}_j\right\|_{\mathbb{P}_{\mathcal{X}}}^2 &= \frac{1}{k}\sum_{i=1}^{k}\|\tilde{h}_i\|_{\mathbb{P}_{\mathcal{X}}}^2-\frac{1}{k^2}\sum_{i,j=1}^{k}\langle\tilde{h}_i,\tilde{h}_j\rangle_{\mathbb{P}_{\mathcal{X}}}\\ &= 1-\frac{1}{k}-\frac{1}{k^2}\sum_{i\neq j}\langle\tilde{h}_i,\tilde{h}_j\rangle_{\mathbb{P}_{\mathcal{X}}}\\ &\overset{(a)}{\geq} 1-\frac{1}{k}-\varepsilon\\ &\overset{(b)}{\geq} 1-\frac{2}{N}-\varepsilon\\ &\overset{(c)}{\geq} 1-2\varepsilon,\end{aligned}$$

where we used (a) $|\langle\tilde{h}_i,\tilde{h}_j\rangle_{\mathbb{P}_{\mathcal{X}}}|\leq\varepsilon$ by assumption, (b) $k\geq N/2$, and (c) $N\geq 2/\varepsilon$ by assumption. This concludes the proof of this theorem. □

This theorem implies that if the SQ dimension of $\mathcal{H}$ is at least N, then any CSQ algorithm requires at least

$$q\geq\frac{N\tau^2}{2}-1$$

queries to achieve squared test error below $1-4/N$. However, Theorem 4.2.5 allows to derive more general lower bound by fixing ε and N independently. For example, suppose there exist exponentially many functions in $\mathcal{H}$ satisfying (4.2.6) with $\varepsilon=1/B$. This implies that $\mathcal{H}$ is not CSQ-learnable unless $\tau^2=O(1/B)$ (that is $n=\Omega(B)$). We will see an example of such a lower bound when considering single-index models in Section 4.5.

Weak versus strong learning. Theorem 4.2.5 characterizes the complexity of *weak learning* in the CSQ model—that is, achieving performance strictly better than a trivial predictor (e.g., the algorithm that always output the 0 function, which achieves squared test error equal to 1).

This result is essentially tight for weak learning of binary functions. Let $\mathcal{H}$ be a class of functions $h:\mathcal{X}\to\{\pm1\}$ and consider a maximal subfamily of functions $\tilde{\mathcal{H}}=\{h_1,\dots,h_N\}$ satisfying $|\langle h_i,h_j\rangle_{\mathbb{P}_{\mathcal{X}}}|\leq\varepsilon$ for all $i\neq j$. Thus for any function $h\in\mathcal{H}$, there exists $h_j\in\tilde{\mathcal{H}}$ such that $|\langle h_i,h_j\rangle_{\mathbb{P}_{\mathcal{X}}}|>\varepsilon$. Fix $\tau=\varepsilon/3$, compute the correlation with the N functions in $\tilde{\mathcal{H}}$ and select the

function with response $|v| \geq \varepsilon/3$. By construction, it is guaranteed to have absolute correlation at least $\varepsilon/3$.

However, weak learning does not imply *strong learning*—achieving arbitrarily small test error—in the distribution-specific case. Various alternative of 'strong' SQ dimension have been introduced to capture strong learnability. We refer the reader to [Szö09, Fel12] for discussions.

Remark 4.2.6 (SQ dimension versus Dimension LB, weak learning)**.** ***[TBD]*** *Weak learning between CSQ and kernel methods. If SQ dimension N, needs $q/\tau^2 = \Omega(N)$ to learn with edge $1/N$. taking these N functions, we get dimension lower bound $\Omega(N)$. LB on query complexity versus number of samples. But one is adaptive, and strong learning very different.*

Example of parity functions. We illustrate the SQ dimension with a canonical example: learning parity functions. Let $\mathcal{X} = \{\pm 1\}^d$ with uniform distribution $\mathbb{P}_{\mathcal{X}} = \nu_d = \mathrm{Unif}(\{\pm 1\}^d)$ and consider the class of all parity functions:

$$\mathcal{H} = \left\{ \chi_S(\boldsymbol{x}) = \prod_{i \in S} x_i : S \subseteq [d] \right\}. \tag{4.2.9}$$

Since $\mathcal{H}$ consists of binary-valued functions, learning $\mathcal{H}$ with CSQ or SQ are equivalent in this setting.

The class $\mathcal{H}$ contains 2^d orthogonal functions with unit norm (see Definition 2.4.4). Thus, its SQ dimension is exactly 2^d. Applying Theorem 4.2.5 immediately yields the following classical result:

Theorem 4.2.7 (SQ-hardness of learning parities [BFJ+94, Kea98])**.** *Any SQ algorithm that learns parity functions under the uniform distribution over the hypercube must make at least $2^{\Omega(d)}$ queries or have tolerance $\tau = 2^{-\Omega(d)}$.*

More precisely, Theorem 4.2.5 implies that no SQ algorithm can achieve nontrivial prediction error unless $q/\tau^2 = \Omega(2^d)$. We can also consider the class of k-sparse parity functions, defined as:

$$\mathcal{H}_k := \{\chi_S : S \subseteq [d], |S| = k\}, \qquad \text{with SQ dimension } |\mathcal{H}_k| = \binom{d}{k}.$$

This yield a lower bound $q/\tau^2 = \Omega(d^k)$ (for k fixed).

Remark 4.2.8 (Hardness of learning parities)**.** *Does this SQ lower bound reflect the true complexity of learning parity functions?*

- (Efficient learning without noise.) *Parities without noise can be learned efficiently using non-SQ algorithms. Indeed, each χ_S corresponds to a linear function over $\mathbb{Z}_2$: writing $\tilde{y} = (y+1)/2 \in \{0,1\}$ and $\boldsymbol{z} = (\boldsymbol{x}+\mathbf{1})/2 \in \{0,1\}^d$,*

$$\tilde{y} = \frac{\chi_S(\boldsymbol{x})+1}{2} = \boldsymbol{z}^\mathsf{T}\boldsymbol{c} \mod 2, \qquad \boldsymbol{c} = (\mathbb{1}[i \in S])_{i\in[d]}.$$

Thus, given n labeled examples, one can solve the linear system $\tilde{\boldsymbol{y}} = \boldsymbol{Z}\boldsymbol{c}$ over $\mathbb{Z}_2$, where $\boldsymbol{Z} \in \{\pm 1\}^{n\times d}$ and $\tilde{\boldsymbol{y}} \in \{0,1\}^n$ are the n samples transformed as above. As soon as $n = \Omega(d)$, with high probability this system is invertible and Gaussian elimination recovers $\boldsymbol{c}$ efficiently. However, this method is not robust and fails as soon as there is some noise in the labels: it is not noise-tolerant and does not fall within the SQ model.

- (Noisy settings.) *Does SQ correctly captures the complexity of noise-tolerant algorithms? Not quite. Blum, Kalai, and Wasserman [BKW03] showed that noisy parities are learnable in time $2^{\Theta(d/\log d)}$, which is much better than the $2^{\Omega(d)}$ bound in Theorem 4.2.7. However, in many noisy settings, SQ correctly captures learning hardness.*

As an application of Theorem 4.2.7, let us show SQ hardness of learning two-layer neural network under the uniform distribution over the hypercube. We first show that parity functions can be represented exactly by two-layer neural networks with ReLu activations.

Lemma 4.2.9. *Any parity function $\chi_S(\boldsymbol{x})$ with $|S| = k$ can be represented by a two-layer neural network with ReLu activation and k neurons.*

Proof. Let $X = \sum_{i\in S} x_i$. Observe that we can write

$$\chi_S(\boldsymbol{x}) = \prod_{i\in S} x_i = g_k(X),,$$

where g_k is the sawtooth function depicted in Figure 4.1 with $g_k(k-2i) = (-1)^i$ for $i = 0,\dots,k$. One can verify that

$$g_k(X) = (-1)^k + (-1)^{k+1}(X+k)_+ + 2\sum_{i=1}^{k-1}(-1)^{k-i+1}(X+k-2i)_+, \quad (4.2.10)$$

which expresses $g_k(X)$ as a linear combination of k ReLu activations. Finally, note that we can write $X = \langle \mathbf{1}_S, \boldsymbol{x}\rangle$ where $\mathbf{1} = (\mathbb{1}[i \in S])_{i\in[d]}$. Thus substituting $X = \langle \mathbf{1}_S, \boldsymbol{x}\rangle$ in Equation (4.2.10) yields the desired representation. □

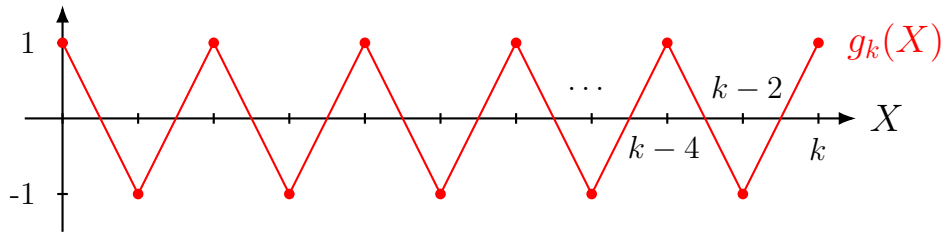

Figure 4.1: Sawtooth function $g_k(X)$.

Define $\mathcal{H}_{2\mathsf{NN},k}$ to be the class of two-layer neural networks with ReLu activation and k neurons. The previous lemma implies that $\mathcal{H}_k \subseteq \mathcal{H}_{2\mathsf{NN},k}$, so the SQ dimension of $\mathcal{H}_{2\mathsf{NN},k}$ is at least $\binom{d}{k}$.

Corollary 4.2.10. *The class $\mathcal{H}_{2\mathsf{NN},k}$ is not SQ-learnable under the uniform distribution over the hypercube whenever $k = \omega_d(1)$.*

Note that this lower bound holds even for improper learning. Compare this with the hardness result of Klivans and Sherstov (Theorem 1.4.4, Chapter 1), which shows that intersections of halfspaces—and hence two-layer ReLU networks (with ReLu output) and $k = d^{\varepsilon}$ neurons (it can be improved to $k = \omega_d(1)$ [DLSS14])—cannot be learned in polynomial time under the assumption that SVP is hard. That result rules out any polynomial-time algorithm (not just SQ) but holds in the worst case over the input distribution. In contrast, our SQ lower bound applies to algorithms that work under a fixed input distribution (uniform over the hypercube), but only rules out statistical query algorithms.

4.3 Lower bounds via Statistical Dimension

In the previous section, we saw that we can study the SQ complexity of binary classification by restricting ourselves to CSQ algorithms. In that setting, proving lower bounds (for weak learning) reduces to identifying a set of nearly orthogonal functions. This idea led to the notion of *SQ dimension*.

However, in more complex settings such as real-valued labels, the CSQ model is much less powerful than the SQ model, and the SQ dimension fails to capture SQ complexity. In this section, we will show that a closely related idea extends to the general SQ framework: proving lower bounds reduces again to identifying a set of distributions—but now with *likelihood ratios* that are

nearly orthogonal. This leads to a generalization of the SQ dimension known as the *statistical dimension with average correlation* (SDA), introduced by Feldman et al. [FGR+17].

We follow the framework developed in [FGR+17], which provides a unified approach to proving SQ lower bounds across a wide range of problems. In particular, it introduces the notion of *search problems over distributions*, which includes most standard learning tasks, such as regression, hypothesis testing, and strong learning. Our goal in this section is to convey the main ideas behind this framework rather than to state the most general or tightest results. For further refinements, generalizations, or applications, we refer the reader to [FGR+17, FPV15, Fel17].

Distributional search problems. We start by formally defining the class of problems we will consider in this section:

Definition 4.3.1 (Search problems over distributions)**.** *Fix a domain $\mathcal{Z}$ (measurable space). A* search problem over distributions *is defined by*

(i) A set of distributions $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Z})$;

(ii) A set of solutions $\mathcal{F}$;

(iii) A mapping $\varphi : \mathcal{H} \to 2^{\mathcal{F}}$.

For $\mathcal{D} \in \mathcal{H}$, $\varphi(\mathcal{D}) \subseteq \mathcal{F}$ is the (non-empty) set of valid solutions given $\mathcal{D}$. The distributional search problem *$(\mathcal{H}, \mathcal{F}, \varphi)$ corresponds to finding a valid solution $f \in \varphi(\mathcal{D})$ given access (to an oracle or samples from) an unknown $\mathcal{D} \in \mathcal{H}$. We will use $\mathcal{H}_f := \{\mathcal{D} \in \mathcal{H} : f \in \varphi(\mathcal{D})\}$ to denote the set of distributions $\mathcal{D}$ for which f is a valid solution.*

Supervised learning is a special case of this search problem: let $\mathcal{Z} = \mathcal{Y} \times \mathcal{X}$ be our domain and consider $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ a class of models $f : \mathcal{X} \to \mathcal{Y}$. Given a loss ℓ and an accuracy $\varepsilon > 0$, we define the set of valid solutions to be

$$\varphi(\mathcal{D}) := \{f \in \mathcal{F} : \mathbb{E}_{\mathcal{D}}[\ell(y, f(\boldsymbol{x}))] \leq \varepsilon\}.$$

In other words, the goal of the distributional search problem here is to find a model $f \in \mathcal{F}$ with test error at most ε given data from an unknown $\mathcal{D} \in \mathcal{H}$.

Example 4.3.2 (Detection problem)**.** *Consider the detection problem* (4.1.3) *with $\mathcal{H} = \mathcal{H}_1 \cup \{\mathcal{D}_0\}$, where $\mathcal{D}_0$ is the null and $\mathcal{H}_1$ is the set of alternative*

distributions. In this case we can set $\mathcal{F} = \{0, 1\}$ *and* $\varphi : \mathcal{H} \to 2^{\mathcal{F}}$ *as*

$$\varphi(\mathcal{D}) = \begin{cases} \{1\} & \text{if } \mathcal{D} \in \mathcal{H}_1, \\ \{0\} & \text{if } \mathcal{D} = \mathcal{D}_0. \end{cases}$$

The search problem in Definition 4.3.1 includes many other popular classes of problems, such as random constraint satisfaction or stochastic optimization. See [FGR+17, Fel17] for further examples and applications.

Statistical dimension for detection. We will characterize the SQ complexity of solving $(\mathcal{H}, \mathcal{F}, \varphi)$ as the hardest detection problem implicit in the search problem. The idea is that if an algorithm can solve the search problem, then for any subset $\mathcal{H}_1 \subseteq \mathcal{H}$ and null $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$ (sufficiently different from $\mathcal{H}_1$), the algorithm can distinguish between $\mathcal{H}_1$ and $\mathcal{D}_0$.

This is a common approach for proving lower bounds in general, and [Fel17] argues that it is essentially tight: the hardest detection problem is (roughly) as hard as the search problem. Therefore, we start by proving a lower bound on general detection problems. The lower bound on the search problem will be obtained by taking the worst of these lower bounds over all possible detection problems implicit in that search problem.

Below, we define the *statistical dimension with average correlation* (SDA) which will capture the difficulty of the detection problem. The difficulty of distinguishing an alternative $\mathcal{D}_1$ from a null $\mathcal{D}_0$ is controlled by

$$\mathbb{E}_{\mathcal{D}_1}[\phi] - \mathbb{E}_{\mathcal{D}_0}[\phi] = \mathbb{E}_{\mathcal{D}_0}\left[\frac{\mathrm{d}\mathcal{D}_1}{\mathrm{d}\mathcal{D}_0}\phi\right] - \mathbb{E}_{\mathcal{D}_0}[\phi] = \left\langle \frac{\mathrm{d}\mathcal{D}_1}{\mathrm{d}\mathcal{D}_0} - 1, \phi \right\rangle_{\mathcal{D}_0}.$$

This naturally leads to the definition of relative pairwise correlations.

Definition 4.3.3 (Average relative correlations)**.** *Given two distributions* $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{P}(\mathcal{Z})$ *and a reference distribution* $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$*, their* relative pairwise correlation *is defined by*

$$\chi_{\mathcal{D}_0}(\mathcal{D}_1, \mathcal{D}_2) := \left| \left\langle \frac{\mathrm{d}\mathcal{D}_1}{\mathrm{d}\mathcal{D}_0} - 1, \frac{\mathrm{d}\mathcal{D}_2}{\mathrm{d}\mathcal{D}_0} - 1 \right\rangle_{\mathcal{D}_0} \right| = \left| \left\langle \frac{\mathrm{d}\mathcal{D}_1}{\mathrm{d}\mathcal{D}_0}, \frac{\mathrm{d}\mathcal{D}_2}{\mathrm{d}\mathcal{D}_0} \right\rangle_{\mathcal{D}_0} - 1 \right|. \quad (4.3.1)$$

The average correlation $\rho(\mathcal{H}, \mathcal{D}_0)$ *of a class* $\mathcal{H}$ *relative to* $\mathcal{D}_0$ *is defined as*

$$\rho(\mathcal{H}, \mathcal{D}_0) := \frac{1}{|\mathcal{H}|^2} \sum_{\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{H}} \chi_{\mathcal{D}_0}(\mathcal{D}_1, \mathcal{D}_2). \quad (4.3.2)$$

In particular, if we can find a set of N distributions such that $\chi_{\mathcal{D}_0}(\mathcal{D}_i, \mathcal{D}_i) \leq C$ and $\chi_{\mathcal{D}_0}(\mathcal{D}_i, \mathcal{D}_j) \leq 1/N$ for all $i \neq j$, then $\rho(\mathcal{H}, \mathcal{D}_0) = O(1/N)$ (see later Definition 4.3.11 of pairwise (γ, β)-correlation).

Definition 4.3.4 (Statistical dimension, detection)**.** *For domain $\mathcal{Z}$, a finite set of distributions $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Z})$, and a reference distribution $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$, the* statistical dimension *of $\mathcal{H}$ relative to $\mathcal{D}$ with average correlation $\bar{\gamma}$ is defined to be the largest value d such that for any subset $\widetilde{\mathcal{H}} \subseteq \mathcal{H}$ with $|\widetilde{\mathcal{H}}| \geq |\mathcal{H}|/d$, we have $\rho(\widetilde{\mathcal{H}}, \mathcal{D}_0) \leq \bar{\gamma}$. We denote it by $\mathsf{SDA}(\mathcal{H}, \mathcal{D}_0, \bar{\gamma})$.*

Proposition 4.3.5 (SDA lower bound, detection)**.** *Let $\mathcal{H} = \mathcal{H}_1 \cup \{\mathcal{D}_0\} \subseteq \mathcal{P}(\mathcal{Z})$ be a detection problem with null $\mathcal{D}_0$ and a finite set of alternatives $\mathcal{H}_1$. For $\bar{\gamma} > 0$, if an SQ algorithm solves this problem with q queries and tolerance $\sqrt{\bar{\gamma}}$, then we must have*

$$q \geq \mathsf{SDA}(\mathcal{H}_1, \mathcal{D}_0, \bar{\gamma}). \tag{4.3.3}$$

Proof. This result follows from the same argument as in the proof with SQ dimension (Theorem 4.2.5). Denote $d = \mathsf{SDA}(\mathcal{H}_1, \mathcal{D}_0, \bar{\gamma})$.

Let's consider the sequence of queries $\phi_1, \ldots, \phi_q$ associated to the responses $v_t = \mathbb{E}_{\mathcal{D}_0}[\phi_t]$ (exact responses of the oracle under $\mathcal{D}_0$), and upper bound how many hypotheses in $\mathcal{H}_1$ are eliminated after q queries. For each $t \in [q]$, we define again

$$S_t := \left\{\mathcal{D} \in \mathcal{H}_1 : |\mathbb{E}_{\mathcal{D}}[\phi_t] - \mathbb{E}_{\mathcal{D}_0}[\phi_t]| > \sqrt{\bar{\gamma}}\right\},$$

that is, the subset of $\mathcal{H}_1$ that we are able to rule out with query ϕ_t. Denote $L_{\mathcal{D}} := \frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}$ and let $s_{t,\mathcal{D}} = \mathrm{sign}(\langle L_{\mathcal{D}} - 1, \phi_t\rangle)$. By definition of S_t,

$$\begin{aligned} |S_t|\sqrt{\bar{\gamma}} &< \sum_{\mathcal{D} \in S_t} s_{t,\mathcal{D}} \langle L_{\mathcal{D}} - 1, \phi_t\rangle_{\mathcal{D}_0} \\ &\leq \sqrt{\sum_{\mathcal{D}_1, \mathcal{D}_2 \in S_t} s_{t,\mathcal{D}_1} s_{t,\mathcal{D}_2} \langle L_{\mathcal{D}_1} - 1, L_{\mathcal{D}_2} - 1\rangle_{\mathcal{D}_0}} \leq |S_t|\sqrt{\rho(S_t, \mathcal{D}_0)}. \end{aligned}$$

Therefore $\bar{\gamma} < \rho(S_t, \mathcal{D}_0)$ and we must have $|S_t| \leq |\mathcal{H}_1|/d$: indeed, if $|S_t| > |\mathcal{H}_1|/d$, then by definition of SDA, we would have $\rho(S_t, \mathcal{D}_0) \leq \bar{\gamma}$.

Thus, any SQ algorithm rules out at most $q|\mathcal{H}_1|/d$ distributions using q queries. If the SQ algorithm distinguishes between all $\mathcal{D} \in \mathcal{H}_1$ and $\mathcal{D}_0$, then we must have $q|\mathcal{H}_1|/d \geq |\mathcal{H}_1|$, that is, $q \geq d$, which concludes the proof. □

Statistical dimension for search. We are now ready to present the general lower bound for any search problem. We define the statistical dimension of the search problem by reducing it to the worst statistical dimension over detection problems.

Definition 4.3.6 (Statistical dimension, search)**.** *For domain $\mathcal{Z}$ and search problem $(\mathcal{H}, \mathcal{F}, \varphi)$, the* statistical dimension *of $(\mathcal{H}, \mathcal{F}, \varphi)$ with average correlation $\bar{\gamma}$ is defined to be the largest value d such that there exist a reference distribution $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$ and a finite set of distributions $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ with the following property: for any solution $f \in \mathcal{F}$, the set $\tilde{\mathcal{H}}^f = \tilde{\mathcal{H}} \setminus \mathcal{H}_f$ is non-empty and $\mathsf{SDA}(\tilde{\mathcal{H}}^f, \mathcal{D}_0, \bar{\gamma}) \geq d$. We denote it by $\mathsf{SDA}(\mathcal{H}, \bar{\gamma})$.*

Intuitively, SDA for search problems is the worst SDA over 'well-posed' detection problems $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ versus $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$. Specifically, if an SQ algorithm output solution $f \in \mathcal{F}$ on the null $\mathcal{D}_0$, then f is not a valid solution for any of the alternatives $f \notin \bigcup_{\mathcal{D} \in \mathcal{H}} \varphi(\mathcal{D})$. Thus an SQ algorithm that solves the search problem must distinguish between $\mathcal{D}_0$ and $\tilde{\mathcal{H}}$.

Theorem 4.3.7 (SDA lower bound, search)**.** *Let $(\mathcal{H}, \mathcal{F}, \varphi)$ be a search problem with $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Z})$. For $\bar{\gamma} > 0$, if an SQ algorithm solves this problem with q queries and tolerance $\sqrt{\bar{\gamma}}$, then we must have*

$$q \geq \mathsf{SDA}(\mathcal{H}, \bar{\gamma}). \tag{4.3.4}$$

Proof. Let $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$ and $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ be a set of distributions that achieve $d = \mathsf{SDA}(\mathcal{H}, \bar{\gamma})$ in the Definition 4.3.6. Let $\mathcal{A}$ be a SQ algorithm that uses q queries of tolerance $\sqrt{\bar{\gamma}}$ to solve $(\mathcal{H}, \mathcal{F}, \varphi)$. Consider the sequence of queries $\phi_1, \ldots, \phi_q$ associated to the responses $v_t = \mathbb{E}_{\mathcal{D}_0}[\phi_t]$, and let $f \in \mathcal{F}$ be the output of $\mathcal{A}$ after seeing these q responses. Let $\tilde{\mathcal{H}}^f := \tilde{\mathcal{H}} \setminus \mathcal{H}_f$, so that f is not a solution to any of the alternative distributions $\tilde{\mathcal{H}}^f$. Thus $\mathcal{A}$ must solve the detection problem $\tilde{\mathcal{H}}^f$ versus $\mathcal{D}_0$. By definition of SDA, we have $\mathsf{SDA}(\tilde{\mathcal{H}}^f, \mathcal{D}_0, \bar{\gamma}) \geq d$. Thus, applying Proposition 4.3.5, we must have $q \geq d$, which concludes the proof. □

The above definition of SDA looks abstract and intimidating. We will show at the end of this section a simpler notion, based on pairwise correlations, which is easier to apply and will be sufficient in many settings.

VSTAT oracle. By Hoeffding's inequality, we saw that the $\mathsf{STAT}(\tau)$ oracle can be computed using an empirical average over $O(1/\tau^2)$ samples. However, if ϕ is very biased, e.g., $\phi(y, \boldsymbol{x}) = 0$ with high probability, $O(1/\tau^2)$ is too pessimistic and we can implement the oracle with far fewer queries.

To make the correspondence between the number of samples n and the tolerance τ more precise, Feldman et al [FGR+17] proposed a strengthening of the STAT oracle which incorporates the variance of the query:

Definition 4.3.8 ($\mathsf{VSTAT}(t)$ oracle)**.** *Let $\mathcal{D} \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be the data distribution. Given a parameter $t > 0$, the oracle $\mathsf{VSTAT}(t)$ takes as input a query $\phi : \mathcal{Y} \times \mathcal{X} \to [0, 1]$ and returns a value in the range $[p - \tau, p + \tau]$ where*

$$p = \mathbb{E}_{\mathcal{D}}[\phi], \quad \text{and} \quad \tau = \max\left\{\frac{1}{t}, \sqrt{\frac{p(1-p)}{t}}\right\}. \tag{4.3.5}$$

First, note that the value returned by $\mathsf{VSTAT}(t)$ is always within $1/\sqrt{t}$ from the expectation and the range is always at least $1/t$. Thus $\mathsf{VSTAT}(t)$ is no weaker than $\mathsf{STAT}(1/\sqrt{t})$ and no stronger than $\mathsf{STAT}(1/t)$. Second, $p(1-p)/t$ corresponds to the variance of the empirical mean with t samples when $\phi(y, \boldsymbol{x}) \in \{0, 1\}$ is boolean. More generally, for any $\phi : \mathcal{Y} \times \mathcal{X} \to [0, 1]$, the variance of the empirical average is upper bounded by

$$\mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(y_i, \boldsymbol{x}_i)\right) = \frac{\mathbb{E}_{\mathcal{D}}[\phi^2] - \mathbb{E}_{\mathcal{D}}[\phi]^2}{n} \le \frac{\mathbb{E}_{\mathcal{D}}[\phi] - \mathbb{E}_{\mathcal{D}}[\phi]^2}{n} = \frac{p(1-p)}{n}.$$

Thus, we can interpret $\mathsf{VSTAT}(t)$ as incorporating variance information about the query, with t the number size.

Remark 4.3.9 (Analyzing VSTAT)**.** *The oracle VSTAT is more complex to analyze than STAT: the expression is asymmetric in the response and $p = \mathbb{E}_{\mathcal{D}}[\phi]$, and the range depends on the data distribution through p. However, Feldman [Fel17] showed that $\mathsf{VSTAT}(t)$ is equivalent (up to a factor 3) to a simpler oracle that outputs a response $|\sqrt{v} - \sqrt{\mathbb{E}_{\mathcal{D}}[\phi]}| \le 1/\sqrt{t}$. This version is often much easier to analyze (see [Fel17, Lemma 5.2]).*

The proof of Proposition 4.3.5 can be adapted to $\mathsf{VSTAT}(t)$ oracle (see [FGR+17, Lemma 3.3]) to obtain the same lower bound $q \ge \mathsf{SDA}(\mathcal{H}_1, \mathcal{D}_0, \bar{\gamma})$ for SQ algorithms that make q query calls to $\mathsf{VSTAT}(1/(3\bar{\gamma}))$. In particular, we immediately obtain the following lower bound for search problem with VSTAT oracle:

Corollary 4.3.10 (SDA lower bound on VSTAT [FGR+17, Theorem 3.2])**.** *Let $(\mathcal{H}, \mathcal{F}, \varphi)$ be a search problem with $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Z})$. For $\bar{\gamma} > 0$, if an SQ algorithm solves this problem with q query calls to $\mathsf{VSTAT}(1/(3\bar{\gamma}))$, then we must have*

$$q \geq \mathsf{SDA}(\mathcal{H}, \bar{\gamma}). \tag{4.3.6}$$

Pairwise correlation. Finally, we introduce a simpler notion than SDA, that is more similar to our definition of the SQ dimension in Section 4.2. While SDA is based on average correlations, the following definition impose a bound on every pairwise correlation:

Definition 4.3.11 (Relative pairwise (γ, β)-correlation)**.** *We say that a set of N distributions $\mathcal{H} = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$ is* (γ, β)-correlated *relative to a distribution $\mathcal{D}_0$ if:*

$$|\chi_{\mathcal{D}_0}(\mathcal{D}_i, \mathcal{D}_j)| \leq \begin{cases} \beta & \text{for } i = j \in [N], \\ \gamma & \text{for } i \neq j \in [N]. \end{cases}$$

The statistical dimension with (γ, β)-correlation of a detection problem $\mathcal{H}_1$ versus $\mathcal{D}_0$ is the largest N such that there exists a subset $\tilde{\mathcal{H}}_1 \subseteq \mathcal{H}_1$ of size N that is (γ, β)-correlated relative to $\mathcal{D}_0$. We denote it $\mathsf{SDC}(\mathcal{H}_1, \mathcal{D}_0, \gamma, \beta)$. We can similarly define a statistical dimension for search problems:

Definition 4.3.12 (SDC with (γ, β)-correlation)**.** *The* statistical dimension *of $(\mathcal{H}, \mathcal{F}, \varphi)$ with (γ, β)-correlation is the largest value N such that there exist $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Z})$ and a finite set $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ with the following property: for any solution $f \in \mathcal{F}$, the set $\tilde{\mathcal{H}}^f = \tilde{\mathcal{H}} \setminus \mathcal{H}_f$ has size $|\tilde{\mathcal{H}}^f| \geq N$ and is (γ, β)-correlated relative to $\mathcal{D}_0$. We denote it by $\mathsf{SDC}(\mathcal{H}, \gamma, \beta)$.*

The statistical dimension with (γ, β)-correlation implies a lower bound on the statistical dimension with average correlation:

Lemma 4.3.13 ([FGR+17, Lemma 3.10])**.** *Let $\mathcal{H}$ be a set of N distributions that is (γ, β)-correlated relative to $\mathcal{D}_0$. Then, for every $\bar{\gamma} > \gamma$,*

$$\mathsf{SDA}(\mathcal{H}, \mathcal{D}_0, \bar{\gamma}) \geq N \frac{\bar{\gamma} - \gamma}{\beta - \gamma}. \tag{4.3.7}$$

Proof. Denote $d = N(\bar{\gamma} - \gamma)/(\beta - \gamma)$. Let us show that $\mathsf{SDA}(\mathcal{H}, \mathcal{D}_0, \bar{\gamma}) \geq d$: for any set of distributions $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ with

$$|\tilde{\mathcal{H}}| \geq \frac{|\mathcal{H}|}{d} = \frac{N}{d} = \frac{\beta - \gamma}{\bar{\gamma} - \gamma}, \tag{4.3.8}$$

we have

$$\rho(\tilde{\mathcal{H}}, \mathcal{D}_0) = \frac{1}{|\tilde{\mathcal{H}}|^2} \sum_{\mathcal{D}_1, \mathcal{D}_2 \in \tilde{\mathcal{H}}} |\chi_{\mathcal{D}_0}(\mathcal{D}_1, \mathcal{D}_2)| \leq \frac{\beta - \gamma}{|\tilde{\mathcal{H}}|} + \gamma \leq \bar{\gamma},$$

where we used (4.3.8) in the last inequality. This concludes the proof. □

In particular, this immediately imply the same lower bound on SDA for search problems: $\mathsf{SDA}(\mathcal{H}, \bar{\gamma}) \geq N(\bar{\gamma} - \gamma)/(\beta - \gamma)$. Thus, together with Theorem 4.3.7 and Corollary 4.3.10, we obtain the following lower bound based on statistical dimension with (γ, β)-correlation:

Corollary 4.3.14 (SDC lower bound, search [FGR+17, Corollary 3.12])**.** *Let $(\mathcal{H}, \mathcal{F}, \varphi)$ be a search problem with $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Z})$. For $\gamma, \beta > 0$, let $N = \mathsf{SDC}(\mathcal{H}, \gamma, \beta)$. For any $\bar{\gamma} > \gamma$, any SQ algorithm requires at least*

$$q \geq N \frac{\bar{\gamma} - \gamma}{\beta - \gamma}$$

query calls to $\mathsf{STAT}(\sqrt{\bar{\gamma}})$ or $\mathsf{VSTAT}(1/(3\bar{\gamma}))$ oracles to solve $(\mathcal{H}, \mathcal{F}, \varphi)$.

Hence, to show a SQ lower bounds on search problems (with either STAT or VSTAT oracle), it is sufficient to identify a set of distributions $\tilde{\mathcal{H}} \subseteq \mathcal{H}$ and a reference distribution $\mathcal{D}_0$ such that

- $\tilde{\mathcal{H}}$ is (γ, β)-correlated relative to $\mathcal{D}_0$;
- For any solution $f \in \mathcal{F}$, $|\tilde{\mathcal{H}} \setminus \mathcal{H}_f| \geq N$.

We will show an example of such a construction in Section 4.5 when studying Gaussian single index models.

4.4 SQ and noisy gradient descent

Before turning to examples, we briefly return to the connection between noisy GD and SQ algorithms discussed in Section 4.1. These two frameworks can be viewed as learning under restricted access to data with two different noise models: additive Gaussian noise for noisy GD and additive adversarial noise for SQ. Both are meant as simplified models to capture statistical noise.

Among the two, the SQ model represents a more conservative choice In particular, any algorithm that succeeds with adversarial SQ access is guaranteed to succeed under statistical (or Gaussian) noise—but not vice versa. While Gaussian noise and statistical noise are generally incomparable, the Gaussian model appears to be a more realistic assumption in certain scenarios, such as online learning, where fresh independent samples are drawn for each query. In such settings, Gaussian noise may offer a more accurate model than adversarial noise.

In this section, we further develop the connection between Gaussian and adversarial noise models. Recall that Theorem 4.2.3 showed a similar lower bound for CSQ algorithms as for noisy GD, in terms of a *correlation alignment complexity*. We extend this connection in two directions:

- We define a new SQ oracle based on additive Gaussian noise and adapt the junk flow technique to prove lower bounds against algorithms accessing this oracle. This generalizes the hardness results for noisy GD, ruling out a broader class of algorithms under the Gaussian noise model.
- Conversely, we establish an alternative lower bound on noisy GD by reducing a CSQ algorithm to noisy GD and transferring CSQ lower bounds to noisy GD. Although this yields a weaker lower bound than the junk flow argument, it illustrates an interesting approach: transferring hardness across different restricted models of computation via reductions.

Gaussian statistical query oracle. We consider the following oracle which replaces the adversarial noise from $\mathsf{STAT}(\tau)$ to additive Gaussian noise.

Definition 4.4.1 ($\mathsf{GSQ}(\tau)$ oracle)**.** *Let $\mathcal{D} \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be the data distribution. Given noise level $\tau > 0$, the* Gaussian noise oracle $\mathsf{GSQ}(\tau)$ *takes as input a query functions $\phi : \mathcal{Y} \times \mathcal{X} \to [-1, 1]$ and returns a random value with distribution*

$$v \sim \mathcal{N}(\mathbb{E}_{\mathcal{D}}[\phi], \tau^2). \tag{4.4.1}$$

A (deterministic) GSQ algorithm with q queries is determined by q functions $\Phi_t : \mathbb{R}^{t-1} \to [-1, 1]^{\mathcal{Y} \times \mathcal{X}}$, which given previous responses $v_1, \ldots, v_{t-1}$ outputs a new query function $\phi_t := \Phi_t(v_1, \ldots, v_{t-1}) : \mathcal{Y} \times \mathcal{X} \to [-1, 1]$, and an output function $\mathcal{A} : \mathbb{R}^q \to \mathcal{F}$, which given all q responses $v_1, \ldots, v_q$, outputs a solution $\mathcal{A}(v_1, \ldots, v_q) \in \mathcal{F}$. Given a source distribution $\mathcal{D}$, define sequentially the random query functions $\Phi_t(\mathcal{D})$ and the random scalar

variables $v_t(\mathcal{D})$ as $\Phi_t(\mathcal{D}) := \Phi_t(v_1(\mathcal{D}), \ldots, v_{t-1}(\mathcal{D}))$ and $v_t(\mathcal{D})$ the output of $\mathsf{GSQ}(\tau)$ with input $\Phi_t(\mathcal{D})$. In particular

$$v_t(\mathcal{D})\big|_{\Phi_t(\mathcal{D})} \sim \mathcal{N}\left(\mathbb{E}_{\mathcal{D}}[\Phi_t(\mathcal{D})], \tau^2\right).$$

Further, denote $\mathcal{A}(\mathcal{D})$ the random variable in $\mathcal{F}$ as

$$\mathcal{A}(\mathcal{D}) := \mathcal{A}(v_1(\mathcal{D}), \ldots, v_q(\mathcal{D})).$$

We show a lower bound on detection with GSQ algorithms analogous to Theorem 4.1.6 with adversarial SQ noise.

Theorem 4.4.2. *Let $\mathcal{H} \subseteq \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ be a class of target distributions, $\mu_{\mathcal{H}}$ a prior on $\mathcal{H}$, and $\mathcal{D}_0 \in \mathcal{P}(\mathcal{Y} \times \mathcal{X})$ a null distribution. For any $\eta \in (0,1)$ and any GSQ algorithm $\mathcal{A}$ with q queries and noise level τ, if*

$$\frac{q}{\tau^2} \leq 4\eta^2 \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0), \tag{4.4.2}$$

then, there exists a joint coupling between $\mathcal{A}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D}_0)$ such that

$$\mathbb{P}_{\mathcal{D}\sim\mu_{\mathcal{H}},\mathcal{A}(\mathcal{D}),\mathcal{A}(\mathcal{D}_0)}\left[\mathcal{A}(\mathcal{D}) = \mathcal{A}(\mathcal{D}_0)\right] \geq 1 - \eta. \tag{4.4.3}$$

Proof. This result follows by a straightforward adaptation of the proof of Lemma 3.2.3. We can write

$$\begin{aligned}
&\mathsf{KL}(\mathcal{L}(\mathcal{A}(\mathcal{D}_0))||\mathcal{L}(\mathcal{A}(\mathcal{D}))) \\
&= \sum_{t=0}^{q-1} \mathbb{E}_{\Phi_t\sim\mathcal{L}(\Phi_t(\mathcal{D}_0))}\left[\mathsf{KL}\left(\mathcal{L}(v_{t+1}(\mathcal{D}_0)|\Phi_t(\mathcal{D}_0) = \Phi_t)||\mathcal{L}(v_{t+1}(\mathcal{D})|\Phi_t(\mathcal{D}) = \Phi_t)\right)\right] \\
&= \frac{1}{2\tau^2}\sum_{t=0}^{q-1} \mathbb{E}_{\Phi_t\sim\mathcal{L}(\Phi_t(\mathcal{D}_0))}\left[\left(\mathbb{E}_{\mathcal{D}}[\Phi_t] - \mathbb{E}_{\mathcal{D}_0}[\Phi_t]\right)^2\right].
\end{aligned}$$

By change of measure and definition of $\mathsf{Align}_{\mathsf{SQ}}$,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\mathbb{E}_{\Phi_t\sim\mathcal{L}(\Phi_t(\mathcal{D}_0))}\left[\left(\mathbb{E}_{\mathcal{D}}[\Phi_t] - \mathbb{E}_{\mathcal{D}_0}[\Phi_t]\right)^2\right] \\
\leq &\sup_{\|\phi\|_{\mathcal{D}_0}\leq 1} \mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left[\langle L_{\mathcal{D}} - 1, \phi\rangle_{\mathcal{D}_0}^2\right] = \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0)^{-1}.
\end{aligned}$$

Thus we obtain the upper bound

$$\mathbb{E}_{\mathcal{D}\sim\mu_{\mathcal{H}}}\left[\mathsf{KL}(\mathcal{L}(\mathcal{A}(\mathcal{D}_0))||\mathcal{L}(\mathcal{A}(\mathcal{D})))\right] \leq \frac{q}{2\tau^2}\cdot\frac{1}{\mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0)}.$$

The rest of the proof is identical to the proof of Lemma 3.2.3. □

In other words, any GSQ algorithm fails to distinguish between $\mathcal{D}$ and $\mathcal{D}_0$ with probability at least $1-\eta$ over $\mathcal{D} \sim \mu_{\mathcal{H}}$ and the Gaussian noise in the queries, when

$$q/\tau^2 = O\left(\eta^2 \mathsf{Align}_{\mathsf{SQ}}(\mu_{\mathcal{H}}, \mathcal{D}_0)\right).$$

This is the same query complexity as for SQ model (Theorem 4.1.6).

However, the lower bound based on SDA in Proposition 4.3.5 heavily relies on the adversarial noise assumption and deterministic responses: GSQ requires taking an union bound over $h \in \mathcal{H}$, which makes the bound vacuous in this model.

Reduction from CSQ to Noisy GD. We consider a reduction from CSQ to noisy GD, but the same argument readily gives a reduction from the SQ model to the GSQ model.

Fix an input distribution $\mathbb{P}_{\mathcal{X}}$ and a class of target functions $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$. We consider two algorithms:

- **Noisy GD on target h:** Let $\boldsymbol{\theta}_h^0, \dots, \boldsymbol{\theta}_h^T$ be the trajectory of noisy gradient descent with $\boldsymbol{\theta}_h^0 \sim \rho_0$ and updated the parameters via

$$\boldsymbol{\theta}_h^{k+1} = \boldsymbol{\theta}_h^k - \eta_k \boldsymbol{g}_h(\boldsymbol{\theta}_h^k) + \eta_k \boldsymbol{\xi}^k, \quad \text{where } \boldsymbol{\xi}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p). \tag{4.4.4}$$

  Assume that the gradient has each coordinate clipped to 1, that is, $\|\nabla_{\boldsymbol{\theta}} f(\cdot; \boldsymbol{\theta})\|_\infty \le 1$ for all $\boldsymbol{\theta} \in \mathbb{R}^p$.

- **CSQ algorithm on target h:** Let $\tilde{\boldsymbol{\theta}}_h^0, \dots, \tilde{\boldsymbol{\theta}}_h^T$ be the trajectory constructed by a (random) CSQ algorithm with tolerance $\bar{\tau}$ as follows: let $\tilde{\boldsymbol{\theta}}_h^0 \sim \rho_0$ and iteratively define

$$\tilde{\boldsymbol{\theta}}_h^{k+1} = \tilde{\boldsymbol{\theta}}_h^k - \eta_k \boldsymbol{v}_h(\tilde{\boldsymbol{\theta}}_h^k) + \eta_k \tilde{\boldsymbol{\xi}}^k, \quad \text{where } \tilde{\boldsymbol{\xi}}^k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_p), \tag{4.4.5}$$

  where $\boldsymbol{v}_h(\tilde{\boldsymbol{\theta}}_h^k)$ is the vector of p responses from $\mathsf{CSQ}(\bar{\tau})$ oracle, when given the p inputs $\boldsymbol{g}_h(\tilde{\boldsymbol{\theta}}_h^k)$. Recall that gradient on squared loss can indeed be implemented within CSQ (by equation (4.2.2)). In particular,

$$\|\boldsymbol{v}_h(\tilde{\boldsymbol{\theta}}_h^k) - \boldsymbol{g}_h(\tilde{\boldsymbol{\theta}}_h^k)\|_\infty \le 2\bar{\tau}. \tag{4.4.6}$$

  Note that the total number of query call of this algorithm is pT.

We can show the following bound between the two trajectories:

Lemma 4.4.3. *Fix $\mathbb{P}_{\mathcal{X}}$ the input distribution and a target function $h : \mathcal{X} \to \mathbb{R}$. Then, there exists a joint coupling between $\boldsymbol{\theta}_h^{\leq T}$ the noisy GD trajectory (4.4.4) and $\tilde{\boldsymbol{\theta}}_h^{\leq T}$ the CSQ trajectory (4.4.5), such that*

$$\mathbb{P}\left[\boldsymbol{\theta}_h^{\leq k} = \tilde{\boldsymbol{\theta}}_h^{\leq k}\right] \geq 1 - \sqrt{pT}\frac{\bar{\tau}}{\tau}. \tag{4.4.7}$$

Proof. Again the result follows by adapting the proof of Lemma 3.2.3. We have

$$\begin{aligned}\mathsf{KL}(\mathcal{L}(\boldsymbol{\theta}_h^{\leq T})||\mathcal{L}(\tilde{\boldsymbol{\theta}}_h^{\leq T})) &= \frac{1}{2\tau^2}\sum_{k=0}^{T-1}\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{L}(\boldsymbol{\theta}_h^k)}\left[\left\|\boldsymbol{v}_h(\tilde{\boldsymbol{\theta}}_h^k) - \boldsymbol{g}_h(\tilde{\boldsymbol{\theta}}_h^k)\right\|_2^2\right] \\ &\leq 2\frac{Tp\bar{\tau}^2}{\tau^2}.\end{aligned}$$

The rest of the proof is identical and we omit it. □

Set $\bar{\tau} = \frac{\delta\tau}{\sqrt{pT}}$. Then the above lemma states that with probability at least $1-\delta$, we have $\boldsymbol{\theta}_h^T = \tilde{\boldsymbol{\theta}}_h^T$. Thus, if any CSQ algorithms fail to solve a problem with Tp queries and tolerance $\bar{\tau} = \frac{\delta\tau}{\sqrt{pT}}$, so does noisy GD with probability $1-\delta$. For example, CSQ fails to solve the detection problem if

$$\frac{Tp}{\bar{\tau}} < \mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}; g) \implies T < \frac{\delta\tau}{p}\left(\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}; g)\right)^{1/2},$$

and noisy GD fails with this number of steps with probability at least $1-\delta$. For $\tau, \delta = \Theta(1)$, this is much worse than the bound we derived with junk flow:

$$T \leq 4\delta^2\tau^2\mathsf{Align}_{\mathsf{Cor}}(\mu_{\mathcal{H}}; g).$$

However, the advantage of this approach is that one can directly transfer hardness of CSQ to hardness of noisy GD. For example, using CSQ-hardness of learning parities (Theorem 4.2.7), we directly deduce the following interesting result:

Corollary 4.4.4. *Noisy GD on a polynomially-sized neural network must either take $2^{\Omega(d)}$ many steps or have noise level $\tau = 2^{-\Omega(d)}$ in order to learn parity functions.*

4.5 Examples

[TBD]

Correlation Alignment complexity already done for both. Mention that it extends to CSQ (or SQ if binary valued).

4.5.1 Gaussian single-index models

we can do way much better by taking non-correlation queries (changing loss) or reusing queries.

generative exponent.

discuss difference with info exponent (e.g., polynomials: we can always transform them to be generative leap 1 or 2)

SQ query lower bound $q/\tau^2 \geq d^{k_*/2}$ SQ alignment

But also on the number of samples. Computational statistical gap.

Matching algo

4.5.2 Sparse functions on the hypercube

[TBD] Leap complexity SQ-leap complexity

Give an example wehre it lower the leap

Matching algo.

Chapter 5

Low-degree polynomials

In this chapter, we provide a brief introduction to the *low-degree polynomial* (LDP) method, a framework developed to study statistical-computational tradeoffs in high-dimensional inference [BHK+19, HS17, HKP+17, Hop18]. The LDP method has proven particularly effective in predicting statistical-computational gaps for a wide range of planted problems, including planted clique, sparse PCA, tensor PCA, sparse linear regression, and many others.

Formally, low-degree polynomials can be viewed as a restricted model of computation, in which the output of algorithms can be represented as a multivariate polynomial of bounded degree in the input data. These algorithms enjoy several desirable properties: they can be computed efficiently and are robust to noise in the data. Lower bounds within this framework rule out all algorithms whose outputs can be approximated by low-degree polynomials, thus providing evidence of computational hardness.

Remarkably, many popular algorithms—including spectral methods and approximate message passing—can be captured within this framework, and it was shown that low-degree polynomials are as powerful as the best-known polynomial-time algorithms for many canonical problems. This observation has led to the formulation of the so-called *low-degree conjecture*, which posits that for a broad class of inference problems, low-degree polynomials capture the full power of polynomial-time algorithms. That is, for 'nice' problems, an efficient algorithm exists if and only if the problem can be solved by a low-degree polynomial algorithm.

In this chapter, we focus on the application of the LDP framework to detection problems, and illustrate its power by deriving a lower bound for Gaussian single-index models. For a recent and comprehensive overview of the LDP framework and its applications, we refer the reader to [Wei25]; see

also [KWB19] for a pedagogical introduction.

5.1 Background on hypothesis testing

A majority of the works on the LDP framework concerns the *detection problem*. As discussed in Chapter 4, a lower bound for detection implies a lower bound for recovery (weak or strong learning). However, the converse does not necessarily hold: recovery may still be hard even when detection is easy. Recent work by Schramm and Wein [SW22] has extended the LDP framework to the recovery setting, but the theory is comparatively less developed and we will not pursue this direction.

In this section, we formally define our detection problem and review some classical results on hypothesis testing.

Sequence of detection problems. Throughout this chapter, we are interested in high-dimensional detection problems, which we formalize as a sequence of hypothesis testing tasks indexed by some integer d (e.g., the dimension of the input space). We will be interested in how the computational complexity of detection scales with d as $d \to \infty$.

Formally, let $\mathcal{Z} = (\mathcal{Z}^{(d)})_{d \geq 1}$ be a sequence of measurable spaces, for example $\mathcal{Z}^{(d)} := \mathcal{Y} \times \mathbb{R}^d$, and let $\mathcal{D} = \{\mathcal{D}^{(d)}\}_{d \geq 1}$ and $\mathcal{D}_0 = \{\mathcal{D}_0^{(d)}\}$ be two sequences of distributions on $\mathcal{Z}^{(d)}$. For each d, the task is to distinguish between the null hypothesis $H_0 : Z \sim \mathcal{D}_0^{(d)}$ and the alternative hypothesis $H_1 : Z \sim \mathcal{D}^{(d)}$. An algorithm that solves this problem defines a sequence of *tests*, that is, functions $T^{(d)} : \mathcal{Z}^{(d)} \to \{0, 1\}$, that takes a sample Z from either $\mathcal{D}^{(d)}$ or $\mathcal{D}$ and outputs $T(Z) = 1$ if it guesses that the sample comes from $\mathcal{D}^{(d)}$ and $T(Z) = 0$ for $\mathcal{D}_0^{(d)}$.

For notational simplicity, we will sometimes drop the superscript and leave the dependency on d implicit.

Likelihood ratio and optimal test. In statistical parlance, a test makes two possible types of errors:

- A *type I error* when the test falsely rejects the null (i.e., $T(Z) = 1$ when $Z \sim \mathcal{D}_0$);
- A *type II error* when the test fails to reject the null despite the alternative being true (i.e., $T(Z) = 0$ when $Z \sim \mathcal{D}$).

We define the type I and type II error probabilities as:

$$\begin{aligned} \alpha(T) &:= \mathbb{P}_{Z\sim\mathcal{D}_0}(T(Z)=1), \\ \beta(T) &:= \mathbb{P}_{Z\sim\mathcal{D}}(T(Z)=0), \end{aligned} \tag{5.1.1}$$

and refer to $1-\beta(T)$ as the *power* of the test.

There is a fundamental tradeoff between type I and type II errors. For instance, the constant test $T\equiv 0$ minimizes the type I error ($\alpha=0$) but has no power ($\beta=1$), while $T\equiv 1$ does the opposite. In statistics, one typically fixes a tolerance $\alpha\in[0,1]$ for the type I error and seeks a test maximizing the power subject to $\alpha(T)\le\alpha$.

A foundational result—the Neyman–Pearson lemma—characterizes the optimal solution to this tradeoff via the likelihood ratio test.

Definition 5.1.1 (Likelihood ratio test)**.** *Let $\mathcal{D}\ll\mathcal{D}_0$ and define the likelihood ratio as the Radon-Nikodym derivative $L(Z):=\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}(Z)$. The* likelihood ratio test *with threshold $\eta\ge 0$ is the test defined as*

$$\mathrm{LR}_\eta(Z) := \begin{cases} 1 & \text{if } L(Z)>\eta, \\ 0 & \text{if } L(Z)\le\eta. \end{cases} \tag{5.1.2}$$

Lemma 5.1.2 (Neyman-Pearson lemma [NP33])**.** *For $\eta\ge 0$, let $\alpha_*(\eta):=\alpha(\mathrm{LR}_\eta)$. Among all tests T such that $\alpha(T)\le\alpha_*(\eta)$, the lijelihood ratio test LR_η maximizes the power $1-\beta(T)$.*

In other words, for any fixed level $\alpha\in[0,1]$, the most powerful test that controls the type I error at level α is (possibly a randomized version of) the likelihood ratio test with a suitable threshold η such that $\alpha(\mathrm{LR}_\eta)=\alpha$.

Weak and strong detection. We consider two notions of success for our detection problem in high dimensions:

Definition 5.1.3 (Weak or Strong detection)**.** *Let $\mathcal{D}_0=\{\mathcal{D}_0^{(d)}\}_{d\ge1}$ and $\mathcal{D}=\{\mathcal{D}^{(d)}\}$ be sequences of distributions corresponding to the null and alternative hypotheses, and let $T=(T^{(d)})_{d\ge1}$ be a sequence of hypothesis tests.*

(i) (Strong detection.) *The test T achieves* strong detection *if it is correct with high probability, that is, type I and II errors vanish asymptotically:*

$$\alpha(T)+\beta(T)=o_d(1). \tag{5.1.3}$$

(ii) (Weak detection.) *We say that the test achieves* weak detection *if it performs strictly better than random guessing, with asymptotically non-vanishing advantage:*

$$\alpha(T) + \beta(T) = 1 - \Omega_d(1). \tag{5.1.4}$$

The asymptotic (in)distinguishability of two sequences of distributions can be characterized by the following notion of *contiguity*, an asymptotic analogue of absolute continuity between measures:

Definition 5.1.4 (Contiguity)**.** *A sequence of distributions $\mathcal{D} = (\mathcal{D}^{(d)})_{d\geq 1}$ is said to be* contiguous *to another sequence $\mathcal{D}_0 = (\mathcal{D}_0^{(d)})_{d\geq 1}$, denoted $\mathcal{D} \lhd \mathcal{D}_0$, if for any sequence of events $(A_d)_{d\geq 1}$ with $\mathcal{D}_0^{(d)}(A_d) \to 0$, it holds that $\mathcal{D}^{(d)}(A_d) \to 0$ as well.*

Lemma 5.1.5. *If either $\mathcal{D} \lhd \mathcal{D}_0$ or $\mathcal{D}_0 \lhd \mathcal{D}$, then strong detection is impossible. Conversely, if no test achieves weak detection, then both $\mathcal{D} \lhd \mathcal{D}_0$ and $\mathcal{D}_0 \lhd \mathcal{D}$.*

Proof. We identify a test $T^{(d)}$ with the event $A_d := \{T^{(d)} = 1\}$. Suppose $\mathcal{D} \lhd \mathcal{D}_0$ (the other case is symmetric), and assume for contradiction that there exists a test T achieving strong detection. Then,

$$\alpha(T) = \mathcal{D}_0^{(d)}(A_d) \to 0, \quad \text{and} \quad \beta(T) = \mathcal{D}^{(d)}(A_d^c) \to 0.$$

Since $\mathcal{D}_0^{(d)}(A_d) \to 0$, contiguity implies $\mathcal{D}^{(d)}(A_d) \to 0$, hence $\mathcal{D}^{(d)}(A_d^c) \to 1$, contradicting $\beta(T) \to 0$.

Conversely, suppose no test achieves weak detection. Then for all measurable sequences $T = (A_d)_{d\geq 1}$,

$$\alpha(T) + \beta(T) = \mathcal{D}_0^{(d)}(A_d) + \mathcal{D}^{(d)}(A_d^c) \geq 1 - o_d(1).$$

This implies

$$\mathcal{D}^{(d)}(A_d) \leq \mathcal{D}_0^{(d)}(A_d) + o_d(1),$$

and therefore, $\mathcal{D} \lhd \mathcal{D}_0$. The same argument shows $\mathcal{D}_0 \lhd \mathcal{D}$. □

Thus, to rule out strong detection, it suffices to show that either $\mathcal{D} \lhd \mathcal{D}_0$ or $\mathcal{D}_0 \lhd \mathcal{D}$ (as shown below, it will often be easier to show $\mathcal{D} \lhd \mathcal{D}_0$). Thanks to Neyman-Pearson lemma, we know that the optimal test is based on the likelihood ratio. Hence, it is natural to seek a criterion for weak and strong detection in terms of the likelihood ratio. This leads to the following classical criterion for detectability:

Proposition 5.1.6. *Let $L := (L_d)_{d\geq 1}$ be the sequence of likelihood ratios. Then:*

(a) If $\|L\|^2_{\mathcal{D}_0} = O_d(1)$, then strong detection is impossible.

(b) If $\|L\|^2_{\mathcal{D}_0} = 1 + o_d(1)$, then weak detection is impossible.

Proof. Let's show that $\|L\|^2_{\mathcal{D}_0} = O_d(1)$ implies $\mathcal{D} \lhd \mathcal{D}_0$, and thus, strong detection is impossible by Lemma 5.1.5. Take any sequence of events $(A_d)_{d\geq 1}$ such that $\mathcal{D}_0(A_d) \to 0$. By change of measure and Cauchy–Schwarz inequality:

$$\mathcal{D}(A_d) = \mathbb{E}_{\mathcal{D}_0}\left[L_d(Z)\mathbb{1}[Z \in A_d]\right] \leq \|L\|_{\mathcal{D}_0}\sqrt{\mathcal{D}_0(A_d)} \to 0.$$

Hence $\mathcal{D} \lhd \mathcal{D}_0$.

For part (b), consider the total variation:

$$2 \cdot \mathsf{TV}(\mathcal{D}, \mathcal{D}_0) = \mathbb{E}_{\mathcal{D}_0}[|L-1|] \leq \|L-1\|_{\mathcal{D}_0} = \sqrt{\|L\|^2_{\mathcal{D}_0} - 1} \to 0,$$

where we used that $\langle L, 1\rangle_{\mathcal{D}_0} = 1$ and therefore $\|L-1\|^2_{\mathcal{D}_0} = \|L\|^2_{\mathcal{D}_0} - 1$. By definition of the total variation distance, for any test T,

$$\alpha(T) + \beta(T) = \mathcal{D}_0(T=1) + 1 - \mathcal{D}(T=1) \geq 1 - \mathsf{TV}(\mathcal{D}, \mathcal{D}_0) = 1 - o_d(1),$$

and no test achieves weak detection. □

Proposition 5.1.6 is often referred to as the *second moment method*, and provides a practical criterion for proving contiguity. In particular, to prove impossibility of weak or strong detection, it is sufficient to control the L^2-norm of the likelihood ratio. However, the criteria in Proposition 5.1.6 are not necessary: there are situations where strong detection is impossible even though $\|L\|_{\mathcal{D}_0} \to \infty$, due to rare "bad events" where the likelihood ratio takes very large values. In such cases, more refined arguments—such as conditioning on the complement of these bad events in the second moment method—are required to establish contiguity.

An example: Principal Component Analysis. Let's illustrate Proposition 5.1.6 with a simple example. We consider the standard *spiked Gaussian Wigner model* with uniform prior, and the task of detecting the signal. Specifically, we consider $\mathcal{Z} = \mathrm{Sym}_d(\mathbb{R})$ the space of $d \times d$ symmetric matrices:

H_0: The data under the null $\boldsymbol{Y} \sim \mathcal{D}_0$ is drawn as

$$\boldsymbol{Y} = \boldsymbol{W}, \tag{5.1.5}$$

where $\boldsymbol{W}$ is a $d \times d$ random symmetric matrix with entries $W_{ii} \sim \mathcal{N}(0, 2/d)$ and $W_{ij} \sim \mathcal{N}(0, 1/d)$ independently for $i < j$.

H_1: The data under the alternative $\boldsymbol{Y} \sim \mathcal{D}$ is drawn as

$$\boldsymbol{Y} = \lambda \boldsymbol{u}\boldsymbol{u}^\mathsf{T} + \boldsymbol{W}, \tag{5.1.6}$$

where $\lambda > 0$, $\boldsymbol{u} \sim \pi := \mathrm{Unif}(\mathbb{S}^{d-1})$, and $\boldsymbol{W} \sim \mathcal{D}_0$.

When can we detect the signal, that is, distinguish between distributions with $\lambda > 0$ and $\lambda = 0$? The following lemma bound the second moment of this detection problem

Lemma 5.1.7. *If $\lambda < 1$, then $\|L\|^2_{\mathcal{D}_0} = O_d(1)$ and strong detection is impossible.*

Proof. The likelihood ratio in this setting simplifies to

$$\begin{aligned}\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}(\boldsymbol{Y}) &= \frac{\mathbb{E}_{\boldsymbol{u}\sim\pi}\left[\exp\left(-\frac{d}{4}\|\boldsymbol{Y} - \lambda\boldsymbol{u}\boldsymbol{u}^\mathsf{T}\|_F^2\right)\right]}{\exp\left(-\frac{d}{4}\|\boldsymbol{Y}\|_F^2\right)} \\ &= \mathbb{E}_{\boldsymbol{u}\sim\pi}\left[\exp\left(\frac{d\lambda}{2}\boldsymbol{u}^\mathsf{T}\boldsymbol{Y}\boldsymbol{u} - \frac{d\lambda^2}{4}\right)\right].\end{aligned}$$

Now passing to the second moment:

$$\begin{aligned}\left\|\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\mathcal{D}_0}\right\|^2_{\mathcal{D}_0} &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{v}\sim\pi}\mathbb{E}_{\boldsymbol{Y}\sim\mathcal{D}_0}\left[\exp\left(\frac{d\lambda}{2}[\boldsymbol{u}^\mathsf{T}\boldsymbol{Y}\boldsymbol{u} + \boldsymbol{v}^\mathsf{T}\boldsymbol{Y}\boldsymbol{v}] - \frac{d\lambda^2}{2}\right)\right] \\ &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{v}\sim\pi}\left[\exp\left(\frac{d\lambda^2}{4}\|\boldsymbol{u}\boldsymbol{u}^\mathsf{T} + \boldsymbol{v}\boldsymbol{v}^\mathsf{T}\|_F^2 - \frac{d\lambda^2}{2}\right)\right] \\ &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{v}\sim\pi}\left[\exp\left(\frac{d\lambda^2}{2}\langle\boldsymbol{u},\boldsymbol{v}\rangle^2\right)\right],\end{aligned} \tag{5.1.7}$$

where on the second line we used that $\boldsymbol{u}^\mathsf{T}\boldsymbol{Y}\boldsymbol{u} + \boldsymbol{v}^\mathsf{T}\boldsymbol{Y}\boldsymbol{v}$ is a Gaussian random variable with variance $2\|\boldsymbol{u}\boldsymbol{u}^\mathsf{T} + \boldsymbol{v}\boldsymbol{v}^\mathsf{T}\|_F^2/d$, and we simplified the Gaussian moment-generating function (MGF).

Under the uniform distribution over the sphere, $\langle \boldsymbol{u}, \boldsymbol{u}' \rangle^2 \sim \text{Beta}(\frac{1}{2}, \frac{d-1}{2})$. The second moment simply corresponds to the MGF of a Beta distribution at $t = d\lambda^2/2$, and we can compute it exactly in terms of a confluent hypergeometric function. Here, we only need to prove an upper bound uniform in d, and we use instead that $\mathbb{P}(|\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \geq t) \leq 2\exp(-dt^2/2)$ (that is, $(1/d)$-sub-Gaussianity tails):

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{u},\boldsymbol{v}\sim\pi}\left[\exp\left(\frac{d\lambda^2}{2}\langle \boldsymbol{u}, \boldsymbol{v}\rangle^2\right)\right] &= \int_0^\infty \mathbb{P}\left[\exp\left(\frac{d\lambda^2}{2}\langle \boldsymbol{u}, \boldsymbol{v}\rangle^2\right) \geq t\right] \mathrm{d}t \\
&= \int_0^\infty \mathbb{P}\left[|\langle \boldsymbol{u}, \boldsymbol{v}\rangle| \geq \sqrt{\frac{2\log t}{d\lambda^2}}\right] \mathrm{d}t \\
&\leq \int_0^\infty 2t^{-1/\lambda^2} \mathrm{d}t,
\end{aligned}$$

which is finite for $\lambda < 1$. □

In fact, one can show that for $\lambda < 1$, $\|L\|^2_{\mathcal{D}_0}$ converges to $(1-\lambda^2)^{-1/2}$, while $\|L\|^2_{\mathcal{D}_0}$ diverges for $\lambda > 1$. By Proposition 5.1.6, strong detection is impossible for $\lambda < 1$. On the other hand, strong detection is possible for $\lambda > 1$ by simply computing the top eigenvalue $\lambda_{\max} := \lambda_{\max}(\boldsymbol{Y})$:

- If $\boldsymbol{Y} \sim \mathcal{D}$ with $\lambda \leq 1$ (including the null $\lambda = 0$), then $\lambda_{\max}$ converges almost surely to 2.
- If $\boldsymbol{Y} \sim \mathcal{D}$ with $\lambda > 1$, then $\lambda_{\max}$ converges almost surely to $\lambda + 1/\lambda > 2$.

Thus the test that simply thresholds the top eigenvalue $\{\lambda_{\max}(\boldsymbol{Y}) \geq \eta\}$ for any $2 < \eta < \lambda + 1/\lambda$ has vanishing type I and II errors asymptotically.

The critical value $\lambda = 1$ is often referred to as the Baik-Ben Arous-Péché (BBP) transition [BBAP05, BS06, Péc06]. Below this value, $\mathcal{D} \lhd \mathcal{D}_0$, while above this value, $\mathcal{D}$ is not contiguous to $\mathcal{D}_0$ because strong detection is possible. For the interested reader that might be wondering what happens at $\lambda = 1$, Johnstone and Onatski [JO20] showed that strong detection is possible at this critical value.

Polynomial-time detection. We have seen that the likelihood ratio test is statistically optimal, and that its L^2-norm provides a convenient way to certify the impossibility of weak or strong detection. However, computing the likelihood ratio is generally intractable, as it often requires summing over

an exponentially large space. Thus, while the likelihood ratio characterizes information-theoretic (statistical) limits, it may not correspond to any feasible algorithm. Instead, we would like to restrict our attention to 'feasible' tests, that is, tests that are computable in polynomial time (polynomial in d). This leads to computational analogues of weak and strong distinguishability, where we require the test to be efficiently implementable.

Clearly, requiring distinguishability via a polynomial-time algorithm is a much stronger constraint than information-theoretic distinguishability. In the spiked Gaussian Wigner model, we saw that as soon as strong detection is possible, an efficient test exists, based on thresholding the top eigenvalue. However, in general, statistical distinguishability does not imply the existence of a polynomial-time algorithm. This phenomenon is known as a *statistical–computational gap*. The planted clique problem is a canonical example:

Problem 5.1.8 (Planted Clique)**.** *Let $\mathcal{Z}_d$ denote the space of undirected graphs on d vertices, represented as binary vectors $\boldsymbol{e} = (e_{ij})_{i<j\in[d]} \in \{0,1\}^{d(d-1)/2}$, where $e_{ij} = 1$ indicates an edge between vertices i and j. Consider the following hypothesis testing problem:*

H_0: Under the null, we observe an Erdös-Rényi graph $\mathcal{G}(d, 1/2)$, where each edge appears independently with probability $1/2$.

H_1: Under the alternative, we first choose a random subset $S \subset [d]$ of size $|S| = k$. The graph is then drawn by placing edges with probability 1 between all pairs $\{i, j\} \subseteq S$ (i.e., forming a planted clique), and with probability $1/2$ between all other pairs.

Under the alternative, the graph is a random Erdös-Rényi graph with a planted k-clique. When are these two distributions distinguishable? And when does there exist a polynomial-time algorithm that can perform the task?

- **Statistical threshold:** The largest clique in a random graph $\mathcal{G}(d, 1/2)$ has size approximately $k_{\text{stat}} = 2\log_2(d)$ with high probability. Thus, if $k \geq (1+\varepsilon)k_{\text{stat}}$, one can achieve strong detection by finding the largest clique: the maximum clique will, with high probability, correspond to the planted one. Conversely, if $k \leq (1-\varepsilon)k_{\text{stat}}$, it can be shown that no test (even computationally unbounded) can distinguish the hypotheses.

- **Computational threshold:** The brute-force algorithm that finds the largest clique requires enumerating all $\binom{d}{k}$ subsets and is computationally infeasible for large d. Is there an efficient algorithm for planted clique detection? Despite decades of work, the best known polynomial-time algorithms succeed only when $k = \Omega(\sqrt{d})$ [Kuč95, AKS98]. This leads to the conjecture that the *computational threshold* is $k_{\text{comp}} \approx \sqrt{d}$, and that for $k_{\text{stat}} \ll k \ll k_{\text{comp}}$, detection is information-theoretically possible but computationally intractable.

How can we provide evidence for the conjectured computational threshold k_{comp}? Earlier, we saw that statistical indistinguishability can be established by bounding the L^2-norm of the likelihood ratio. But to reason about the limits of *efficient* algorithms, we need a different framework—one that captures the power of polynomial-time computation.

This is precisely the motivation behind the *low-degree (polynomial) method*. The idea is to restrict attention to test statistics that are multivariate polynomials in the input of bounded degree D, and analyze their performance. In particular, we will examine the *low-degree likelihood ratio*, which is the projection of the likelihood ratio onto degree-D polynomials. We will see that bounding its L^2-norm is conjectured to provide a sharp criterion for computational detectability.

5.2 Low-degree polynomials

We now describe the *low-degree method*. The idea is to use low-degree multivariate polynomials as a proxy for polynomial-time computable functions. For many 'nice enough' problems, low-degree polynomials appear to capture the power of all known polynomial-time algorithms, that is, the problem is solvable by an efficient algorithm if, and only if, it is solvable by a low-degree polynomial. Thus, lower bounds against low-degree polynomials are taken as evidence that no efficient algorithm exists. This is formalized as the *low-degree conjecture*.

Low degree polynomials. Let $\mathcal{Z} \subseteq \mathbb{R}^N$ be the input space for our detection problem. We denote by $\mathbb{R}_{\leq D}[Z]$ the space of real-valued polynomials in $Z =$

$[z_1, \dots, z_N]$ of total degree at most D, i.e.,

$$\mathbb{R}_{\le D}[Z] = \operatorname{span}\left\{ \prod_{i\in[N]} z_i^{\beta_i} \;:\; (\beta_1,\dots,\beta_N) \in \mathbb{Z}^N_{\ge 0},\ \sum_{i=1}^N \beta_i \le D \right\}.$$

We further denote $\mathcal{V}_{\le D} \subseteq L^2(\mathcal{D}_0)$ the subspace of polynomials of degree at most D in $L^2(\mathcal{D}_0)$ (which induces an inner-product $\langle\cdot,\cdot\rangle_{\mathcal{D}_0}$ structure on $\mathbb{R}_{\le D}[Z]$). Let $\mathsf{P}_{\le D}$ denote the orthogonal projection onto $\mathcal{V}_{\le \mathcal{D}}$ in $L^2(\mathcal{D}_0)$.

The low-degree likelihood ratio. In the previous section, we saw that the likelihood ratio provides an information-theoretic optimal test to distinguish between distributions. Below, we introduce an analogous optimal test, now restricted to low-degree polynomials:

Definition 5.2.1 (Low-degree likelihood ratio)**.** *Consider a likelihood ratio function $L = \mathrm{d}\mathcal{D}/\mathrm{d}\mathcal{D}_0 \in L^2(\mathcal{D}_0)$. Then, the D-*low-degree likelihood ratio *(D-LDLR) is defined as $L^{\le D} := \mathsf{P}_{\le D} L$.*

Let's provide some high level motivation for this quantity, before discussing more formal justifications later in this section. With unbounded computation, the L^2-norm of the likelihood ratio characterizes indistinguishability: we can write this criterion in variational form as

$$\|L\|_{\mathcal{D}_0} = \sup_{f:\mathcal{Z}\to\mathbb{R}} \frac{\mathbb{E}_{\mathcal{D}}[f(Z)]}{\sqrt{\mathbb{E}_{\mathcal{D}_0}[f(Z)^2]}}, \tag{5.2.1}$$

with the supremum uniquely attained at $f = L/\|L\|_{\mathcal{D}_0}$ (this follows by rewriting the numerator as $\langle L, f\rangle_{\mathcal{D}_0}$). This suggests to measure the quality for a test statistics f using the following 'signal-to-noise ratio':

$$\frac{\mathbb{E}_{\mathcal{D}}[f(Z)]}{\|f\|_{\mathcal{D}_0}}.$$

To find an optimal computable test, we could therefore restrict the optimization in (5.2.1) to only maximizing over f that are polynomial-time computable. Restricting to low-degree polynomials, the following proposition shows that $L^{\le D}$ is indeed optimal following this heuristic:

Proposition 5.2.2. *Let $L^{\le D}$ be the D-LDLR. Then*

$$\|L^{\le D}\|_{\mathcal{D}_0} = \sup_{f \in \mathbb{R}_{\le D}[Z]} \frac{\mathbb{E}_{Z\sim\mathcal{D}}[f(Z)]}{\sqrt{\mathbb{E}_{Z\sim\mathcal{D}_0}[f(Z)^2]}}, \tag{5.2.2}$$

and the maximum is uniquely achieved at $f_* := L^{\le D}/\|L^{\le D}\|_{\mathcal{D}_0}$.

Proof. Simply rewrite the variational problem as

$$\sup_{f:\mathcal{Z}\to\mathbb{R}} \frac{\mathbb{E}_{\mathcal{D}}[\mathsf{P}_{\le D} f(Z)]}{\|\mathsf{P}_{\le D} f\|_{\mathcal{D}_0}} = \sup_{f:\mathcal{Z}\to\mathbb{R}} \frac{\langle \mathsf{P}_{\le D} L, \mathsf{P}_{\le D} f\rangle_{\mathcal{D}_0}}{\|\mathsf{P}_{\le D} f\|_{\mathcal{D}_0}}.$$

The proposition follows by Cauchy-Schwarz. □

The following conjecture, central to the low-degree method, posits that the second-moment criterion for statistical indistinguishability has a computational analogue when restricted to low-degree polynomials. That is, if the low-degree likelihood ratio has small norm, then no polynomial-time algorithm can distinguish the two distributions:

Conjecture 5.2.3 (Low-degree conjecture, informal)**.** *Let $\mathcal{D}$ and $\mathcal{D}_0$ be "nice enough" sequences of distributions. If there exists $\varepsilon >$ and a sequence of degrees $D = (D_d)_{d\ge 1}$ with $D_d \ge (\log(d))^{1+\varepsilon}$ for which the following holds:*

(a) If $\|L^{\le D}\|_{\mathcal{D}_0} = O_d(1)$, then no polynomial-time algorithm can achieve strong detection;

(b) If $\|L^{\le D}\|_{\mathcal{D}_0} = 1 + o_d(1)$, then no polynomial-time algorithm can achieve weak detection.

Informally, this conjecture suggests that polynomials of degree at most $\log(d)$ can be used as proxy for any polynomial-time algorithm. We will return to this conjecture later in this section: in particular, we will discuss what is meant by "nice enough", why the degree D scales as $\log(d)$, and what evidence supports (or challenges) the low-degree conjecture.

Weak and strong separation. We saw in Chapter 4 that a lower bound on SQ rules out success of any SQ algorithm under adversarial noise. This is somewhat believed to rule out noise-robust algorithm under statistical noise. What about low-degree polynomials? What algorithm can we formally rule out with the criterion in Conjecture 5.2.3?

Ideally, by analogy to the likelihood ratio and the above construction, we would like to rule out weak or strong detection for any test that can be written as $\{f(Z) \geq t\}$ where f is a degree-D polynomial. Let's introduce the following weaker notion of success for low-degree polynomials:

Definition 5.2.4 (Weak and strong separation)**.** *We say that a sequence of polynomials $f = (f_d)_{d\geq 1}$, $f_d : \mathcal{Z}^{(d)} \to \mathbb{R}$, strongly separates distributions $\mathcal{D}$ and $\mathcal{D}_0$ if*

$$\sqrt{\max\left\{\mathrm{Var}_{\mathcal{D}}(f), \mathrm{Var}_{\mathcal{D}_0}(f)\right\}} = o_d\left(\left|\mathbb{E}_{\mathcal{D}}[f] - \mathbb{E}_{\mathcal{D}_0}[f]\right|\right), \tag{5.2.3}$$

and weakly separates $\mathcal{D}$ and $\mathcal{D}_0$ if

$$\sqrt{\max\left\{\mathrm{Var}_{\mathcal{D}}(f), \mathrm{Var}_{\mathcal{D}_0}(f)\right\}} = O_d\left(\left|\mathbb{E}_{\mathcal{D}}[f] - \mathbb{E}_{\mathcal{D}_0}[f]\right|\right). \tag{5.2.4}$$

Proposition 5.2.5. *If a sequence of functions f strongly (resp. weakly) separates $\mathcal{D}$ and $\mathcal{D}_0$, then it achieves strong (resp. weak) detection.*

Proof. Suppose f strongly separates $\mathcal{D}$ and $\mathcal{D}_0$. Denote $\delta = \mathbb{E}_{\mathcal{D}}[f] - \mathbb{E}_{\mathcal{D}_0}[f]$ and assume $\delta > 0$ without loss of generality. Consider the test that rejects the null if $f(Z) \geq \mathbb{E}_{\mathcal{D}_0}[f] + \delta/2$. Then, by Markov's inequality

$$\begin{aligned}\mathbb{P}_{\mathcal{D}_0}\left(f(Z) \geq \mathbb{E}_{\mathcal{D}_0}[f] + \delta/2\right) &\leq \mathbb{P}_{\mathcal{D}_0}\left(|f(Z) - \mathbb{E}_{\mathcal{D}_0}[f]| \geq \delta/2\right) \\ &\leq 4\frac{\mathrm{Var}_{\mathcal{D}_0}(f)}{\delta^2} = o_d(1),\end{aligned}$$

where we used (5.2.3). Similarly,

$$\begin{aligned}\mathbb{P}_{\mathcal{D}}\left(f(Z) < \mathbb{E}_{\mathcal{D}_0}[f] + \delta/2\right) &\leq \mathbb{P}_{\mathcal{D}}\left(|f(Z) - \mathbb{E}_{\mathcal{D}}[f]| \geq \delta/2\right) \\ &\leq 4\frac{\mathrm{Var}_{\mathcal{D}}(f)}{\delta^2} = o_d(1).\end{aligned}$$

We conclude that this test achieves strong detection.

For weak separation, a similar thresholding test $\{f(Z) \geq t\}$ achieves weak detection, but now with t not necessarily the midpoint. The proof is more involved and we refer to [KSWY25, Proposition 3.2] for details. □

Thus, strong and weak separations give simple, sufficient conditions for weak and strong detection, based only on the two first moments of the polynomial function f. The following proposition shows that L^2-norm bounds on the low-degree likelihood ratio rules out this weaker version of detectability:

Proposition 5.2.6. *Let $L^{\leq D}$ be the sequence of low-degree likelihood ratios as defined in Definition 5.2.1, for some sequence of degrees $D = (D_d)_{d\geq 1}$. Then:*

(a) If $\|L^{\leq D}\|^2_{\mathcal{D}_0} = O_d(1)$, then no sequence of degree-D polynomials strongly separates $\mathcal{D}$ and $\mathcal{D}_0$.

(b) If $\|L^{\leq D}\|^2_{\mathcal{D}_0} = 1+o_d(1)$, then no sequence of degree-D polynomials weakly separates $\mathcal{D}$ and $\mathcal{D}_0$.

Proof. From the definition, showing that no degree-D polynomial weakly or strongly separates $\mathcal{D}$ and $\mathcal{D}_0$ is equivalent to showing that

$$\sup_{f\in\mathbb{R}_{\leq D}[Z]} \frac{\mathbb{E}_{\mathcal{D}}[f] - \mathbb{E}_{\mathcal{D}_0}[f]}{\sqrt{\max\{\mathrm{Var}_{\mathcal{D}}(f), \mathrm{Var}_{\mathcal{D}_0}(f)\}}}$$

is $o_d(1)$ or $O_d(1)$ respectively. Without loss of generality, we can impose $\mathbb{E}_{\mathcal{D}_0}[f] = 0$. Let us further relax the denominator, which is lower bounded by $\mathrm{Var}_{\mathcal{D}_0}(f)^{1/2} = \|f\|_{\mathcal{D}_0}$. Thus the above display is upper bounded by

$$\begin{aligned}\sup_{f\in\mathbb{R}_{\leq D}[Z],\mathbb{E}_{\mathcal{D}_0}[f]=0} \frac{\mathbb{E}_{\mathcal{D}}[f]}{\|f\|_{\mathcal{D}_0}} &\leq \sup_{f\in\mathbb{R}_{\leq D}[Z]} \frac{\mathbb{E}_{\mathcal{D}}[f] - \mathbb{E}_{\mathcal{D}_0}[f]}{\|f\|_{\mathcal{D}_0}} \\ &= \sup_f \frac{\langle L-1, \mathsf{P}_{\leq D} f\rangle_{\mathcal{D}_0}}{\|\mathsf{P}_{\leq D} f\|_{\mathcal{D}_0}} \\ &= \|L^{\leq D} - 1\|_{\mathcal{D}_0} = \sqrt{\|L^{\leq D}\|^2_{\mathcal{D}_0} - 1}.\end{aligned}$$

Assuming either $\|L^{\leq D}\|^2_{\mathcal{D}_0} = O_d(1)$ or $\|L^{\leq D}\|^2_{\mathcal{D}_0} = 1 + o_d(1)$ directly implies failure of strong and weak separation respectively. □

In Proposition 5.2.6, the statement is not bidirectional: there are examples where $\|L^{\leq D}\|_{\mathcal{D}_0}$ is diverging but strong separation is impossible [DMW25]. Indeed, in the proof we only show part of the strong separation criterion: we do not control the variance under the alternative $\mathrm{Var}_{\mathcal{D}}(f)$. Since counterexamples have been found, now people prefer directly use weak and strong separations directly as criterion of hardness: the threshold under this separation criterion indeed match the (conjectured) computational threshold in these counterexamples.

Summary. Convenient criterion that can be computed in many settings.

If we have an orthonormal basis of $\mathbb{R}[Z]_{\leq D}$ with respect to $\mathcal{D}_0$. Then we have simply

$$\|L^{\leq D}\|^2_{\mathcal{D}_0} = \sum_{i=0}^{k} \mathbb{E}_{Z\sim\mathcal{D}}[\phi_i(Z)]^2.$$

How to compute with basis. Use example.

5.2.1 Discussion on the low-degree conjecture

We refer to [Wei25]. [TBD]

Computational complexity of degree-D algo. why log d and spectral algorithms.

Robustness to noise. Why low-degree polynomials: Properties of low-degree polynomials: insensitive to low probability evens + adding noise makes functions become low-degree.

Ruling out low degree threshold functions The LDLR bound only rules out strong and weak separation. A more natural class of test to rule out would be all polynomial threshold function $\{f(Z) \geq t\}$ for f polynomial. However, our current lower bounds only rules out a control with second moment method.

$$\mathbb{P}_{Z\sim\mathcal{D}}(f(Z) - \mathbb{E}_{\mathcal{D}}[f] \geq t) \leq \mathbb{P}(|f(Z) - \mathbb{E}_{\mathcal{D}}[f]| \geq t)$$

[KWB19, Theorem 4.3].

Formal conjecture. What are the nice problems for the conjecture to hold.

5.3 Example: Gaussian single-index models

[TBD]

For sparse function, no statistical-computational gaps.

There is some evidence that gradient descent will not be able to achieve (online SGD sufficient $O(d^{k_*-1})$) but it remains open either way.

Bibliography

[AAM22] Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz, *The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks*, Conference on Learning Theory, PMLR, 2022, pp. 4782–4887.

[AB09] Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.

[ABA22] Emmanuel Abbe and Enric Boix-Adsera, *On the non-universality of deep learning: quantifying the cost of symmetry*, Advances in Neural Information Processing Systems **35** (2022), 17188–17201.

[ACHM22] Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Christopher Marquis, *An initial alignment between neural network and target is needed for gradient descent to learn*, International Conference on Machine Learning, PMLR, 2022, pp. 33–52.

[AKM+21] Emmanuel Abbe, Pritish Kamath, Eran Malach, Colin Sandon, and Nathan Srebro, *On the power of differentiable learning versus pac and sq learning*, Advances in Neural Information Processing Systems **34** (2021), 24340–24351.

[AKS98] Noga Alon, Michael Krivelevich, and Benny Sudakov, *Finding a large hidden clique in a random graph*, Random Structures & Algorithms **13** (1998), no. 3-4, 457–466.

[Aro50] Nachman Aronszajn, *Theory of reproducing kernels*, Transactions of the American mathematical society **68** (1950), no. 3, 337–404.

[AS20] Emmanuel Abbe and Colin Sandon, *Poly-time universality and limitations of deep learning*, arXiv preprint arXiv:2001.02992 (2020).

[AZL20] Zeyuan Allen-Zhu and Yuanzhi Li, *Backward feature correction: How deep learning performs deep learning*, arXiv:2001.04413 (2020).

[Bac17] Francis Bach, *Breaking the curse of dimensionality with convex neural networks*, The Journal of Machine Learning Research **18** (2017), no. 1, 629–681.

[Bar93] Andrew R Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information theory **39** (1993), no. 3, 930–945.

[BBAP05] Jinho Baik, Gérard Ben Arous, and Sandrine Péché, *Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices.*

[BBV06] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala, *Kernels as features: On kernels, margins, and low-dimensional mappings*, Machine Learning **65** (2006), 79–94.

[BF02] Nader H Bshouty and Vitaly Feldman, *On using extended statistical queries to avoid membership queries*, Journal of Machine Learning Research **2** (2002), no. Feb, 359–395.

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich, *Weakly learning dnf and characterizing statistical query learning using fourier analysis*, Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, 1994, pp. 253–262.

[BFT17] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky, *Spectrally-normalized margin bounds for neural networks*, Advances in neural information processing systems **30** (2017).

[BHK+19] Boaz Barak, Samuel Hopkins, Jonathan Kelner, Pravesh K Kothari, Ankur Moitra, and Aaron Potechin, *A nearly tight*

sum-of-squares lower bound for the planted clique problem, SIAM Journal on Computing **48** (2019), no. 2, 687–735.

[BKM+19] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborová, *Optimal errors and phase transitions in high-dimensional generalized linear models*, Proceedings of the National Academy of Sciences **116** (2019), no. 12, 5451–5460.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman, *Noise-tolerant learning, the parity problem, and the statistical query model*, Journal of the ACM (JACM) **50** (2003), no. 4, 506–519.

[BLLT20] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler, *Benign overfitting in linear regression*, Proceedings of the National Academy of Sciences **117** (2020), no. 48, 30063–30070.

[BMR21] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin, *Deep learning: a statistical viewpoint*, Acta numerica **30** (2021), 87–201.

[BR88] Avrim Blum and Ronald Rivest, *Training a 3-node neural network is np-complete*, Advances in neural information processing systems **1** (1988).

[BS06] Jinho Baik and Jack W Silverstein, *Eigenvalues of large sample covariance matrices of spiked population models*, Journal of multivariate analysis **97** (2006), no. 6, 1382–1408.

[BTA11] Alain Berlinet and Christine Thomas-Agnan, *Reproducing kernel hilbert spaces in probability and statistics*, Springer Science & Business Media, 2011.

[CDV07] Andrea Caponnetto and Ernesto De Vito, *Optimal rates for the regularized least-squares algorithm*, Foundations of Computational Mathematics **7** (2007), no. 3, 331–368.

[CMM21] Michael Celentano, Theodor Misiakiewicz, and Andrea Montanari, *Minimum complexity interpolation in random features models*, arXiv:2103.15996 (2021).

[COB19] Lenaic Chizat, Edouard Oyallon, and Francis Bach, *On lazy training in differentiable programming*, NeurIPS 2019-33rd Conference on Neural Information Processing Systems, 2019, pp. 2937–2947.

[DLSS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz, *From average case complexity to improper learning complexity*, Proceedings of the forty-sixth annual ACM symposium on Theory of computing, 2014, pp. 441–448.

[DM20] Amit Daniely and Eran Malach, *Learning parities with neural networks*, Advances in Neural Information Processing Systems **33** (2020), 20356–20365.

[DMW25] Abhishek Dhawan, Cheng Mao, and Alexander S Wein, *Detection of dense subhypergraphs by low-degree polynomials*, Random Structures & Algorithms **66** (2025), no. 1, e21279.

[DPVLB24] Alex Damian, Loucas Pillaud-Vivien, Jason Lee, and Joan Bruna, *Computational-statistical gaps in gaussian single-index models*, The Thirty Seventh Annual Conference on Learning Theory, PMLR, 2024, pp. 1262–1262.

[Duc24] John Duchi, *Lecture notes for statistics and information theory*, Personal Website (2024).

[DV21] Amit Daniely and Gal Vardi, *From local pseudorandom generators to hardness of learning*, Conference on Learning Theory, PMLR, 2021, pp. 1358–1394.

[DW18] Edgar Dobriban and Stefan Wager, *High-dimensional asymptotics of prediction: Ridge regression and classification*, The Annals of Statistics **46** (2018), no. 1, 247–279.

[Fel11] Vitaly Feldman, *Distribution-independent evolvability of linear threshold functions*, Proceedings of the 24th Annual Conference on Learning Theory, JMLR Workshop and Conference Proceedings, 2011, pp. 253–272.

[Fel12] ———, *A complete characterization of statistical query learning with applications to evolvability*, Journal of Computer and System Sciences **78** (2012), no. 5, 1444–1459.

[Fel17] ———, *A general characterization of the statistical query complexity*, Conference on learning theory, PMLR, 2017, pp. 785–830.

[FGR+17] Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh S Vempala, and Ying Xiao, *Statistical algorithms and a lower bound for detecting planted cliques*, Journal of the ACM (JACM) **64** (2017), no. 2, 1–37.

[FH23] Vincent Froese and Christoph Hertrich, *Training neural networks is np-hard in fixed dimension*, Advances in Neural Information Processing Systems **36** (2023), 44039–44049.

[FPV15] Vitaly Feldman, Will Perkins, and Santosh Vempala, *On the complexity of random satisfiability problems with planted solutions*, Proceedings of the forty-seventh annual ACM symposium on Theory of Computing, 2015, pp. 77–86.

[GJ02] Michael R Garey and David S Johnson, *Computers and intractability*, vol. 29, wh freeman New York, 2002.

[GMMM21] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari, *Linearized two-layers neural networks in high dimension*, The Annals of Statistics **49** (2021), no. 2, 1029–1054.

[HKP+17] Samuel B Hopkins, Pravesh K Kothari, Aaron Potechin, Prasad Raghavendra, Tselil Schramm, and David Steurer, *The power of sum-of-squares for detecting hidden structures*, 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2017, pp. 720–731.

[Hop18] Samuel Hopkins, *Statistical inference and the sum of squares method*, Cornell University, 2018.

[HS17] Samuel B Hopkins and David Steurer, *Bayesian estimation from few samples: community detection and related problems*, arXiv:1710.00264 (2017).

[HSSVG21] Daniel Hsu, Clayton H Sanford, Rocco Servedio, and Emmanouil Vasileios Vlatakis-Gkaragkounis, *On the approximation*

power of two-layer networks of random relus, Conference on Learning Theory, PMLR, 2021, pp. 2423–2461.

[Hsu21] Daniel Hsu, *Dimension lower bounds for linear approaches to function approximation*, Daniel Hsu's homepage (2021).

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, Advances in neural information processing systems **31** (2018).

[JKMS25] Nirmit Joshi, Hugo Koubbi, Theodor Misiakiewicz, and Nathan Srebro, *Learning single-index models via harmonic decomposition*, arXiv:2506.09887 (2025).

[JMS24] Nirmit Joshi, Theodor Misiakiewicz, and Nathan Srebro, *On the complexity of learning sparse functions with statistical and gradient queries*, arXiv preprint arXiv:2407.05622 (2024).

[JO20] Iain M Johnstone and Alexei Onatski, *Testing in high-dimensional spiked models*, The Annals of Statistics **48** (2020), no. 3.

[Jud87] J Stephen Judd, *Learning in networks is hard*, Proc. of 1st International Conference on Neural Networks, San Diego, California, June 1987, IEEE, 1987.

[Kea98] Michael Kearns, *Efficient noise-tolerant learning from statistical queries*, Journal of the ACM (JACM) **45** (1998), no. 6, 983–1006.

[Kha93] Michael Kharitonov, *Cryptographic hardness of distribution-specific learning*, Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, 1993, pp. 372–381.

[KMS20] Pritish Kamath, Omar Montasser, and Nathan Srebro, *Approximate is good enough: Probabilistic variants of dimensional and margin complexity*, Conference on Learning Theory, PMLR, 2020, pp. 2236–2262.

[KS09] Adam R Klivans and Alexander A Sherstov, *Cryptographic hardness for learning intersections of halfspaces*, Journal of Computer and System Sciences **75** (2009), no. 1, 2–12.

[KSWY25] Dmitriy Kunisky, Daniel A Spielman, Alexander S Wein, and Xifan Yu, *Statistical inference of a ranked community in a directed graph*, Proceedings of the 57th Annual ACM Symposium on Theory of Computing, 2025, pp. 2107–2117.

[Kuč95] Luděk Kučera, *Expected complexity of graph partitioning problems*, Discrete Applied Mathematics **57** (1995), no. 2-3, 193–212.

[KWB19] Dmitriy Kunisky, Alexander S Wein, and Afonso S Bandeira, *Notes on computational hardness of hypothesis testing: Predictions using the low-degree likelihood ratio*, ISAAC Congress (International Society for Analysis, its Applications and Computation), Springer, 2019, pp. 1–50.

[LZZ24] Shuchen Li, Ilias Zadik, and Manolis Zampetakis, *On the hardness of learning one hidden layer neural networks*, arXiv preprint arXiv:2410.03477 (2024).

[McC84] Peter McCullagh, *Generalized linear models*, European Journal of Operational Research **16** (1984), no. 3, 285–292.

[Mis22] Theodor Misiakiewicz, *Spectrum of inner-product kernel matrices in the polynomial regime and multiple descent phenomenon in kernel ridge regression*, arXiv:2204.10425 (2022).

[MM24] Theodor Misiakiewicz and Andrea Montanari, *Six lectures on linearized neural networks*, Journal of Statistical Mechanics: Theory and Experiment **2024** (2024), no. 10, 104006.

[MMM22] Song Mei, Theodor Misiakiewicz, and Andrea Montanari, *Generalization error of random feature and kernel methods: hypercontractivity and kernel matrix concentration*, Applied and Computational Harmonic Analysis **59** (2022), 3–84.

[MS24] Theodor Misiakiewicz and Basil Saeed, *A non-asymptotic theory of kernel ridge regression: deterministic equivalents, test error, and gcv estimator*, arXiv:2403.08938 (2024).

[NP33] Jerzy Neyman and Egon Sharpe Pearson, *Ix. on the problem of the most efficient tests of statistical hypotheses*, Philosophical

Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **231** (1933), no. 694-706, 289–337.

[Péc06] Sandrine Péché, *The largest eigenvalue of small rank perturbations of hermitian random matrices*, Probability Theory and Related Fields **134** (2006), no. 1, 127–173.

[PV88] Leonard Pitt and Leslie G Valiant, *Computational limitations on learning from examples*, Journal of the ACM (JACM) **35** (1988), no. 4, 965–984.

[Rey20] Lev Reyzin, *Statistical queries and statistical algorithms: Foundations and applications*, arXiv:2004.00557 (2020).

[RR07] Ali Rahimi and Benjamin Recht, *Random features for large-scale kernel machines*, Advances in neural information processing systems **20** (2007).

[SH20] Johannes Schmidt-Hieber, *Nonparametric regression using deep neural networks with relu activation function*, The Annals of Statistics **48** (2020), no. 4, 1875.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge university press, 2014.

[SVWX17] Le Song, Santosh Vempala, John Wilmes, and Bo Xie, *On the complexity of learning neural networks*, Advances in neural information processing systems **30** (2017).

[SW22] Tselil Schramm and Alexander S Wein, *Computational barriers to estimation from low-degree polynomials*, The Annals of Statistics **50** (2022), no. 3, 1833–1858.

[Szö09] Balázs Szörényi, *Characterizing statistical query learning: simplified notions and proofs*, International Conference on Algorithmic Learning Theory, Springer, 2009, pp. 186–200.

[Val84] Leslie G Valiant, *A theory of the learnable*, Communications of the ACM **27** (1984), no. 11, 1134–1142.

[Val12] Gregory Valiant, *Finding correlations in subquadratic time, with applications to learning parities and juntas*, 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, IEEE, 2012, pp. 11–20.

[Vem97] Santosh Vempala, *A random sampling based algorithm for learning the intersection of half-spaces*, Proceedings 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997, pp. 508–513.

[VW19] Santosh Vempala and John Wilmes, *Gradient descent for one-hidden-layer neural networks: Polynomial convergence and sq lower bounds*, Conference on Learning Theory, PMLR, 2019, pp. 3115–3117.

[Wai19] Martin J Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48, Cambridge university press, 2019.

[Wei25] Alexander S Wein, *Computational complexity of statistics: New insights from low-degree polynomials*, arXiv:2506.10748 (2025).

[Yan05] Ke Yang, *New lower bounds for statistical query learning*, Journal of Computer and System Sciences **70** (2005), no. 4, 485–509.

Appendix A

Technical results

[TBD]

A.1 Function space on the sphere

[TBD]

The function space $L^2(\bar{\gamma}_d)$ admits the following orthogonal decomposition

$$L^2(\bar{\gamma}_d) = \bigoplus_{k=0}^{\infty} \bar{\Omega}_{d,k}, \tag{A.1.1}$$

where $\bar{\Omega}_{d,k}$ is the subspace of degree-k spherical harmonics, that is, the space of degree-k polynomials that are orthogonal (in $L^2(\bar{\gamma}_d)$) to polynomials of degree at most $k-1$. For each $k \in \mathbb{N}$, denote $\bar{B}_{d,k} = \dim(\bar{\Omega}_{d,k}) = \Theta_d(d^k)$ and let $\{Y_{ks}\}_{s\in[\bar{B}_{d,k}]}$ be an orthonormal basis of $\bar{\Omega}_{d,k}$.

Denote $\bar{\gamma}_{d,1}$ the distribution of $\langle \boldsymbol{z}, \boldsymbol{e}_1\rangle$ when $\boldsymbol{z} \sim \bar{\gamma}_d$, that is, the marginal distribution of one coordinate of a unit vector uniformly distributed on $\mathbb{S}^{d-1}$. Introduce $\{Q_k\}_{k\geq 0}$ the orthonormal basis of Gegenbauer polynomials on $L^2(\bar{\gamma}_{d,1})$, such that

$$\mathbb{E}_{Z\sim\bar{\gamma}_{d,1}}[Q_k(Z)Q_{k'}(Z)] = \mathbb{E}_{\boldsymbol{z}\sim\bar{\gamma}_d}[Q_k(\langle \boldsymbol{w}, \boldsymbol{z}\rangle)Q_{k'}(\langle \boldsymbol{w}, \boldsymbol{z}\rangle)] = \delta_{kk'}.$$

A celebrated property of these polynomials is their correspondence to degree-k spherical harmonics: for any $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{S}^{d-1}$, we have

$$Q_k(\langle \boldsymbol{w}, \boldsymbol{z}\rangle) = \frac{1}{\sqrt{\bar{B}_{d,k}}} \sum_{s=1}^{\bar{B}_{d,k}} Y_{ks}(\boldsymbol{w})Y_{ks}(\boldsymbol{z}).$$

In particular, we deduce the following reproducing property of Gegenbauer polynomials

$$\mathbb{E}_{\boldsymbol{z}}[Q_k(\langle \boldsymbol{w}, \boldsymbol{z}\rangle)Q_{k'}(\langle \boldsymbol{z}, \boldsymbol{u}\rangle)] = \frac{\delta_{kk'}}{\sqrt{\bar{B}_{d,k}}} Q_k(\langle \boldsymbol{w}, \boldsymbol{u}\rangle). \tag{A.1.2}$$

A.2 Dimension lower bound

[TBD]

Specifically, consider the polar decomposition $\boldsymbol{x} = r\boldsymbol{z}$ with $r = \|\boldsymbol{x}\|_2 \sim \chi_2$ and $\boldsymbol{z} = \boldsymbol{x}/\|\boldsymbol{x}\|_2 \sim \bar{\gamma}_d := \mathrm{Unif}(\mathbb{S}^{d-1})$ independently.

Let us go back to our original function space $L^2(\gamma_d)$. We thus have the following alternative orthogonal decomposition

$$L^2(\gamma_d) = L^2(\chi_2) \otimes L^2(\bar{\gamma}_d) = \bigoplus_{k=0}^{\infty} L^2(\chi_2) \otimes \bar{\Omega}_{d,k} = \bigoplus_{k=0}^{\infty} \tilde{\Omega}_{d,k}, \tag{A.2.1}$$

where we denoted $\tilde{\Omega}_{d,k} := L^2(\chi_2) \otimes \bar{\Omega}_{d,k}$. That is $g \in \tilde{\Omega}_{d,k}$ if we can decompose it as

$$g(\boldsymbol{x}) = \sum_{s=1}^{\bar{B}_{d,k}} c_s(\|\boldsymbol{x}\|) Y_{ks}(\boldsymbol{x}/\|\boldsymbol{x}\|_2).$$

Proposition A.2.1. *We have*

$$n$$

A.3 Alignment complexity

[TBD]