# Fundamentals: Definition, Notation, and Principles

Lecturer: Qiang Sun — Lecture 01
January, 2026

## 1 Welcome

Welcome to STAD80, Analysis of Big Data or ABD for short. This course is a topic course, whose contents evolve from year to year. The course usually consists of both theoretical and computational components. This year, we will focus on three pillars:

1. fundamentals such as statisticasl principles and models;
2. optimization;
3. generative modeling and learning.

## 2 Overview

In recent years, artificial intelligence (AI) has undergone a dramatic shift. Beyond predicting outcomes (e.g., classification or regression), many modern AI systems can now *generate* new content—images, text, audio, and video—conditioned on a prompt. This distinction matters: traditional predictive models aim to estimate an unknown target from observed data, whereas generative models aim to produce realistic *samples* that resemble draws from a complex data distribution. The ability to generate, not just predict, is a defining feature of the current Generative AI (GenAI) revolution. For example, image generators such as Stable Diffusion can create images in many styles, video models can synthesize short clips, and large language models such as ChatGPT can produce fluent responses to text prompts.

This shift toward generative AI is tightly linked to big data and computation. Learning to generate realistic samples requires approximating extremely complex distributions over high-dimensional objects (images, text, audio, video). Doing so reliably typically demands (i) massive training datasets and (ii) large-scale optimization. In other words, GenAI is not only a modeling breakthrough; it is also a *computational* breakthrough, enabled by modern hardware (GPUs/TPUs), distributed systems, and scalable stochastic optimization algorithms. From a data science perspective, GenAI sits at the intersection of data engineering (collecting and curating large multimodal datasets), statistical modeling (specifying probabilistic objectives), and optimization (training models efficiently at scale).

More broadly, the move from prediction to generation often coincides with a move from "small data" to "big data" regimes. Predictive models can sometimes succeed with modest sample sizes and carefully chosen features; generative models must capture much richer structure and variability, which typically requires orders of magnitude more data. Moreover, because training objectives involve expectations over high-dimensional distributions, learning relies on Monte Carlo approximations via minibatches and repeated passes through data. As a result, the success of generative AI is inseparable from scalable data pipelines and efficient numerical optimization.

### 2.1 Generative Modeling as Sampling

We begin by thinking about different data types, or *modalities*, and how to represent them numerically. The key point is that, once we choose a representation, each data object can be viewed as a point in a (typically high-dimensional) vector space.

1. **Image.** Consider an image with height $H$ and width $W$ pixels, with three color channels (RGB). Each pixel-channel entry is an intensity value (often scaled to $[0,1]$ or $[0,255]$). A convenient representation is a tensor
$$Z \in \mathbb{R}^{H\times W\times 3}.$$
2. **Video.** A video is a sequence of images over time. If the video has $T$ frames, then a natural representation is
$$Z \in \mathbb{R}^{T\times H\times W\times 3}.$$
3. **Molecular structure.** A simple representation of a molecule with $N$ atoms is to record the 3D coordinates of each atom:
$$Z = (Z^1, \dots, Z^N) \in \mathbb{R}^{3\times N}, \qquad Z^i \in \mathbb{R}^3.$$
(In practice, more structured representations are often used, such as graphs with node/edge features, which better encode chemical constraints and symmetries.)

In all of these examples, the object we wish to model or generate can be encoded as a vector after *flattening* (i.e., stacking entries into a single column). Thus, throughout this document we will treat data as vectors:
$$Z \in \mathbb{R}^d,$$
where $d$ may be extremely large (e.g., $d = H \cdot W \cdot 3$ for images and $d = T \cdot H \cdot W \cdot 3$ for videos). A *generative model* specifies a probability distribution over such objects and allows us to *sample* new synthetic data points that resemble the observed data.

## 3 Fundamentals

### 3.1 Concepts and Notations

We begin with fundamental concepts and notation.

**Definition 3.1 (Random Samples)**

We call random variables $X_1, \dots, X_n$ a *random sample* if they are independent and identically distributed (i.i.d.) from some distribution with density (or mass) function $p(x)$:
$$X_1, \dots, X_n \overset{\text{i.i.d.}}{\sim} p(x).$$

Throughout the course, capital letters denote random variables and lower-case letters denote observed values. We also write $X_{1:n} = (X_1, \dots, X_n)$ and $x_{1:n} = (x_1, \dots, x_n)$.

**Definition 3.2 (Realization / Observed Values)**
The observed values $x_1, \ldots, x_n$ are called *realizations* of the random variables $X_1, \ldots, X_n$.

**Definition 3.3 (Statistics)**
A statistic is any measurable function of the sample $X_{1:n}$.

**Definition 3.4 (Cumulative Distribution Function (CDF))**
The CDF of a random variable $X$ is

$$F(x) = \mathbb{P}(X \leq x), \qquad x \in \mathbb{R}.$$

**Definition 3.5 (Probability Density Function (PDF))**
If $X$ is absolutely continuous and $F$ is differentiable, its density is

$$p(x) = \frac{d}{dx}F(x), \qquad x \in \mathbb{R}.$$

**Remark 3.1**
For discrete $X$, $p(x)$ typically denotes the probability mass function (pmf). For a parametric family we write $p_\theta(x)$, where $\theta \in \Theta$ indexes the distribution.

**Definition 3.6 (Statistical Model)**
A statistical model is a family of distributions indexed by a parameter set $\Theta$:

$$\mathcal{P} = \{p_\theta : \theta \in \Theta\}.$$

**Definition 3.7 (Parametric Model)**
If $\Theta$ is finite-dimensional (e.g., $\Theta \subseteq \mathbb{R}^d$ for some finite $d$), then $\mathcal{P}$ is called *parametric*.

**Definition 3.8 (Nonparametric Model)**
If no finite-dimensional parameterization is available, then $\mathcal{P}$ is called *nonparametric*.

**Example 3.1 (Parametric Model)**
The Gaussian family

$$\mathcal{P} = \left\{ p_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(x-\mu)^2}{2\sigma^2} \right) : \ \mu \in \mathbb{R}, \ \sigma^2 > 0 \right\}$$

is parametric with $\theta = (\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}^+$.

**Example 3.2 (Nonparametric Model)**
Let

$$\mathcal{P} = \{\text{all PDFs on } \mathbb{R}\}.$$

This model is nonparametric: it cannot be indexed by finitely many real parameters.

In what follows, we will focus primarily on parametric models, since they provide a clean and accessible starting point. Later in the course, we will return to nonparametric models, which offer greater flexibility but typically require more advanced tools.

## 3.2 Statistical Estimators

**Definition 3.9 (Point Estimation)**
Given $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} p_\theta(x)$, point estimation means producing a single value intended to approximate $\theta$.

**Definition 3.10 (Estimator)**
An estimator is a rule (a statistic) for estimating $\theta$ from data:

$$\widehat{\theta}_n = g(X_1, \ldots, X_n),$$

where $g : \mathbb{R}^n \to \Theta$ is measurable.

**Definition 3.11 (Consistent Estimator)**
$\widehat{\theta}_n$ is consistent for $\theta$ if $\widehat{\theta}_n \xrightarrow{P} \theta$ as $n \to \infty$.

**Definition 3.12 (Bias)**
The bias of $\widehat{\theta}_n$ is

$$\operatorname{Bias}(\widehat{\theta}_n) = \mathbb{E}[\widehat{\theta}_n] - \theta,$$

where the expectation is under the joint distribution of $X_{1:n}$.

**Definition 3.13 (Unbiased Estimator)**
$\widehat{\theta}_n$ is unbiased if $\operatorname{Bias}(\widehat{\theta}_n) = 0$.

Unbiasedness and consistency are different properties, and neither implies the other.

**Example 3.3**
Suppose $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} N(\mu, 1)$.

(a) $\widehat{\mu}^{(1)} = X_1$ is unbiased but not consistent (it does not use more information as $n$ grows).

(b) $\widehat{\mu}_n^{(2)} = \frac{1}{n}\sum_{i=1}^n X_i$ is unbiased and consistent.

(c) $\widehat{\mu}_n^{(3)} = \frac{1}{n+1}\sum_{i=1}^n X_i$ is biased but consistent, since $\mathbb{E}[\widehat{\mu}_n^{(3)}] = \frac{n}{n+1}\mu \to \mu$ and $\widehat{\mu}_n^{(3)} = \frac{n}{n+1}\widehat{\mu}_n^{(2)} \xrightarrow{P} \mu$.

## 3.3 Maximum Likelihood Estimators

Maximum likelihood is a general-purpose method for estimating parameters in a chosen model.

**Definition 3.14 (Likelihood (Single Observation))**
Given a model $p_\theta(\cdot)$ and an observed value $x$, the likelihood is the function of $\theta$

$$L(\theta; x) = p_\theta(x).$$

**Definition 3.15 (Joint Log-Likelihood)**
For observed data $x_{1:n}$, the joint likelihood is

$$L_n(\theta; x_{1:n}) = p_\theta(x_{1:n}) = \prod_{i=1}^{n} p_\theta(x_i),$$

where the last equality uses independence. The joint likelihood is

$$\ell_n(\theta; x_{1:n}) = \log L_n(\theta; x_{1:n}) = \sum_{i=1}^{n} \log p_\theta(x_i).$$

**Definition 3.16 (Maximum Likelihood Estimator (MLE))**
An estimator $\widehat{\theta}_n$ is an MLE if it maximizes the likelihood:

$$\widehat{\theta}_n \in \arg\max_{\theta \in \Theta} L_n(\theta; x_{1:n}) \qquad \text{equivalently} \qquad \widehat{\theta}_n \in \arg\max_{\theta \in \Theta} \ell_n(\theta; x_{1:n}).$$

**Example 3.4 (Gaussian Distribution)**
If $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} N(\mu, \sigma^2)$, then the MLEs are

$$\widehat{\mu}_n = \bar{X}_n, \qquad \widehat{\sigma}_n^2 = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X}_n)^2.$$

Why is the MLE important? It provides a unified framework for estimation and often has strong large-sample properties when the model is correctly specified.

**Theorem 3.17 (Asymptotic Normality of the MLE (Informal))**
Under regularity conditions,

$$\sqrt{n}\,(\widehat{\theta}_n - \theta) \ \xrightarrow{D}\ N\big(0,\, I(\theta)^{-1}\big),$$

where $I(\theta)$ is the Fisher information (per observation). For a scalar parameter,

$$I(\theta) = \mathbb{E}_\theta\left[\left(\frac{\partial}{\partial\theta} \log p_\theta(X)\right)^2\right] = -\mathbb{E}_\theta\left[\frac{\partial^2}{\partial\theta^2} \log p_\theta(X)\right].$$

Moreover, under conditions, unbiased estimators satisfy a Cramér–Rao type lower bound $\mathrm{Var}(\widetilde{\theta}_n) \geq 1/(nI(\theta))$.

## 3.4 Generative Models: Generation as Sampling

Let us make precise what it means to "generate" an object. Suppose, for example, that we want to generate an image of a dog. There is no single "best" dog image; instead, there are many acceptable images, with varying degrees of realism and diversity. A convenient way to formalize this idea is to posit an (unknown) data distribution, or a probability density function, over the space of objects. We denote this density function by $p_{\text{data}}$.

Intuitively, $p_{\text{data}}$ assigns higher probability to objects that look more like valid data (e.g., images that look like dogs) and lower probability to implausible objects. With this viewpoint, the vague question "How good is this generated sample?" is replaced by the more precise question "How likely is it under $p_{\text{data}}$?" Therefore, we can use the maximum likelihood estiamtors as introduced in the previous subsection to estimate this data distribution.

**Idea 3.1 (Generation as Sampling)**
Generating an object $Z$ can be modeled as sampling from the (unknown) data distribution:

$$Z \sim p_{\text{data}}.$$

A *generative model* is a machine learning model that aims to approximate $p_{\text{data}}$ well enough that we can generate realistic samples. Of course, we do not observe $p_{\text{data}}$ directly. Instead, we observe a finite dataset, which we typically idealize as i.i.d. samples from $p_{\text{data}}$.

**Idea 3.2 (Dataset)**
A dataset consists of a finite collection of samples

$$Z_1, \ldots, Z_n \overset{\text{i.i.d.}}{\sim} p_{\text{data}}.$$

In other words, a dataset consists of random samples following $p_{\text{data}}$.

For images, such a dataset might be constructed from large collections of publicly available photos. For videos, one might use curated video repositories. For molecular structures (or proteins), one may use experimentally measured databases such as the Protein Data Bank (PDB). As $n$ grows, the empirical dataset typically becomes a better approximation of the underlying distribution $p_{\text{data}}$.

### 3.4.1 Conditional Generation

In many applications, we want to generate an object *conditioned* on some input $y$. For example, we might want an image conditioned on a text prompt, or a molecular structure conditioned on desired chemical properties. This can be formalized through a conditional distribution.

**Idea 3.3 (Conditional Generation)**

Conditional generation is modeled as sampling

$$Z \sim p_{\mathrm{data}}(\,\cdot \mid y),$$

where $y$ is a conditioning variable (e.g., a class label, a text prompt, or side information).

We call $p_{\mathrm{data}}(\cdot \mid y)$ the *conditional data distribution.* In practice, the goal is often to learn a *single* generative model that can condition on many possible values of $y$ (for example, many different text prompts), rather than training a separate model for each fixed choice of $y$. Many techniques for unconditional generation extend naturally to the conditional setting; therefore, we will often introduce ideas first in the unconditional case and then discuss how they generalize.

**From Noise to Data.** So far, we have discussed the what of generative modeling: generating samples from $p_{\mathrm{data}}$. Here, we will briefly discuss the how. For this, we assume that we have access to some initial distribution $p_{\mathrm{init}}$ that we can easily sample from, such as the Gaussian $p_{\mathrm{init}} = \mathcal{N}(0, I_d)$. The goal of generative modeling is then to transform samples from $x \sim p_{\mathrm{init}}$ into samples from $p_{\mathrm{data}}$. We note that $p_{\mathrm{init}}$ does not have to be so simple as a Gaussian. As we shall see, there are interesting use cases for leveraging this flexibility. Despite this, in the majority of applications we take it to be a simple Gaussian and it is important to keep that in mind.