BAYESIAN INFERENCE

& NAIVE BAYES — FROM BEGINNER TO EXPERT

A complete, visual, application-driven course — from Bayes' theorem and the prior–posterior update through conjugacy, Naive Bayes, graphical models, and MCMC, with a capstone on how Bayesian thinking powers Machine Learning, Deep Learning, and AI.

What you will be able to do

Update beliefs with Bayes' theorem; choose priors and derive posteriors by conjugacy; build and apply Naive Bayes classifiers; reason with Bayesian networks; estimate posteriors with MCMC and variational inference; compare models with the evidence; and explain why Bayesian thinking underlies regularization, uncertainty, Bayesian optimization, and modern probabilistic AI.

Format: self-paced reference & course | **Prerequisites:** basic probability, a little calculus

Part of the Comprehensive Machine Learning & Deep Learning Curriculum.

Preface: how to use this book

There is one equation behind this entire book, and it fits on a single line:

$$P(\mathcal{H} \mid D) = \frac{P(D \mid \mathcal{H})\, P(\mathcal{H})}{P(D)}\,.$$

Bayes' theorem is a recipe for learning from evidence: start with what you believe (the *prior*), see how well each possibility explains the data (the *likelihood*), and end with an updated belief (the *posterior*). That is all of Bayesian inference — and, remarkably, it is also a lens through which much of modern machine learning comes into focus. Regularization is a prior. The loss is a negative log-likelihood. Uncertainty is a posterior. A spam filter is Bayes' theorem with an independence shortcut. This book builds the idea from a coin flip all the way to Bayesian neural networks.

Who this is for

This book starts from the very beginning — what a probability *means* — and climbs to conjugate analysis, Naive Bayes classifiers, Bayesian networks, Markov chain Monte Carlo, variational inference, hierarchical models, and Bayesian model comparison. It is for the self-learner who wants both **fluency** ("I can set up a prior, compute a posterior, and make a decision") and **understanding** ("I know why the posterior is a compromise, when the prior matters, and how this connects to the rest of statistics and ML"). Every major idea gets a picture, a worked example, and a forward link to machine learning.

The central idea

All of Bayesian reasoning orbits one move and its consequences:

- **Probability is degree of belief.** We assign probabilities to hypotheses, not just to repeatable events — so we can talk about the probability that a parameter lies in a range, or that a model is true.
- **Learning is conditioning.** New data updates belief by Bayes' theorem: posterior $\propto$ likelihood $\times$ prior. Yesterday's posterior is today's prior.
- **Answers are distributions, not points.** The output is a full posterior, carrying its own uncertainty — from which we extract estimates, intervals, predictions, and decisions.

The structure

- **Part I — Foundations** (beginner): Bayesian thinking; Bayes' theorem and its classic puzzles.
- **Part II — The Bayesian Engine** (core): prior, likelihood, and posterior; conjugate updating; choosing priors; summarizing the posterior and making decisions.
- **Part III — Classification & Comparison** (advanced): Naive Bayes classifiers; Bayesian versus frequentist inference.
- **Part IV — Structure & Computation** (advanced/expert): Bayesian networks; MCMC and variational inference; Bayesian regression and hierarchical models.
- **Part V — Frontier**: model comparison and Occam's razor; and a capstone on Bayes 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*, work it before reading on; 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 or analogy behind the math.
- **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 / scikit-learn).

Key idea

Hold one picture above all others: a Bayesian **starts with a belief, sees data, and ends with a sharper belief**. The prior is where you begin, the likelihood is what the data says, and the posterior is the negotiated result — always a compromise between the two, tilting toward the data as evidence accumulates. Everything else — conjugacy, Naive Bayes, MCMC, Bayesian deep learning — is machinery for carrying out this one update when the math gets hard.

Contents

PART I

Foundations

1

What Is Bayesian Thinking?

Level: Beginner

Before any formula, Bayesian inference is a *way of reasoning*: hold beliefs as probabilities, and revise them in proportion to the evidence. This chapter sets up that worldview — probability as degree of belief, the contrast with the frequentist view, and the single update rule that the rest of the book makes precise — so that every later technique reads as a variation on one natural idea.

1.1 Two meanings of "probability"

Definition 1.1. There are two dominant interpretations of probability:

- **Frequentist:** a probability is the long-run frequency of an event in repeated trials. "The coin is fair" means it lands heads 50% of the time over infinitely many flips.
- **Bayesian:** a probability is a **degree of belief** about a proposition, given what you know. "There is a 70% chance it will rain tomorrow" or "a 95% chance this parameter lies between 2 and 4" — statements about *one-off* events and unknown quantities.

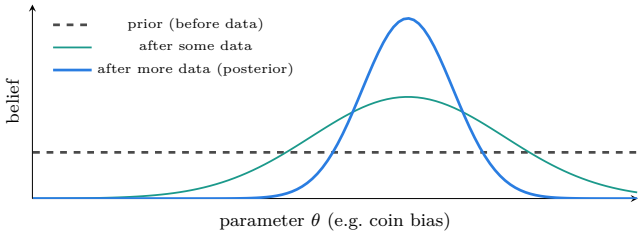
Key idea

The Bayesian view lets you assign probabilities to *hypotheses and parameters*, not just to repeatable events. This is the move that makes everything else possible: if a parameter can have a probability distribution, then data can *update* that distribution. "What is the probability this model is right?" is a meaningful Bayesian question and a forbidden frequentist one.

1.2 Learning as belief revision

Intuition

You already think like a Bayesian. You hear a strange noise at night; your prior says "probably the wind." You hear it again, rhythmic and close; the evidence shifts your belief toward "someone is at the door." You did not discard your prior — you *combined* it with new data. Bayesian inference formalizes exactly this: belief after = belief before, reweighted by how well each possibility predicts what you observed.



As evidence accumulates the belief curve marches toward the truth and *narrows* — uncertainty shrinking as data grows. That sharpening is the visual signature of Bayesian learning, and we will see it again and again.

1.3 The update rule, in words

Key idea

Every Bayesian analysis has the same three ingredients and one rule:

$$\underbrace{P(\mathcal{H} \mid D)}_{\text{posterior}} \propto \underbrace{P(D \mid \mathcal{H})}_{\text{likelihood}} \times \underbrace{P(\mathcal{H})}_{\text{prior}}.$$

Prior: what you believed about hypothesis $\mathcal{H}$ before. **Likelihood**: how well $\mathcal{H}$ predicts the data D. **Posterior**: your updated belief. The posterior is always a *compromise* between prior and likelihood — and as data accumulates, the likelihood dominates. Chapter 2 derives this; the whole book is its consequences.

1.4 Why think this way?

- **Uncertainty is first-class.** The answer is a distribution, so you always know how sure you are — crucial for risk, science, and safety-critical AI.
- **Prior knowledge is usable.** You can fold in physics, past studies, or expert judgment instead of starting from scratch.
- **It is coherent.** Under mild axioms (Cox, de Finetti), degree-of-belief reasoning *must* obey the probability rules — otherwise a clever bettor can guarantee a profit against you (a "Dutch book").
- **It updates sequentially.** Today's posterior is tomorrow's prior, so learning is naturally online and incremental.

Common pitfall

Bayesian reasoning is powerful but not magic. The conclusions depend on the prior you choose, so a poorly justified prior can bias results — you must be explicit and check sensitivity (Chapter 5). And exact computation is often intractable, which is why much of the field (Chapter 10) is about *approximating* the posterior. "Subjective prior" is a feature, not a bug, but it demands honesty.

How this powers ML / AI

This worldview is the backbone of probabilistic machine learning. Treating model parameters as random variables with priors gives **Bayesian neural networks** and **Gaussian processes**; the prior-as-belief idea is exactly **regularization** (Chapter 13); and "answers are distributions" is the foundation of **uncertainty quantification**, without which a

model cannot say "I don't know." Sequential updating underlies online learning, Bayesian optimization, and reinforcement learning. Even when a system is not explicitly Bayesian, the Bayesian lens explains *why* it works.

Try it in NumPy

```python
import numpy as np
from scipy.stats import beta
# Belief about a coin's bias theta, updated as flips arrive (Beta-Binomial, Ch.4)
a, b = 1, 1                        # flat prior: all biases equally plausible
flips = "HHTHHHTHHH"               # observed data
for f in flips:
    a += (f == "H"); b += (f == "T")
    lo, hi = beta.ppf([0.025, 0.975], a, b)
    print(f"after {f}: mean={a/(a+b):.2f}  95% CI=[{lo:.2f},{hi:.2f}]")
# Belief shifts toward heads-bias and the interval tightens with more data
```

Chapter summary

- **Frequentist** probability is long-run frequency; **Bayesian** probability is degree of belief — and can be assigned to hypotheses and parameters.
- Learning is **belief revision**: combine prior belief with the likelihood of the data to get a posterior.
- The master rule is **posterior** $\propto$ **likelihood** $\times$ **prior**; the posterior is a compromise that tilts toward the data as it accumulates.
- Benefits: first-class uncertainty, usable prior knowledge, coherence, and sequential updating.
- Caveats: results depend on the prior, and exact computation is often intractable — the motivations for Chapters 5 and 10.

2

Bayes' Theorem

Level: Beginner

Now we make the update rule exact. **Bayes' theorem** is a short consequence of the definition of conditional probability, but its implications are deep enough to have reshaped statistics, science, and AI. This chapter derives it, names its parts, and works the classic medical-test example that exposes the most common reasoning error about probability — the base-rate fallacy.

2.1 From conditional probability to Bayes

Definition 2.1. The **conditional probability** of A given B is $P(A \mid B) = \dfrac{P(A \cap B)}{P(B)}$.
Writing the joint probability two ways, $P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$, and rearranging gives **Bayes' theorem**:

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

Key idea

Bayes' theorem **reverses a conditional**. It converts $P(B \mid A)$ — often the thing we can model or measure (how the cause produces the effect) — into $P(A \mid B)$, the thing we want to know (the cause, given the observed effect). In inference we write it with a hypothesis $\mathcal{H}$ and data D:

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

2.2 The four parts

Prior $P(\mathcal{H})$.
Belief in the hypothesis before seeing data.

Likelihood $P(D \mid \mathcal{H})$.
How probable the observed data is *if* the hypothesis holds. (Read as a function of $\mathcal{H}$ for fixed D; it is not a probability distribution over $\mathcal{H}$.)

Evidence $P(D)$.
The total probability of the data under all hypotheses, $P(D) = \sum_i P(D \mid \mathcal{H}_i)P(\mathcal{H}_i)$ (or an integral). It is a normalizing constant ensuring the posterior sums to one.

Posterior $P(\mathcal{H} \mid D)$.
The updated belief — the goal of the analysis.

Intuition

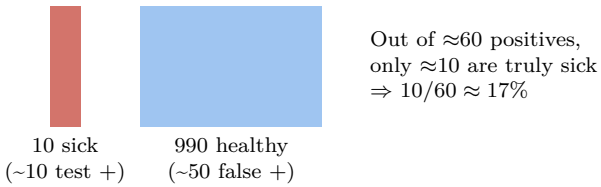
Because the evidence $P(D)$ does not depend on $\mathcal{H}$, it only rescales; the *shape* of the posterior comes entirely from likelihood × prior. That is why the working form of Bayes is the proportionality $P(\mathcal{H} \mid D) \propto P(D \mid \mathcal{H})P(\mathcal{H})$ — compute that, then normalize at the end. This single simplification saves enormous effort and is the reason conjugate priors (Chapter 4) are so convenient.

2.3 The medical-test example

Example 2.2. A disease affects 1% of a population. A test is 99% sensitive ($P(+ \mid \text{disease}) = 0.99$) and 95% specific ($P(- \mid \text{healthy}) = 0.95$, so the false-positive rate is 5%). You test positive. What is the probability you have the disease?

$$P(\text{dis} \mid +) = \frac{P(+ \mid \text{dis})P(\text{dis})}{P(+ \mid \text{dis})P(\text{dis}) + P(+ \mid \text{healthy})P(\text{healthy})} = \frac{0.99 \times 0.01}{0.99 \times 0.01 + 0.05 \times 0.99} \approx 0.167.$$

Despite the "99% accurate" test, there is only a **17%** chance you are actually sick.



Common pitfall

This is the **base-rate fallacy**: ignoring the prior $P(\mathcal{H})$ and confusing $P(D \mid \mathcal{H})$ with $P(\mathcal{H} \mid D)$. The test's accuracy ($P(+ \mid \text{dis})$) is *not* the chance of disease given a positive ($P(\text{dis} \mid +)$). When the base rate is low, even a good test produces mostly false positives, because there are so many more healthy people to generate them. Forgetting the base rate corrupts reasoning in medicine, law (the "prosecutor's fallacy"), and spam filtering alike — always anchor on the prior.

2.4 Sequential updating

Key idea

Bayes' theorem chains. Apply it to datum D_1 to get a posterior; treat that posterior as the prior for D_2; and so on. With independent data the result is the same as updating on all data at once:

$$P(\mathcal{H} \mid D_1, D_2) \propto P(D_2 \mid \mathcal{H})\, P(D_1 \mid \mathcal{H})\, P(\mathcal{H}).$$

Today's posterior is tomorrow's prior — which makes Bayesian learning naturally

online and incremental.

How this powers ML / AI

Bayes' theorem is the literal engine of many ML systems. The original **spam filters** scored emails by $P(\text{spam} \mid \text{words})$ via Bayes' rule (Chapter 7). **Bayesian filtering** (Kalman filters, particle filters) tracks robots and prices by repeated predict-update steps. In probabilistic ML the same equation links observations to latent variables, and the awkward denominator $P(D)$ — the *evidence* — is precisely the intractable quantity that MCMC and variational inference (Chapter 10) exist to approximate, and that doubles as the score for Bayesian model comparison (Chapter 12).

Try it in NumPy

```
def posterior_disease(prior, sens, spec):
    p_pos = sens*prior + (1-spec)*(1-prior)          # total prob of
    testing +
    return sens*prior / p_pos
print(round(posterior_disease(0.01, 0.99, 0.95), 3))   # 0.167
# Retest positive again -> yesterday's posterior becomes today's
    prior
p1 = posterior_disease(0.01, 0.99, 0.95)
print(round(posterior_disease(p1, 0.99, 0.95), 3))     # ~0.799 after
    2nd positive
```

Chapter summary

- Bayes' theorem, $P(\mathcal{H} \mid D) = P(D \mid \mathcal{H})P(\mathcal{H})/P(D)$, **reverses a conditional**: from "effect given cause" to "cause given effect."
- Its parts are **prior**, **likelihood**, **evidence** (normalizer), and **posterior**; in practice use posterior $\propto$ likelihood $\times$ prior.
- The **base-rate fallacy**: a 99%-accurate test on a 1%-prevalent disease yields only $\sim$17% posterior — the prior cannot be ignored.
- Bayes **chains**: today's posterior is tomorrow's prior, enabling online learning.
- The evidence $P(D)$ is the hard-to-compute term that motivates MCMC/VI and scores models.

PART II

The Bayesian Engine

3

The Prior, the Likelihood, and the Posterior

Level: Core

In Chapters 1–2 the hypotheses were discrete (sick or healthy). The real power of Bayesian inference appears when the unknown is a *continuous parameter* — a probability, a rate, a mean — so the prior and posterior become *distributions over a parameter*. This chapter is the conceptual engine of the book: how the three ingredients combine for continuous parameters, why the posterior is a compromise, and how it sharpens with data.

3.1 Inference about a parameter

Definition 3.1. Let θ be an unknown parameter (say, the probability a coin lands heads). Bayesian inference produces the full **posterior density**

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)}, \qquad p(D) = \int p(D \mid \theta)\, p(\theta)\, d\theta,$$

where $p(\theta)$ is the prior density, $p(D \mid \theta)$ the likelihood, and the integral $p(D)$ the evidence. The answer is the whole curve $p(\theta \mid D)$, not a single number.

3.2 The likelihood function

Key idea

Fix the data and view $p(D \mid \theta)$ as a function of θ: this is the **likelihood function**. It scores every parameter value by how well it predicts what we saw. For n independent observations it is a product,

$$p(D \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta),$$

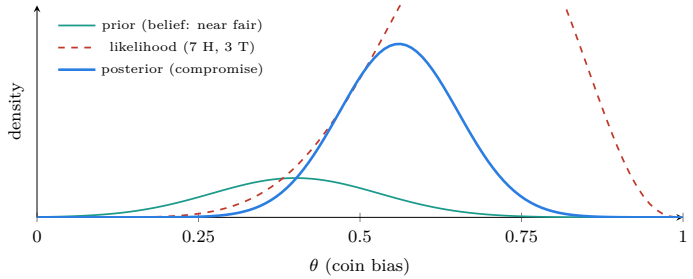
so we usually work with the *log*-likelihood (sums beat products numerically). The likelihood is the data's entire contribution to inference — the bridge between model and evidence.

Intuition

For a coin with h heads in n flips, the likelihood is $\theta^h(1-\theta)^{n-h}$ — peaked at $\theta = h/n$, the value that best explains the data. The *maximum likelihood estimate* (MLE) sits at that peak. A Bayesian does not just take the peak; they multiply the whole curve by the prior

and keep the result as a distribution, retaining the uncertainty the peak alone discards.

3.3 Prior × likelihood = (unnormalized) posterior



Key idea

The posterior sits **between** the prior and the likelihood — a weighted compromise. Its location is pulled toward whichever is more confident (more sharply peaked), and its spread is narrower than either, because combining two sources of information always reduces uncertainty. This single picture explains the entire behavior of Bayesian updating.

3.4 How the posterior evolves with data

Intuition

Two limits make the compromise concrete:

- **Little data:** the likelihood is broad, so the prior matters a lot — the posterior stays close to the prior.
- **Much data:** the likelihood becomes sharp and dominates; the posterior concentrates near the MLE and the prior's influence fades (the *Bernstein–von Mises* phenomenon).

So priors are training wheels: indispensable when data is scarce, almost irrelevant once data is abundant. This is why reasonable people with different priors converge once they see enough evidence.

3.5 The predictive distribution

Definition 3.2. To predict a new observation $\tilde{x}$, average the likelihood over the posterior — the **posterior predictive distribution**:

$$p(\tilde{x} \mid D) = \int p(\tilde{x} \mid \theta)\, p(\theta \mid D)\, d\theta.$$

Key idea

This is a defining Bayesian move: instead of plugging in one "best" θ, we **marginalize** (average) over all values weighted by their posterior probability. Predictions therefore inherit parameter uncertainty — a Bayesian forecast is automatically wider when the parameter is poorly known, which is exactly the calibrated behavior we want.

Common pitfall

Do not collapse the posterior to its peak too early. Reporting only the MAP or posterior mean throws away the uncertainty that is the whole point; and *plugging* a point estimate into predictions (rather than marginalizing) systematically *understates* predictive uncertainty. Keep the distribution as long as you can, and integrate, not maximize, when you predict.

How this powers ML / AI

The likelihood is the same object that supervised learning minimizes: **negative log-likelihood is the loss function** (MSE for Gaussian, cross-entropy for categorical). Maximum likelihood is the non-Bayesian special case of taking the likelihood peak; adding a prior and taking the peak gives **MAP = regularized training** (Chapter 13). And the posterior-predictive "marginalize, don't maximize" principle is precisely what **Bayesian neural networks**, **deep ensembles**, and **MC-dropout** approximate to get trustworthy uncertainty — averaging predictions over many plausible parameter settings rather than betting on one.

Try it in NumPy

```
import numpy as np
from scipy.stats import beta
h, n = 7, 10                          # 7 heads in 10 flips
grid = np.linspace(0, 1, 1001)
prior = beta(4, 4).pdf(grid)          # belief: probably near fair
like  = grid**h * (1-grid)**(n-h)     # binomial likelihood (unnormalized)
post  = prior * like
post /= np.trapz(post, grid)          # normalize via the evidence integral
print("posterior mean:", round(np.trapz(grid*post, grid), 3))   # compromise value
# Posterior-predictive prob the next flip is heads = posterior mean of theta
```

Chapter summary

- For a continuous parameter, prior and posterior are **densities**; $p(\theta \mid D) \propto p(D \mid \theta)p(\theta)$ with evidence $p(D) = \int p(D \mid \theta)p(\theta)d\theta$.
- The **likelihood** $p(D \mid \theta)$ scores parameters by fit; its peak is the MLE, and it is a product (use log-likelihood).
- The posterior is a **compromise** between prior and likelihood, pulled toward the sharper one and narrower than both.
- With more data the likelihood dominates and the prior's influence fades; priors matter most when data is scarce.
- Predict by the **posterior predictive** — marginalize over θ, don't plug in one value — which is what Bayesian deep learning approximates.

4

Conjugate Priors and Updating

Level: Core

Computing the posterior means doing the integral $p(D) = \int p(D \mid \theta)p(\theta)\, d\theta$ — usually hard. **Conjugate priors** are a beautiful special case where the integral is free: prior and posterior belong to the same family, so updating is just *arithmetic on the parameters*. The three classic conjugate pairs in this chapter are the workhorses of analytic Bayes and the clearest illustration of "the posterior is prior plus data."

4.1 Conjugacy

Definition 4.1. A prior is **conjugate** to a likelihood if the resulting posterior is in the *same family* as the prior. Then Bayesian updating reduces to mapping the prior's parameters to the posterior's parameters — no integration required.

Key idea

Conjugacy turns the calculus of Bayes' theorem into bookkeeping. You start with a prior parameterized by a few numbers ("pseudo-counts"), you observe data summarized by a few sufficient statistics, and the posterior parameters are simply prior + data. The intuition is irresistible: *the prior acts like data you have already seen.*

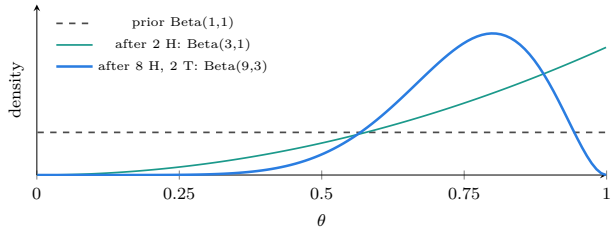
4.2 Beta–Binomial (a probability)

Definition 4.2. For a success probability θ with a Binomial likelihood, the conjugate prior is the **Beta** distribution. With prior $\theta \sim \text{Beta}(\alpha, \beta)$ and data of h successes in n trials:

$$\theta \mid D \;\sim\; \text{Beta}(\alpha + h,\ \beta + n - h).$$

Intuition

Read α as "prior successes" and β as "prior failures" — pseudo-counts encoding belief before data. Updating just *adds the observed counts*. The posterior mean $\frac{\alpha+h}{\alpha+\beta+n}$ is a weighted average of the prior mean and the data proportion h/n, with the prior's weight set by $\alpha + \beta$ (its "confidence in trials"). A flat $\text{Beta}(1, 1)$ prior gives posterior mean $\frac{h+1}{n+2}$ — *Laplace's rule of succession*, the original smoothing.



4.3 Gamma–Poisson (a rate)

Definition 4.3. For a rate λ of events with a Poisson likelihood, the conjugate prior is the **Gamma**. With prior $\lambda \sim \text{Gamma}(\alpha, \beta)$ (shape α, rate β) and observed total count $\sum x_i$ over n periods:

$$\lambda \mid D \;\sim\; \text{Gamma}\Big(\alpha + \textstyle\sum_i x_i,\; \beta + n\Big).$$

Intuition

Here α is "prior total events" and β is "prior periods observed." Updating adds the events seen to α and the periods elapsed to β — again, prior pseudo-counts plus data. The posterior mean rate $\frac{\alpha+\sum x_i}{\beta+n}$ blends the prior rate with the empirical rate. Use this for counts: server requests per minute, shark attacks per year, photons per pixel.

4.4 Normal–Normal (a mean)

Definition 4.4. For an unknown mean μ with known variance σ^2 and a Normal likelihood, the conjugate prior is **Normal**. With prior $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ and n observations of sample mean $\bar{x}$, the posterior is Normal with precision (inverse variance) *adding*:

$$\frac{1}{\tau_n^2} = \frac{1}{\tau_0^2} + \frac{n}{\sigma^2}, \qquad \mu_n = \tau_n^2\left(\frac{\mu_0}{\tau_0^2} + \frac{n\bar{x}}{\sigma^2}\right).$$

Key idea

Precisions add, and the posterior mean is a precision-weighted average of the prior mean and the data mean. Each new observation adds $1/\sigma^2$ to the precision, so certainty grows linearly with n and the posterior standard deviation shrinks like $1/\sqrt{n}$ — the Bayesian echo of the standard error.

4.5 Why it works: the exponential family

Conjugacy is not a coincidence: every likelihood in the **exponential family** (Bernoulli, Binomial, Poisson, Normal, Gamma, …) has a conjugate prior, because multiplying prior and likelihood just adds their natural parameters. This is the same exponential-family structure that unifies generalized linear models — statistics' deep patterns rhyme.

Common pitfall

Conjugate priors are chosen for *mathematical convenience*, which can conflict with honest belief. If your real prior knowledge is not Beta/Gamma/Normal-shaped, forcing conjugacy

distorts it. Conjugacy also covers only simple models; realistic multi-parameter models rarely have closed-form posteriors — which is exactly why MCMC and variational inference (Chapter 10) were invented. Treat conjugacy as a valuable special case and a sanity check, not the whole toolbox.

How this powers ML / AI

Conjugate updates are the analytic core of many ML methods. **Naive Bayes** text classifiers use Beta/Dirichlet smoothing of word counts — Laplace smoothing *is* a Beta prior (Chapter 7). **Thompson sampling**, the elegant solution to multi-armed bandits and the basis of online recommendation and A/B testing, maintains a Beta posterior per arm and updates it with one addition per observation. **Bayesian optimization**, **Kalman filters** (Normal–Normal updates in disguise), and **latent Dirichlet allocation** for topic modeling all lean on conjugacy for fast, closed-form belief updates.

Try it in NumPy

```
from scipy.stats import beta, gamma
# Beta-Binomial: prior Beta(2,2), observe 8 heads in 10
a, b = 2 + 8, 2 + (10 - 8)
print("Beta-Binomial posterior mean:", round(a/(a+b), 3))     # 0.571
# Gamma-Poisson: prior Gamma(2, 1), observe 30 events over 10 periods
shape, rate = 2 + 30, 1 + 10
print("Gamma-Poisson posterior mean rate:", round(shape/rate, 3))  #
    2.909
```

Chapter summary

- A **conjugate** prior yields a posterior in the same family, so updating is parameter arithmetic — no integral.
- **Beta–Binomial**: $\text{Beta}(\alpha, \beta) \to \text{Beta}(\alpha + h, \beta + n - h)$; α, β are pseudo-counts; flat prior gives Laplace's rule.
- **Gamma–Poisson**: $\text{Gamma}(\alpha, \beta) \to \text{Gamma}(\alpha + \sum x, \beta + n)$ for rates.
- **Normal–Normal**: precisions add; the posterior mean is a precision-weighted average; SD shrinks like $1/\sqrt{n}$.
- Conjugacy follows from the **exponential family**; it is convenient but limited, motivating MCMC/VI. It underlies Naive Bayes smoothing, Thompson sampling, and Kalman filters.

5

Choosing Priors

Level: Core

The prior is what makes Bayesian inference both powerful and controversial: it lets you inject knowledge, but it also forces you to be explicit about your assumptions. This chapter is a practical guide to choosing priors — informative, weakly informative, and "non-informative" — and to the honest practice of checking that your conclusions do not hinge on an arbitrary choice.

5.1 A spectrum of priors

Definition 5.1. Priors range from strongly opinionated to deliberately vague:

- **Informative:** encodes real knowledge (past studies, physics, expert elicitation). Narrow; strongly shapes the posterior when data is limited.
- **Weakly informative:** rules out the absurd (e.g. a human height is not 5 m) but stays open. The modern default — it regularizes gently without dominating.
- **Non-informative / reference:** aims to "let the data speak" (flat, Jeffreys). Useful as a baseline, but trickier than it looks.

Key idea

There is no such thing as *no* prior — choosing to do frequentist maximum likelihood is itself the choice of a flat prior. The real question is not "prior or no prior?" but "*which* prior, and how much does it matter?" A good default is a **weakly informative** prior: confident enough to stabilize estimates and prevent nonsense, humble enough to be overruled by data.

5.2 Non-informative priors and their subtleties

Intuition

A **flat** prior ($p(\theta) \propto 1$) looks neutral but is not: flatness is not preserved under reparameterization, so "uniform on θ" is informative about θ^2 or $\log\theta$. The **Jeffreys prior**, $p(\theta) \propto \sqrt{I(\theta)}$ (the square root of Fisher information), fixes this by being invariant to how you parameterize — for a Binomial probability it is Beta$(\frac{1}{2}, \frac{1}{2})$, gently favoring the extremes. Such priors can also be **improper** (not integrate to one); that is acceptable only if the resulting posterior is proper.

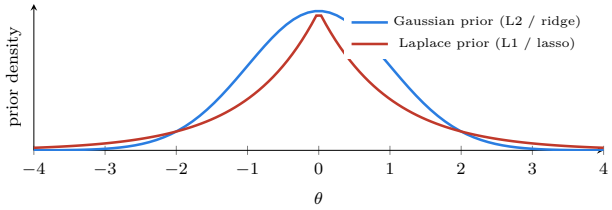
5.3 Priors as regularization

Key idea

A prior pulls estimates toward sensible values — which is exactly what regularization does. The posterior-mode (MAP) estimate is

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \; \big[\log p(D \mid \theta) + \log p(\theta)\big],$$

i.e. log-likelihood *plus* a penalty $\log p(\theta)$. A **Gaussian prior** gives the L_2 penalty (ridge / weight decay); a **Laplace prior** gives the L_1 penalty (lasso / sparsity). Every penalty you add to a loss is, secretly, a prior belief about the parameters.



5.4 Eliciting and stress-testing priors

A practical workflow:

1. **Put parameters on an interpretable scale** (standardize predictors) so a generic prior like $\mathcal{N}(0, 1)$ is meaningful.
2. **Use prior predictive checks:** simulate fake data from the prior; if it produces impossible datasets, the prior is wrong.
3. **Run a sensitivity analysis:** refit with a few different reasonable priors and confirm the conclusions are stable.

Common pitfall

Two opposite errors. (1) An **overconfident prior** (too narrow, centered wrong) can stubbornly override the data and bias the posterior — dangerous when you do not actually have that much prior knowledge. (2) A **vague/flat prior** is not automatically "objective": it can put most of its mass on absurd values, cause improper posteriors, or behave badly under transformation. When a result changes drastically with a slightly different reasonable prior and little data, report that fragility rather than hiding it.

How this powers ML / AI

The prior-as-regularizer equivalence is one of the most useful bridges between Bayesian statistics and deep learning. **Weight decay** is a zero-mean Gaussian prior on the weights; L_1 **regularization** is a Laplace prior driving sparsity; **early stopping** and **dropout** act as implicit priors. In **Bayesian neural networks** the choice of weight prior (Gaussian, heavy-tailed Student-t, or function-space priors) is an active research area precisely because, with millions of parameters and finite data, the prior still shapes generalization. Understanding priors lets you read regularization choices as statements of belief.

Try it in NumPy

```python
import numpy as np
from scipy.stats import beta
h, n = 2, 3                              # tiny dataset: prior will matter a lot
grid = np.linspace(0, 1, 1001)
for (a, b, name) in [(1,1,"flat"), (0.5,0.5,"Jeffreys"), (20,20,"strong=fair")]:
    post = beta(a+h, b+n-h)
    print(f"{name:>12} prior -> posterior mean {post.mean():.3f}")
# Conclusions differ with little data -> always do this sensitivity check
```

Chapter summary

- Priors span **informative**, **weakly informative** (the modern default), and **non-informative/reference**.
- There is always a prior; flat priors are not truly neutral (not reparameterization-invariant), which motivates the **Jeffreys** prior.
- **MAP = penalized likelihood**: Gaussian prior $\to$ L_2/ridge, Laplace prior $\to$ L_1/lasso — penalties are priors.
- Elicit on interpretable scales; use **prior predictive checks** and **sensitivity analysis**.
- Beware overconfident priors (bias) and vague priors (absurd mass, impropriety); report fragility honestly.

6

Summarizing the Posterior and Bayesian Decisions

Level: Core

The posterior is a full distribution, but people need answers: a single estimate, an interval, a yes/no decision. This chapter shows how to *summarize* a posterior responsibly (point estimates and credible intervals) and how Bayesian **decision theory** converts beliefs into optimal actions by minimizing expected loss — the principled bridge from "what do I believe?" to "what should I do?"

6.1 Point estimates

Definition 6.1. Three common single-number summaries of a posterior $p(\theta \mid D)$:

- **Posterior mean** $\mathbb{E}[\theta \mid D]$ — minimizes expected squared error.
- **Posterior median** — minimizes expected absolute error; robust to skew.
- **Posterior mode (MAP)** $\arg\max_\theta p(\theta \mid D)$ — the single most probable value.

Intuition

Which to report depends on the *loss* you care about (next section) and the posterior's shape. For a symmetric, unimodal posterior all three coincide. For a skewed posterior they diverge, and the mean can sit in a low-density region — so always look at the whole distribution before collapsing it. The MAP is the cheapest to compute (just optimize), which is why it dominates large-scale ML, but it ignores the posterior's width.

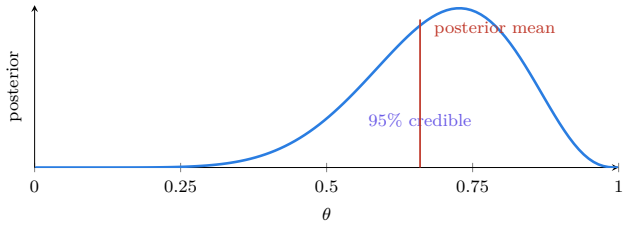
6.2 Credible intervals

Definition 6.2. A 95% **credible interval** is a range containing 95% of the posterior probability. Two common forms: the *equal-tailed* interval (2.5% in each tail) and the **highest posterior density (HPD)** interval — the shortest interval containing 95% of the mass.

Key idea

A credible interval means exactly what people *wish* a confidence interval meant: "given the data and prior, there is a 95% probability the parameter lies in this range." That is a direct probability statement about θ — legitimate because, to a Bayesian, θ has a distribution. Contrast this with the frequentist confidence interval (Chapter 8), whose 95% refers to the

long-run coverage of the *procedure*, not to your one interval.



6.3 Bayesian decision theory

Definition 6.3. A **loss function** $L(\theta, a)$ gives the cost of taking action a when the truth is θ. The **Bayes action** minimizes the *posterior expected loss*:

$$a^\star = \arg\min_a \ \mathbb{E}_{\theta|D}\big[L(\theta, a)\big] = \arg\min_a \int L(\theta, a)\, p(\theta \mid D)\, d\theta.$$

Key idea

This is the payoff of being Bayesian: a single, coherent recipe for optimal action under uncertainty — *average the loss over everything you don't know, then pick the best action.* Different losses recover different estimators automatically: squared-error loss ⇒ posterior mean; absolute-error loss ⇒ posterior median; 0/1 loss ⇒ posterior mode. The estimate is not arbitrary; it is dictated by what mistakes cost you.

Intuition

Decisions should depend on consequences, not just beliefs. If missing a disease is far worse than a false alarm, the optimal action treats at a posterior probability well below 50%. Bayesian decision theory makes this asymmetry explicit through the loss function, rather than hiding it in an arbitrary threshold — the same logic that sets classification thresholds in ML.

Common pitfall

Beware reporting the **MAP as if it summarized the posterior**: in high dimensions the mode can be wildly unrepresentative of where the mass actually is (the "mean is not the mode" problem, severe for skewed or multimodal posteriors). Also, a credible interval is only as trustworthy as the model and prior that produced it — 95% credibility under a bad model is false comfort. And do not confuse credible intervals (Bayesian, about θ) with confidence intervals (frequentist, about the procedure); they answer different questions.

How this powers ML / AI

Decision theory is the hidden backbone of applied ML. Choosing a **classification threshold** to balance precision and recall is minimizing expected loss under an asymmetric cost matrix. **Bayes-optimal prediction** — the theoretical best any classifier can do — is exactly the Bayes action under 0/1 loss. **Active learning** and **Bayesian optimization** choose the next query to minimize expected future loss (maximize expected information or improvement). And in reinforcement learning, acting to maximize expected return is

decision theory with the posterior over outcomes. Whenever an AI system "decides," it is (ideally) computing a Bayes action.

Try it in NumPy

```python
import numpy as np
from scipy.stats import beta
post = beta(9, 4)                          # example posterior over theta
print("mean:", round(post.mean(), 3), " median:", round(post.median(), 3))
lo, hi = post.ppf([0.025, 0.975])          # 95% equal-tailed credible interval
print("95% CI:", (round(lo,3), round(hi,3)))
# Decision: treat if expected loss of not-treating exceeds that of treating
c_fn, c_fp = 10.0, 1.0                      # cost of false negative vs false positive
p_disease = 1 - post.cdf(0.5)               # posterior prob theta>0.5, say
print("treat?", c_fn*p_disease > c_fp*(1-p_disease))
```

Chapter summary

- Summarize a posterior with the **mean** (squared-error optimal), **median** (absolute-error optimal), or **mode/MAP** (0/1 optimal).
- A **credible interval** is a direct probability statement about θ — what a confidence interval is often mistaken for.
- **Bayesian decision theory**: the optimal action minimizes *posterior expected loss*; the loss determines the estimator.
- Asymmetric losses justify thresholds away from 50% — decisions follow consequences, not just beliefs.
- Beware the MAP in high dimensions and credibility under a bad model; this machinery underlies thresholds, Bayes-optimal prediction, and active learning.

PART III

Classification and Comparison

7

Naive Bayes Classifiers

Level: Advanced

Naive Bayes is Bayes' theorem turned into a fast, surprisingly effective classifier. Despite an assumption that is almost always false — that features are independent given the class — it powers spam filters, document classifiers, and countless baselines, often beating far more complex models on text. This chapter explains the model, its three flavors, the smoothing that makes it robust, and exactly why "naive" still works.

7.1 From Bayes' theorem to a classifier

Definition 7.1. To classify an input with features $\mathbf{x} = (x_1, \ldots, x_d)$ into class C, apply Bayes' theorem:

$$P(C \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C)\, P(C)}{P(\mathbf{x})} \;\propto\; P(C)\, P(\mathbf{x} \mid C).$$

The **naive** assumption is conditional independence of features given the class:

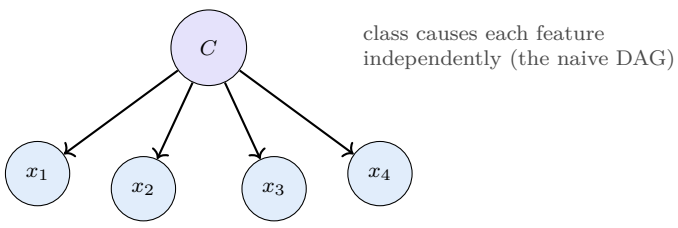
$$P(\mathbf{x} \mid C) = \prod_{j=1}^{d} P(x_j \mid C),$$

so the prediction is

$$\hat{C} = \arg\max_{C} \; P(C) \prod_{j=1}^{d} P(x_j \mid C).$$

Key idea

The naive assumption replaces one impossibly hard quantity — the joint distribution of all features given the class — with a *product of easy one-dimensional ones*. Estimating $P(x_j \mid C)$ for each feature separately needs little data and no high-dimensional density estimation. That is the entire trick: trade a false-but-convenient independence assumption for enormous statistical and computational savings.



7.2 Three flavors

Gaussian NB (continuous features).
Model each $P(x_j \mid C)$ as a Gaussian; estimate per-class mean and variance. Good for real-valued data.

Multinomial NB (counts).
Model features as word/event counts; $P(x_j \mid C)$ comes from class word frequencies. The classic for document/topic classification.

Bernoulli NB (binary).
Features are present/absent flags; explicitly models non-occurrence too. Good for short texts and presence-of-word features.

7.3 Laplace smoothing

Key idea

If a word never appears in the training spam, $P(\text{word} \mid \text{spam}) = 0$, and because the model *multiplies* probabilities, a single zero annihilates the whole product — one unseen feature vetoes the class. **Laplace (add-α) smoothing** fixes this by adding pseudo-counts:

$$P(x_j = w \mid C) = \frac{\text{count}(w, C) + \alpha}{\text{count}(C) + \alpha\, V},$$

where V is the vocabulary size. This is exactly a **Beta/Dirichlet prior** from Chapter 4 — smoothing *is* Bayesian priors on the word probabilities, ensuring nothing has probability zero just because it was unseen.

7.4 Working in log space

Common pitfall

Multiplying hundreds of small probabilities causes **numerical underflow** (the product rounds to zero). Always compute in logs, turning the product into a sum:

$$\log P(C \mid \mathbf{x}) = \log P(C) + \sum_{j=1}^{d} \log P(x_j \mid C) + \text{const.}$$

This is standard practice and the reason library implementations report log-probabilities. Forgetting it silently breaks classification on long documents.

7.5 Why "naive" works

Intuition

The independence assumption is usually false — in text, "New" and "York" are highly dependent. Yet Naive Bayes classifies well because *accurate ranking does not require accurate probabilities*. Even if the estimated $P(C \mid \mathbf{x})$ is poorly calibrated (often pushed toward 0 or 1 by double-counting correlated features), the *argmax* — the predicted class — is frequently still correct. Naive Bayes is a low-variance, high-bias model: it cannot overfit

much, so with limited data it often beats flexible models, and it trains in a single pass.

Common pitfall

Two cautions. (1) **Probabilities are unreliable**: correlated features make Naive Bayes overconfident, so use it for the *decision*, not for a trustworthy probability, unless you calibrate. (2) **Correlated/redundant features** (the same signal entered twice) get double-counted and bias the result — deduplicate features when you can. The class *ranking* is robust; the numbers are not.

How this powers ML / AI

Naive Bayes is the original probabilistic text classifier and remains a standard, hard-to-beat **baseline** for spam detection, sentiment analysis, and document categorization — fast to train, tiny to store, and effective in high dimensions with little data. In modern NLP pipelines it is the sanity check a fancy Transformer must beat to justify its cost. Conceptually it foreshadows deep generative classifiers: model $P(\mathbf{x} \mid C)$ and invert with Bayes. And its smoothing is a clean, intuitive first encounter with the prior-as-pseudo-counts idea that recurs throughout probabilistic ML (e.g. Dirichlet priors in topic models and label smoothing in deep nets).

Try it in NumPy

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
X = ["win money now", "cheap meds online", "let us meet tomorrow",
     "project deadline tomorrow", "free prize claim now", "lunch
    meeting today"]
y = ["spam", "spam", "ham", "ham", "spam", "ham"]
vec = CountVectorizer(); Xc = vec.fit_transform(X)
clf = MultinomialNB(alpha=1.0).fit(Xc, y)        # alpha = Laplace
    smoothing (a prior)
test = vec.transform(["free money meeting"])
print(clf.predict(test), clf.predict_proba(test).round(3))
```

Chapter summary

- Naive Bayes applies Bayes' theorem with a **conditional-independence** assumption: $P(\mathbf{x} \mid C) = \prod_j P(x_j \mid C)$.
- This turns a hard joint density into a product of easy 1-D estimates — cheap, low-variance, great in high dimensions.
- Flavors: **Gaussian** (continuous), **Multinomial** (counts/text), **Bernoulli** (binary).
- **Laplace smoothing** (a Beta/Dirichlet prior) prevents zero probabilities; compute in **log space** to avoid underflow.
- It works because correct *ranking* survives wrong independence assumptions; probabilities are overconfident, so trust the class, not the number. A standard, strong text baseline.

8

Bayesian versus Frequentist Inference

Level: Advanced

The two great schools of statistics answer the same questions in fundamentally different ways. Understanding the contrast sharpens your grasp of *both* — and clears up the most persistent confusions in applied statistics, like what a confidence interval and a p-value actually mean. This chapter compares the philosophies, the interval estimates, and the approaches to hypothesis testing, then argues that the practical question is "which tool for which job?"

8.1 Two philosophies

Definition 8.1. The split is about what is random:

- **Frequentist:** parameters are *fixed unknown constants*; data is random. Probability describes the long-run behavior of procedures over hypothetical repeated samples.
- **Bayesian:** parameters are *random* (we model our uncertainty about them); the observed data is fixed. Probability describes degree of belief.

Frequentist
θ fixed, data random; $P(\text{data} \mid \theta)$; estimators, p-values, confidence intervals; no prior.

Bayesian
θ random, data fixed; $P(\theta \mid \text{data})$; posteriors, credible intervals, Bayes factors; prior required.

8.2 Confidence vs. credible intervals

Key idea

This is the most misunderstood pair in statistics.

- A **95% confidence interval** (frequentist): if we repeated the experiment many times, 95% of the intervals so constructed would contain the true θ. The probability is a property of the *procedure*, not of your one interval — which either contains θ or does not.
- A **95% credible interval** (Bayesian): given the data and prior, there is a 95% probability that θ lies in *this* interval — a direct statement about the parameter.

People almost always *want* the credible interpretation; only the Bayesian framework licenses it.

8.3 Hypothesis testing

Definition 8.2. Frequentists test a null hypothesis H_0 via a ***p*-value**: $P(\text{data at least this extreme} \mid H_0)$. Small p "rejects" H_0. Bayesians compare hypotheses with the **Bayes factor**, the ratio of evidences:

$$\mathrm{BF}_{10} = \frac{P(D \mid H_1)}{P(D \mid H_0)},$$

which multiplies the prior odds into the posterior odds: $\frac{P(H_1|D)}{P(H_0|D)} = \mathrm{BF}_{10} \cdot \frac{P(H_1)}{P(H_0)}$.

Common pitfall

The ***p*-value is not** the probability that H_0 is true, nor the probability your results are due to chance. It is $P(\text{data} \mid H_0)$, *not* $P(H_0 \mid \text{data})$ — confusing the two is the inverse fallacy of Chapter 2. A p-value also says nothing about effect size, depends on the (often unstated) sampling intention, and is rampantly abused through "p-hacking." The Bayes factor answers the question people actually ask — "how much should I believe H_1 over H_0?" — but it depends on the prior and can be sensitive to it (the Jeffreys–Lindley paradox).

8.4 When they agree, and when they differ

Intuition

With lots of data and a weak prior, Bayesian and frequentist answers usually *numerically coincide* — the likelihood dominates, and a credible interval matches a confidence interval. They diverge when (a) data is scarce, where the prior matters; (b) prior information is genuinely available and valuable; or (c) the question is inherently about belief in a hypothesis or about a one-off event. The frameworks are complementary lenses, not rivals to be ranked once and for all.

	Frequentist	**Bayesian**
Parameter	fixed constant	random (belief)
Needs a prior?	no	yes
Interval	confidence (procedure)	credible (parameter)
Test tool	p-value	Bayes factor / posterior
Strength	no prior, fast, calibrated	uncertainty, prior info, intuitive
Cost	misinterpreted, no prior info	prior choice, computation

How this powers ML / AI

Machine learning is thoroughly *pragmatic* and uses both. Plain **maximum likelihood** training and **cross-validation** are frequentist in spirit; **regularization**, **Bayesian neural networks**, **Gaussian processes**, and **Bayesian optimization** are Bayesian. The most useful stance is to know which question you are asking: "what procedure has good average performance?" (frequentist, e.g. test-set accuracy and CV) versus "how uncertain am I about this prediction?" (Bayesian, e.g. predictive posteriors and calibration). Modern probabilistic ML freely mixes them — e.g. empirical Bayes and frequentist evaluation of Bayesian models.

Try it in NumPy

```python
import numpy as np
from scipy import stats
rng = np.random.default_rng(0)
data = rng.normal(1.0, 2.0, size=30)
# Frequentist 95% confidence interval for the mean
m, se = data.mean(), stats.sem(data)
ci = stats.t.interval(0.95, len(data)-1, loc=m, scale=se)
print("frequentist 95% CI:", np.round(ci, 3))
# Bayesian 95% credible interval (Normal–Normal, vague prior) ~ coincides here
tau = 1/(1/1e6 + len(data)/data.var()); mu = tau*(len(data)*m/data.var())
print("bayesian 95% CrI:", np.round(stats.norm(mu, tau**0.5).ppf([.025,.975]), 3))
```

Chapter summary

- **Frequentist**: parameters fixed, data random, no prior; **Bayesian**: parameters random, data fixed, prior required.
- A **confidence** interval is a statement about the procedure's long-run coverage; a **credible** interval is a probability statement about the parameter.
- p**-value** $= P(\text{data} \mid H_0)$, not $P(H_0 \mid \text{data})$; the **Bayes factor** compares evidence for hypotheses but depends on priors.
- With ample data and weak priors the two usually agree; they diverge when data is scarce or prior information matters.
- ML uses both pragmatically: MLE/CV (frequentist) and regularization/BNNs/GPs (Bayesian) — match the tool to the question.

PART IV

Structure and Computation

9

Bayesian Networks and Graphical Models

Level: Advanced

Real problems have many interacting variables. **Bayesian networks** (directed graphical models) make such joint distributions tractable by drawing the dependencies as a graph: an arrow means "directly influences," a missing arrow means "conditionally independent." The graph is both a picture of your assumptions and a computational engine for answering probabilistic queries. Naive Bayes was the simplest example; this chapter is the general theory.

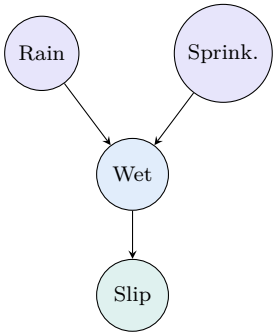
9.1 Factorizing a joint distribution

Definition 9.1. A **Bayesian network** is a directed acyclic graph (DAG) whose nodes are random variables. It encodes the joint distribution as a product of **local conditionals**, each variable given its parents:

$$P(X_1, \dots, X_n) = \prod_{i=1}^{n} P\big(X_i \mid \text{parents}(X_i)\big).$$

Key idea

This factorization is the whole payoff. A joint over n binary variables needs $2^n - 1$ numbers; a Bayesian network needs only the small conditional tables for each node given its (few) parents. By asserting that most variables *do not* directly depend on most others, the graph turns an exponential object into a compact, interpretable one — and makes inference feasible.



$$P(R,S,W,Sl) = P(R)\,P(S)\,P(W\,|\,R,S)\,P(Sl\,|\,W)$$

9.2 Conditional independence and d-separation

Intuition

The graph's missing edges encode **conditional independencies**, read off by *d-separation*. Three canonical structures:

- **Chain** $A \to B \to C$: A and C are independent *given* B (B blocks the path).
- **Fork** $A \leftarrow B \to C$: a common cause; A, C dependent but independent given B.
- **Collider** $A \to B \leftarrow C$: A, C independent, but conditioning on B (or its descendants) *creates* dependence — "explaining away."

Intuition

"Explaining away" is the subtle, important one: if wet grass can be caused by rain *or* the sprinkler, learning it rained *lowers* the probability the sprinkler was on, even though rain and sprinkler are independent a priori. Observing a shared effect couples its causes. This is why conditioning on the wrong variable (a collider) can create spurious correlations — the graphical version of the causality pitfalls in the regression literature.

9.3 Inference on the graph

Definition 9.2. Inference means computing a query like $P(\text{Rain} \mid \text{Slip} = \text{true})$ by summing the factorized joint over the unobserved variables. Exact methods (variable elimination, belief propagation, the junction-tree algorithm) exploit the graph structure; when the graph is too complex, we fall back on the approximate methods of Chapter 10.

Key idea

Inference flows *both directions* along the arrows. Arrows point from cause to effect (the generative story), but Bayes' theorem lets us reason from observed effects back to hidden causes — *diagnosis*. A Bayesian network thus answers predictive queries (causes→effects) and diagnostic queries (effects→causes) within one coherent model. This bidirectional reasoning is what makes graphical models so powerful for expert systems and probabilistic AI.

9.4 The wider family

Bayesian networks are one branch of **probabilistic graphical models**. Undirected **Markov random fields** model symmetric relationships (pixels in an image, spins in a lattice); **hidden Markov models** and **Kalman filters** are dynamic graphical models for sequences; **factor graphs** unify them for message-passing inference. All share the core idea: structure the joint, then compute with it.

Common pitfall

Two recurring mistakes. (1) **Arrows are not automatically causal.** A Bayesian network encodes conditional independence; reading edges as cause-and-effect requires extra assumptions (interventions, no hidden confounders) — that is the leap to *causal* graphical models. (2) **Exact inference is NP-hard** in general; densely connected graphs blow up, so know when to switch to approximate inference. And learning structure from data is

hard — many graphs fit equally well.

How this powers ML / AI

Graphical models are a foundational language for probabilistic AI. **Hidden Markov models** drove speech recognition and bioinformatics for decades; **Kalman filters** (a Gaussian graphical model) power navigation, tracking, and control. **Conditional random fields** were the workhorse of sequence labeling before neural nets. Modern **latent-variable models** — VAEs, diffusion models, topic models (LDA) — are graphical models whose conditionals are parameterized by neural networks, trained with the variational inference of the next chapter. The factorization idea also underlies probabilistic programming languages (Stan, PyMC, Pyro) that let you specify a graph and infer automatically.

Try it in NumPy

```python
# Sprinkler network: P(Rain | Slip) by enumerating the factorized joint
import itertools
P_R = {1: 0.2, 0: 0.8}
P_S = {1: 0.4, 0: 0.6}
P_W = {(1,1):0.99,(1,0):0.9,(0,1):0.9,(0,0):0.0}   # P(Wet=1 | R, S)
P_Sl = {1: 0.8, 0: 0.05}                            # P(Slip=1 | Wet)
num = den = 0.0
for r, s, w in itertools.product([0,1],[0,1],[0,1]):
    pw = P_W[(r,s)] if w else 1-P_W[(r,s)]
    joint = P_R[r]*P_S[s]*pw*P_Sl[w]                # P(R,S,W,Slip=1)
    den += joint; num += joint if r==1 else 0
print("P(Rain | Slip=1) =", round(num/den, 3))
```

Chapter summary

- A **Bayesian network** (DAG) factorizes a joint as $\prod_i P(X_i \mid \text{parents})$, shrinking an exponential object to compact local tables.
- Missing edges encode **conditional independence**, read via d-separation: chains and forks block, **colliders** couple ("explaining away").
- **Inference** sums the factorized joint over hidden variables and flows both ways: prediction (causes→effects) and diagnosis (effects→causes).
- The family includes MRFs, HMMs, Kalman filters, and factor graphs.
- Arrows are not automatically causal; exact inference is NP-hard in general. Graphical models underlie HMMs, CRFs, VAEs, diffusion, and probabilistic programming.

10

Computational Bayes: MCMC and Variational Inference

Level: Expert

Conjugacy is a luxury; most real posteriors have no closed form because the evidence integral $p(D) = \int p(D \mid \theta)p(\theta)\,d\theta$ is intractable. The modern Bayesian toolkit sidesteps it two ways: **MCMC** draws samples from the posterior without ever computing the normalizer, and **variational inference** replaces sampling with optimization. Together they unlocked Bayesian methods for complex models and made Bayesian deep learning possible.

10.1 The intractable normalizer

Key idea

We can almost always evaluate the *unnormalized* posterior $\tilde{p}(\theta) = p(D \mid \theta)p(\theta)$ — it is just likelihood times prior. What we cannot do is compute the constant $p(D)$ that makes it integrate to one, because that integral is high-dimensional and has no closed form. Every computational Bayesian method is a way to answer questions about the posterior *using only the unnormalized* $\tilde{p}$. That is the central trick of the entire field.

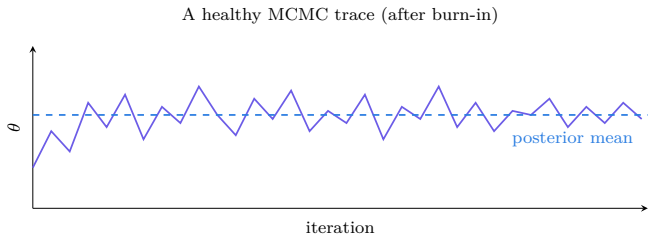
10.2 Markov chain Monte Carlo

Definition 10.1. MCMC constructs a Markov chain whose stationary distribution *is* the posterior. Running the chain and collecting its states yields (correlated) samples from $p(\theta \mid D)$, from which any expectation, mean, or interval is estimated by averaging.

Metropolis–Hastings.
Propose a move $\theta \to \theta'$; accept with probability $\min\!\big(1, \frac{\tilde{p}(\theta')q(\theta \mid \theta')}{\tilde{p}(\theta)q(\theta' \mid \theta)}\big)$. The normalizer cancels in the ratio — which is why we never need it. Always go uphill; sometimes go downhill, so the chain explores.

Gibbs sampling.
When full conditionals $p(\theta_i \mid \theta_{-i}, D)$ are known, sample each parameter in turn from its conditional. No tuning, ideal for conjugate substructure.

Hamiltonian Monte Carlo (HMC) / NUTS.
Use the gradient of $\log \tilde{p}$ to propose distant, high-acceptance moves via simulated physical dynamics. The state of the art for continuous parameters; the engine inside Stan and PyMC.



Common pitfall

MCMC is exact in the limit but tricky in practice. Discard early **burn-in** samples (before the chain reaches the posterior). Check **convergence** — run multiple chains and compare with the $\hat{R}$ statistic ($\hat{R} \approx 1$ is good), inspect trace plots, and watch **effective sample size** (correlated draws carry less information). A chain stuck in one mode of a multimodal posterior can look perfectly converged while being completely wrong. Never trust a single short chain.

10.3 Variational inference

Key idea

Variational inference (VI) turns inference into *optimization*: pick a tractable family $q_\phi(\theta)$ (e.g. Gaussians) and find the member closest to the true posterior by minimizing the KL divergence $\mathrm{KL}(q_\phi \,\|\, p(\cdot \mid D))$. Since we cannot evaluate that KL directly, we equivalently *maximize* the **evidence lower bound (ELBO)**:

$$\mathrm{ELBO}(\phi) = \mathbb{E}_{q_\phi}\big[\log p(D,\theta)\big] - \mathbb{E}_{q_\phi}\big[\log q_\phi(\theta)\big] \;\le\; \log p(D).$$

Intuition

The ELBO splits into "fit the data well" minus "stay spread out" (an entropy term) — a built-in Occam's razor. VI is typically *much faster* than MCMC and scales to huge models and datasets via stochastic gradients, which is why it dominates Bayesian deep learning. The cost: q is an approximation, and the usual mean-field q tends to *underestimate* posterior variance (it finds one mode and hugs it), making VI overconfident relative to MCMC.

MCMC — sample the posterior
asymptotically exact, slower, gold standard; HMC/NUTS, Gibbs, Metropolis.

Variational — optimize an approximation
fast, scalable, biased; maximize the ELBO; mean-field, Bayes-by-backprop.

How this powers ML / AI

These algorithms are what make Bayesian deep learning real. **Bayes by Backprop** trains a **Bayesian neural network** by maximizing the ELBO over weight distributions; **MC dropout** reinterprets dropout at test time as variational inference, giving cheap uncertainty from an ordinary net. **Stochastic-gradient MCMC** (SGLD, SG-HMC) scales sampling to mini-batches and big models. The **reparameterization trick** that makes the ELBO differentiable is the same one that trains **variational autoencoders**. Probabilistic programming systems (Stan, PyMC, Pyro, NumPyro) provide NUTS and VI as push-button inference, so practitioners specify a model and let these algorithms compute the posterior.

Try it in NumPy

```python
import numpy as np
rng = np.random.default_rng(0)
data = rng.normal(2.0, 1.0, size=50)
def log_post(mu):                          # unnormalized: Normal likelihood + N(0,10^2) prior
    return -0.5*np.sum((data-mu)**2) - 0.5*(mu/10)**2
# Metropolis sampler -- note we only ever use the UNNORMALIZED log-posterior
mu, samples = 0.0, []
for _ in range(20000):
    prop = mu + rng.normal(0, 0.3)
    if np.log(rng.uniform()) < log_post(prop) - log_post(mu):
        mu = prop
    samples.append(mu)
s = np.array(samples[2000:])               # discard burn-in
print("posterior mean ~", round(s.mean(), 3), " 95% CrI", np.round(np.percentile(s,[2.5,97.5]),3))
```

Chapter summary

- Posteriors are usually intractable because the **evidence** integral has no closed form; methods use only the **unnormalized** posterior.
- **MCMC** builds a Markov chain whose stationary distribution is the posterior: Metropolis–Hastings, Gibbs, and gradient-based **HMC/NUTS**.
- Diagnose MCMC with burn-in, multiple chains, $\hat{R}$, and effective sample size; beware multimodality.
- **Variational inference** maximizes the **ELBO** to fit a tractable q — fast and scalable but typically underestimates variance.
- These power Bayes by Backprop, MC dropout, SG-MCMC, VAEs, and probabilistic programming — the machinery of Bayesian deep learning.

11

Bayesian Regression and Hierarchical Models

Level: Expert

Now we put the machinery to work on the models practitioners use every day. **Bayesian regression** replaces point-estimate coefficients with full posteriors, delivering principled uncertainty on every prediction. **Hierarchical (multilevel) models** go further, sharing information across related groups through a shared prior — the elegant idea of "partial pooling" that underlies much of modern applied Bayesian statistics.

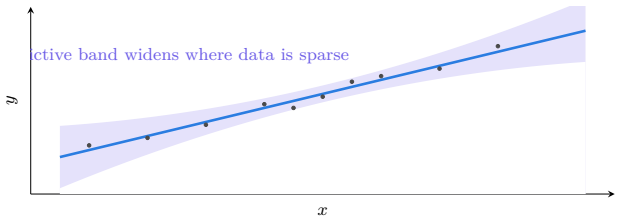
11.1 Bayesian linear regression

Definition 11.1. Place a prior on the regression coefficients and infer their posterior. With a Gaussian prior $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \tau^2 \mathbf{I})$ and Gaussian noise, the posterior over $\boldsymbol{\beta}$ is Gaussian (conjugate), and predictions use the **posterior predictive**

$$p(\tilde{y} \mid \tilde{\mathbf{x}}, D) = \int p(\tilde{y} \mid \tilde{\mathbf{x}}, \boldsymbol{\beta})\, p(\boldsymbol{\beta} \mid D)\, d\boldsymbol{\beta}.$$

Key idea

Bayesian linear regression returns a *distribution over lines*, not one line. Two consequences follow for free: the posterior **mean** coefficients equal the **ridge** estimate (the Gaussian prior is the L_2 penalty of Chapter 5), and the predictive variance *grows where data is sparse* — the model is appropriately unsure when extrapolating. You get regularization and calibrated uncertainty from one coherent recipe.



11.2 Hierarchical models and partial pooling

Definition 11.2. A **hierarchical (multilevel) model** gives each group g its own parameters θ_g, drawn from a shared *population* prior whose **hyperparameters** are themselves estimated:

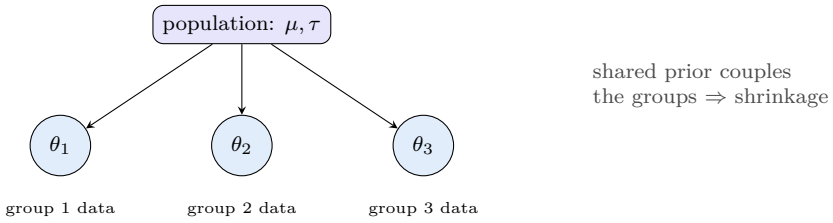
$$\theta_g \mid \mu, \tau \sim \mathcal{N}(\mu, \tau^2), \qquad \mu, \tau \sim \text{hyperprior}.$$

Key idea

This formalizes **partial pooling**, the middle path between two bad extremes. *Complete pooling* ignores groups (one estimate for all — too rigid); *no pooling* fits each group alone (noisy, overfit for small groups). The hierarchical model *learns* how much to pool from the data: groups with little data are pulled (*shrunk*) toward the population mean, groups with abundant data keep their own estimate. It borrows strength across groups automatically.

Intuition

This is why a baseball player with 5 at-bats should not be projected as a .800 hitter: the hierarchical model shrinks that noisy estimate toward the league average, by an amount the data dictates. The same logic stabilizes estimates for small schools, rare diseases, new products — anywhere you have many related but unevenly-sampled groups. Hierarchical shrinkage is the principled version of "regularize the small-sample estimates."



11.3 Bayesian logistic and generalized models

The same recipe extends to **Bayesian logistic regression** and other GLMs: keep the link and likelihood, put priors on the coefficients, and infer the posterior (no longer conjugate, so use MCMC or VI from Chapter 10). The reward is posterior distributions over odds ratios and calibrated predictive probabilities with honest uncertainty — valuable in medicine, risk, and any setting where overconfidence is costly.

Common pitfall

Hierarchical models have characteristic traps. The **funnel**: when a group has little data, the geometry of θ_g versus τ becomes pathological and samplers struggle — a *non-centered reparameterization* usually fixes it. Choosing the **hyperprior on the group variance** τ matters: too tight forces over-pooling, too loose allows over-fitting; half-Normal or half-Cauchy priors are common defaults. And always run **posterior predictive checks** (Chapter 12) to confirm the model can reproduce the data's structure.

How this powers ML / AI

Hierarchical Bayes is the statistical ancestor of ideas everywhere in ML. **Shrinkage toward a shared prior** is exactly transfer learning and **multi-task learning**, where related tasks share parameters or a prior. **Empirical Bayes** — estimating the prior from data — underlies modern **meta-learning** ("learning to learn," e.g. MAML as hierarchical Bayes) and automatic relevance determination. **Partial pooling** is the principled cousin of regularization and is used in large-scale recommendation and experimentation platforms to stabilize millions of per-item or per-user estimates. And Bayesian regression with predictive uncertainty is the backbone of **Bayesian optimization** and active learning.

Try it in NumPy

```
import numpy as np
# James-Stein-style shrinkage: pull noisy group means toward the grand mean
rng = np.random.default_rng(0)
true = rng.normal(0, 1, 8)                  # 8 groups' true effects
obs  = true + rng.normal(0, 1, 8)           # noisy observations (n=1 each)
grand = obs.mean()
tau2, s2 = obs.var(), 1.0                    # pop variance vs noise variance
w = tau2 / (tau2 + s2)                       # data-driven pooling weight
shrunk = grand + w*(obs - grand)             # partial pooling
print("MSE no-pool:", round(((obs-true)**2).mean(),3),
      " MSE shrunk:", round(((shrunk-true)**2).mean(),3))   # shrinkage usually wins
```

Chapter summary

- **Bayesian regression** infers a posterior over coefficients; the mean is the **ridge** estimate and predictive variance grows where data is sparse.
- Predictions use the **posterior predictive** (marginalize over coefficients) for calibrated uncertainty.
- **Hierarchical models** draw group parameters from a shared prior, giving **partial pooling**: small groups shrink toward the population mean.
- Watch the **funnel** (use non-centered parameterization) and the variance hyperprior; check with posterior predictive checks.
- Hierarchical/empirical Bayes underlies transfer and multi-task learning, meta-learning, and large-scale shrinkage in ML.

PART V

Frontier and Applications

12

Model Comparison, Evidence, and Occam's Razor

Level: Expert

Inference within a model assumes the model is right. But which model should we use? Bayesian model comparison answers this with the very quantity we worked so hard to avoid — the **evidence** $p(D)$ — which turns out to automatically embody **Occam's razor**, preferring models that are no more complex than the data demands. This chapter covers the evidence, Bayes factors, practical predictive criteria, and the indispensable habit of checking models against reality.

12.1 The evidence as a model score

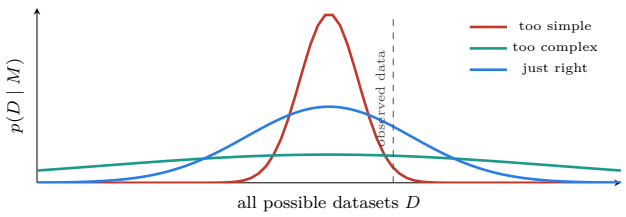
Definition 12.1. For a model M, the **marginal likelihood** (evidence) is the probability it assigns to the data, averaged over its own prior:

$$p(D \mid M) = \int p(D \mid \theta, M)\, p(\theta \mid M)\, d\theta.$$

This is the normalizer from Bayes' theorem — ignorable *within* a model, decisive *between* models.

Key idea

The evidence rewards models that predict the observed data well *before* seeing it — averaged over all parameter settings the prior allows. A model too simple cannot fit the data (low likelihood everywhere); a model too complex spreads its prior probability thinly over many datasets it *could* have produced, so it assigns little to the one we actually saw. The evidence peaks at the model of *just enough* complexity — **Occam's razor falls out of the mathematics**, with no ad hoc penalty added.



12.2 Bayes factors

Definition 12.2. The **Bayes factor** comparing models M_1, M_0 is the ratio of their evidences,

$$\mathrm{BF}_{10} = \frac{p(D \mid M_1)}{p(D \mid M_0)},$$

which updates the prior odds on the models into posterior odds. Conventionally $\mathrm{BF} > 10$ is strong, > 100 decisive evidence for M_1.

Common pitfall

The evidence is powerful but treacherous. It is **sensitive to the prior**: unlike the posterior, $p(D \mid M)$ does not wash out a vague prior even with infinite data, so a carelessly diffuse prior can arbitrarily penalize a model (the *Jeffreys–Lindley paradox*). It is also **hard to compute** — that intractable integral again — and naive estimators (like the harmonic-mean estimator) are notoriously unreliable. For these reasons many practitioners prefer predictive criteria below for model *selection*, reserving Bayes factors for carefully specified hypotheses.

12.3 Predictive criteria: WAIC and cross-validation

Intuition

A robust, practical alternative is to score models by their *out-of-sample predictive accuracy*, estimated without a separate test set:

- **WAIC** (widely applicable information criterion): uses the full posterior predictive and an effective-parameter penalty — a fully Bayesian generalization of AIC.
- **PSIS-LOO**: approximate leave-one-out cross-validation computed cheaply from MCMC samples via importance sampling.

These target predictive performance directly, are far less prior-sensitive than the evidence, and come with diagnostics — the recommended default for everyday model comparison.

12.4 Posterior predictive checks

Key idea

Before comparing models, ask whether a model is *adequate at all*. A **posterior predictive check** simulates datasets from the fitted model and compares them to the real data: if the model cannot reproduce key features (the spread, the tails, the number of zeros), it is wrong regardless of its evidence. "All models are wrong" — predictive checks tell you whether yours is *useful*, and they catch failures that any single summary score hides.

Intuition

There is a deep alternative to selection: **model averaging**. Rather than pick one model, weight each model's predictions by its posterior probability and average. This propagates model uncertainty into predictions and often beats any single model — the Bayesian justification for why *ensembles* work so well in machine learning.

How this powers ML / AI

These ideas reappear throughout ML, sometimes in disguise. The **evidence** is the basis of **empirical Bayes** and of **Gaussian process** hyperparameter tuning (maximizing the marginal likelihood, a.k.a. *type-II maximum likelihood* or "the evidence framework"), and of the **evidence lower bound (ELBO)** that trains VAEs and Bayesian neural networks — the ELBO is a tractable surrogate for $\log p(D)$. Occam's-razor-by-evidence explains why Bayesian methods resist overfitting. **Bayesian model averaging** is the principled story behind **ensembles** and **deep ensembles**, and predictive scoring (LOO/WAIC) mirrors the cross-validation at the heart of model selection in ML.

Try it in NumPy

```
import numpy as np
# Evidence (marginal likelihood) via simple Monte Carlo over the prior:
# does a quadratic beat a line on curved data?
rng = np.random.default_rng(0)
x = np.linspace(-2, 2, 25); y = 0.8*x**2 + rng.normal(0, 0.5, 25)
def evidence(degree, S=20000):
    coefs = rng.normal(0, 3, size=(S, degree+1))    # samples from the prior
    Phi = np.vstack([x**k for k in range(degree+1)]).T
    pred = coefs @ Phi.T
    loglik = -0.5*((y - pred)**2).sum(1)/0.5**2      # Gaussian likelihood
    return np.log(np.mean(np.exp(loglik - loglik.max()))) + loglik.max()
print("log evidence  line:", round(evidence(1), 1),
      " quadratic:", round(evidence(2), 1))          # quadratic wins on curved data
```

Chapter summary

- The **evidence** $p(D \mid M)$ scores a model; it automatically embodies **Occam's razor**, favoring just-enough complexity.
- **Bayes factors** compare evidences but are **prior-sensitive** (Jeffreys–Lindley) and hard to compute.
- Prefer predictive criteria — **WAIC**, **PSIS-LOO** — for robust, low-prior-sensitivity model selection.
- Always run **posterior predictive checks**: can the model reproduce the data? **Model averaging** beats selection and justifies ensembles.
- The evidence underlies empirical Bayes, GP hyperparameter tuning, and the ELBO; these explain Bayesian resistance to overfitting.

13

Bayes in Machine Learning, Deep Learning, and AI

Level: Capstone

This capstone gathers the threads into one claim: *Bayesian reasoning is the theory of learning under uncertainty, and it quietly underwrites much of machine learning.* The loss is a likelihood. Regularization is a prior. Training finds a posterior mode. Ensembles approximate posterior averaging. Uncertainty estimates are posteriors. From the spam filter to the diffusion model, the same prior→likelihood→posterior logic recurs. We revisit each idea from the book and show where it lives in modern AI.

13.1 The grand correspondence

Key idea

Maximum-a-posteriori estimation reveals the bridge in one line. Training a model minimizes a regularized loss, and

$$\hat{\theta} = \arg\max_{\theta} \Big[\underbrace{\log p(D \mid \theta)}_{-\text{ loss}} + \underbrace{\log p(\theta)}_{-\text{ regularizer}} \Big] = \arg\min_{\theta} \Big[\text{loss}(\theta) + \lambda\, \Omega(\theta) \Big].$$

So **negative log-likelihood = loss** (MSE ↔ Gaussian, cross-entropy ↔ categorical) and **negative log-prior = regularizer** (Gaussian ↔ L_2/weight decay, Laplace ↔ L_1). Standard deep-learning training is *MAP estimation* — a Bayesian computation that keeps only the peak.

Bayesian concept	ML / DL counterpart	Where it appears
Likelihood $p(D \mid \theta)$	loss function	MSE, cross-entropy (Ch. 3)
Prior $p(\theta)$	regularizer	weight decay (L_2), lasso (L_1) (Ch. 5)
MAP estimate	regularized training	most deep-net training
Posterior predictive	ensemble / averaging	deep ensembles, MC dropout
Bayes' theorem	probabilistic classifier	Naive Bayes, spam filters (Ch. 7)
Evidence $p(D)$	marginal likelihood	GP tuning, ELBO (Ch. 12)
Conjugate update	online belief update	Thompson sampling (Ch. 4)
Hierarchical prior	transfer / multi-task	meta-learning (Ch. 11)

13.2 Naive Bayes and probabilistic classifiers

The chapter on Naive Bayes (Chapter 7) is the most direct application: a fast, strong baseline for text and a template for all generative classifiers, which model $p(\mathbf{x} \mid C)$ and invert with Bayes'

rule. Its smoothing introduced the prior-as-pseudo-counts idea that recurs as Dirichlet priors in topic models and label smoothing in deep nets.

13.3 Bayesian deep learning

Key idea

A **Bayesian neural network** places a distribution over weights and infers their posterior $p(\mathbf{w} \mid D)$, so predictions average over many plausible networks: $p(\tilde{y} \mid \tilde{\mathbf{x}}, D) = \int p(\tilde{y} \mid \tilde{\mathbf{x}}, \mathbf{w})\, p(\mathbf{w} \mid D)\, d\mathbf{w}$. Because that posterior is hopelessly intractable, we use the approximations of Chapter 10.

Variational inference / Bayes by Backprop.
Learn a Gaussian over each weight by maximizing the ELBO — a distribution of networks for the price of doubling the parameters.

MC dropout.
Keep dropout on at test time and average several forward passes — a remarkably cheap variational approximation that turns any dropout net into a Bayesian one.

Laplace approximation.
Fit a Gaussian at a trained network's MAP using curvature (the Hessian) — post-hoc uncertainty with no retraining.

SG-MCMC & deep ensembles.
Stochastic-gradient HMC/Langevin sample the weight posterior; deep ensembles (train several nets) are a simple, strong approximation to posterior averaging.

Intuition

The point of all this is **knowing what you don't know**. A standard network is dangerously confident on inputs unlike its training data; a Bayesian network's predictive variance *grows* there, flagging out-of-distribution inputs. This **epistemic** uncertainty (reducible with more data) is distinguished from **aleatoric** uncertainty (inherent noise) — a distinction that matters enormously for safety-critical AI, medicine, and autonomous systems.

13.4 Bayesian optimization and sequential decisions

Key idea

Bayesian optimization tunes expensive black-box functions (hyperparameters, experiments, materials) by fitting a probabilistic surrogate — usually a Gaussian process — and using its *uncertainty* to decide where to sample next, balancing exploration and exploitation through an acquisition function. **Thompson sampling**, the elegant bandit strategy behind A/B testing and recommendation, simply samples from the posterior and acts greedily — conjugate Bayes (Chapter 4) in an online loop. Both are decision theory (Chapter 6) made sequential.

13.5 Generative models as Bayesian inference

How this powers ML / AI

Modern generative AI is shot through with Bayesian structure. **Variational autoencoders** are latent-variable graphical models (Chapter 9) trained by maximizing the ELBO (Chapter 10) — inference of latent codes *is* Bayesian posterior inference. **Diffusion models** can be read as hierarchical latent-variable models with a variational training objective, learning to reverse a noising process. **Latent Dirichlet allocation** for topics, and Bayesian **nonparametrics** (Gaussian and Dirichlet processes) that grow with data, are explicitly Bayesian. Even **in-context learning** in large language models has been analyzed as *implicit Bayesian inference*: the model behaves as if forming a posterior over tasks from the prompt. The probabilistic-programming ecosystem (Stan, PyMC, Pyro, NumPyro) lets practitioners build such models declaratively and infer with NUTS or VI.

13.6 The thread, end to end

Key idea

Trace one idea through the book into AI: *start with a belief, see data, update coherently, and act under the remaining uncertainty.* Chapter 2 gave the update; Chapters 3–4 carried it for parameters; Chapter 5 revealed priors as regularizers; Chapter 6 turned beliefs into decisions; Chapter 7 built a classifier; Chapter 10 made it computable at scale. A Bayesian neural network is all of these at once. Whether or not a system explicitly computes a posterior, the Bayesian lens explains *why it works* — and points to how to make it know what it does not know. **Bayes is where probabilistic AI begins, and the frame in which it makes sense.**

Try it in NumPy

```
import numpy as np
# MC-dropout-style uncertainty: average stochastic forward passes.
# Here we mimic the idea with an ensemble of noisy linear models.
rng = np.random.default_rng(0)
xtr = np.linspace(-2, 2, 30); ytr = np.sin(xtr) + rng.normal(0, 0.1,
    30)
def fit_noisy():                          # one "sampled" model (
    bootstrap + jitter)
    idx = rng.integers(0, 30, 30)
    return np.polyfit(xtr[idx], ytr[idx], 3)
ens = [fit_noisy() for _ in range(200)]  # a crude posterior over
    functions
xt = np.linspace(-3, 3, 50)              # includes out-of-
    distribution region
preds = np.array([np.polyval(c, xt) for c in ens])
mean, std = preds.mean(0), preds.std(0)
print("uncertainty in-domain vs out-of-domain:",
      round(std[20:30].mean(), 3), "<", round(std[:5].mean(), 3))  #
    wider OOD
```

Chapter summary

- **MAP = regularized training**: negative log-likelihood is the loss, negative log-prior is the regularizer (L_2/weight decay, L_1/lasso).
- **Naive Bayes** is the direct classifier; **posterior predictive** averaging is what ensembles and MC dropout approximate.
- **Bayesian neural networks** (VI/Bayes-by-Backprop, MC dropout, Laplace, SG-MCMC, deep ensembles) provide **epistemic uncertainty** — knowing what you don't know.
- **Bayesian optimization** and **Thompson sampling** use posterior uncertainty for sequential decisions.
- **VAEs**, **diffusion**, LDA, Bayesian nonparametrics, and even in-context learning are Bayesian inference — the frame in which probabilistic AI makes sense.

A

Distributions, Conjugate Pairs, and Formulas

A compact reference to the book's key results.

A.1 The core identities

Quantity	Formula
Bayes' theorem	$P(\mathcal{H} \mid D) = \dfrac{P(D \mid \mathcal{H})\,P(\mathcal{H})}{P(D)}$
Working form	$p(\theta \mid D) \propto p(D \mid \theta)\,p(\theta)$
Evidence (marginal likelihood)	$p(D) = \int p(D \mid \theta)\,p(\theta)\,d\theta$
Posterior predictive	$p(\tilde{x} \mid D) = \int p(\tilde{x} \mid \theta)\,p(\theta \mid D)\,d\theta$
MAP estimate	$\hat{\theta} = \arg\max_\theta \left[\log p(D \mid \theta) + \log p(\theta)\right]$
Bayes action	$a^\star = \arg\min_a \mathbb{E}_{\theta \mid D}[L(\theta, a)]$
Bayes factor	$\mathrm{BF}_{10} = p(D \mid M_1)/p(D \mid M_0)$
ELBO	$\mathbb{E}_q[\log p(D, \theta)] - \mathbb{E}_q[\log q(\theta)] \le \log p(D)$
Naive Bayes	$\hat{C} = \arg\max_C P(C) \prod_j P(x_j \mid C)$
Laplace smoothing	$P(w \mid C) = \dfrac{\mathrm{count}(w, C) + \alpha}{\mathrm{count}(C) + \alpha V}$

A.2 Conjugate prior cheat sheet

Likelihood	Parameter	Conjugate prior	Posterior update
Bernoulli / Binomial	prob. θ	$\mathrm{Beta}(\alpha, \beta)$	$\mathrm{Beta}(\alpha + h,\ \beta + n - h)$
Poisson	rate λ	$\mathrm{Gamma}(\alpha, \beta)$	$\mathrm{Gamma}(\alpha + \sum x,\ \beta + n)$
Normal (known σ^2)	mean μ	$\mathcal{N}(\mu_0, \tau_0^2)$	precision-weighted (see Ch. 4)
Multinomial	probs. $\boldsymbol{\theta}$	$\mathrm{Dirichlet}(\boldsymbol{\alpha})$	$\mathrm{Dirichlet}(\boldsymbol{\alpha} + \text{counts})$
Normal (known μ)	variance σ^2	Inverse-Gamma	Inverse-Gamma (updated)
Exponential	rate λ	$\mathrm{Gamma}(\alpha, \beta)$	$\mathrm{Gamma}(\alpha + n,\ \beta + \sum x)$

A.3 Choosing a computational method

Situation	Recommended approach
Conjugate model	closed-form update (Ch. 4)
Low-dim, want exactness	MCMC: HMC / NUTS (Ch. 10)
Conjugate full conditionals	Gibbs sampling
Large model / dataset, speed	variational inference (ELBO)
Deep network uncertainty	MC dropout, deep ensembles, Laplace, SG-MCMC
Discrete graphical model	variable elimination / belief propagation (Ch. 9)
Expensive black-box tuning	Bayesian optimization (GP surrogate)
Many related groups	hierarchical model + partial pooling (Ch. 11)

A.4 Workflow checklist

1. State the model: likelihood (data-generating story) and priors on all parameters.
2. Run a **prior predictive check**; revise priors that imply absurd data.
3. Compute the posterior (closed form, MCMC, or VI); for MCMC check $\hat{R}$, ESS, and trace plots.
4. Run a **posterior predictive check**: can the model reproduce the data?
5. Summarize with means/credible intervals; make decisions by minimizing expected loss.
6. Compare models with LOO/WAIC; do a **prior sensitivity analysis**; consider model averaging.

A.5 Symbols

Symbol	Meaning
θ, $\mathbf{w}$	parameter(s) / network weights
D, $\mathcal{D}$	observed data
$\mathcal{H}$, M	hypothesis; model
$p(\theta)$ / $p(\theta \mid D)$	prior / posterior
$p(D \mid \theta)$	likelihood
$p(D)$	evidence (marginal likelihood)
α, β	prior hyperparameters (pseudo-counts)
$q_\phi(\theta)$	variational approximation
$\mathrm{KL}(q \,\|\, p)$	Kullback–Leibler divergence