PROBABILITY & STATISTICS

FROM BEGINNER TO EXPERT

A complete, visual, application-driven course — from sample spaces and Bayes' theorem through random variables, limit theorems, and statistical inference, with a capstone on how probability and statistics power Machine Learning, Deep Learning, and AI.

What you will be able to do

Reason quantitatively about uncertainty; model data with the right distribution; estimate parameters by maximum likelihood and Bayesian inference; test hypotheses and quantify confidence; and explain why MLE, cross-entropy, regularization, and uncertainty estimates are the statistical backbone of modern AI.

Format: self-paced reference & course | **Prerequisites:** basic calculus (integrals) and a little linear algebra

Part of the Comprehensive Machine Learning & Deep Learning Curriculum.

Preface: how to use this book

If linear algebra is the language of machine learning and calculus is its engine, then probability and statistics are its *worldview*. Machine learning is, at its core, the discipline of drawing reliable conclusions from noisy, incomplete data — which is exactly what statistics was invented to do. Every prediction a model makes is a probabilistic statement; every loss function it minimizes is, on closer inspection, a likelihood; every claim that a model "generalizes" is a statistical claim about unseen data. This book builds that worldview from first principles and carries it to the frontier of modern AI.

Who this is for

This book starts from the very beginning — what a probability *is* — and climbs to maximum likelihood, Bayesian inference, Markov chain Monte Carlo, and information theory. It is written for the self-learner who wants both **fluency** ("I can compute the posterior, run the test, fit the model") and **understanding** ("I know what these numbers mean and when they lie"). Every major idea is motivated with a picture, a worked example, and a forward link to how it is used in machine learning.

The two halves

The subject has two complementary halves, and you need both:

- **Probability** (Parts I–III): given a model of the world, what data should we expect? This is the *forward* direction — from assumptions to predictions.
- **Statistics** (Parts IV–V): given the data, what model of the world is plausible? This is the *inverse* direction — from observations back to the underlying truth. It is the direction machine learning lives in.

The structure

- **Part I — Foundations of Probability** (beginner): the rules of probability; conditioning and Bayes' theorem.
- **Part II — Random Variables & Distributions** (core): random variables and expectation; the common distributions; joint distributions and the multivariate Gaussian.
- **Part III — Limit Theorems** (core): inequalities, the law of large numbers, and the central limit theorem.
- **Part IV — Statistical Inference** (advanced): estimation and maximum likelihood; confidence intervals and hypothesis testing; Bayesian inference; regression.
- **Part V — Advanced & Computational**: stochastic processes, Markov chains and Monte Carlo; information theory; and a capstone on probability and statistics in ML/DL/AI.

How to read it

Read with a pen and a computer. When you meet a *Definition*, restate it; when you meet an *Example*, try it before reading the solution; when you meet a *Try it in code* box, run it. The colored boxes recur throughout:

- **Key idea** — the one sentence to remember.

- **Intuition** — the picture behind the formula.
- **Common pitfall** — the mistakes that bite everyone.
- **How this powers ML / AI** — the forward link to applications.
- **Try it in code** — a hands-on check in Python/NumPy/SciPy.

Key idea

Hold one idea above all others: **learning from data is inference under uncertainty**. A model never *knows* the truth; it maintains beliefs and updates them as evidence arrives. Probability is the calculus of those beliefs, and statistics is the practice of forming them from data. Master this and the entire training loop of any model — from a coin-flip estimator to a diffusion transformer — becomes one coherent story.

Contents

PART I

Foundations of Probability

1

What Is Probability?

Level: Beginner

Probability is the mathematics of uncertainty — a precise language for reasoning about things we cannot predict with certainty. A coin flip, tomorrow's weather, whether an email is spam, what word comes next in a sentence: all are uncertain, and probability lets us quantify that uncertainty and compute with it. This chapter lays the foundation: the vocabulary of outcomes and events, the three axioms that govern all of probability, the art of counting, and the two great schools of thought about what a probability *means*.

1.1 Sample spaces, outcomes, and events

Definition 1.1. A **random experiment** is any process with an uncertain outcome. The **sample space** Ω is the set of all possible outcomes. An **event** A is a subset of Ω — a collection of outcomes we care about.

Example 1.2. Roll a six-sided die. The sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. The event "even" is $A = \{2, 4, 6\}$; the event "greater than 4" is $B = \{5, 6\}$. Events combine with set operations: $A \cup B$ ("A or B"), $A \cap B$ ("A and B"), and A^c ("not A").

Intuition

Think of Ω as the space of everything that could happen, and an event as a region carved out of it. Asking "what is the probability of A?" is asking "how much of the total possibility lands in region A?" All the logic of probability is the logic of sets — unions, intersections, complements — with a number attached that measures size.

1.2 The axioms of probability

Modern probability rests on three rules, due to Kolmogorov (1933). Everything else is derived from them.

Definition 1.3 (Kolmogorov's axioms)**.** A **probability** P assigns to each event a number such that

1. **Nonnegativity:** $P(A) \geq 0$ for every event A.
2. **Normalization:** $P(\Omega) = 1$ — something must happen.
3. **Additivity:** if $A_1, A_2, \ldots$ are mutually exclusive (disjoint), then $P(\bigcup_i A_i) = \sum_i P(A_i)$.

From these follow all the familiar rules:

$$P(A^c) = 1 - P(A), \qquad P(\varnothing) = 0, \qquad P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Common pitfall

The last rule — **inclusion–exclusion** — is where beginners stumble: you may only add probabilities directly when events are *disjoint*. For overlapping events you must subtract the double-counted intersection. Forgetting this is the single most common probability error, and it reappears in subtle forms (e.g. assuming features are independent when they are not).

1.3 Equally likely outcomes and counting

When a finite sample space has N equally likely outcomes, probability reduces to counting:

$$P(A) = \frac{\text{number of outcomes in } A}{\text{total number of outcomes}} = \frac{|A|}{|\Omega|}.$$

So we need to count well. The two workhorses are **permutations** (ordered arrangements) and **combinations** (unordered selections):

$$\text{permutations: } \frac{n!}{(n-k)!}, \qquad \text{combinations: } \binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

Example 1.4. A standard deck has $\binom{52}{5} = 2{,}598{,}960$ possible five-card poker hands. The probability of a specific hand is one in that many. Counting the hands of a given type (e.g. a flush) and dividing gives its probability.

Intuition

Combinations $\binom{n}{k}$ count "how many ways to choose k things from n when order doesn't matter." They appear far beyond card games: the binomial distribution (Chapter 4) counts the ways a fixed number of successes can occur, and the same coefficients show up whenever we sum over subsets — including the combinatorial terms in model-counting and ensemble methods.

1.4 What does a probability mean? Two interpretations

The axioms tell us how probabilities *behave*, but not what they *mean*. Two schools answer differently — and the split runs straight through machine learning.

Key idea

Frequentist: a probability is a long-run frequency. "$P(\text{heads}) = 0.5$" means that in many repeated flips, the fraction of heads approaches 0.5. Probability is a property of the world, revealed by repetition.
Bayesian: a probability is a degree of belief. "$P(\text{rain tomorrow}) = 0.3$" is a coherent statement of confidence given current information — even though tomorrow happens only once. Probability is a property of the *observer's knowledge*, updated as evidence arrives.

Intuition

Neither view is "correct"; they are different tools. Frequentist reasoning underlies hypothesis tests and confidence intervals (Chapter 8); Bayesian reasoning underlies posterior updating and most modern probabilistic ML (Chapter 9). Deep learning quietly uses both: it estimates parameters by (frequentist) maximum likelihood, yet interprets a softmax output as a (Bayesian-flavored) degree of belief over classes. You will become fluent in both dialects.

How this powers ML / AI

Every prediction a model makes is a probability. A classifier does not output "cat"; it outputs a distribution like $P(\text{cat}) = 0.83$, $P(\text{dog}) = 0.12, \ldots$ obeying exactly the axioms above (nonnegative, summing to one). The sample space is the set of classes; the softmax layer is a machine for producing a valid probability over it. Generative models go further and place a probability over *all possible images or sentences*. Getting the foundations right — what is an event, when may probabilities be added, what does the number mean — is what separates a model that reasons soundly about uncertainty from one that fools itself.

Try it in NumPy

```python
import numpy as np
from math import comb
# Frequentist view: long-run frequency approaches the probability
flips = np.random.randint(0, 2, size=1_000_000)   # 0/1 = tails/heads
print(flips.mean())                                # -> ~0.5

# Counting: probability of being dealt a flush-shaped 5-card combo
total = comb(52, 5)
print(total)                                       # 2,598,960
# inclusion-exclusion: P(A or B) with overlap
PA, PB, PAB = 0.5, 0.4, 0.2
print(PA + PB - PAB)                               # 0.7, NOT 0.9
```

Chapter summary

- The **sample space** Ω holds all outcomes; an **event** is a subset; probability measures the "size" of events.
- **Kolmogorov's axioms** (nonnegativity, normalization, additivity for disjoint events) generate all the rules.
- Add probabilities only for **disjoint** events; otherwise use $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
- For equally likely outcomes, probability is **counting**: permutations (ordered) and combinations $\binom{n}{k}$ (unordered).
- **Frequentist** (long-run frequency) and **Bayesian** (degree of belief) interpretations both run through machine learning.

2

Conditional Probability and Bayes' Theorem

Level: Beginner

Almost all useful probability is *conditional*: we want to know the chance of something *given* what we already know. How likely is disease given a positive test? Spam given the word "free"? The next word given the sentence so far? Conditioning is how probability incorporates evidence, and **Bayes' theorem** is the engine that reverses it — turning "probability of evidence given cause" into "probability of cause given evidence." This is the single most important chapter for understanding machine learning as inference.

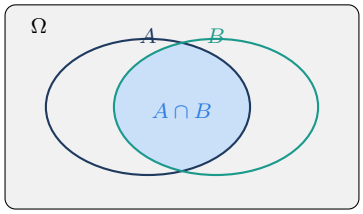
2.1 Conditional probability

Definition 2.1. The **conditional probability** of A given B (with $P(B) > 0$) is

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

Intuition

Conditioning *shrinks the sample space*. Once you know B happened, B becomes your new universe of possibilities; you ask what fraction of that universe also has A. Dividing by $P(B)$ renormalizes so the conditional probabilities still sum to one within B. Learning "B occurred" is exactly how a model updates when it sees data.



2.2 The multiplication and chain rules

Rearranging the definition gives the **multiplication rule** $P(A \cap B) = P(A \mid B)P(B)$, which extends to the **chain rule**:

$$P(A_1 \cap \cdots \cap A_n) = P(A_1)\, P(A_2 \mid A_1)\, P(A_3 \mid A_1, A_2) \cdots P(A_n \mid A_1, \ldots, A_{n-1}).$$

This factorization — breaking a complex joint probability into a product of simpler conditionals — is the backbone of probabilistic models, from naive Bayes to autoregressive language models that generate text one token at a time.

2.3 Independence

Definition 2.2. Events A and B are **independent** ($A \perp\!\!\!\perp B$) if knowing one tells you nothing about the other:

$$P(A \cap B) = P(A)\,P(B), \quad \text{equivalently} \quad P(A \mid B) = P(A).$$

Common pitfall

Independence is a strong, often-violated assumption — and "independent" is not the same as "mutually exclusive." Disjoint events with positive probability are in fact *maximally dependent* (if one happens, the other cannot). Also beware **conditional** independence: A and B can be dependent overall yet independent given C (or vice versa). Naive Bayes assumes features are conditionally independent given the class; it works surprisingly well even though the assumption is usually false.

2.4 The law of total probability

To find the probability of A, split the world into exhaustive, disjoint cases $B_1, \ldots, B_n$ and average:

$$P(A) = \sum_i P(A \mid B_i)\,P(B_i).$$

This "condition on every possibility and recombine" move is ubiquitous — it is the discrete cousin of marginalizing (integrating out) a variable, which is how we compute the evidence term below.

2.5 Bayes' theorem

Theorem 2.3 (Bayes' theorem)**.**

$$\underbrace{P(H \mid E)}_{\text{posterior}} = \frac{\overbrace{P(E \mid H)}^{\text{likelihood}}\ \overbrace{P(H)}^{\text{prior}}}{\underbrace{P(E)}_{\text{evidence}}}, \qquad P(E) = \sum_i P(E \mid H_i)P(H_i).$$

Key idea

Bayes' theorem is how rational belief updates with evidence. Start with a **prior** $P(H)$ (your belief in hypothesis H before data); multiply by the **likelihood** $P(E \mid H)$ (how well H predicts the evidence E); normalize by the **evidence** $P(E)$; out comes the **posterior** $P(H \mid E)$ (your updated belief). *Posterior* $\propto$ *likelihood* $\times$ *prior*. This one line is the foundation of Bayesian inference, of spam filters, of medical diagnosis, and of the entire probabilistic view of learning.

Example 2.4 (The base-rate fallacy)**.** A disease affects 1% of people. A test is 99% accurate (both sensitivity and specificity). You test positive — what is the chance you are

sick? Intuition screams "99%," but Bayes says otherwise. With prior $P(H) = 0.01$:

$$P(H \mid +) = \frac{0.99 \times 0.01}{0.99 \times 0.01 + 0.01 \times 0.99} = \frac{0.0099}{0.0198} = 0.5.$$

Only 50%. Because the disease is rare, false positives from the healthy 99% are as numerous as true positives. Ignoring the **base rate** (the prior) is one of the most consequential reasoning errors — in medicine, in law, and in ML systems that flag rare events.

Intuition

The lesson of the base-rate fallacy: *the prior matters.* Evidence shifts belief, but it shifts it *from* where you started. A very rare hypothesis needs very strong evidence to become probable. This is exactly why class imbalance wrecks naive classifiers, why anomaly detectors drown in false alarms, and why calibrated priors are essential whenever the thing you are detecting is rare.

How this powers ML / AI

Bayes' theorem *is* the probabilistic template for learning. Replace H with model parameters $\boldsymbol{\theta}$ and E with data $\mathcal{D}$:

$$P(\boldsymbol{\theta} \mid \mathcal{D}) \;\propto\; P(\mathcal{D} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta}).$$

Maximizing the likelihood $P(\mathcal{D} \mid \boldsymbol{\theta})$ alone is **maximum likelihood estimation** (Chapter 7) — the origin of cross-entropy and squared-error losses. Maximizing likelihood times a prior is **MAP estimation** — the origin of weight decay and other regularizers (Chapter 9). The chain rule $P(x_1, \ldots, x_n) = \prod_t P(x_t \mid x_{<t})$ is precisely how autoregressive language models (GPT-style) factor the probability of a sentence. Conditioning, independence, and Bayes are not background math — they are the specification of what these models compute.

Try it in NumPy

```
# Base-rate fallacy: P(sick | positive) with a rare disease
prior_sick = 0.01
sens = 0.99          # P(positive | sick)
spec = 0.99          # P(negative | healthy)  => P(positive | healthy) = 0.01
p_pos = sens*prior_sick + (1-spec)*(1-prior_sick)   # law of total probability
posterior = sens*prior_sick / p_pos
print(round(posterior, 3))     # 0.5  -- only 50%, not 99%!

# Bayesian updating: a second independent positive test
prior2 = posterior
p_pos2 = sens*prior2 + (1-spec)*(1-prior2)
print(round(sens*prior2 / p_pos2, 3))   # 0.99 -- evidence accumulates
```

Chapter summary

- **Conditional probability** $P(A \mid B) = P(A \cap B)/P(B)$ shrinks the sample space to B.

- The **chain rule** factors joint probabilities into products of conditionals — the basis of autoregressive models.
- **Independence** means $P(A \cap B) = P(A)P(B)$; it differs from mutual exclusivity, and conditional independence is subtle.
- The **law of total probability** averages over exhaustive cases; it computes the evidence term.
- **Bayes' theorem** (posterior $\propto$ likelihood $\times$ prior) updates belief with evidence; ignoring the prior causes the base-rate fallacy. With $\boldsymbol{\theta}$ and $\mathcal{D}$ it becomes the template for MLE and MAP learning.

PART II

Random Variables and Distributions

3

Random Variables and Expectation

Level: Core

So far events have been qualitative ("it rained"). A **random variable** attaches numbers to outcomes, letting us do arithmetic with uncertainty: average it, measure its spread, add independent copies. Random variables and their two summary numbers — **expectation** (the center) and **variance** (the spread) — are the daily vocabulary of statistics and machine learning.

3.1 Random variables

Definition 3.1. A **random variable** X is a function from the sample space to the real numbers, $X : \Omega \to \mathbb{R}$. It is **discrete** if it takes countably many values (a die, a word count) and **continuous** if it takes values in a continuum (a height, a measurement).

A random variable is described by its **distribution**, which says how probability is spread over its values:

- **Discrete:** the **probability mass function** (PMF) $p(x) = P(X = x)$, with $p(x) \geq 0$ and $\sum_x p(x) = 1$.
- **Continuous:** the **probability density function** (PDF) $f(x) \geq 0$ with $\int f(x)\,\mathrm{d}x = 1$; probabilities are areas, $P(a \leq X \leq b) = \int_a^b f(x)\,\mathrm{d}x$.
- **Either:** the **cumulative distribution function** (CDF) $F(x) = P(X \leq x)$, non-decreasing from 0 to 1.

Common pitfall

For a continuous variable, $f(x)$ is a *density*, not a probability — it can exceed 1, and $P(X = x) = 0$ for any single point. Only *areas* (integrals) are probabilities. Confusing density with probability is the classic continuous-variable error; it is why we never write "$P(X = x)$" for a continuous X but instead work with $f(x)\,\mathrm{d}x$ over a region.

3.2 Expectation

Definition 3.2. The **expected value** (mean) of X is its probability-weighted average:

$$\mathbb{E}[X] = \sum_x x\,p(x) \quad \text{(discrete)}, \qquad \mathbb{E}[X] = \int x\,f(x)\,\mathrm{d}x \quad \text{(continuous)}.$$

Intuition

The expectation is the **balance point** of the distribution — where it would balance if the probability were physical weight on a ruler. It is the long-run average value over many repetitions (a fact the law of large numbers will make precise in Chapter 6). It need not be a value X can actually take: the expected roll of a die is 3.5.

Proposition 3.3 (Linearity of expectation)**.** For *any* random variables and constants,

$$\mathbb{E}[aX + bY + c] = a\,\mathbb{E}[X] + b\,\mathbb{E}[Y] + c.$$

This holds **even when X and Y are dependent** — a remarkably powerful fact.

The **law of the unconscious statistician** (LOTUS) lets us take expectations of functions without finding their distribution: $\mathbb{E}[g(X)] = \sum_x g(x)p(x)$ or $\int g(x)f(x)\,\mathrm{d}x$.

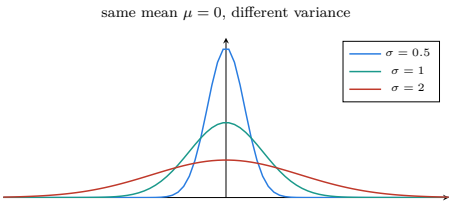
3.3 Variance and standard deviation

Definition 3.4. The **variance** measures spread around the mean:

$$\mathrm{Var}[X] = \mathbb{E}\big[(X - \mathbb{E}[X])^2\big] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \;\geq 0.$$

Its square root $\sigma = \sqrt{\mathrm{Var}[X]}$ is the **standard deviation**, in the same units as X.

Key scaling rules: $\mathrm{Var}[aX + b] = a^2\,\mathrm{Var}[X]$ (shifts don't change spread; scaling squares it), and for *independent* X, Y, $\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$.



3.4 Higher moments and standardization

The mean and variance are the first two **moments**. Higher moments describe shape: **skewness** (asymmetry) and **kurtosis** (tail heaviness). **Standardizing** a variable,

$$Z = \frac{X - \mu}{\sigma},$$

gives mean 0 and variance 1 — exactly the operation behind feature normalization and batch/layer normalization in neural networks, which keeps activations on a stable scale.

How this powers ML / AI

Random variables are the data of machine learning, and expectation is the form of nearly every objective. A **loss** is a random variable depending on a random data point; we minimize its expectation, the **risk** $\mathbb{E}$[loss], estimated by the average loss over a mini-batch (Chapter 6). **Linearity of expectation** justifies summing per-example gradients. **Variance** is the language of the bias–variance tradeoff (Chapter 10), of gradient-estimate noise in SGD, and of uncertainty in predictions. **Standardization** (Z-scoring) is the prepro-

cessing and normalization that makes deep nets trainable. Even a model's output is a random variable: a distribution we summarize by its mean (the prediction) and variance (the confidence).

Try it in NumPy

```python
import numpy as np
# A fair die: PMF, mean (3.5), variance (35/12 ~ 2.917)
x = np.arange(1, 7); p = np.ones(6)/6
mean = (x*p).sum()
var  = ((x**2)*p).sum() - mean**2
print(mean, round(var, 3))            # 3.5  2.917

# Linearity & standardization on samples
s = np.random.normal(loc=5, scale=2, size=100_000)
z = (s - s.mean())/s.std()
print(round(s.mean(),2), round(s.var(),2), round(z.mean(),2), round(z
    .std(),2))  # ~5, ~4, ~0, ~1
```

Chapter summary

- A **random variable** maps outcomes to numbers; describe it by PMF (discrete), PDF (continuous), or CDF.
- A continuous **density** is not a probability — only areas (integrals) are; $P(X = x) = 0$ pointwise.
- **Expectation** $\mathbb{E}[X]$ is the probability-weighted average (balance point); it is **linear** even for dependent variables.
- **Variance** $\mathbb{E}[X^2] - \mathbb{E}[X]^2$ measures spread; $\mathrm{Var}[aX + b] = a^2 \mathrm{Var}[X]$, and variances add for independent sums.
- **Standardizing** $Z = (X - \mu)/\sigma$ underlies feature and batch normalization; risk = expected loss is the ML objective.

4

Common Probability Distributions

Level: Core

A handful of distributions describe most of the randomness we ever model. Each arises from a simple, recurring story — a coin flip, a count of rare events, a sum of many small effects — and learning those stories lets you recognize at a glance which distribution fits a problem. This chapter is your field guide to the essential discrete and continuous distributions, with special attention to the Gaussian, which dominates statistics and machine learning.

4.1 Discrete distributions

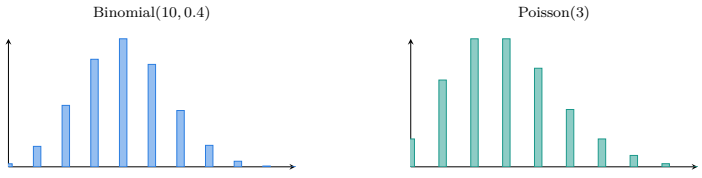
Bernoulli(p) — a single yes/no trial.
$X \in \{0, 1\}$ with $P(X = 1) = p$. Mean p, variance $p(1-p)$. The atom of binary classification: a single labeled example is a Bernoulli outcome.

Binomial(n, p) — number of successes in n independent trials.
$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$, mean np, variance $np(1-p)$. Counts of correct predictions, conversions, defects.

Geometric(p) — trials until the first success.
Models waiting times in discrete steps; "memoryless."

Poisson(λ) — count of rare events in a fixed interval.
$P(X = k) = \dfrac{\lambda^k e^{-\lambda}}{k!}$, mean = variance = λ. Arrivals, clicks, mutations, word counts. It is the limit of Binomial when $n \to \infty$, $p \to 0$ with $np = \lambda$ fixed.

Categorical / Multinomial — one of K classes (and counts thereof).
The multi-class generalization of Bernoulli/Binomial; the target distribution of every softmax classifier.



4.2 Continuous distributions

Uniform(a, b) — all values in a range equally likely.
Density $\frac{1}{b-a}$. The basis of random number generation and uninformative priors.

Exponential(λ) — waiting time for a continuous-time rare event.
Density $\lambda e^{-\lambda x}$, mean $1/\lambda$. Memoryless; the continuous cousin of the geometric, paired with the Poisson process.

Gaussian / Normal(μ, σ^2) — the bell curve.
Density $\dfrac{1}{\sigma\sqrt{2\pi}} \exp\left(-\dfrac{(x-\mu)^2}{2\sigma^2}\right)$. The most important distribution in all of statistics (next section).

Beta(α, β) — a distribution over probabilities, on $[0, 1]$.
The natural prior for a Bernoulli/Binomial rate; conjugate (Chapter 9).

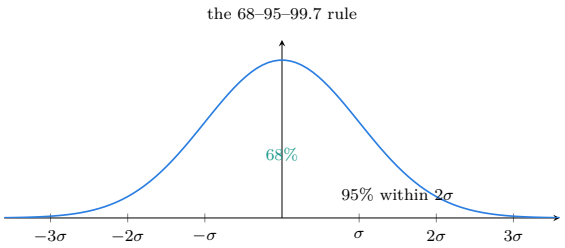
Gamma and $\boldsymbol{\chi^2}$, **Student's** $\boldsymbol{t}$ — positive quantities and small-sample inference.
The t has heavy tails and governs tests when σ is unknown (Chapter 8).

4.3 The Gaussian and why it is everywhere

The normal distribution earns its dominance for three reasons.

Key idea

(1) The Central Limit Theorem (Chapter 6): sums and averages of many independent effects are approximately Gaussian, *regardless* of the original distribution. Since measurement noise, aggregate behavior, and averaged gradients are all such sums, Gaussians appear by default. **(2) Maximum entropy**: among all distributions with a given mean and variance, the Gaussian is the most "spread out" / least presumptuous — the natural choice when you know only the first two moments. **(3) Mathematical convenience**: it is closed under linear maps, conditioning, and marginalization, and its log is a simple quadratic — which is exactly why squared-error loss corresponds to a Gaussian noise model.



About 68% of a Gaussian's mass lies within one standard deviation of the mean, 95% within two, and 99.7% within three — the rule of thumb behind "2σ" error bars and many confidence intervals.

4.4 Relationships between distributions

These distributions form a connected family, not a list to memorize: Bernoulli trials sum to a Binomial; the Binomial tends to a Poisson (rare events) and to a Gaussian (many events, by the CLT); the Exponential is the continuous Geometric; the Beta is a prior for the Bernoulli rate; sums of squared Gaussians are χ^2. Seeing the web of relationships is what turns a catalog into understanding.

How this powers ML / AI

Choosing a distribution *is* choosing a model, and that choice fixes the loss function (Chapter 7). Modeling labels as **Categorical** and maximizing likelihood yields **cross-entropy** loss — the default for classification. Modeling targets as **Gaussian** yields **mean-squared-error** loss — the default for regression. Modeling counts as **Poisson** yields Poisson regression. The **Beta** and **Gaussian** serve as conjugate priors for Bayesian models; the **Bernoulli** underlies dropout; the **Gaussian** is the latent prior in VAEs and the noise model in diffusion. When you pick a distribution, you are telling the model what kind of randomness generated the data.

Try it in NumPy

```python
import numpy as np
from scipy import stats
# Binomial -> Gaussian (CLT) and Binomial -> Poisson (rare events)
print(stats.binom.pmf(4, 10, 0.4))          # ~0.2508
print(stats.poisson.pmf(3, 3))              # ~0.2240
# 68-95-99.7 rule for the standard normal
for k in (1, 2, 3):
    print(k, round(stats.norm.cdf(k) - stats.norm.cdf(-k), 4))   # .6827 .9545 .9973
```

Chapter summary

- Each distribution encodes a **story**: Bernoulli (one trial), Binomial (count of successes), Poisson (rare events), Exponential (waiting time), Gaussian (sums/averages).
- The **Gaussian** dominates via the CLT, maximum entropy, and analytic convenience; remember the 68–95–99.7 rule.
- Distributions are **related** (Bernoulli→Binomial→Poisson/Gaussian; Beta as a rate prior), not isolated.
- Choosing a distribution chooses the **loss**: Categorical → cross-entropy, Gaussian → MSE, Poisson → Poisson regression.

5

Joint Distributions, Covariance, and the Multivariate Gaussian

Level: Core

Real data is rarely one number; it is vectors of many interrelated quantities — pixels, features, tokens. To model them we need **joint distributions** that capture how variables vary *together*, the **covariance** that quantifies their linear relationship, and the **multivariate Gaussian**, the workhorse distribution over vectors. This chapter is where probability meets the linear algebra of the companion volume.

5.1 Joint, marginal, and conditional distributions

Definition 5.1. The **joint distribution** of X and Y gives the probability of value pairs: $p(x, y) = P(X = x, Y = y)$ (discrete) or a joint density $f(x, y)$ (continuous). From it we recover:

$$\textbf{marginal: } p(x) = \sum_{y} p(x, y), \qquad \textbf{conditional: } p(y \mid x) = \frac{p(x, y)}{p(x)}.$$

Intuition

The joint distribution is the complete probabilistic description of several variables at once — a table (or surface) over all combinations. **Marginalizing** sums/integrates out the variables you don't care about (collapsing the table along a dimension); **conditioning** fixes a variable and renormalizes (taking a slice). Every probabilistic model is, in the end, a recipe for a joint distribution that we then marginalize and condition.

5.2 Covariance and correlation

Definition 5.2. The **covariance** of X and Y measures how they move together:

$$\operatorname{Cov}(X, Y) = \mathbb{E}\big[(X - \mathbb{E}X)(Y - \mathbb{E}Y)\big] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

The **correlation** normalizes it to $[-1, 1]$: $\operatorname{Corr}(X, Y) = \dfrac{\operatorname{Cov}(X, Y)}{\sigma_X \sigma_Y}$.

Positive covariance: they tend to rise together; negative: one rises as the other falls; zero: no *linear* relationship. For independent variables, $\operatorname{Cov} = 0$ and $\operatorname{Var}[X + Y] = \operatorname{Var}[X] + \operatorname{Var}[Y]$.

Common pitfall

Two famous traps. (1) **Correlation is not causation**: a strong correlation may reflect a hidden common cause, not influence. (2) **Zero correlation is not independence**: covariance detects only *linear* association. $Y = X^2$ with X symmetric about 0 has $\mathrm{Cov}(X, Y) = 0$ yet Y is completely determined by X. Independence implies zero covariance, never the reverse (except for jointly Gaussian variables, where they coincide).

5.3 The covariance matrix

For a random vector $\mathbf{x} = (X_1, \ldots, X_n)$, all pairwise covariances assemble into the **covariance matrix**

$$\boldsymbol{\Sigma} = \mathbb{E}\big[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top\big], \qquad \Sigma_{ij} = \mathrm{Cov}(X_i, X_j).$$

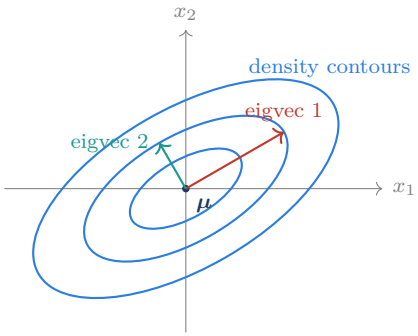
It is symmetric and positive semidefinite; its diagonal holds the variances. This single matrix is the bridge to linear algebra: its eigenvectors are the principal axes of the data cloud, and its eigendecomposition *is* **Principal Component Analysis** (PCA).

5.4 The multivariate Gaussian

Definition 5.3. The **multivariate normal** $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ has density

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big(-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\Big).$$

Its contours of constant density are **ellipses** (ellipsoids) centered at $\boldsymbol{\mu}$, oriented along the eigenvectors of $\boldsymbol{\Sigma}$, with widths set by the square roots of its eigenvalues.



Key idea

The multivariate Gaussian is governed entirely by a mean vector $\boldsymbol{\mu}$ (the center) and a covariance matrix $\boldsymbol{\Sigma}$ (the shape). The quadratic form $(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$ in the exponent is the squared **Mahalanobis distance** — distance measured in units of standard deviations along each principal direction. Spherical $\boldsymbol{\Sigma} = \sigma^2 I$ gives circular contours and recovers ordinary Euclidean distance; a general $\boldsymbol{\Sigma}$ stretches and rotates them. It is the one continuous distribution over vectors that is fully analytically tractable, which is why it is everywhere in ML.

Remarkably, the multivariate Gaussian is closed under the operations we care about: marginals are Gaussian, conditionals are Gaussian, and linear transformations of a Gaussian are Gaussian.

This closure is what makes Gaussian processes, Kalman filters, and linear-Gaussian models exactly solvable.

How this powers ML / AI

The covariance matrix and the multivariate Gaussian sit at the heart of classical and modern ML. **PCA** — the most-used dimensionality reduction — is the eigendecomposition of $\boldsymbol{\Sigma}$: keep the top eigenvectors (directions of greatest variance). **Whitening** transforms data to $\boldsymbol{\Sigma} = I$. The **multivariate Gaussian** is the latent prior $\mathcal{N}(\mathbf{0}, I)$ in VAEs, the noise model in diffusion, the function prior in **Gaussian processes**, and the assumption behind Gaussian discriminant analysis. The **Mahalanobis distance** powers anomaly detection. And whenever a model reports a covariance over its predictions, it is reporting correlated uncertainty in exactly this language.

Try it in NumPy

```
import numpy as np
# Sample a correlated 2-D Gaussian; recover its covariance and principal axes
mu = np.array([0.0, 0.0])
Sigma = np.array([[3.0, 1.5], [1.5, 1.0]])
X = np.random.multivariate_normal(mu, Sigma, size=200_000)
print(np.cov(X.T).round(2))                  # ~ [[3, 1.5], [1.5, 1]]
vals, vecs = np.linalg.eigh(np.cov(X.T))     # eigenvectors = PCA directions
print(vals.round(2))                         # variances along principal axes
# correlation != independence: Y = X^2 has zero covariance
x = np.random.randn(100_000); y = x**2
print(round(np.corrcoef(x, y)[0,1], 3))      # ~0, yet y depends fully on x
```

Chapter summary

- The **joint distribution** describes several variables together; **marginalize** (sum out) and **condition** (slice + renormalize) to query it.
- **Covariance/correlation** measure *linear* co-movement; zero correlation $\neq$ independence, and correlation $\neq$ causation.
- The **covariance matrix** $\boldsymbol{\Sigma}$ is symmetric PSD; its eigendecomposition is **PCA**.
- The **multivariate Gaussian** $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ has ellipsoidal contours and is closed under marginals, conditionals, and linear maps.
- These objects underlie PCA, whitening, Gaussian processes, VAE/diffusion priors, and Mahalanobis anomaly detection.

PART III

Limit Theorems

6

Inequalities and Limit Theorems

Level: Core

Why does averaging more data give a better answer? Why are so many things bell-shaped? Why can we estimate an expectation by a sample mean — the move that justifies mini-batch training? The **limit theorems** answer these questions. They are the theoretical bridge between probability (the model) and statistics (the data), and they are the reason machine learning works at all.

6.1 Probability inequalities

Before the limit theorems, two inequalities let us bound probabilities knowing only a mean or variance.

Proposition 6.1 (Markov and Chebyshev)**.** For a nonnegative random variable and any $a > 0$: $P(X \geq a) \leq \dfrac{\mathbb{E}[X]}{a}$ (Markov). For any variable with finite variance: $P(|X - \mu| \geq k\sigma) \leq \dfrac{1}{k^2}$ (Chebyshev).

Intuition

These are crude but assumption-light: Chebyshev says *no* distribution can put more than $1/k^2$ of its mass beyond k standard deviations — at most 25% past 2σ, for any shape whatsoever. Sharper "concentration inequalities" (Hoeffding, Bernstein) tighten this for sums of bounded variables and are the backbone of generalization bounds in learning theory: they bound how far training error can stray from true error.

Proposition 6.2 (Jensen's inequality)**.** For a convex function φ, $\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$. (Reversed for concave φ.)

Jensen is the workhorse behind the ELBO in variational inference and the non-negativity of KL divergence (Chapter 12).

6.2 The Law of Large Numbers

Theorem 6.3 (Law of Large Numbers)**.** Let $X_1, X_2, \ldots$ be i.i.d. with mean μ. The sample mean $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ converges to μ as $n \to \infty$:

$$\bar{X}_n \longrightarrow \mu.$$

Key idea

The LLN is the bedrock guarantee of statistics: **averages of more data converge to the truth**. It is what makes the frequentist interpretation coherent (long-run frequency → probability) and what licenses **Monte Carlo estimation** — approximating an expectation $\mathbb{E}[g(X)]$ by a sample average $\frac{1}{N}\sum g(x_i)$. Every time we estimate a loss, a gradient, or an integral by averaging samples, we are leaning on the LLN.

6.3 The Central Limit Theorem

The LLN says the sample mean converges; the CLT says *how* — and reveals why the Gaussian is universal.

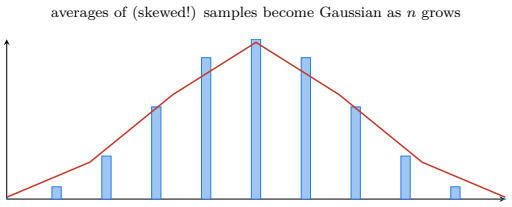
Theorem 6.4 (Central Limit Theorem)**.** For i.i.d. X_i with mean μ and finite variance σ^2, the standardized sample mean converges to a standard normal:

$$\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \longrightarrow \mathcal{N}(0,1).$$

Equivalently, $\bar{X}_n$ is approximately $\mathcal{N}\big(\mu,\, \sigma^2/n\big)$ for large n.

Intuition

The CLT is one of the most astonishing facts in mathematics: *sum up many independent random effects and the result is bell-shaped, no matter what the individual effects look like* — uniform, skewed, discrete, anything (with finite variance). This is why measurement errors, heights, and averaged quantities are Gaussian. Note the $\sqrt{n}$: the standard error of the mean shrinks like $1/\sqrt{n}$, so to halve your error you need *four times* the data — the fundamental, sometimes painful, exchange rate of statistics.



6.4 Modes of convergence (brief)

Probabilists distinguish convergence *in probability* (the LLN), *in distribution* (the CLT), and *almost surely* (the strong LLN). The distinctions matter for proofs; for our purposes the headline is enough: sample averages converge to expectations, and their fluctuations are Gaussian of size $\sigma/\sqrt{n}$.

How this powers ML / AI

The limit theorems are the silent justification of everyday ML. **Mini-batch SGD** replaces the true gradient (an expectation over all data) with an average over a batch — valid by the LLN, with noise of order $1/\sqrt{\text{batch size}}$ by the CLT, which is why larger batches give smoother, lower-variance gradient estimates. **Monte Carlo** estimates of integrals (the reparameterization trick, dropout-as-Bayesian-inference, RL returns) converge by the LLN.

Confidence intervals and error bars on metrics use the CLT's $\sigma/\sqrt{n}$. **Concentration inequalities** give the generalization bounds of statistical learning theory. And the CLT explains why Gaussian assumptions are so often safe: averaged quantities are bell-shaped by law.

Try it in NumPy

```python
import numpy as np
# LLN: sample mean of a skewed (exponential) variable -> true mean 1.0
x = np.random.exponential(scale=1.0, size=2_000_000)
print(round(x.mean(), 3))                    # ~1.0

# CLT: averages of n exponential draws look Gaussian; SE shrinks like 1/sqrt(n)
for n in (1, 10, 100, 1000):
    means = np.random.exponential(1.0, size=(100_000, n)).mean(axis=1)
    print(n, round(means.std(), 4))          # ~ 1/sqrt(n): 1, 0.316, 0.1, 0.0316
```

Chapter summary

- **Markov/Chebyshev** bound tail probabilities from a mean/variance alone; concentration inequalities sharpen this for generalization bounds.
- **Jensen's inequality** ($\varphi(\mathbb{E}X) \le \mathbb{E}[\varphi(X)]$ for convex φ) underlies the ELBO and KL ≥ 0.
- The **Law of Large Numbers**: sample averages converge to the true mean — the basis of Monte Carlo and mini-batch estimation.
- The **Central Limit Theorem**: standardized sample means become $\mathcal{N}(0,1)$; the standard error shrinks like $\sigma/\sqrt{n}$.
- Together they justify SGD's batch gradients, Monte Carlo objectives, and confidence intervals.

PART IV

Statistical Inference

7

Estimation and Maximum Likelihood

Level: Advanced

We now cross from probability into statistics — from "given the model, predict the data" to "given the data, infer the model." The central task is **estimation**: using a sample to guess an unknown parameter. The dominant method, **maximum likelihood**, is not just a statistical technique; it is the principle from which nearly every loss function in machine learning is derived. This is one of the most important chapters in the book.

7.1 Estimators and their quality

Definition 7.1. An **estimator** $\hat{\theta}$ is a function of the data used to estimate an unknown parameter θ. Because it depends on a random sample, $\hat{\theta}$ is itself a random variable with a **sampling distribution**.

We judge estimators by three properties:

- **Bias:** $\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$. Unbiased means correct *on average*.
- **Variance:** $\text{Var}(\hat{\theta})$ — how much the estimate jitters across samples.
- **Mean squared error:** $\text{MSE}(\hat{\theta}) = \mathbb{E}[(\hat{\theta} - \theta)^2] = \text{Bias}^2 + \text{Var}$.

Key idea

The decomposition $\text{MSE} = \text{Bias}^2 + \text{Var}$ is the seed of the **bias–variance tradeoff** (Chapter 10) that governs all of machine learning. A more flexible estimator/model lowers bias but raises variance; a simpler one does the reverse. The best estimator is not always unbiased — accepting a little bias to cut variance often lowers total error, which is exactly what regularization does.

Example 7.2 (The n vs. $n-1$ in sample variance)**.** The sample variance with divisor n is biased low; dividing by $n-1$ instead (**Bessel's correction**) makes it unbiased. The maximum-likelihood estimate of a Gaussian's variance uses n and is therefore slightly biased — a concrete reminder that MLE is not automatically unbiased.

7.2 The likelihood function

Definition 7.3. Given data $\mathcal{D} = \{x_1, \ldots, x_n\}$ and a model with parameter θ, the **likeli-**

hood is the probability of the observed data viewed as a function of θ:

$$L(\theta) = P(\mathcal{D} \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta) \quad \text{(i.i.d. data)}.$$

Common pitfall

Likelihood is *not* a probability distribution over θ — it does not integrate to 1 over θ, and "$L(\theta)$" answers "how well does this θ explain the data I saw," not "how probable is θ." Treating likelihood as a posterior is the error that Bayes' theorem (and a prior) exists to correct. The probability is over the *data*; we merely read it as a function of θ.

7.3 Maximum likelihood estimation

Definition 7.4. The **maximum likelihood estimator** is the parameter that makes the observed data most probable:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \sum_{i=1}^{n} \log p(x_i \mid \theta).$$

We maximize the **log-likelihood** because the log turns the product into a sum (numerically stable, easy to differentiate) without moving the maximum.

Intuition

MLE asks a beautifully simple question: "of all the settings of θ, which one would have made my data least surprising?" Pick that one. Geometrically, you slide θ until the model's probability mass piles up where your data actually fell. Maximizing is done with the calculus of the companion volume: set the derivative of the log-likelihood to zero, or descend its negative by gradient descent.

Example 7.5 (MLE for a coin)**.** Observe k heads in n flips. The log-likelihood is $k \log p + (n-k)\log(1-p)$; setting its derivative to zero gives the unsurprising $\hat{p} = k/n$, the observed frequency. MLE recovers common sense — and then generalizes it to models far too complex for intuition.

Proposition 7.6 (Why statisticians love the MLE)**.** Under regularity conditions the MLE is **consistent** (converges to the truth as $n \to \infty$), **asymptotically normal**, and **asymptotically efficient** (attains the Cramér–Rao lower bound — the smallest possible variance). The **Fisher information** quantifies how sharply peaked the likelihood is and sets that minimum variance.

How this powers ML / AI

Maximum likelihood is where loss functions come from. Minimizing the negative log-likelihood (NLL) is maximizing likelihood, and the choice of probability model determines the loss:

- Categorical labels $\Rightarrow$ NLL = **cross-entropy loss** (every classifier, every language

model).
- Gaussian targets ⇒ NLL = **mean-squared error** (regression).
- Poisson counts ⇒ Poisson regression loss; and so on.

So training a neural network by minimizing cross-entropy *is* maximum likelihood estimation of its parameters. The bias–variance decomposition explains why pure MLE overfits on small data, motivating the regularizers (= priors, Chapter 9) that every real system adds. The MLE skeleton is underneath essentially all of supervised deep learning.

Try it in NumPy

```python
import numpy as np
from scipy.optimize import minimize_scalar
# MLE for a coin: maximize log-likelihood of k heads in n flips
k, n = 7, 10
nll = lambda p: -(k*np.log(p) + (n-k)*np.log(1-p))   # negative log-likelihood
res = minimize_scalar(nll, bounds=(1e-6, 1-1e-6), method='bounded')
print(round(res.x, 3))                  # 0.7 = k/n, the observed frequency

# MLE for a Gaussian recovers the sample mean and (biased) variance
x = np.random.normal(5, 2, size=10_000)
print(round(x.mean(),3), round(x.var(),3))   # mu_hat ~5, sigma^2_hat ~4 (divisor n)
```

Chapter summary

- An **estimator** is a random function of data; judge it by **bias**, **variance**, and MSE = $\text{Bias}^2 + \text{Var}$.
- The **likelihood** $L(\theta) = P(\mathcal{D} \mid \theta)$ is the data's probability read as a function of θ — not a distribution over θ.
- **MLE** picks θ maximizing the (log-)likelihood; it is consistent, asymptotically normal, and efficient, but can be biased.
- Minimizing **negative log-likelihood** yields cross-entropy (categorical) and MSE (Gaussian) — the standard ML losses.
- The bias–variance decomposition foreshadows why regularization (priors) is needed beyond pure MLE.

8

Confidence Intervals and Hypothesis Testing

Level: Advanced

A point estimate alone is dangerous: it hides its own uncertainty. This chapter adds the error bars. **Confidence intervals** express how precise an estimate is; **hypothesis tests** decide whether an observed effect is real or could be chance. These frequentist tools are how we compare models honestly, report results responsibly, and avoid being fooled by noise — including the noise in our own benchmark numbers.

8.1 Sampling distributions and the standard error

An estimator like the sample mean has a **sampling distribution**; by the CLT (Chapter 6) it is approximately Gaussian for large n, centered at the true value with **standard error**

$$\mathrm{SE}(\bar{X}) = \frac{\sigma}{\sqrt{n}} \quad \text{(estimated by } s/\sqrt{n}\text{)}.$$

The standard error is the typical distance between estimate and truth — and it shrinks only as $1/\sqrt{n}$.

8.2 Confidence intervals

Definition 8.1. A 95% **confidence interval** is a data-computed range produced by a procedure that, over repeated samples, contains the true parameter 95% of the time. For a mean (large n):

$$\bar{X} \pm 1.96\,\frac{s}{\sqrt{n}}.$$

Common pitfall

The correct interpretation is subtle and constantly mangled. A 95% CI does **not** mean "there is a 95% probability the true value is in *this* interval" — in the frequentist view the true value is fixed and the interval is random, so a specific interval either contains it or not. The 95% refers to the long-run success rate of the *procedure*. (The statement you *want* to make — probability over the parameter — is exactly what a Bayesian **credible interval** provides, Chapter 9.)

Intuition

A confidence interval is the estimate plus an honest margin of error. Wider intervals mean less certainty (small n, high variance); narrower means more. When two models' performance intervals overlap heavily, the apparent winner may just be lucky — which is why serious benchmarking reports intervals, not bare numbers.

8.3 Hypothesis testing

Definition 8.2. A **hypothesis test** pits a **null hypothesis** H_0 ("no effect," the skeptical default) against an **alternative** H_1. We compute a **test statistic** from the data and ask how extreme it would be *if* H_0 *were true*.

Definition 8.3. The **p-value** is the probability, *assuming* H_0 *is true*, of observing data at least as extreme as what we saw. A small p-value means the data would be surprising under H_0, casting doubt on it. If $p < \alpha$ (commonly 0.05), we "reject H_0."

Common pitfall

The p-value is the most misinterpreted number in science. It is **not** the probability that H_0 is true, nor the probability your result is a fluke, nor one minus the probability of replication. It is $P(\text{data this extreme} \mid H_0)$ — a statement about the data given the hypothesis, not the hypothesis given the data. Confusing the two is, once again, the inversion that only Bayes' theorem (with a prior) can legitimately perform. Beware also **p-hacking**: test enough things and some will cross 0.05 by chance.

8.4 Errors, power, and multiple testing

Every test risks two errors:

	H_0 **true**	H_0 **false**
Reject H_0	Type I error (rate α)	correct (power $= 1 - \beta$)
Fail to reject	correct	Type II error (rate β)

Power is the chance of detecting a real effect; it grows with sample size and effect size. Common tests include the **z-** and **t-tests** (means; the t handles unknown variance and small samples via its heavy tails), the **χ^2 test** (categorical counts), and **ANOVA** (several groups). When running many tests, correct for **multiple comparisons** (e.g. Bonferroni) or the false-discovery rate, or chance alone will manufacture "significant" findings.

8.5 Resampling: the bootstrap

When formulas are hard, **bootstrapping** estimates uncertainty by resampling the data with replacement many times and recomputing the statistic. The spread of those replicates approximates the sampling distribution — a computational shortcut to standard errors and confidence intervals for almost any statistic, and a close cousin of the bagging/ensemble idea in ML.

How this powers ML / AI

These tools are how ML practitioners avoid fooling themselves. A model that scores 91.2% vs. a baseline's 90.8% has likely won *nothing* if the confidence interval on the gap straddles zero — benchmark leaderboards demand error bars and significance tests for exactly this reason. **A/B testing** of deployed models is hypothesis testing. **Bootstrapping** produces confidence intervals on metrics like AUC and powers **bagging** and random forests. Power analysis sizes experiments. And the p-value pitfalls reappear as the replication and "SOTA-chasing" crises in ML research, where multiple-comparison and selection effects inflate apparent gains.

Try it in NumPy

```python
import numpy as np
from scipy import stats
# 95% CI for a mean and a two-sample t-test
a = np.random.normal(0.0, 1.0, 200)
b = np.random.normal(0.3, 1.0, 200)
se = a.std(ddof=1)/np.sqrt(len(a))
print((a.mean()-1.96*se, a.mean()+1.96*se))      # ~ CI around 0
t, p = stats.ttest_ind(a, b)
print(round(t,2), round(p,4))                     # small p => means differ

# Bootstrap CI for the median (no formula needed)
x = np.random.exponential(1.0, 500)
boot = [np.median(np.random.choice(x, len(x), replace=True)) for _ in range(5000)]
print(np.percentile(boot, [2.5, 97.5]).round(3))  # 95% bootstrap interval
```

Chapter summary

- The **standard error** $\sigma/\sqrt{n}$ is the typical estimate–truth distance; it shrinks only as $1/\sqrt{n}$.
- A **confidence interval** reflects the long-run coverage of a *procedure*, not the probability that this interval contains the truth.
- A **p-value** is $P(\text{data this extreme} \mid H_0)$ — not $P(H_0 \mid \text{data})$; mind Type I/II errors, power, and multiple testing.
- Use $z/t/\chi^2$/ANOVA as appropriate; **bootstrap** for uncertainty without formulas.
- In ML these give honest model comparison, A/B tests, metric confidence intervals, and the basis of bagging.

9 Bayesian Inference

Level: Advanced

Frequentist statistics treats parameters as fixed unknowns and data as random. **Bayesian inference** flips the emphasis: it treats parameters as random variables with probability distributions that encode our beliefs, and updates those beliefs with data via Bayes' theorem. The result is a complete distribution over what we don't know — not a single point — which is precisely the language of uncertainty that modern, safety-conscious AI needs.

9.1 The Bayesian recipe

Definition 9.1. Bayesian inference computes the **posterior** distribution of parameters given data:

$$\underbrace{p(\theta \mid \mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{likelihood}}\ \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{evidence}}}, \qquad p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta) p(\theta)\, \mathrm{d}\theta.$$

In words: *posterior* $\propto$ *likelihood* $\times$ *prior*.

Key idea

The shift from MLE to Bayes is the shift from "the single best θ" to "the full distribution of plausible θ." The **prior** $p(\theta)$ states what you believe before seeing data; the **likelihood** updates it; the **posterior** is your refreshed belief. With more data the likelihood dominates and the prior washes out — but with little data the prior is what keeps you sane (and what prevents overfitting). Everything you might want — a point estimate, an interval, a prediction — is read off the posterior.

9.2 Conjugate priors: Bayes in closed form

For certain prior–likelihood pairs the posterior stays in the same family as the prior — a **conjugate** prior — making the update pure algebra.

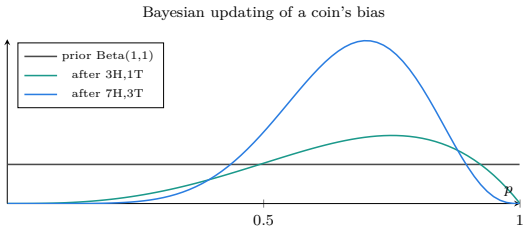
Example 9.2 (Beta–Bernoulli)**.** To estimate a coin's bias p, use a $\mathrm{Beta}(\alpha, \beta)$ prior. After observing k heads in n flips, the posterior is simply

$$p \mid \mathcal{D} \sim \mathrm{Beta}(\alpha + k,\ \beta + n - k).$$

The prior parameters act like "pseudo-counts" of heads and tails seen in advance. Start with $\mathrm{Beta}(1, 1)$ (uniform) and the posterior mean is $\frac{k+1}{n+2}$ — **Laplace smoothing**, the same

+1 that keeps naive Bayes and language models from assigning zero probability to unseen events.

The Gaussian is conjugate to itself (Gaussian prior + Gaussian likelihood ⇒ Gaussian posterior), which is why linear-Gaussian models and Kalman filters update in closed form.



9.3 Point estimates from the posterior

The posterior is a whole distribution; to summarize it:

- The **posterior mean** $\mathbb{E}[\theta \mid \mathcal{D}]$ minimizes expected squared error.
- The **MAP estimate** $\hat{\theta}_{\text{MAP}} = \arg\max_\theta p(\theta \mid \mathcal{D}) = \arg\max_\theta \left[\log p(\mathcal{D} \mid \theta) + \log p(\theta)\right]$ is the posterior's peak.

Key idea

MAP = MLE + prior penalty, and that penalty *is* regularization. With a Gaussian prior $\theta \sim \mathcal{N}(0, \tau^2)$, $\log p(\theta)$ contributes a term $\propto -\|\theta\|^2$ — exactly **L2 regularization / weight decay** (ridge regression). With a Laplace prior, it contributes $\propto -\|\theta\|_1$ — exactly **L1 / lasso**, which induces sparsity. So every weight penalty in deep learning is, secretly, a prior belief that parameters should be small. This is the single most important bridge between Bayesian statistics and practical ML.

9.4 Credible intervals and prediction

A **credible interval** is an interval containing, say, 95% of the posterior probability — and it means exactly what people *wish* a confidence interval meant: "given the data and prior, there is a 95% probability the parameter lies here." To predict new data we use the **posterior predictive**, averaging over all parameter values weighted by the posterior:

$$p(x_{\text{new}} \mid \mathcal{D}) = \int p(x_{\text{new}} \mid \theta)\, p(\theta \mid \mathcal{D})\, \mathrm{d}\theta.$$

This *marginalization over uncertainty* is what gives Bayesian predictions their honest, often better-calibrated confidence.

9.5 Bayesian vs. frequentist

Intuition

The two schools answer different questions. Frequentist: "what procedure has good long-run error rates?" Bayesian: "what should I believe given this data and my prior?" Frequentist methods need no prior and dominate hypothesis testing; Bayesian methods handle small data, prior knowledge, and uncertainty propagation gracefully but require choosing a prior and (usually) heavy computation for the evidence integral. Modern ML is pragmat-

ically both: point estimates by penalized MLE (a MAP in disguise), with Bayesian ideas supplying the regularizers, the uncertainty estimates, and the generative models.

How this powers ML / AI

Bayesian thinking is woven through ML even when no one says "Bayes." **Weight decay is a Gaussian prior**; **L1 sparsity is a Laplace prior**; **dropout** approximates Bayesian model averaging; **label smoothing** is a prior on the label distribution. **Bayesian neural networks**, **MC dropout**, and **deep ensembles** approximate the posterior predictive to produce calibrated uncertainty — crucial for medicine, autonomy, and active learning. **Variational inference** (Chapter 12) makes the intractable evidence integral tractable and trains VAEs. **Gaussian processes** are fully Bayesian nonparametric models. When a system says "I am 80% sure," and means it, there is a posterior underneath.

Try it in NumPy

```python
import numpy as np
from scipy import stats
# Beta–Bernoulli conjugate update for a coin's bias
alpha, beta = 1, 1                      # uniform prior
heads, tails = 7, 3
post = stats.beta(alpha+heads, beta+tails)
print(round(post.mean(), 3))            # (7+1)/(10+2) = 0.667  (Laplace smoothing)
print(post.interval(0.95))              # 95% CREDIBLE interval for p

# MAP with a Gaussian prior == ridge: argmax [loglik - lambda*||theta||^2]
# (the prior's log-term is exactly the L2 penalty)
```

Chapter summary

- Bayesian inference computes a **posterior** $\propto$ likelihood $\times$ prior — a full distribution over parameters.
- **Conjugate priors** (Beta–Bernoulli, Gaussian–Gaussian) give closed-form updates; pseudo-counts yield Laplace smoothing.
- **MAP = MLE + prior**: Gaussian prior $\to$ L2/weight decay, Laplace prior $\to$ L1/lasso. Regularization is a prior.
- **Credible intervals** mean what CIs are mistaken for; the **posterior predictive** marginalizes over uncertainty.
- Bayesian ideas power weight decay, dropout, ensembles, BNNs, variational inference, and Gaussian processes — the machinery of calibrated uncertainty.

10

Statistical Models and Regression

Level: Advanced

Regression is where probability and statistics become predictive machine learning. A **statistical model** describes how outputs depend on inputs plus randomness; **regression** fits that dependence to data. Linear and logistic regression are the simplest such models — and, not coincidentally, the conceptual templates for the entire supervised-learning toolbox, including deep networks. This chapter also formalizes the **bias–variance tradeoff** that governs every fitting decision you will ever make.

10.1 The linear model

Definition 10.1. Linear regression models a continuous target as a linear function of features plus Gaussian noise:

$$y = \mathbf{w}^\top \mathbf{x} + b + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

Key idea

Fitting this model by **maximum likelihood** is identical to minimizing **least squares**. Because the noise is Gaussian, the log-likelihood of the data is $-\frac{1}{2\sigma^2}\sum_i (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$ plus constants — so maximizing likelihood *is* minimizing the sum of squared errors. "Least squares," invented by Gauss and Legendre, turns out to be MLE under a Gaussian noise assumption. The probability model and the loss function are two views of one thing.

Minimizing squared error has the famous closed form (the *normal equations* of the linear-algebra companion), $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ — statistics and linear algebra meeting exactly.

10.2 Logistic regression

For classification, we model the *probability* of a class rather than a continuous value.

Definition 10.2. Logistic regression passes a linear score through the sigmoid to produce a probability:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Fitting it by maximum likelihood means minimizing **cross-entropy** (the Bernoulli NLL). There is no closed form, so we use gradient descent — and the gradient has the clean "predicted

minus actual" form from the calculus companion. The multi-class version uses the softmax and is exactly the final layer of a neural-network classifier.

Intuition

Logistic regression is a one-layer neural network with a sigmoid output. Stack more layers with nonlinearities and you get deep learning; the loss (cross-entropy), the fitting principle (maximum likelihood), and the optimizer (gradient descent) are unchanged. This is why logistic regression is called the conceptual ancestor of modern classifiers — understand it probabilistically and you understand the output layer of essentially every deep model.

10.3 Generalized linear models

Linear and logistic regression are special cases of **generalized linear models** (GLMs): pick a distribution for the target (Gaussian, Bernoulli, Poisson, …) and a **link function** connecting its mean to a linear predictor $\mathbf{w}^\top \mathbf{x}$. The distribution determines the loss (via MLE), unifying a whole family of models under one probabilistic principle — and explaining why so many losses look related.

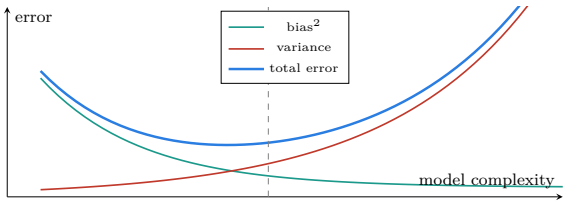
10.4 The bias–variance tradeoff

Theorem 10.3 (Bias–variance decomposition)**.** For squared-error loss, the expected test error of a model decomposes as

$$\mathbb{E}\big[(y - \hat{f}(\mathbf{x}))^2\big] = \underbrace{\text{Bias}^2}_{\text{too rigid}} + \underbrace{\text{Var}}_{\text{too sensitive}} + \underbrace{\sigma^2}_{\text{irreducible noise}}.$$

Key idea

This is the central tension of model fitting. A model that is too simple (high **bias**) underfits — it cannot capture the pattern. A model that is too flexible (high **variance**) overfits — it memorizes the training noise and varies wildly across samples. Generalization error is minimized in between. **Regularization** (= a prior, Chapter 9) deliberately adds a little bias to remove a lot of variance; **cross-validation** estimates the test error so we can tune the balance.



How this powers ML / AI

This chapter *is* the skeleton of supervised learning. **Linear regression = MLE under Gaussian noise = least squares**; **logistic/softmax regression = MLE under Bernoulli/Categorical = cross-entropy** — and the latter is exactly the output layer of a deep classifier. **GLMs** explain the menagerie of losses as one principle. The **bias–variance tradeoff** is the lens for every capacity decision: network size, depth, regularization strength, early stopping, data augmentation, and ensembling all move you along this curve. (Deep learning adds a famous twist — the "double descent" where massively

overparameterized models generalize well anyway — but the decomposition remains the foundational vocabulary.)

Try it in NumPy

```python
import numpy as np
# Linear regression by least squares == Gaussian-noise MLE (normal equations)
X = np.column_stack([np.ones(200), np.random.randn(200, 2)])
w_true = np.array([1.0, 2.0, -0.5])
y = X @ w_true + np.random.normal(0, 0.3, 200)
w_hat = np.linalg.lstsq(X, y, rcond=None)[0]
print(w_hat.round(2))                          # ~ [1, 2, -0.5]

# Ridge = MAP with a Gaussian prior: add lambda*I before inverting
lam = 1.0
w_ridge = np.linalg.solve(X.T@X + lam*np.eye(3), X.T@y)
print(w_ridge.round(2))                        # shrunk slightly toward 0
```

Chapter summary

- **Linear regression** = MLE under Gaussian noise = least squares (closed-form normal equations).
- **Logistic/softmax regression** = MLE under Bernoulli/Categorical = cross-entropy = a neural net's output layer.
- **GLMs** unify these: choose a distribution + link, and MLE fixes the loss.
- The **bias–variance tradeoff** (MSE = $\text{Bias}^2 + \text{Var} + \sigma^2$) governs over/underfitting; regularization trades bias for variance.
- **Cross-validation** estimates test error to tune the balance; these ideas drive every model-capacity decision in ML.

PART V

Advanced and Computational

11

Stochastic Processes, Markov Chains, and Monte Carlo

Level: Expert

So far our randomness has been static. A **stochastic process** adds time: a sequence of random variables that evolve and depend on one another. The most important kind — the **Markov chain** — forgets everything but the present, yet is rich enough to model language, web surfing, and physical systems. Markov chains also power **Markov chain Monte Carlo** (MCMC), the algorithm that made Bayesian inference practical and that quietly underlies modern generative models.

11.1 Stochastic processes

Definition 11.1. A **stochastic process** is a collection of random variables $\{X_t\}$ indexed by time t. Examples: a random walk, the price of a stock, the sequence of words in a sentence, the state of a game.

11.2 Markov chains

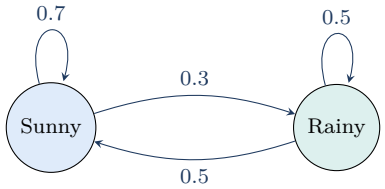
Definition 11.2. A process has the **Markov property** if the future depends on the past only through the present:

$$P(X_{t+1} \mid X_t, X_{t-1}, \ldots, X_0) = P(X_{t+1} \mid X_t).$$

A **Markov chain** on a finite state space is specified by a **transition matrix P** whose entry $P_{ij} = P(X_{t+1} = j \mid X_t = i)$; each row sums to 1.

Intuition

"Memoryless" does not mean the past is irrelevant — it means the present state *summarizes* all relevant history. Where you go next depends only on where you are now, not the route that brought you. This is a modeling assumption (often approximate), and it is exactly the trade that makes the math tractable: one matrix **P** captures the entire dynamics, and multiplying by it advances the distribution one step.



11.3 Stationary distributions and convergence

Definition 11.3. A distribution $\boldsymbol{\pi}$ over states is **stationary** if it is unchanged by a transition: $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$. It is the left eigenvector of $\mathbf{P}$ with eigenvalue 1 — linear algebra reappearing.

Key idea

For a well-behaved (irreducible, aperiodic) chain, the distribution over states *converges to the stationary distribution* $\boldsymbol{\pi}$ regardless of where it started — the chain "forgets" its initial condition. This convergence is the magic that MCMC exploits: design a chain whose stationary distribution is the one you want to sample from, run it long enough, and its states become samples from that target. **PageRank** is exactly this — the stationary distribution of a random web-surfer's Markov chain.

11.4 Monte Carlo and MCMC

Monte Carlo methods estimate quantities by random sampling: to approximate $\mathbb{E}[g(X)]$, draw samples and average (justified by the LLN, Chapter 6). But how do you sample from a complicated, high-dimensional distribution — like a Bayesian posterior whose normalizing evidence integral is intractable?

Definition 11.4 (MCMC)**. Markov chain Monte Carlo** builds a Markov chain whose stationary distribution is the target p, then uses the chain's trajectory as (correlated) samples. The **Metropolis–Hastings** algorithm proposes a move and accepts it with a probability that depends only on a *ratio* $p(x')/p(x)$ — so the intractable normalizing constant cancels. **Gibbs sampling** updates one coordinate at a time from its conditional.

Intuition

The brilliance of Metropolis–Hastings is that you never need the normalizing constant — only ratios of densities, which you *can* compute. This is what made full Bayesian inference practical: you can sample from a posterior known only up to proportionality (likelihood $\times$ prior), exactly the situation Chapter 9 left us in. Run the chain, discard an initial "burn-in," and the remaining states approximate the posterior.

How this powers ML / AI

Markov chains and Monte Carlo are everywhere in modern AI. **MCMC** is the classical engine of Bayesian machine learning, sampling posteriors that have no closed form. **Markov chains** model sequences — the n-gram language models that preceded neural ones are literally Markov chains over words, and **reinforcement learning** formalizes the world as a Markov decision process. Most strikingly, **diffusion models** — today's leading image generators — are Markov chains: a forward chain gradually adds Gaussian noise, and a

learned reverse chain removes it step by step to generate samples. The "sampling" in a diffusion or autoregressive model is a Monte Carlo walk through a learned probability distribution.

Try it in NumPy

```python
import numpy as np
# Stationary distribution as the eigenvector of P^T with eigenvalue 1
P = np.array([[0.7, 0.3],
              [0.5, 0.5]])
vals, vecs = np.linalg.eig(P.T)
pi = np.real(vecs[:, np.argmin(np.abs(vals-1))]); pi = pi/pi.sum()
print(pi.round(3))                  # ~ [0.625, 0.375]

# Metropolis-Hastings sampling from an unnormalized target p(x) ∝ e^{-x^2/2}
logp = lambda x: -x**2/2            # only need it up to a constant!
x, samples = 0.0, []
for _ in range(200_000):
    xp = x + np.random.normal(0, 1)
    if np.log(np.random.rand()) < logp(xp) - logp(x):
        x = xp
    samples.append(x)
print(round(np.mean(samples), 2), round(np.std(samples), 2))   # ~0, ~1 (standard normal)
```

Chapter summary

- A **stochastic process** is random variables evolving in time; a **Markov chain** depends on the past only through the present.
- A chain is described by its **transition matrix P**; the **stationary distribution** solves $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$.
- Well-behaved chains **converge** to $\boldsymbol{\pi}$ regardless of start — the basis of PageRank and MCMC.
- **Monte Carlo** estimates expectations by sampling; **MCMC** (Metropolis–Hastings, Gibbs) samples hard distributions using only density *ratios*.
- These power Bayesian posterior sampling, n-gram/RL/MDP models, and the forward/reverse chains of diffusion models.

12

Information Theory

Level: Expert

Information theory, born from Claude Shannon's 1948 work on communication, turns probability into a measure of *information* and *surprise*. Its quantities — entropy, cross-entropy, KL divergence, mutual information — are not exotic add-ons to machine learning; they *are* its loss functions and objectives. This short chapter makes precise the cross-entropy you have met repeatedly and reveals that training a classifier is, literally, minimizing the information gap between the model and the world.

12.1 Entropy: the measure of uncertainty

Definition 12.1. The **entropy** of a distribution p measures its average surprise (in bits, using $\log_2$, or nats, using ln):

$$\mathrm{H}(p) = -\sum_x p(x) \log p(x) = \mathbb{E}_p[-\log p(X)].$$

Intuition

Define the **surprise** of an outcome as $-\log p(x)$: rare events (p small) are very surprising, certain events ($p = 1$) are unsurprising ($-\log 1 = 0$). Entropy is the *expected* surprise — how uncertain you are on average before seeing the outcome. It is maximized by the uniform distribution (maximum ignorance) and zero for a deterministic one (no uncertainty). Shannon showed it is also the minimum average number of bits needed to encode samples from p — uncertainty and compressibility are the same quantity.

12.2 Cross-entropy and KL divergence

Definition 12.2. The **cross-entropy** between a true distribution p and a model q is the average surprise of p's outcomes *under* q:

$$\mathrm{H}(p, q) = -\sum_x p(x) \log q(x).$$

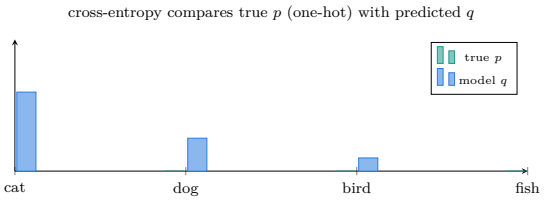
The **Kullback–Leibler divergence** is the excess:

$$\mathrm{KL}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = \mathrm{H}(p, q) - \mathrm{H}(p) \;\geq 0,$$

with equality iff $p = q$ (a consequence of Jensen's inequality, Chapter 6).

Key idea

KL divergence is the "distance" from your model q to the truth p — the information you waste by using the wrong distribution. Since $\mathrm{H}(p)$ is a fixed property of the data, **minimizing cross-entropy over the model q is exactly minimizing** $\mathrm{KL}(p \,\|\, q)$: pulling the model toward the data distribution. This is why "cross-entropy loss" and "maximum likelihood" and "minimize KL to the data" are three names for the same training objective. KL is *not* symmetric ($\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$), and which direction you minimize matters: one is mean-seeking, the other mode-seeking.



With a one-hot true label, cross-entropy collapses to $-\log q(\text{correct class})$ — punishing the model exactly for the probability it failed to assign to the right answer.

12.3 Mutual information

Definition 12.3. The **mutual information** between X and Y is how much knowing one reduces uncertainty about the other:

$$\mathrm{I}(X;Y) = \mathrm{KL}\left(p(x,y) \,\|\, p(x)p(y)\right) = \mathrm{H}(X) - \mathrm{H}(X \mid Y) \;\geq 0.$$

It is zero exactly when X and Y are independent, and unlike correlation it captures *nonlinear* dependence. It underlies feature selection, representation learning objectives (InfoNCE, the InfoMax principle), and the information-bottleneck view of deep networks.

How this powers ML / AI

Information theory supplies the objectives of modern ML almost verbatim. **Cross-entropy loss** — the default for classification and language modeling — is this chapter's $\mathrm{H}(p,q)$, and minimizing it is maximum likelihood / minimizing KL to the data. The **ELBO** that trains **VAEs** is an expected log-likelihood minus a KL term that regularizes the latent toward its prior. **Variational inference** minimizes KL between an approximate and true posterior. **Diffusion models** are trained with KL terms between forward and reverse steps. **Mutual information** drives contrastive self-supervised learning (InfoNCE) and disentanglement. **Knowledge distillation** matches a student's distribution to a teacher's via cross-entropy. When you minimize a loss in deep learning, you are almost always minimizing a divergence between distributions.

Try it in NumPy

```python
import numpy as np
# entropy, cross-entropy, KL (in nats)
p = np.array([0.5, 0.5])
q = np.array([0.9, 0.1])
H  = -(p*np.log(p)).sum()
```

```
CE = -(p*np.log(q)).sum()
KL = (p*np.log(p/q)).sum()
print(round(H,3), round(CE,3), round(KL,3))     # CE = H + KL; KL >= 0
print(round(CE - (H + KL), 6))                  # ~0, the identity holds

# classification cross-entropy with a one-hot label collapses to -log q[correct]
q_pred = np.array([0.6, 0.25, 0.1, 0.05]); correct = 0
print(round(-np.log(q_pred[correct]), 3))       # 0.511
```

Chapter summary

- **Entropy** $\mathrm{H}(p) = \mathbb{E}[-\log p]$ is average surprise / minimum code length; maximal for uniform, zero for certain.
- **Cross-entropy** $\mathrm{H}(p, q)$ scores model q on truth p; **KL** $= \mathrm{H}(p, q) - \mathrm{H}(p) \geq 0$ is the gap, zero iff $p = q$.
- Minimizing cross-entropy = minimizing $\mathrm{KL}(p\|q)$ = maximum likelihood — one objective, three names. KL is asymmetric.
- **Mutual information** measures (nonlinear) dependence and is zero iff independent.
- These define classification loss, the VAE ELBO, variational inference, diffusion training, contrastive learning, and distillation.

13

Probability and Statistics in Machine Learning, Deep Learning, and AI

Level: Capstone synthesis

Here is where everything converges. Machine learning is applied probability and statistics at scale: it posits a probabilistic model of data, estimates its parameters by (penalized) maximum likelihood, quantifies uncertainty, and generalizes from a sample to a population. Linear algebra and calculus give us the *tools* to compute; probability and statistics give us the *reason* those computations are the right ones. This chapter assembles the whole picture and shows that the training of any model — from a coin estimator to a diffusion transformer — is one statistical idea repeated.

13.1 Learning is statistical inference

A model defines a probability $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ of outputs given inputs and parameters. Training estimates $\boldsymbol{\theta}$ from data $\mathcal{D}$; prediction marginalizes or maximizes over outputs. The two estimation philosophies of this book map directly onto two views of training:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) \qquad \text{vs.} \qquad p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}).$$

The first is frequentist point estimation (most of deep learning); the second is the Bayesian posterior (uncertainty-aware ML). Both start from the same likelihood.

13.2 Where the loss functions come from

Every standard loss is a negative log-likelihood under some noise model (Chapters 7, 10):

Probabilistic model	**= Loss (NLL)**	**Used by**
Gaussian targets	mean-squared error	regression, autoencoders
Bernoulli / Categorical	cross-entropy	classifiers, language models
Poisson counts	Poisson loss	count regression
Laplace noise	mean-absolute error	robust regression

Key idea

Choosing how you model the randomness chooses your loss function. Cross-entropy is not an arbitrary formula — it is the negative log-likelihood of a categorical model and equivalently the KL divergence from the model to the data (Chapter 12). Squared error

is the NLL of Gaussian noise. So minimizing a loss is maximizing likelihood is minimizing a divergence to the true data distribution. This single thread ties Chapters 7, 10, and 12 together and runs underneath all supervised learning.

13.3 Regularization is a prior

Pure MLE overfits (the variance side of the bias–variance tradeoff). Real training adds a penalty, which Bayes reveals to be a prior (Chapter 9):

$$\underbrace{\min_{\boldsymbol{\theta}} \; -\log p(\mathcal{D} \mid \boldsymbol{\theta})}_{\text{MLE / data fit}} \; + \; \underbrace{\lambda\, R(\boldsymbol{\theta})}_{-\log p(\boldsymbol{\theta}) \;=\; \text{prior}} .$$

- **L2 / weight decay** = Gaussian prior $\boldsymbol{\theta} \sim \mathcal{N}(0, \tau^2 I)$ (ridge).
- **L1 / lasso** = Laplace prior (induces sparsity).
- **Dropout** $\approx$ Bayesian model averaging; **label smoothing** = a prior on labels; **early stopping** $\approx$ implicit shrinkage.

Every regularizer is a statement of prior belief about the parameters.

13.4 Generalization: from sample to population

A model is trained on a sample but judged on the population — a purely statistical leap. The **bias–variance tradeoff** (Chapter 10) and concentration inequalities (Chapter 6) describe when it succeeds; **cross-validation** estimates population error; **confidence intervals and significance tests** (Chapter 8) keep model comparisons honest. "This model generalizes" is a claim that training error is a good estimate of true risk — exactly the kind of claim statistics was built to evaluate.

13.5 Uncertainty and calibration

A trustworthy model knows what it does not know. Two kinds of uncertainty matter:

- **Aleatoric** — irreducible noise in the data itself (the σ^2 term).
- **Epistemic** — reducible uncertainty about the parameters, captured by the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$.

Key idea

A model is **calibrated** if its stated confidence matches reality: among predictions made with 80% confidence, about 80% should be correct. Deep networks are notoriously *overconfident*; we measure the gap with **Expected Calibration Error** and fix it with **temperature scaling**, **Bayesian neural networks**, **MC dropout**, and **deep ensembles** — all of which approximate the Bayesian posterior predictive (Chapter 9) to turn raw scores into honest probabilities. In medicine, autonomy, and finance, calibration matters as much as accuracy.

13.6 Generative models: learning the whole distribution

Discriminative models learn $p(\mathbf{y} \mid \mathbf{x})$; **generative models** learn the full data distribution $p(\mathbf{x})$ and sample from it. Each major family is a probabilistic construction from this book:

- **VAEs**: maximize the ELBO — expected log-likelihood minus a KL to a Gaussian latent prior (Chapters 9, 12); trained with the reparameterization trick.

- **Diffusion models**: a Markov chain (Chapter 11) of Gaussian noising, reversed by a learned chain; the objective is a sum of KL/score-matching terms.
- **Autoregressive models** (GPT-style): factor $p(\mathbf{x}) = \prod_t p(x_t \mid x_{<t})$ by the chain rule (Chapter 2) and fit each factor by cross-entropy.
- **GANs**: a two-player game whose equilibrium minimizes a divergence between real and generated distributions.

13.7 The grand summary table

Probability / statistics concept	Ch.	Where it powers AI
Probability axioms; events	1	valid model outputs; softmax as a distribution
Conditional prob. & chain rule	2	autoregressive factorization $p(x) = \prod p(x_t \mid x_{<t})$
Bayes' theorem	2	the template for learning; spam/diagnosis; priors
Random variables; expectation	3	risk $= \mathbb{E}$[loss]; standardization / normalization
Distributions (zoo)	4	choosing a model = choosing a loss
Covariance; multivariate Gaussian	5	PCA, whitening, GPs, VAE/diffusion priors
LLN & CLT	6	mini-batch SGD; Monte Carlo; error bars
Maximum likelihood	7	cross-entropy and MSE losses
Bias–variance; CV	7,10	over/underfitting; capacity & regularization choices
Confidence & tests	8	honest benchmarking; A/B tests; bootstrapped metrics
Bayesian inference; MAP	9	weight decay (L2), lasso (L1), calibrated uncertainty
Regression / GLMs	10	linear/logistic/softmax = a net's output layer
Markov chains & MCMC	11	RL/MDPs; posterior sampling; diffusion chains
Entropy / cross-entropy / KL / MI	12	classification loss; ELBO; VI; contrastive learning

The one-paragraph thesis

Machine learning is the practice of inferring a probabilistic model from data and using it to predict under uncertainty. A model specifies a likelihood; maximum likelihood turns that into a loss (cross-entropy, squared error); a prior turns into a regularizer (weight decay, lasso); the bias–variance tradeoff and the limit theorems govern whether it generalizes; Bayes' theorem and the posterior predictive supply calibrated uncertainty; Markov chains and Monte Carlo generate samples; and information theory measures the gap between model and world. Probability says what to expect, statistics says what to believe, and together they are not a prerequisite to machine learning — they *are* machine learning, with linear algebra and calculus as the tools that make the computations run.

Where to go next

With the companion *Linear Algebra* and *Calculus* volumes, you now hold the full mathematical core of machine learning: the vectors and matrices that represent data, the derivatives that train models, and the probability and statistics that justify the whole enterprise. The natural next steps are statistical learning theory (generalization bounds), the optimization of large models, and the deep-learning curriculum proper — where these distributions, likelihoods, and divergences come alive in classifiers, language models, and generative AI. You now understand *why* learning works; the rest is building.

Chapter summary

- Learning is **statistical inference**: a likelihood model fit by (penalized) maximum likelihood, with prediction by marginalization/maximization.
- **Losses are negative log-likelihoods**; **regularizers are priors**; both follow from the modeling choice.
- **Generalization**, **calibration**, and honest **evaluation** are statistical claims governed by bias–variance, the CLT, and tests.
- **Generative models** (VAEs, diffusion, autoregressive, GANs) are direct constructions from this book's probability.
- Probability and statistics are not background for ML — they are its foundation, with linear algebra and calculus as the computational tools.

A

Distributions and Identities Reference

A compact reference for the notation, distributions, and identities used throughout the book.

A.1 Notation

Symbol	Meaning
$\Omega,\ A,\ A^c$	sample space, event, complement
$P(A),\ P(A \mid B)$	probability, conditional probability
$X,\ p(x),\ f(x),\ F(x)$	random variable, PMF, PDF, CDF
$\mathbb{E}[X],\ \mathrm{Var}[X],\ \sigma$	expectation, variance, standard deviation
$\mathrm{Cov}(X,Y),\ \mathrm{Corr}(X,Y)$	covariance, correlation
$\boldsymbol{\mu},\ \boldsymbol{\Sigma}$	mean vector, covariance matrix
$\mathcal{N}(\mu,\sigma^2)$	normal (Gaussian) distribution
$\hat{\theta},\ L(\theta)$	estimator, likelihood
$p(\theta \mid \mathcal{D})$	posterior distribution
$\mathrm{H}(p),\ \mathrm{KL}(p\|q),\ \mathrm{I}(X;Y)$	entropy, KL divergence, mutual information
$X \perp\!\!\!\perp Y$	independence

A.2 Core identities

$$P(A \cup B) = P(A) + P(B) - P(A \cap B), \qquad P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

Bayes: $P(H \mid E) = \dfrac{P(E \mid H)P(H)}{P(E)}$, **Total prob.:** $P(E) = \displaystyle\sum_i P(E \mid H_i)P(H_i)$.

$$\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y], \qquad \mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2,$$

$$\mathrm{Var}[aX + b] = a^2\,\mathrm{Var}[X], \qquad \mathrm{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

LLN: $\bar{X}_n \to \mu$, **CLT:** $\dfrac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \to \mathcal{N}(0,1)$.

A.3 Distribution reference

Distribution	PMF/PDF	Mean	Variance	Use
Bernoulli(p)	$p^x(1-p)^{1-x}$	p	$p(1-p)$	binary label
Binomial(n,p)	$\binom{n}{k}p^k(1-p)^{n-k}$	np	$np(1-p)$	success counts
Poisson(λ)	$\frac{\lambda^k e^{-\lambda}}{k!}$	λ	λ	rare events
Uniform(a,b)	$\frac{1}{b-a}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$	RNG, priors
Exp(λ)	$\lambda e^{-\lambda x}$	$1/\lambda$	$1/\lambda^2$	waiting time
$\mathcal{N}(\mu,\sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2	noise, CLT
Beta(α,β)	$\propto x^{\alpha-1}(1-x)^{\beta-1}$	$\frac{\alpha}{\alpha+\beta}$	—	rate prior

A.4 Inference & information cheat-sheet

$$\hat{\theta}_{\text{MLE}} = \arg\max_\theta \sum_i \log p(x_i \mid \theta), \qquad \hat{\theta}_{\text{MAP}} = \arg\max_\theta \Big[\sum_i \log p(x_i \mid \theta) + \log p(\theta)\Big].$$

$$\text{L2 / weight decay} \;\equiv\; \text{Gaussian prior}, \qquad \text{L1 / lasso} \;\equiv\; \text{Laplace prior}.$$

$$\text{MSE}(\hat{\theta}) = \text{Bias}(\hat{\theta})^2 + \text{Var}(\hat{\theta}), \qquad 95\%\text{ CI (mean):}\;\; \bar{X} \pm 1.96\,\frac{s}{\sqrt{n}}.$$

$$\text{H}(p) = -\sum p\log p, \quad \text{H}(p,q) = -\sum p\log q, \quad \text{KL}(p\|q) = \text{H}(p,q) - \text{H}(p) \geq 0.$$

$$\text{minimize cross-entropy} \iff \text{maximize likelihood} \iff \text{minimize KL}(p\,\|\,q).$$

A.5 Minimal NumPy / SciPy reference

Task	Call
sample / pdf / cdf	`scipy.stats.<dist>.rvs / pdf / cdf`
mean, var, std	`np.mean`, `np.var`, `np.std`
covariance / correlation	`np.cov`, `np.corrcoef`
multivariate normal	`np.random.multivariate_normal`
t-test / χ^2 test	`scipy.stats.ttest_ind`, `chi2_contingency`
fit by MLE (optimize NLL)	`scipy.optimize.minimize`
bootstrap CI	resample with `np.random.choice(..., replace=True)`

End of book. Probability says what to expect; statistics says what to believe.