

Parametric Models

Index terms: parametric statistics, estimators, error decomposition, time-data trade-off

1 Introduction

In the previous lecture, we discussed the elements of statistical learning using a Vapnik's block diagram, which is depicted in Figure 1. There is a generator providing data features \mathbf{a}_i , a supervisor providing real numbers (or labels, probabilities, etc.) b_i , and the learning machine's job is to predict or estimate what the supervisor is doing. We conceptualized the mapping from data to predictions as functions, and discussed how to estimate those functions.

We now move on to the powerful idea of *parametric models*, in which the (unknown) true function is assumed to lie in a family of functions parameterized by a finite-dimensional vector. In doing so, we reduce the search space from a potentially infinite-dimensional function space to a finite-dimensional parameter space. This chapter discusses classic examples of such parametric models and illustrates how maximum likelihood estimation (and more general M -estimators) unify the viewpoints of modeling and optimization.

We'll dive into several key parametric models including linear regression with Gaussian noise, logistic regression for classification, Poisson regression often used in modeling count data, and the broader framework of generalized linear models that unifies these approaches. We'll explore how these models are formulated, learned from data using the principle of maximum likelihood estimation, and ultimately used to draw inferences.

As we progress, we'll see that while the models differ in their specifics - the type of data they model and the link function that relates the linear predictor to the response variable - they share core similarities. Each model can be seen as an instantiation of the general approach of defining a probabilistic model, estimating its parameters by optimizing the likelihood function, and using the learned model to make predictions or decisions.

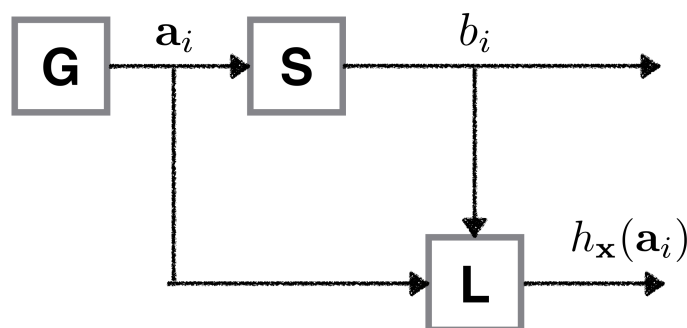


Figure 1: A basic statistical learning framework by Vapnik. It commonly consists of three elements: a generator that produces samples $\mathbf{a}_i \in \mathbb{R}^p$ of random variable $\mathbf{a} \sim \mathbb{P}_{\mathbf{a}}$, a supervisor that generates a sample b_i with an unknown conditional probability distribution $\mathbb{P}_{b|\mathbf{a}}$, and a learning machine $h \in \mathcal{H}^\circ$ that takes \mathbf{a}_i as an input. In the context of parametric estimation models, the function space \mathcal{H}° is characterized by parameters \mathbf{x} such that finding an estimate for h is computationally feasible.

2 Parametric Statistics

Parametric statistics is centered around the assumption that data are generated by an (unknown) distribution belonging to a *parametric family*, typically indexed by some finite-dimensional parameter vector. Before we get into specific models, let's establish the key components that define a parametric statistical model:

1. A **parameter space** $\mathcal{X} \subseteq \mathbb{R}^p$: This is the set of all possible values that the model's parameters can take. It's often a continuous space, defined by the number of parameters p .
2. The **true parameter** $\mathbf{x}^\dagger \in \mathcal{X}$: While we don't know its value, we assume that there exists a true set of parameters that generated our observed data.
3. A **class of probability distributions** $\mathcal{P}_{\mathcal{X}} = \{\mathbb{P}_{\mathbf{x}} : \mathbf{x} \in \mathcal{X}\}$: Each possible parameter value \mathbf{x} defines a probability distribution $\mathbb{P}_{\mathbf{x}}$ over the data space. This captures our assumption about how the data is generated given the parameters.
4. The **observed data** $(\mathbf{a}_i, b_i)_{i=1, \dots, n}$: We have n data points, each consisting of an input \mathbf{a}_i and an output or response b_i . We assume that each b_i is generated from the distribution $\mathbb{P}_{\mathbf{x}^\dagger, \mathbf{a}_i}$ - that is, the true distribution at \mathbf{x}^\dagger conditioned on the input \mathbf{a}_i .

With these elements in place, the goal of parametric statistical estimation is to learn an estimate $\hat{\mathbf{x}}$ of the true parameters \mathbf{x}^\dagger given the model setup $(\mathcal{X}, \mathcal{P}_{\mathcal{X}})$ and observed data $(\mathbf{a}_i, b_i)_{i=1, \dots, n}$. We'll see how this is done using maximum likelihood estimation.

2.1 Estimators

Definition 1 (Estimator). *An estimator \mathbf{x}^* is a mapping that takes $\mathcal{X}, \mathcal{P}_{\mathcal{X}}, (\mathbf{a}_i, b_i)_{i=1, \dots, n}$ as inputs and outputs a value in \mathcal{X} .*

A statistical *estimator* is a procedure (or a function) that takes in the problem ingredients and produces an estimate of the true parameter. Because the input to this estimator might be random, the output is also generally a random variable. An estimator can be deterministic or randomized:

- If the estimator is deterministic, once the data is fixed, the output (the parameter estimate) is fixed.
- If the estimator is randomized, even for fixed data, the estimator's output can still be random.

The output of an estimator is *not necessarily equal to the true parameter*. As an example, consider the *devil's advocate* or the *zero* estimator, which always outputs zero, regardless of the data. Although obviously simplistic, such constant estimators can sometimes be surprisingly competitive in certain structured estimation problems.

2.1.1 Estimation as an optimization problem

Last lecture, we introduced the maximum likelihood estimator (MLE). The main idea is picking the parameters that assign the highest probability to the observed data. For simplicity, suppose we have n independent and identically distributed (i.i.d.) observations. Then the probability of the entire dataset $(\mathbf{a}_i, b_i)_{i=1}^n$ is the product of the probability (or probability density) of each individual observation:

$$P_{\mathbf{x}}(b_1, \dots, b_n | \mathbf{a}_1, \dots, \mathbf{a}_n) = \prod_{i=1}^n P_{\mathbf{x}}(b_i | \mathbf{a}_i). \quad (1)$$

where P is a probability distribution parametrized by \mathbf{x} . The maximum likelihood rule says:

$$\mathbf{x}_{\text{ML}}^* = \arg \max_{\mathbf{x}} \prod_{i=1}^n P_{\mathbf{x}}(b_i | \mathbf{a}_i). \quad (2)$$

Often, it is more convenient to *minimize* the negative log-likelihood (NLL) instead of maximizing the product of probabilities:

$$\mathbf{x}_{\text{ML}}^* = \arg \min_{\mathbf{x}} - \sum_{i=1}^n \log p(\mathbf{B}_i | \mathbf{A}_i, \mathbf{x}). \quad (3)$$

This typically simplifies the algebra, since products become sums and exponentials become logarithms. Note that MLE is not always the best or only estimator; there exist others, such as the James–Stein estimator, which can strictly dominate MLE in some settings.

In the context of the statistical learning framework, MLE implicitly defines a loss function $L(\cdot, \cdot)$ measuring data fidelity as

$$\mathbf{x}_{\text{ML}}^* \in \arg \min_{\mathbf{x} \in \mathcal{X}} \{L(h_{\mathbf{x}}(\mathbf{a}, \mathbf{b})) := -\log p_{\mathbf{x}}(\mathbf{b})\} \quad (4)$$

where $p_{\mathbf{x}}(\cdot)$ denotes the probability density or probability mass function of $\mathbb{P}_{\mathbf{x}}$ for $\mathbf{x} \in \mathcal{X}$. As such, we can write down minimization problems as way of defining parameter estimators. More generally,

Definition 2 (*M-Estimators*). *Estimators can be formulated as optimization problems of the following form:*

$$\mathbf{x}^* \in \arg \min_{\mathbf{x} \in \mathcal{X}} \{F(\mathbf{x})\} \quad (5)$$

with some constraints $\mathcal{X} \in \mathbb{R}^p$. The term "*M-estimator*" denotes "*maximum-likelihood-type estimator*". [3]

2.1.2 Regression estimators via probabilistic models

Let $\mathbf{x} \in \mathbb{R}^p$ (i.e. $\mathcal{X} = \mathbb{R}^p$). Let $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$ be given feature vectors. The sample given by $\mathbf{b} := (b_1, \dots, b_n) \in \mathbb{B}^n$ for some set \mathbb{B} where each b_i follows a distribution $\mathbb{P}_{\mathbf{x}^\natural, \mathbf{a}_i}$ determined by \mathbf{x} and \mathbf{a}_i , and b_1, \dots, b_n are independent. In this particular case, the statistical estimation problem becomes like a *regression* problem as we regress a function (probability distribution) parametrized by \mathbf{x} on given data $(\mathbf{a}_i, b_i)_{1, \dots, n}$. In the sequel, we will discuss the following statistical regression models with examples:

- The *Gaussian linear regression model* is a regression model, where each b_i is a Gaussian random variable with mean $\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle$ and variance σ^2 , for some $\sigma > 0$.
- The *logistic regression model* is a regression model, where each b_i is a Bernoulli random variable with

$$P\{b_i = 1\} = 1 - P\{b_i = -1\} = [1 + \exp(-\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle)]^{-1}$$

- The statistical model for photon-limited imaging systems is a *Poisson regression model*, where each b_i is a Poisson random variable with mean $\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle$.

2.2 Gaussian linear regression model

Gaussian linear model has widespread and diverse applications across numerous real-world scenarios and scientific fields, one particularly notable example being Magnetic Resonance Imaging (MRI). MRI represents a sophisticated and non-invasive medical imaging technique that has become indispensable in modern healthcare, serving as a crucial diagnostic tool that is essential for detecting and identifying potential health problems early in their progression, such as brain tumors, spinal cord injuries, and vascular abnormalities.

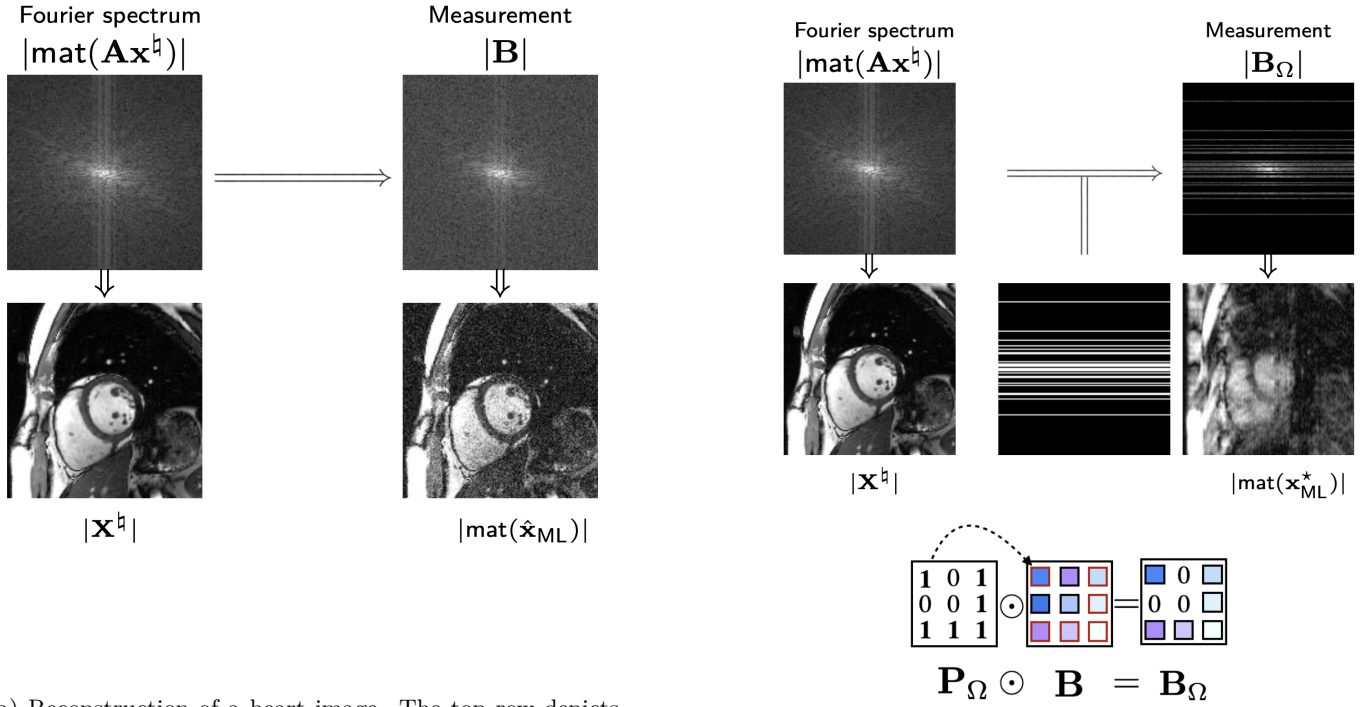
MRI machines work by detecting particle polarizations, which vary depending on the type of tissue being examined. This allows for distinguishing different tissue types within the body. For example, a cross-section of a cardiac MRI, as illustrated in Figure 2a, reveals detailed structures of the heart. The observed signal represents the discrete Fourier transform of the heart's image. The ultimate objective is to reconstruct these tissues accurately, enabling precise medical diagnoses.¹

More formally, let $\mathbf{X}^\natural \in \mathbb{C}^{\sqrt{p} \times \sqrt{p}}$ be a diagnostically meaningful MRI image we want to obtain. Denote $\mathbf{x}^\natural = \text{vec}(\mathbf{X}^\natural) \in \mathbb{C}^p$ as the vectorized image, where $\text{vec} : \mathbb{C}^{a \times b} \rightarrow \mathbb{C}^{ab}$ is a linear operator vectorizing a matrix. Let $\mathbf{A} \in \mathbb{C}^{p \times p}$ as the *discrete Fourier transform* (DFT) matrix. An MRI machine can produce as follows:

$$\mathbf{b} := \mathbf{A}\mathbf{x}^\natural + \mathbf{w} \in \mathbb{C}^p, \quad (6)$$

where $\mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \mathbf{I})$ is the complex Normal distributed noise and \mathbf{b} is the measurement vector with the spectrum $\mathbf{B} \in \mathbb{C}^{\sqrt{p} \times \sqrt{p}}$. The noise \mathbf{w} accounts for additive perturbations during the measurement process (e.g., interference from

¹To learn more on the physics behind MRI, visit www.mriquestions.com.



(a) Reconstruction of a heart image. The top row depicts the element-wise magnitude of complex images $|\cdot|$, where $\text{mat} : \mathbb{C}^{ab} \rightarrow \mathbb{C}^{a \times b}$ is the inverse operator of vec . We observe the measurement $|\mathbf{B}|$, which represents the noisy discrete Fourier transform of the underlying tissue $|\text{mat}(\mathbf{A}\mathbf{x}^h)|$.

(b) Accelerating MRI by scanning only certain points or lines in the Fourier spectrum. The missing elements in the measurements are represented as an element-wise product with a binary mask \mathbf{P}_Ω .

Figure 2: MRI reconstruction examples.

cell tower or radio signals) and is modeled as Gaussian under the assumption of the Central Limit Theorem. The ML estimator is the least squares estimator

$$\mathbf{x}_{\text{ML}}^* = \mathbf{x}_{\text{LS}}^* = \mathbf{A}^\dagger \mathbf{b} = \arg \min_{\mathbf{x}} \left\{ \frac{1}{p} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 : \mathbf{x} \in \mathbb{C}^p \right\} \quad (7)$$

where \mathbf{A}^\dagger is the (pseudo-)inverse of \mathbf{A} since \mathbf{A} is an orthonormal transformation.

The derivation: Since \mathbf{A} and \mathbf{x}^h are fixed, and \mathbf{w} is a complex normal random variable with variance $\sigma^2 \mathbf{I}$, it follows that \mathbf{b} is also a complex normal random variable. Its mean is

$$\mathbb{E}[\mathbf{b}] = \mathbb{E}[\mathbf{A}\mathbf{x}^h + \mathbf{w}] = \mathbf{A}\mathbf{x}^h. \quad (8)$$

Consequently, the probability density function $p_{\mathbf{x}}(\cdot)$ is given by

$$\begin{aligned} p_{\mathbf{x}}(\mathbf{b}) &= \left(\frac{1}{\pi\sigma^2} \right)^p \exp\left(-(\mathbf{b} - \underbrace{\mathbf{A}\mathbf{x}}_{\mu_{\mathbf{b}}})^\dagger \underbrace{\frac{\mathbf{I}}{\sigma^2}}_{\Sigma_{\mathbf{b}}} (\mathbf{b} - \mathbf{A}\mathbf{x}) \right) \\ &= \left(\frac{1}{\pi\sigma^2} \right)^p \exp\left(-\frac{1}{\sigma^2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \right). \end{aligned} \quad (9)$$

Therefore, the maximum likelihood (ML) estimator is

$$\mathbf{x}_{\text{ML}}^* = \arg \min_{\mathbf{x}} \left\{ -\log p_{\mathbf{x}}(\mathbf{b}) = -p \log(\pi\sigma^2) + \frac{1}{\sigma^2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 : \mathbf{x} \in \mathbb{C}^p \right\}. \quad (10)$$

Since the term $p \log(\pi\sigma^2)$ is constant with respect to \mathbf{x} , the above minimization is equivalent to

$$\mathbf{x}_{\text{ML}}^* = \arg \min_{\mathbf{x}} \left\{ \frac{1}{p} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 : \mathbf{x} \in \mathbb{C}^p \right\}. \quad (11)$$

We observe that the ML estimator for the Gaussian linear model coincides with the least squares (LS) estimator. Furthermore, when \mathbf{A} is the orthonormal DFT matrix, this problem has a unique solution.

However, this uniqueness does not hold in general. Depending on the dimensions of \mathbf{A} , a system can be:

- **Underdetermined** (e.g. more columns than rows): Infinitely many solutions exist.
- **Overdetermined** (e.g. more rows than columns): No exact solution exists in most cases.

When infinitely many solutions are possible, additional constraints are often imposed to induce structure (e.g., sparsity priors). These ideas will be explored in later chapters. Below, we address an underdetermined scenario where infinitely many solutions exist.

In some settings, it is advantageous to speed up the measurement process by acquiring only a subset of lines or elements in the measurement matrix \mathbf{B} , since the energy distribution of the Fourier transform is not uniform (see Figure 2b). The resulting measurements have missing entries, which can be modeled by a masking matrix $\mathbf{P}_\Omega \in \mathbb{C}^{\sqrt{p} \times \sqrt{p}}$ that selects a subset Ω of size $n \leq p$ while setting the remaining $p - n$ entries to zero. A basic subsampled MRI model is

$$\mathbf{B}_\Omega := \mathbf{P}_\Omega \odot \text{mat}(\mathbf{A}\mathbf{x}^\natural + \mathbf{w}), \quad (12)$$

where $\mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \mathbf{I})$ is complex Gaussian noise, and $\mathbf{b}_\Omega := \text{vec}(\mathbf{B}_\Omega)$ denotes the measurements in the Fourier domain.

We define the linear operator

$$\mathbf{A}_\Omega := \text{vec} \circ \mathbf{P}_\Omega \circ \text{mat}\mathbf{A}, \quad (13)$$

where “ \circ ” denotes function composition. The ML estimator then becomes

$$\mathbf{x}_{\text{ML}}^* = \mathbf{A}_\Omega^\dagger \mathbf{b}_\Omega \in \arg \min_{\mathbf{x}} \left\{ \frac{1}{n} \|\mathbf{b}_\Omega - \mathbf{A}_\Omega \mathbf{x}\|_2^2 : \mathbf{x} \in \mathbb{C}^p \right\}, \quad (14)$$

where $\mathbf{A}_\Omega^\dagger \mathbf{b}_\Omega$ is the (pseudo-)inverse of \mathbf{A}_Ω . In this underdetermined scenario, the solution is not unique. If \mathbf{A}_Ω does not have full column rank, any vector in its null space can be added to the solution without affecting the objective, thereby yielding infinitely many valid solutions.

2.3 Logistic regression model

Logistic regression is a widely used parameter estimation method for assigning labels to input features. [4] In classification problems, the outputs belong to a discrete, finite set, and the goal is to partition the high-dimensional input space into regions corresponding to these labels. In the following breast cancer detection example, we employ a simplified setup in which labels take values $b = 1$ (disease) or $b = -1$ (no disease). However, the same template extends naturally to more complex problems.

Model setup: Let $\mathbf{x}^\natural \in \mathbb{R}^p$ be an unknown parameter vector, and let $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$ be observed features. The sample is represented by $\mathbf{b} := (b_1, \dots, b_n) \in \{-1, 1\}^n$, where each b_i is a Bernoulli random variable satisfying

$$P\{b_i = 1\} = 1 - P\{b_i = -1\} = \left[1 + \exp(-\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle) \right]^{-1}. \quad (15)$$

The variables b_1, \dots, b_n are assumed to be independent.

Mapping features to probabilities: We can achieve this in two steps:

1. Convert each input feature \mathbf{a} to a scalar via a *score function*.
2. Map this scalar to the interval $(0, 1)$ to interpret it as a probability.

For each feature \mathbf{a} , we define and compute a score $s_{\mathbf{x}}(\mathbf{a}) \in (-\infty, \infty)$. A basic choice is *linear weighting*:

$$\mathbf{a} \mapsto s_{\mathbf{x}}(\mathbf{a}) = \mathbf{x}^{\top} \mathbf{a}. \quad (16)$$

In practice, more sophisticated score functions (e.g., neural networks) are also used in generative modeling and causal inference [10]. We will discuss these in more detail in Lecture 12.

Logistic function: A common way to map real-valued scores to probabilities is via the *logistic function*:

$$t \mapsto h(t) := \frac{1}{1 + \exp(-t)}. \quad (17)$$

Applying it to the score $s_{\mathbf{x}}(\mathbf{a})$ yields

$$P(b = \pm 1 | \mathbf{a}, \mathbf{x}) = h(\pm s_{\mathbf{x}}(\mathbf{a})) \in (0, 1). \quad (18)$$

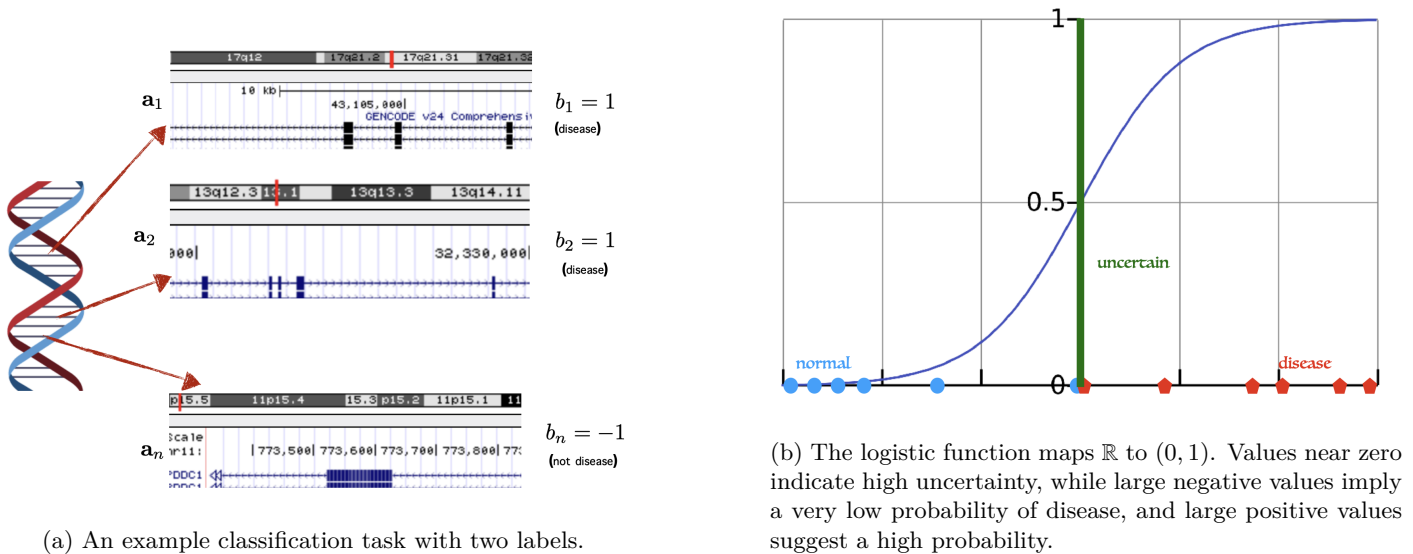


Figure 3: Logistic regression

Figure 3b illustrates the conditional probability of disease given \mathbf{a} . When using a linear score function, we impose a hyperplane in the feature space that separates the two labels. In particular,

$$P(b = 1 | \mathbf{a}, \mathbf{x}) = \begin{cases} > 0.5, & \text{if } s_{\mathbf{x}}(\mathbf{a}) > 0, \\ \leq 0.5, & \text{otherwise.} \end{cases} \quad (19)$$

Thus, the prediction rule becomes

$$\text{Prediction} = \begin{cases} \text{disease,} & \text{if } P(b = 1 | \mathbf{a}, \mathbf{x}) > 0.5, \\ \text{normal,} & \text{if } P(b = 1 | \mathbf{a}, \mathbf{x}) < 0.5, \\ \text{uncertain,} & \text{if } P(b = 1 | \mathbf{a}, \mathbf{x}) = 0.5. \end{cases} \quad (20)$$

The derivation: Because b_1, \dots, b_n are independent and identically distributed Bernoulli random variables with the

above logistic parametrization, the probability mass function is

$$\begin{aligned}
p_{\mathbf{x}}(\mathbf{b}) &= \prod_{i=1}^n \left[1 + \exp(-b_i s_{\mathbf{x}^{\natural}}(\mathbf{a}_i)) \right]^{-1} \\
-\log p_{\mathbf{x}}(\mathbf{b}) &= -\log \prod_{i=1}^n \left[1 + \exp(-b_i s_{\mathbf{x}^{\natural}}(\mathbf{a}_i)) \right]^{-1} \\
&= -\sum_{i=1}^n \log \left[1 + \exp(-b_i s_{\mathbf{x}^{\natural}}(\mathbf{a}_i)) \right]^{-1} \\
&= \sum_{i=1}^n \log \left[1 + \exp(-b_i s_{\mathbf{x}^{\natural}}(\mathbf{a}_i)) \right]
\end{aligned} \tag{21}$$

Hence, the maximum-likelihood (ML) estimator is

$$\mathbf{x}_{\text{ML}}^* \in \arg \min_{\mathbf{x}} \left\{ -\log p_{\mathbf{x}}(\mathbf{b}) = \sum_{i=1}^n \log \left[1 + \exp(-b_i s_{\mathbf{x}^{\natural}}(\mathbf{a}_i)) \right] : \mathbf{x} \in \mathbb{R}^p \right\}. \tag{22}$$

Note that the resulting parameter \mathbf{x}_{ML}^* defines a linear classifier: for a new feature \mathbf{a}_i , the predicted label is $b_i = 1$ if $\langle \mathbf{a}_i, \mathbf{x}_{\text{ML}}^* \rangle \geq 0$, and $b_i = -1$ otherwise. Thus, \mathbf{x}_{ML}^* specifies a hyperplane in \mathbb{R}^p whose two sides correspond to the different labels.

2.4 Poisson regression model

Poisson regression is a useful parametric model for count data. One example arises in photon-limited imaging systems, where the outputs b_1, \dots, b_n are non-negative integers. In confocal imaging, the vectors \mathbf{a}_i can be used to capture lens effects such as blur and spatial low-pass filtering (attributable to the lens's numerical aperture). We then model the photon count b_i as a Poisson random variable with mean $\langle \mathbf{a}_i, \mathbf{x}^{\natural} \rangle$, where $\mathbf{x}^{\natural} \in \mathbb{R}^p$ represents the unknown image [1, 9].

Model setup: Let $\mathbf{x}^{\natural} \in \mathbb{R}^p$ be an unknown parameter vector (i.e., the image). For $i = 1, \dots, n$, let B_i be independent Poisson random variables with mean $\langle \mathbf{a}_i, \mathbf{x}^{\natural} \rangle$. Given observed vectors $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$ and measurement outcomes b_1, \dots, b_n , our goal is to estimate \mathbf{x}^{\natural} .

The derivation: The probability mass function (pmf) of $p_{\mathbf{x}}(\cdot)$ is given by

$$\begin{aligned}
p_{\mathbf{x}}(\mathbf{b}) &= \prod_{i=1}^n \frac{\langle \mathbf{a}_i, \mathbf{x} \rangle^{b_i} \exp(-\langle \mathbf{a}_i, \mathbf{x} \rangle)}{b_i!} \\
-\log p_{\mathbf{x}}(\mathbf{b}) &= -\log \prod_{i=1}^n \frac{\langle \mathbf{a}_i, \mathbf{x} \rangle^{b_i} \exp(-\langle \mathbf{a}_i, \mathbf{x} \rangle)}{b_i!} \\
&= -\sum_{i=1}^n \left[\log \left(\langle \mathbf{a}_i, \mathbf{x} \rangle^{b_i} \exp(-\langle \mathbf{a}_i, \mathbf{x} \rangle) \right) - \log(b_i!) \right] \\
&= \sum_{i=1}^n \left[-b_i \log(\langle \mathbf{a}_i, \mathbf{x} \rangle) + \langle \mathbf{a}_i, \mathbf{x} \rangle + \log(b_i!) \right]
\end{aligned} \tag{23}$$

Consequently, the maximum-likelihood estimator is defined as

$$\mathbf{x}_{\text{ML}}^* \in \arg \min_{\mathbf{x}} \left\{ -\log p_{\mathbf{x}}(\mathbf{b}) = \sum_{i=1}^n \left[\log(b_i!) + \langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \log(\langle \mathbf{a}_i, \mathbf{x} \rangle) \right] : \mathbf{x} \in \mathbb{R}^p \right\} \tag{24}$$

Dropping the term $\sum_{i=1}^n \log(b_i!)$, which does not depend on \mathbf{x} , we obtain the maximum-likelihood (ML) estimator:

$$\mathbf{x}_{\text{ML}}^* \in \arg \min_{\mathbf{x}} \left\{ \sum_{i=1}^n \left[\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \log(\langle \mathbf{a}_i, \mathbf{x} \rangle) \right] : \mathbf{x} \in \mathbb{R}^p \right\} \tag{25}$$

2.5 Language model

In recent years, large language models (LLMs) have gained significant attention for their advanced capabilities and widespread availability. While LLMs are primarily built on neural network architectures, earlier natural language processing models, such as Markov Chains and Hidden Markov Models, followed a more statistical approach. Language models (LMs), which assign probabilities to word sequences, are central to this domain [5]. In this section, we frame language modeling within the context of maximum likelihood estimation.

To process a sentence, it is typically necessary to *tokenize* it. In English, each token approximately corresponds to each word, though exceptions exist; for example, “New York” is treated as a single token. In some tokenization schemes, sub-word tokenization is also used, where words are split into smaller units, such as prefixes, suffixes, or character-level segments (e.g., “playing” might be split into “play” and “ing”). In some languages, such as Chinese or Japanese, where there are no spaces between words, additional sentence segmentation is required for tokenization. Once tokenized, each token is represented using a vector called an *embedding*, which belongs to a predefined token set denoted by \mathcal{V} . We will revisit language models in later chapters.

Given a sentence with T words, $S = w_{1:T} = (w_1, \dots, w_T)$, the chain rule of probability can be applied as follows:

$$P(S) = P(w_{1:T}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \cdots P(w_T|w_{1:T-1}) = \prod_{t=1}^T P(w_t|w_{1:t-1}). \quad (26)$$

For example, if $S = w_{1:3} = \text{'happy new year'}$, then:

$$P(S) = P(\text{'happy'})P(\text{'new'}|\text{'happy'})P(\text{'year'}|\text{'happy new'}). \quad (27)$$

Language model as a maximum likelihood (ML) estimator: A language model can be considered an unsupervised ML estimator, formulated as:

$$\mathbf{x}_{\text{LM}}^* \in \arg \min_{\mathbf{x} \in \mathcal{X}} -\log p_{\mathbf{x}}(S) = -\log p_{\mathbf{x}}(\mathbf{b}_{1:T}), \quad (28)$$

where $\log p_{\mathbf{x}}(S)$ represents the probability mass function for the sentence S , and the embedding is given by $\mathbf{b}_{1:T} = (\mathbf{b}_1, \dots, \mathbf{b}_T)$.

Derivation: A neural network $\mathbf{h}_{\mathbf{x}}$ can model this probability as follows:

$$\begin{aligned} -\log p_{\mathbf{x}}(\mathbf{b}_{1:T}) &= -\log \left(\prod_{t=1}^T p_{\mathbf{x}}(\mathbf{b}_t | \mathbf{b}_{1:t-1}) \right) \\ &= \sum_{t=1}^T (-\log p_{\mathbf{x}}(\mathbf{b}_t | \mathbf{b}_{1:t-1})) \\ &= \sum_{t=1}^T (-\log [\mathbf{h}_{\mathbf{x}}(\mathbf{b}_{1:t-1})]^{\mathbf{b}_t}) \\ &= \text{cross-entropy loss.} \end{aligned} \quad (29)$$

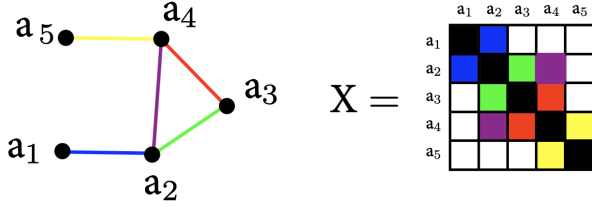
Given a sample in class $k \in [K]$, the probability for each of the K classes is represented as $\mathbf{h}_{\mathbf{x}} \in \mathbb{R}^K$. The cross-entropy loss for this case is defined as:

$$L = -\log \mathbf{h}_{\mathbf{x}}[k]. \quad (30)$$

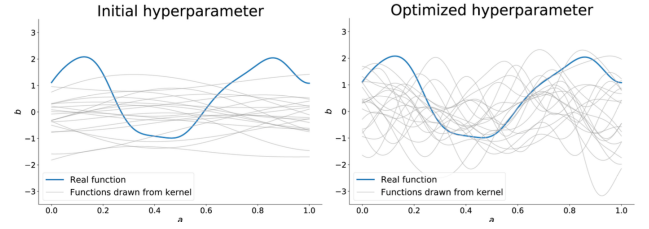
2.6 M -estimator: Graphical model learning

Let $\mathbf{X}^{\natural} \in \mathbb{S}_{++}^{p \times p}$ be a $p \times p$ positive-definite matrix that represents the graph structure as the precision matrix (Figure 4a). Suppose we are given data samples $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$, which are i.i.d. Gaussian random vectors with zero mean and covariance matrix $(\mathbf{X}^{\natural})^{-1}$, drawn from a Markov random field encoding conditional dependencies. The following M -estimator is commonly used for graphical model learning due to its good statistical properties [8]:

$$\mathbf{X}_M^* \in \arg \min_{\mathbf{X}} \left\{ \text{Tr}(\widehat{\Sigma} \mathbf{X}) - \log \det(\mathbf{X}) : \mathbf{X} \in \mathbb{S}_{++}^{p \times p} \right\}, \quad (31)$$



(a) In the given graph, edges determine the relationships between vertices. For example, given a_2 and a_4 , the probability of a_3 is independent of a_5 and a_1 . This structural information can be encoded in the precision matrix \mathbf{X} , which is the inverse of the covariance matrix Σ .



(b) A Gaussian process is a collection of random variables, any finite subset of which follows a joint Gaussian distribution, used to define distributions over functions for modeling and prediction.

Figure 4: Examples for graphical model selection and Gaussian processes

where $\widehat{\Sigma}$ is the empirical covariance matrix, defined as

$$\widehat{\Sigma} := \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^\top. \quad (32)$$

The derivation: The probability density function $p_{\mathbf{X}}(\cdot)$ is given by

$$\begin{aligned} p_{\mathbf{X}}(\mathbf{a}_1, \dots, \mathbf{a}_n) &= \prod_{i=1}^n \left[(2\pi)^{-\frac{p}{2}} \det(\mathbf{X}^{-1})^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i\right) \right] \\ &= (2\pi)^{-\frac{np}{2}} \det(\mathbf{X})^{\frac{n}{2}} \exp\left[-\frac{1}{2} \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i)\right] \\ -\log p_{\mathbf{X}}(\mathbf{a}_1, \dots, \mathbf{a}_n) &= -\log\left((2\pi)^{-\frac{np}{2}} \det(\mathbf{X})^{\frac{n}{2}} \exp\left[-\frac{1}{2} \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i)\right]\right) \\ &= -\log(2\pi)^{-\frac{np}{2}} - \log\left(\det(\mathbf{X})^{\frac{n}{2}}\right) - \log\left(\exp\left[-\frac{1}{2} \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i)\right]\right) \\ &= \frac{np}{2} \log(2\pi) - \frac{n}{2} \log \det(\mathbf{X}) + \frac{1}{2} \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i) \\ &= \frac{np}{2} \log(2\pi) - \frac{n}{2} \log \det(\mathbf{X}) + \frac{1}{2} \sum_{i=1}^n \text{Tr}(\mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i) \\ &= \frac{np}{2} \log(2\pi) - \frac{n}{2} \log \det(\mathbf{X}) + \frac{1}{2} \sum_{i=1}^n \text{Tr}(\mathbf{a}_i \mathbf{a}_i^\top \mathbf{X}) \\ &= \frac{np}{2} \log(2\pi) - \frac{n}{2} \log \det(\mathbf{X}) + \frac{n}{2} \text{Tr}\left(\left(\frac{1}{n} \sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^\top\right) \mathbf{X}\right) \\ &= \frac{np}{2} \log(2\pi) - \frac{n}{2} \log \det(\mathbf{X}) + \frac{n}{2} \left(\text{Tr}(\widehat{\Sigma} \mathbf{X})\right) \end{aligned} \quad (34)$$

Therefore, the ML estimator is defined as

$$\mathbf{X}_M^* \in \arg \min_{\mathbf{X}} \left\{ -\frac{n}{2} \log \det(\mathbf{X}) + \frac{n}{2} \left(\text{Tr}(\widehat{\Sigma} \mathbf{X})\right) : \mathbf{X} \in \mathbb{S}_{++}^{p \times p} \right\}, \quad (35)$$

which is equivalent to the M -estimator of \mathbf{X}_M^* . The M -estimator simplifies to the maximum likelihood (ML) estimator when the data samples \mathbf{a}_i are Gaussian random vectors.

2.7 M -estimator: Gaussian process regression

A Gaussian process (GP) is a *stochastic process*, denoted by:

$$f(\mathbf{a}) \sim \text{GP}(\boldsymbol{\mu}(\mathbf{a}), K(\mathbf{a}, \mathbf{a}')), \quad (36)$$

where $\boldsymbol{\mu}(\mathbf{a}) : \mathbb{R}^p \rightarrow \mathbb{R}$ is the mean function of the GP, and $K(\mathbf{a}, \mathbf{a}') : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ is the covariance or *kernel* function.

An M -estimator for kernel hyperparameter tuning involves finding the maximum-likelihood estimator given noisy targets $b_1, \dots, b_n \in \mathbb{R}$ and training data points $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$. The maximum-likelihood estimator, given the Gaussian process $\text{GP}(\boldsymbol{\mu}(\mathbf{a}), K_{\mathbf{X}}(\mathbf{a}, \mathbf{a}'))$ parametrized by $\mathbf{X} \in \mathbb{R}^m$, satisfies:

$$\mathbf{X}_M^* \in \arg \min_{\mathbf{X}} \left\{ \log \det(\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A})) + \frac{1}{n} \sum_{i=1}^n ((b_i - \mu(\mathbf{a}_i))^\top K_{\mathbf{X}}^{-1}(\mathbf{a}_i, \mathbf{a}_i)(b_i - \mu(\mathbf{a}_i))) \right\}, \quad (37)$$

where $\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A})$ is the kernel matrix with elements $[\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A})]_{ij} = K_{\mathbf{X}}(\mathbf{a}_i, \mathbf{a}_j)$, and $\mathbf{K}_{\mathbf{X}} \in \mathbb{S}_{++}^{n \times n}$ [7].

The derivation:

The probability density function $p_{\mathbf{X}}(\cdot)$ is given by

$$\begin{aligned} p_{\mathbf{X}}(b_1, \dots, b_n) &= \prod_{i=1}^n \left[(2\pi)^{-\frac{p}{2}} \det(\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A}))^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(b_i - \mu_i)^\top K_{i, \mathbf{X}}^{-1}(b_i - \mu_i)\right) \right] \\ &= (2\pi)^{-\frac{np}{2}} \det(\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A}))^{-\frac{n}{2}} \exp\left[-\frac{1}{2} \sum_{i=1}^n (b_i - \mu_i)^\top K_{i, \mathbf{X}}^{-1}(b_i - \mu_i)\right] \end{aligned} \quad (38)$$

where $\mu_i = \mu(\mathbf{a}_i)$ and $K_{i, \mathbf{X}}^{-1} = K_{\mathbf{X}}^{-1}(\mathbf{a}_i, \mathbf{a}_i)$ for brevity. Taking the logarithm, we have

$$-\log p_{\mathbf{X}}(b_1, \dots, b_n) = \underbrace{\frac{np}{2} \log(2\pi)}_{\text{constant}} + \underbrace{\frac{n}{2} \log \det(\mathbf{K}_{\mathbf{X}}(\mathbf{A}, \mathbf{A}))}_{\text{model complexity}} + \underbrace{\frac{1}{2} \sum_{i=1}^n (b_i - \mu_i)^\top K_{i, \mathbf{X}}^{-1}(b_i - \mu_i)}_{\text{mismatch between prior and data}} \quad (39)$$

which is equivalent to our estimator \mathbf{X}_M^* .

2.8 M -estimator: Density estimation

Density estimation is the task of estimating an underlying probability density function $p_{\mathbf{a}}$ from a set of observed data points $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^p$ drawn from that distribution. More formally: Given i.i.d. samples $\mathbf{a}_1, \dots, \mathbf{a}_n \sim p_{\mathbf{a}}$ from an unknown probability density function $p_{\mathbf{a}}$, the goal of density estimation is to learn an estimate $p_{\mathbf{X}}$ of $p_{\mathbf{a}}$ that minimizes some distance metric $d(p_{\mathbf{a}}, p_{\mathbf{X}})$ between the true and estimated distributions.

The distance metric $d(\cdot, \cdot)$ quantifies the dissimilarity between two distributions and could be any valid measure, such as the 1-Wasserstein distance. An M -estimator formulation of the density estimation problem is:

$$\mathbf{x}_M^* \in \arg \min_{\mathbf{x}} d(p_{\mathbf{a}}, p_{\mathbf{x}}) \quad (40)$$

However, since the true distribution $p_{\mathbf{a}}$ is unknown, it must be approximated using the empirical distribution of the observed samples:

$$\hat{p}_{\mathbf{a}} = \frac{1}{n} \sum_{i=1}^n \delta_{\mathbf{a}_i} \quad (41)$$

where $\delta_{\mathbf{a}_i}$ is the Dirac delta function centered at \mathbf{a}_i . Directly substituting the empirical distribution $\hat{p}_{\mathbf{a}}$ for $p_{\mathbf{a}}$ in the M -estimator can significantly alter the optimization problem and lead to overfitting, especially in high dimensions and with limited samples. Therefore, the key challenge in density estimation is to construct estimators that generalize well and provide reliable density estimates despite the finite sample approximation of the true underlying distribution.

2.9 M -estimator: Google PageRank

Google PageRank is an algorithm for ranking web pages based on their importance and relevance. It can be formulated as a least-squares optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})} \quad (42)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the PageRank vector to be estimated, with each element representing the importance score of a web page. The problem is set up as follows:

$$\mathbf{x} = \mathbf{r}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{r} \\ \frac{\gamma}{n} \mathbf{1} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{M} \\ \frac{\gamma}{2n} \mathbf{1} \mathbf{1}^\top \end{bmatrix}, \quad d = n \quad (43)$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the hyperlink matrix capturing the web graph structure, $\gamma \in (0, 1)$ is the teleportation probability, and $\mathbf{1}$ is the all-ones vector. This formulation can be viewed as a linear regression problem, where the goal is to estimate the true PageRank vector \mathbf{x}^\dagger given the hyperlink matrix \mathbf{A} and noisy observations:

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\dagger + \mathbf{w} \quad (44)$$

with \mathbf{w} representing unknown noise. The least-squares estimator minimizes the Euclidean norm of the residual $\mathbf{b} - \mathbf{A}\mathbf{x}^\dagger$, which corresponds to finding the PageRank vector \mathbf{x} that best fits the observed hyperlink structure and importance scores.

The PageRank optimization problem has a unique solution when the hyperlink matrix \mathbf{A} has full column rank. This solution can be computed efficiently using iterative methods such as the power iteration algorithm, which exploits the sparsity and structure of the web graph.

2.10 Generalized Linear Models

The ML estimators for Gaussian linear regression, logistic regression, and Poisson regression are all special cases of a broader class of models known as generalized linear models (GLMs). In this framework, the ML estimator can be expressed as the solution to the following optimization problem:

$$\mathbf{x}_{\text{ML}}^* \in \arg \min_{\mathbf{x} \in \mathbb{R}^p} \left\{ \frac{1}{n} \sum_{i=1}^n [\phi(\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \langle \mathbf{a}_i, \mathbf{x} \rangle)] \right\} \quad (45)$$

The function $\phi(\cdot)$ determines the specific type of GLM:

- $\phi(u) = u^2/2$ results in the ML estimator for linear regression
- $\phi(u) = \log(1 + \exp(u))$ results in the ML estimator for logistic regression
- $\phi(u) = \exp(u)$ results in the ML estimator for Poisson regression

Interestingly, the ML estimators for different GLMs are equivalent up to a scaling constant. This suggests that even under model misspecification, where the data is generated from a different GLM than the one being fit, the estimated parameters may still be meaningful [2]. To illustrate this, consider a dataset generated from a logistic regression model with true parameters \mathbf{x}^\dagger . If we compute the least squares M -estimator \mathbf{x}_{LS}^* for this data, which corresponds to a Gaussian linear regression model, and plot the coefficients of \mathbf{x}^\dagger against those of \mathbf{x}_{LS}^* , we observe a strong linear relationship in Figure 5a. This surprising result implies that the choice of GLM may not be as crucial as one might expect, and that even a misspecified model can yield useful insights about the relationship between the features and the response variable. However, it is still important to carefully consider the appropriate GLM for a given problem, as this can impact the interpretation and quality of the estimates.

3 Role of computation

The performance of an estimator \mathbf{x}^* , as measured by the squared error $\|\mathbf{x}^* - \mathbf{x}^{\natural}\|_2^2$ relative to the true parameters \mathbf{x}^{\natural} , is dependent on the number of available training samples n . As n increases, we generally expect the estimator's performance to improve, as it has access to more information for learning the underlying relationship between the features and the response.

While the squared error $\|\mathbf{x}^* - \mathbf{x}^{\natural}\|_2^2$ provides a useful measure of an estimator's performance, it alone is not sufficient for comprehensively evaluating the effectiveness of a learning machine. Other factors, such as computational efficiency, robustness to noise and outliers, and interpretability of the learned model, should also be considered when assessing the overall quality of a learning machine.

In practice, we can only numerically approximate the solution to the optimization problem that defines the estimator:

$$\mathbf{x}^* \in \arg \min_{\mathbf{x} \in \mathbb{R}^p} F(\mathbf{x}) \quad (46)$$

Various algorithms are employed to iteratively refine an estimate \mathbf{x}^t of the true solution \mathbf{x}^* at each time step t . The quality of this approximation depends on the specific algorithm used, as well as the computational resources allocated to the optimization process.

The practical performance of a learning machine at time t , using n training samples, can be decomposed into two components: the statistical error $\varepsilon(n)$ and the numerical error $\epsilon(t)$. This decomposition is expressed by the following inequality:

$$\underbrace{\|\mathbf{x}^t - \mathbf{x}^{\natural}\|_2^2}_{\bar{\varepsilon}(t,n)} \leq \underbrace{\|\mathbf{x}^t - \mathbf{x}^*\|_2^2}_{\epsilon(t)} + \underbrace{\|\mathbf{x}^* - \mathbf{x}^{\natural}\|_2^2}_{\varepsilon(n)} \quad (47)$$

Here, $\bar{\varepsilon}(t, n)$ represents the total error of the learning machine, which is bounded by the sum of the numerical error $\epsilon(t)$ and the statistical error $\varepsilon(n)$. The numerical error captures the discrepancy between the approximate solution \mathbf{x}^t and the true estimator \mathbf{x}^* , while the statistical error measures the inherent uncertainty in the estimator due to the finite sample size. Understanding and managing these sources of error is crucial for developing effective learning machines that can generalize well to unseen data and provide reliable predictions or insights in practical applications.

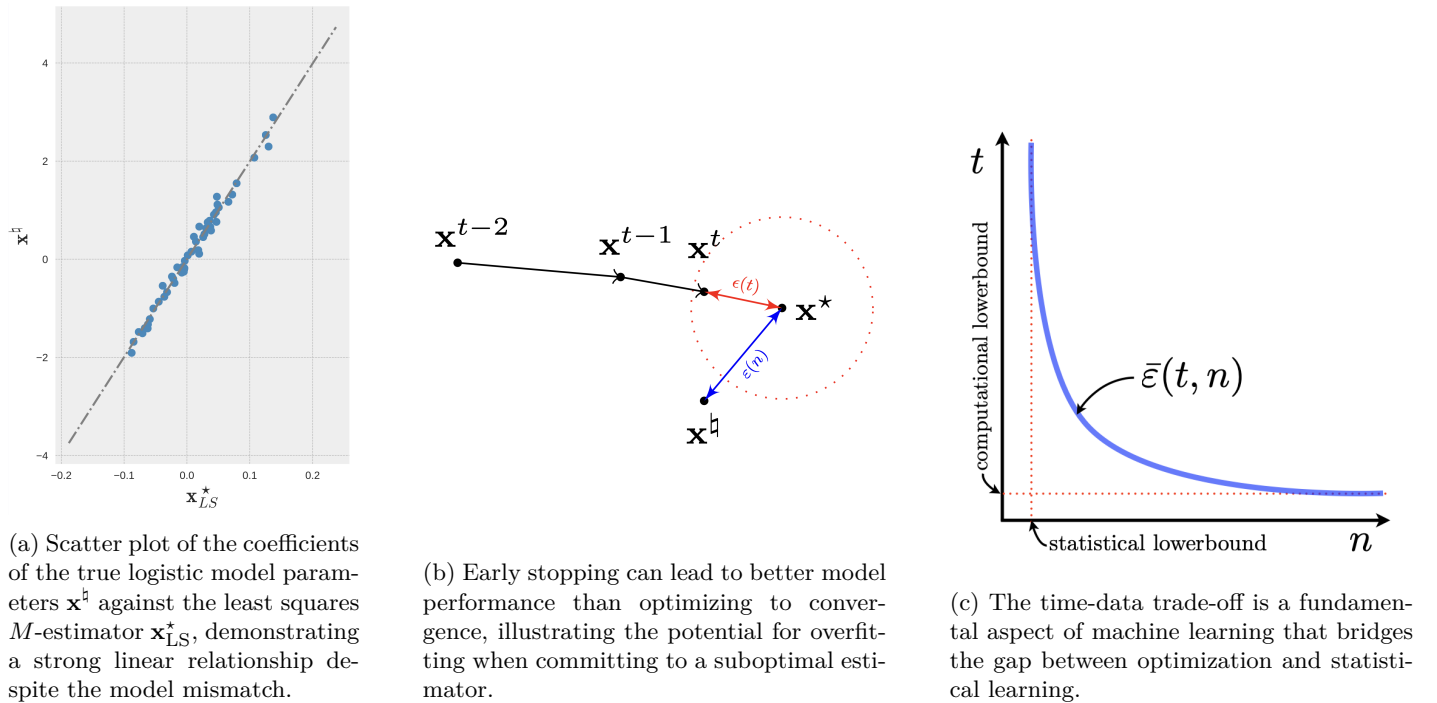


Figure 5: Insights into the interplay between optimization, statistical learning, and the time-data trade-off.

4 Performance of Learning Machines

In this section we discuss the performance of learning machines and the factors that influence their effectiveness. We introduce a general framework for analyzing the various sources of error in a learning machine and provide strategies for reducing these errors.

4.1 Model definitions

Let $d(\cdot, \cdot) : \mathcal{H}^\circ \times \mathcal{H}^\circ \rightarrow \mathbb{R}^+$ be a metric in an extended function space \mathcal{H}° that includes the assumed function class \mathcal{H} , i.e., $\mathcal{H} \subseteq \mathcal{H}^\circ$. We define the following models:

1. $h^\circ \in \mathcal{H}^\circ$: the true, expected risk minimizing model
2. $h^\natural \in \mathcal{H}$: the solution under the assumed function class \mathcal{H}
3. $h^* \in \mathcal{H}$: the estimator solution
4. $h^t \in \mathcal{H}$: the numerical approximation of the algorithm at time t

4.2 Decomposition of practical performance

The practical performance of a learning machine at time t , using n training samples, can be decomposed into three components: the optimization error, the statistical error, and the model error. This decomposition is expressed by the following inequality:

$$\underbrace{d(h^t, h^\circ)}_{\bar{\varepsilon}(t, n)} \leq \underbrace{d(h^t, h^*)}_{\text{optimization error}} + \underbrace{d(h^*, h^\natural)}_{\text{statistical error}} + \underbrace{d(h^\natural, h^\circ)}_{\text{model error}} \quad (48)$$

Here, $\bar{\varepsilon}(t, n)$ represents the total error of the learning machine. The optimization error captures the discrepancy between the approximate solution h^t and the true estimator h^* , the statistical error measures the inherent uncertainty in the estimator due to the finite sample size, and the model error quantifies the difference between the assumed function class and the true underlying model.

To reduce the total error, we can employ the following strategies:

1. Reduce the optimization error by allocating more computational resources to the optimization process.
2. Reduce the statistical error by increasing the number of training samples, using better estimators, or incorporating prior information.
3. Reduce the model error by using more flexible or universal function approximators.

4.3 General setting and performance measures

In the general setting, let $R(h_{\mathbf{x}}) = \mathbb{E}[L(h_{\mathbf{x}}(\mathbf{a}), b)]$ be the risk function and $R_n(h_{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^n L(h_{\mathbf{x}}(\mathbf{a}_i), b_i)$ be its empirical estimate, where L is a loss function. Let $\mathcal{X} \subseteq \mathcal{X}^\circ$ be parameter domains, with \mathcal{X} being known. We define the following models:

1. $\mathbf{x}^\circ \in \arg \min_{\mathbf{x} \in \mathcal{X}^\circ} R(h_{\mathbf{x}})$: true minimum risk model
2. $\mathbf{x}^\natural \in \arg \min_{\mathbf{x} \in \mathcal{X}} R(h_{\mathbf{x}})$: assumed minimum risk model
3. $\mathbf{x}^* \in \arg \min_{\mathbf{x} \in \mathcal{X}} R_n(h_{\mathbf{x}})$: empirical risk minimization (ERM) solution
4. \mathbf{x}^t : numerical approximation of \mathbf{x}^* at time t

Several performance measures can be defined in this context:

1. $R_n(\cdot)$: training error

2. $R(\cdot)$: test error
3. $R(\mathbf{x}^\natural) - R(\mathbf{x}^\circ)$: modeling error
4. $R(\mathbf{x}^*) - R(\mathbf{x}^\natural)$: excess risk
5. $\sup_{\mathbf{x} \in \mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})|$: generalization error
6. $R(\mathbf{x}^t) - R_n(\mathbf{x}^*)$: optimization error

The practical performance can be decomposed as follows:

$$\underbrace{R(\mathbf{x}^t) - R(\mathbf{x}^\circ)}_{\bar{\varepsilon}(t,n)} \geq \underbrace{R_n(\mathbf{x}^t) - R_n(\mathbf{x}^*)}_{\text{optimization error}} + 2 \underbrace{\sup_{\mathbf{x} \in \mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})|}_{\text{generalization error}} + \underbrace{R(\mathbf{x}^\natural) - R(\mathbf{x}^\circ)}_{\text{model error}} \quad (49)$$

As before, the total error $\bar{\varepsilon}(t, n)$ can be reduced by targeting the optimization error, generalization error, and model error.

4.4 Example: Hinge loss and bounded features

Consider a linear model $h_{\mathbf{x}} : \mathbb{R}^p \rightarrow \mathbb{R}$, $h_{\mathbf{x}}(\mathbf{a}) = \mathbf{x}^\top \mathbf{a}$, with the hinge loss $L(h_{\mathbf{x}}(\mathbf{a}), b) = \max(0, 1 - b \cdot \mathbf{x}^\top \mathbf{a})$. Let the parameter domain be $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^p : \|\mathbf{x}\| \leq \lambda\}$, and assume that the features are bounded, i.e., $\|\mathbf{a}\| \leq \sqrt{p}$ almost surely.

Under these conditions, with high probability, the generalization error can be bounded as:

$$\sup_{\mathbf{x} \in \mathcal{X}} |R(\mathbf{x}) - R_n(\mathbf{x})| = \mathcal{O} \left(\lambda \sqrt{\frac{p}{n}} \right) \quad (50)$$

This bound suggests that the generalization error can be reduced by increasing the number of training samples n , reducing the feature dimensionality p , or constraining the parameter norm λ [6].

5 Time-data trade-off

Optimization and statistical learning are two fundamental aspects of machine learning that are often studied independently. However, their goals are tightly connected, and understanding their interplay is crucial for developing efficient and effective learning algorithms. Optimization and statistics have traditionally focused on different objectives:

- In optimization, the goal is to reach a numerical ϵ -accuracy, where the metric is $\|\mathbf{x}^k - \mathbf{x}^*\| \leq \epsilon$. Here, \mathbf{x}^k is the solution at iteration k , and \mathbf{x}^* is the optimal solution.
- In statistics, the task is to learn an ε -accurate model, where the metric is $\|\mathbf{x}^* - \mathbf{x}^\natural\| \leq \varepsilon$. Here, \mathbf{x}^\natural is the true model, and \mathbf{x}^* is the estimator.

The main issue is that ϵ and ε are not the same but should be treated jointly. Focusing solely on one aspect can lead to suboptimal performance in the other.

5.1 Data as a computational resource

A common computational dogma is that the running time of a learning algorithm increases with the size of the data. However, this view overlooks the potential of using data as a computational resource. By carefully managing the trade-off between computation time and sample size, we can develop more efficient learning algorithms.

The time-data trade-off can be formalized as follows:

$$\left\| \mathbf{x}^{k(t)} - \mathbf{x}^\natural \right\| \leq \underbrace{\left\| \mathbf{x}^{k(t)} - \mathbf{x}^* \right\|}_{\varepsilon: \text{ needs "time" } t} + \underbrace{\left\| \mathbf{x}^* - \mathbf{x}^\natural \right\|}_{\varepsilon: \text{ needs "data" } n} \quad (51)$$

Here, $\mathbf{x}^{k(t)}$ is the solution at time t , \mathbf{x}^* is the estimator, \mathbf{x}^\natural is the true model, and $\bar{\varepsilon}(t, n)$ is the actual model precision at time t with n samples.

This formalization highlights the interplay between optimization error (ϵ) and statistical error (ε). To achieve a desired level of accuracy, we can either invest more time in optimization or use more data to improve the statistical properties of the estimator, which is depicted in Figure 5c.

5.2 Impact of model complexity and sample size

Table 1 summarizes the impact of various factors on the performance of learning algorithms. Notably:

- Increasing the model complexity ($\mathcal{X} \rightarrow \mathcal{X}^\circ$) reduces the training error and modeling error but increases the excess risk, generalization error, and computation time.
- Increasing the sample size ($n \uparrow$) reduces the excess risk and generalization error but increases the training error and computation time.
- Increasing the feature dimensionality ($p \uparrow$) reduces the training error but increases the excess risk, generalization error, and computation time. Its effect on the modeling error depends on the specific problem.

	$\mathcal{X} \rightarrow \mathcal{X}^\circ$	$n \uparrow$	$p \uparrow$
Training error	\searrow	\nearrow	\searrow
Excess risk	\nearrow	\searrow	\nearrow
Generalization error	\nearrow	\searrow	\nearrow
Modeling error	\searrow	=	\leftrightarrow
Time	\nearrow	\nearrow	\nearrow

Table 1: Impact of model complexity, sample size, and feature dimensionality on various performance measures.

References

- [1] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Vol. 23. Cambridge University Press, Oct. 2006. ISBN: 9780521175180. DOI: 10.1017/cbo9780511546860. URL: <http://dx.doi.org/10.1017/CB09780511546860>.
- [2] Murat A Erdogdu, Lee H Dicker, and Mohsen Bayati. “Scaled Least Squares Estimator for GLMs in Large-Scale Problems”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/e1696007be4eefb81b1a1d39ce48681b-Paper.pdf.
- [3] Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, Jan. 2009. ISBN: 9780470434697. DOI: 10.1002/9780470434697. URL: <http://dx.doi.org/10.1002/9780470434697>.
- [4] Michael I. Jordan. “Why the logistic function? A tutorial discussion on probabilities and neural networks”. In: 1995. URL: <https://api.semanticscholar.org/CorpusID:14540707>.
- [5] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Third Edition draft. 2024. URL: <https://web.stanford.edu/~jurafsky/slp3/>.
- [6] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. English (US). MIT Press, 2012.
- [7] Carl Edward Rasmussen. “Gaussian Processes in Machine Learning”. In: *Advanced Lectures on Machine Learning*. Springer Berlin Heidelberg, 2004, pp. 63–71. ISBN: 9783540286509. DOI: 10.1007/978-3-540-28650-9_4. URL: http://dx.doi.org/10.1007/978-3-540-28650-9_4.
- [8] Pradeep Ravikumar et al. “High-dimensional covariance estimation by minimizing l_1 -penalized log-determinant divergence”. In: *Electronic Journal of Statistics* 5.none (Jan. 2011). ISSN: 1935-7524. DOI: 10.1214/11-ejs631. URL: <http://dx.doi.org/10.1214/11-EJS631>.

- [9] G. M. P. VAN KEMPEN et al. “A quantitative comparison of image restoration methods for confocal microscopy”. In: *Journal of Microscopy* 185.3 (Mar. 1997), pp. 354–365. ISSN: 1365-2818. DOI: 10.1046/j.1365-2818.1997.d01-629.x. URL: <http://dx.doi.org/10.1046/j.1365-2818.1997.d01-629.x>.
- [10] Zhenyu Zhu, Francesco Locatello, and Volkan Cevher. *Sample Complexity Bounds for Score-Matching: Causal Discovery and Generative Modeling*. 2023. DOI: 10.48550/ARXIV.2310.18123. URL: <https://arxiv.org/abs/2310.18123>.