# A Probabilistic View of ML

*Distributions · sampling · MLE · MAP · KL*

**Lecture 0 · ES 667: Deep Learning** · spans 2 sessions

**Prof. Nipun Batra**

IIT Gandhinagar · Aug 2026

# Why this lecture

You already know how to solve regression and classification. You wrote down a loss, took a gradient, ran SGD. It works.

But two questions you may never have asked ·

1. **Where does the loss come from?** Why MSE for regression and cross-entropy for classification — and not something else?
2. **What is L1 / L2 regularization, really?** They're not just "add this term, weights stay small." They have a meaning.

> REFERENCE
>
> Today's promise · one principle — **maximum likelihood under a probabilistic model** — gives every loss in this course, and a tiny twist on it gives every regularizer. From this lecture onward, "loss design" stops being a bag of tricks and becomes a modeling choice.

# Learning outcomes · Session 1

Framework + likelihood. By end of Session 1 you can ·

DERIVATION

1. **State** the Bernoulli, Categorical, and Normal distributions and the `~` notation.
2. **Read a plate-notation** graphical model and recognize the supervised setup.
3. Explain three reasons why **the Normal shows up everywhere** (CLT, max entropy, closed-under-linear).
4. **Sample** from Bernoulli, Categorical, and Normal — and recognize the **reparameterization trick** as an affine map of a base sample.
5. **Apply Bayes' rule** and identify prior, likelihood, evidence, and posterior.
6. **Derive MSE / BCE / categorical CE** as NLL under Gaussian / Bernoulli / Categorical outputs.

# Learning outcomes · Session 2

MAP + KL + course spine. By end of Session 2 you can ·

DERIVATION

7. **Derive L2** as MAP with a Gaussian prior, **L1** as MAP with a Laplace prior, and explain L1's sparsity geometrically.
8. State and use **KL divergence**, recognize cross-entropy as KL up to a constant, and re-derive MLE/MAP through the KL lens.
9. Distinguish **forward vs reverse KL** and predict their respective failure modes (mode-covering vs mode-seeking).
10. **Connect** these foundations to VAEs, diffusion, RLHF, distillation, and the rest of the course.

PART 0

# Revision · linear & logistic regression

The loss functions we used without asking why

# Linear regression · the model

Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and continuous targets $y_i \in \mathbb{R}$, fit a linear model ·

DERIVATION

$$\hat{y}_i = \boldsymbol{\theta}^\top \mathbf{x}_i$$

(absorb the bias into $\boldsymbol{\theta}$ by appending a $1$ to each $\mathbf{x}_i$).

The prediction is a single real number — a *point estimate* of $y$.

So far, no probability anywhere in sight. We pick weights, we predict a number, we measure how wrong we are.

# Linear regression · loss & training

DERIVATION

**Loss · mean squared error**

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2$$

**Train · two equivalent options ·**

- Solve $\nabla_\theta L = 0$ in closed form · $\hat{\boldsymbol{\theta}} = (X^\top X)^{-1} X^\top \mathbf{y}$.
- Or run gradient descent · $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_\theta L$.

**Open question** · you probably justified MSE as *"penalize big errors more than small ones."* True — but **why squared and not absolute or cubed?** We'll see today.

# Logistic regression · the model

Same setup as linear regression, but now $y_i \in \{0, 1\}$ (binary classification). The output should be a *probability* between 0 and 1.

DERIVATION

$$\hat{p}_i = \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}$$

The **sigmoid** maps any real number to $(0, 1)$ — large positive logits give probabilities near 1; large negative logits give probabilities near 0; zero logit gives $0.5$.

The model output $\hat{p}_i$ is *interpreted* as $P(Y = 1 \mid \mathbf{x}_i)$. (We'll make that interpretation precise in Part 1.)

# Logistic regression · loss & training

DERIVATION

**Loss · binary cross-entropy (BCE)**

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\right]$$

**Train** · gradient descent (no closed form because of the sigmoid).

**Open question** · you justified BCE as *"big penalty when confidently wrong."* True again — but **why this exact form**, not $-(y-\hat{p})^2$ or $-|y-\hat{p}|$? Same question as MSE: where do these specific losses come from?

# Regularization · the other half-mystery

When the model overfits, you added a penalty term:

DERIVATION

**L2 (ridge)** · $L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 = L(\boldsymbol{\theta}) + \lambda \sum_j \theta_j^2$

**L1 (lasso)** · $L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 = L(\boldsymbol{\theta}) + \lambda \sum_j |\theta_j|$

You probably learned ·

- **L2** keeps weights small (smooth shrinkage).
- **L1** drives many weights to **exactly zero** (sparsity).

But **why** does L1 hit zero and L2 doesn't? Why is the penalty *squared* in L2 and *absolute* in L1? We'll derive both from first principles.

# Today's question · the four mysteries

DERIVATION

| OBJECT | RECIPE | WHERE DOES IT COME FROM? |
|---|---|---|
| MSE | $\sum (y - \hat{y})^2$ | ? |
| BCE | $-\sum [y \log \hat{p} + (1-y) \log (1-\hat{p})]$ | ? |
| L2 | $\lambda \sum \theta_j^2$ | ? |
| L1 | $\lambda \sum | \theta_j | $ | ? |

You've used all four — but none was ever *derived* from anything. They were handed to you with the magic words *"this is the loss for regression"*.

Today we replace the magic with a single principle.

# Today's promise · one principle, four answers

KEY IDEA

All four mysteries are **derived consequences** of two ideas ·

1. **The model defines a probability distribution** over $y$ given $x$ — not a single number.
2. **Pick parameters that make the data most likely** (MLE), optionally tempered by a prior on $\theta$ (MAP).

That's the whole lecture in two bullets. Everything else is unpacking.

# Bonus · the dividend in later lectures

The same machinery gives us **KL divergence** — the natural distance between distributions.

KL becomes the central object in ·

- **VAEs (L19)** · the ELBO is a KL between approximate and true posterior.
- **Diffusion (L21)** · score matching ≡ KL minimization between noisy data and model.
- **RLHF / DPO (L16)** · the reward objective is regularized by KL to a reference policy.
- **Distillation (L23)** · student matches teacher distribution by minimizing KL.

One framework today, ten lectures of dividends.

PART 1

# A probabilistic view of ML

The model doesn't predict a number — it predicts a **distribution**

# Random variable · the basic object

A **random variable** $Y$ is a quantity whose value is uncertain. It follows a distribution $p$.

DERIVATION

$$Y \sim p$$

Read · "$Y$ *is* ***distributed as*** $p$". The "$\sim$" is the central notation of this lecture.

Two flavours, depending on the type of value $Y$ takes ·

- **Discrete** $Y \in \{y_1, y_2, \ldots\}$ — described by a probability **mass** function $P(Y = y)$ summing to $1$.
- **Continuous** $Y \in \mathbb{R}$ — described by a probability **density** $p(y)$ integrating to $1$.

We'll use both. The notation is mostly the same.

# Distributions usually have parameters

Most distributions have **parameters** — knobs that shape the distribution. We collect them into a single symbol $\theta$.

DERIVATION

$$Y \sim p(\cdot \mid \theta)$$

Read · *"$Y$ is distributed as $p$, **given** parameters $\theta$."*

The vertical bar "$\mid$" means **"given"** — the same conditional notation as in $P(A \mid B)$ from your probability course.

# Parameters · three concrete examples

The parameter symbol $\theta$ is just a placeholder. For the three distributions we'll meet today ·

DERIVATION

- **Coin** · $\theta = p$ (one parameter, the bias).
  $Y \sim p(\cdot \mid p) = \text{Bernoulli}(p)$.
- **Normal** · $\theta = (\mu, \sigma^2)$ (two parameters).
  $Y \sim p(\cdot \mid \mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2)$.
- **Categorical** · $\theta = \boldsymbol{\pi}$ (a vector of $K$ probabilities summing to 1).

**In ML, $\theta$ ends up being the model's weights** — the things we estimate from data via MLE / MAP later. For now, treat $\theta$ as known.

# IID · the assumption that makes everything work

A dataset $\mathcal{D} = \{Y_1, \ldots, Y_N\}$ is **independent and identically distributed** if ·

DERIVATION

$$Y_i \overset{\text{iid}}{\sim} p(\cdot \mid \theta)$$

- **Identically distributed** · every $Y_i$ comes from the *same* distribution.
- **Independent** · knowing $Y_i$ tells you *nothing* about $Y_j$ for $i \neq j$.

These two assumptions together give us the **product factorization** ·

$$P(\mathcal{D} \mid \theta) = \prod_{i=1}^{N} P(Y_i \mid \theta)$$

This product is what becomes a sum after taking logs — and what becomes the **summed loss over a dataset** in every training loop. IID is the formal license to add up per-example losses.

When IID fails (time series, video frames, sensor logs from one device) we need different math · autoregressive models, state-space models, etc. For this course, treat batches as IID.

# Bernoulli · the coin

Outcome $Y \in \{0, 1\}$, parameter $p \in [0, 1]$ = probability of "heads."

$$Y \sim \text{Bernoulli}(p)$$

DERIVATION

**Probability mass function ·**

$$P(Y = 1 \mid p) = p$$
$$P(Y = 0 \mid p) = 1 - p$$

Two outcomes, two probabilities, summing to 1. This is the simplest non-trivial distribution.

**Examples** · email is spam (Y=1) or not (Y=0) · patient has disease or not · pixel is foreground or background.

# Bernoulli · the compact form we'll reuse

We can fold the two cases of the PMF into a single expression ·

DERIVATION

$$P(Y = y \mid p) = p^{y}\,(1-p)^{1-y}$$

**Sanity check** ·

- $y = 1 \;\Rightarrow\; p^{1}(1-p)^{0} = p$ ✓
- $y = 0 \;\Rightarrow\; p^{0}(1-p)^{1} = 1-p$ ✓

This compact form is what makes the per-example log-likelihood $y \log p + (1-y)\log(1-p)$ work for both classes simultaneously — the **seed of binary cross-entropy**. We'll use it every time we write a BCE loss.

# Bernoulli · moments

DERIVATION

**Mean** · $\mathbb{E}[Y] = 0 \cdot (1-p) + 1 \cdot p = p$

**Variance** · $\mathrm{Var}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 = p - p^2 = p(1-p)$

Variance is largest at $p = 0.5$ (most uncertain) and zero at $p \in \{0, 1\}$ (deterministic).

This will be reused when we derive **logistic regression's gradient —** it has a $\hat{p}(1-\hat{p})$ term that is exactly the Bernoulli variance at the predicted probability.

# Bernoulli · IID worked example

**Setup** · coin with $p = 0.7$. Three flips give $H, T, H$ (i.e. $Y_1, Y_2, Y_3 = 1, 0, 1$).

DERIVATION

Under the IID assumption ·

$$P(\mathcal{D} \mid p) = P(Y_1 = 1) \cdot P(Y_2 = 0) \cdot P(Y_3 = 1)$$
$$= 0.7 \cdot 0.3 \cdot 0.7 = \mathbf{0.147}$$

The **product over independent observations** is the heart of likelihood — coming up in Part 2 when we ask *"which $p$ makes the observed data most likely?"*

# Plate notation · graphical-model conventions

We now have one concrete distribution (Bernoulli). A clean way to *draw* a probabilistic model is **plate notation** — the standard for the rest of this course.

DERIVATION

| SYMBOL | MEANING |
|---|---|
| ○ | a random variable (uncertain) |
| ● (filled) | an **observed** random variable (we see its value) |
| arrow $A \to B$ | $A$ generates $B$ (i.e. $B$ depends on $A$) |
| rectangle (plate) labelled $i = 1 \dots N$ | the contents are repeated $N$ times — independence across $i$ |

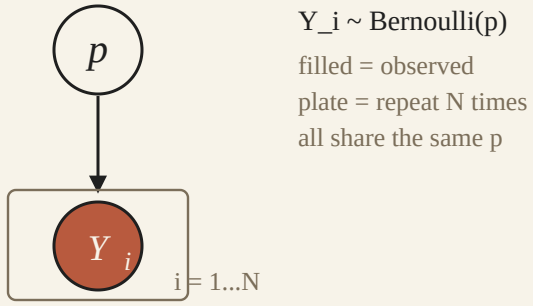
These four symbols compose every probabilistic model in this course. **Bayesian networks**, **HMMs**, **VAEs**, **diffusion models** — all drawn with these conventions.

# Plate notation · IID Bernoulli example

Apply the conventions to the simplest model · $N$ IID Bernoulli observations.

A single Bernoulli observation · $p \rightarrow \bullet Y$.

For $N$ observations, draw a plate around the repeated part ·




The plate says *"draw a fresh $Y_i$ for each $i$, all from the same Bernoulli($p$)."* The single $p$ outside the plate is **shared across all observations** — that is what makes the dataset *identically distributed*.

# Categorical · the K-sided die (formal)

Outcome $Y \in \{1, 2, \ldots, K\}$, parameter vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ with $\pi_k \geq 0$ and $\sum_k \pi_k = 1$.

$$Y \sim \text{Categorical}(\boldsymbol{\pi})$$

DERIVATION

**Probability mass function** · $P(Y = k \mid \boldsymbol{\pi}) = \pi_k$

**One-hot compact form** · let $\mathbf{y} \in \{0, 1\}^K$ with $y_k = 1$ if $Y = k$, else 0. Then ·

$$P(Y \mid \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{y_k}$$

The product collapses · only one $y_k = 1$, so only one factor survives.

**Mean** · $\mathbb{E}[\mathbf{y}] = \boldsymbol{\pi}$. **Bernoulli is the special case** $K = 2$.

# Categorical · a worked example

**MNIST classifier** outputs $\boldsymbol{\pi} = (0.05, 0.7, 0.1, 0.02, \ldots, 0.05)$ for one image (10 components, summing to 1).

DERIVATION

The model says $P(\text{class } k)$ for each digit.

If the true label is $Y = 2$ (digit "2"), then the probability the model assigned to **the truth** is

$$\pi_{Y_{\text{true}}} = \pi_2 = 0.7$$

A perfect model would put all mass on class 2 (i.e. $\boldsymbol{\pi} = (0, 0, 1, 0, \ldots, 0)$). The further the prediction is from a one-hot truth, the *less likely* the data is under it — and the larger the cross-entropy loss.

**The softmax output of *any* classifier IS a Categorical distribution.** Treat it that way and the loss falls out automatically.

# Normal (Gaussian) · the bell curve

Continuous $Y \in \mathbb{R}$ with mean $\mu$ and variance $\sigma^2$.

$$Y \sim \mathcal{N}(\mu, \sigma^2)$$

DERIVATION

**Probability density function (PDF) ·**

$$p(y \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

**Mean & variance** · $\mathbb{E}[Y] = \mu, \quad \mathrm{Var}[Y] = \sigma^2$

The most important continuous distribution in all of statistics — and the seed of the MSE loss.

# Normal · three properties to memorize

DERIVATION

1. **Centred at $\mu$**, spread controlled by $\sigma$.
2. Density falls off **exponentially in the squared distance** $(y-\mu)^2$.
3. The squared exponent will be the **seed of MSE** in Part 4.

Property 2 is what we'll lean on most. Let's unpack it.

# Normal · how fast the bell decays

Property 2 says density falls off **exponentially** in $(y-\mu)^2$. How fast?

DERIVATION

| DISTANCE FROM MEAN | EXPONENT | DENSITY FACTOR |
|---|---|---|
| $1\sigma$ | $1/2$ | $e^{-0.5} \approx 0.61$ |
| $2\sigma$ | $2$ | $e^{-2} \approx 0.14$ |
| $4\sigma$ | $8$ | $e^{-8} \approx 3 \times 10^{-4}$ |

This **squared, exponential decay** is what makes Gaussians "tightly concentrated" — almost all the mass sits within a few $\sigma$ of the mean.

**Empirical rule** · 68% within $1\sigma$, 95% within $2\sigma$, 99.7% within $3\sigma$.

# Normal · a worked numeric example

**House prices** modelled as $\mathcal{N}(50, 5^2)$ lakh.

DERIVATION

| SAMPLE VALUE $y$ | DENSITY $p(y)$ | DISTANCE FROM MEAN |
|---|---|---|
| $50$ (the mean) | $1/\sqrt{2\pi \cdot 25} \approx 0.0798$ | $0\sigma$ |
| $55$ | $\approx 0.0484$ | $1\sigma$ |
| $60$ | $\approx 0.0108$ | $2\sigma$ |
| $70$ | $\approx 2.7 \times 10^{-5}$ | $4\sigma$ |

A house priced at the mean ($Y = 50$) is the most likely. A house priced at $Y = 70$ — four standard deviations away — is vanishingly unlikely under this model.

This **squared-distance penalty** $(y - \mu)^2$ is the exact form that becomes MSE when we maximize the likelihood over data — covered in Part 4.

# Multivariate Normal · briefly

For $\mathbf{Y} \in \mathbb{R}^d$ ·

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

DERIVATION

$$p(\mathbf{y} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \, |\Sigma|}} \exp\!\left(-\tfrac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{y}-\boldsymbol{\mu})\right)$$

- $\boldsymbol{\mu} \in \mathbb{R}^d$ — mean vector.
- $\Sigma \in \mathbb{R}^{d \times d}$ — covariance matrix (symmetric, positive definite).

If $\Sigma = \sigma^2 I$ (isotropic), the components are independent. **This is the case in diffusion (L21)** — every noise step samples isotropic Gaussian noise. We'll come back to this when we get there.

# Why does the Normal show up everywhere?

Three deep reasons — each comes back later in the course.

DERIVATION

1. **Central Limit Theorem (CLT)** · the sum of many small independent things is Normal. So measurement noise, sensor jitter, biological variation all *empirically* look Gaussian.
2. **Maximum entropy** · among all distributions with given mean and variance, Normal has the largest entropy → "the most agnostic choice when all you know is mean & variance."
3. **Closed under linear operations** · sum, scaling, and conditioning on linear maps of Gaussians stay Gaussian. Bayesian updates with Gaussian prior + Gaussian likelihood give a Gaussian posterior — *conjugacy*.

These three properties together explain why the Gaussian dominates classical statistics, signal processing, **diffusion models**, Kalman filters, and Bayesian neural nets.

# Why Normal · the Central Limit Theorem

**Statement (informal)** · let $X_1, X_2, \ldots, X_N$ be IID with mean $\mu$ and variance $\sigma^2$. Define the standardized sum ·

DERIVATION

$$Z_N = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \frac{X_i - \mu}{\sigma}$$

Then as $N \to \infty$, $Z_N \to \mathcal{N}(0, 1)$ — **regardless of the original distribution** of $X_i$.

**Implication** · any quantity arising as the *aggregate of many small effects* looks Gaussian. Sensor noise, human height, daily temperature deviation — all approximately Normal because the underlying causes are sums of many small contributions.

# CLT · a worked numeric

A clean way to see CLT in action ·

DERIVATION

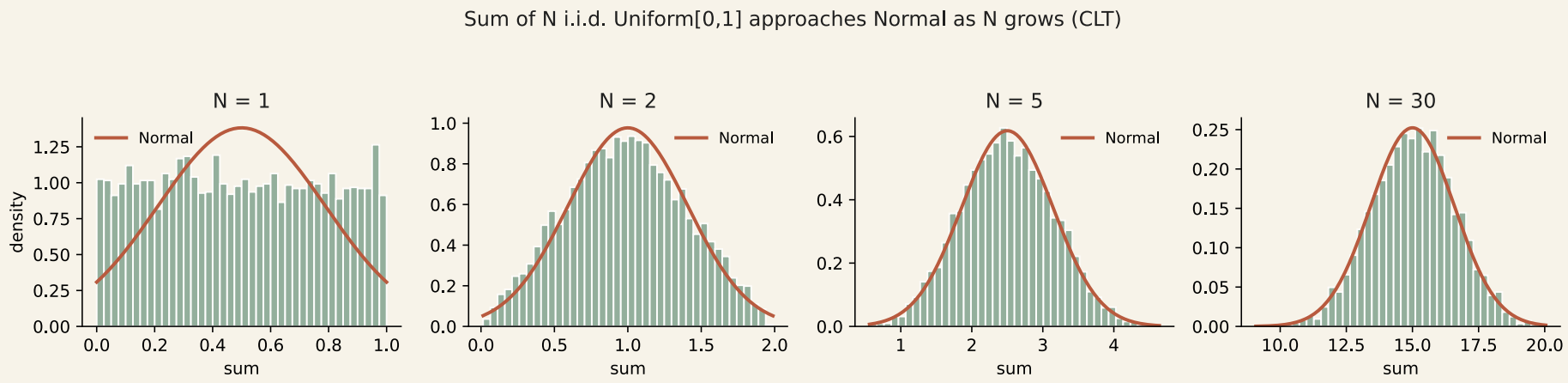
Sum of 12 IID $\mathrm{Uniform}[0, 1]$ samples ·

$$S = \sum_{i=1}^{12} U_i, \quad U_i \overset{\text{iid}}{\sim} \mathrm{Uniform}[0, 1]$$

- **Mean** of $S \cdot 12 \cdot 0.5 = 6$.
- **Variance** of $S \cdot 12 \cdot 1/12 = 1$.
- **Distribution** of $S$ · approximately $\mathcal{N}(6, 1)$.

Historically used to *generate* Gaussian samples before better algorithms (Box-Muller) existed · subtract 6, you get a draw from $\mathcal{N}(0, 1)$.

# CLT in pictures · sum of N uniforms




DERIVATION

Sum of $N$ IID $\mathrm{Uniform}[0, 1]$ samples, repeated 5000 times. **One uniform** is uniform; **two uniforms summed** form a triangular density; by **N = 30** the sum is essentially indistinguishable from a Gaussian. The CLT is not a special property of any one distribution — it's an *attractor* that almost any IID sum flows to.

# Why Normal · maximum entropy

**Entropy** quantifies "how spread-out / uncertain" a distribution is. Among all distributions with a *given* mean $\mu$ and variance $\sigma^2$ ·

DERIVATION

$$\boxed{\arg\max_{p} \; H(p) \;\; \text{s.t.} \;\; \mathbb{E}_p[Y] = \mu, \; \mathbb{E}_p[(Y-\mu)^2] = \sigma^2 \quad \Longrightarrow \quad p = \mathcal{N}(\mu, \sigma^2)}$$

**Reading** · *"if all you know about a quantity is its first two moments, the least committal probability model is Gaussian."* This is **Occam's razor for distributions** — don't bake in assumptions you can't justify.

# Other max-entropy distributions

The same max-entropy principle picks out other familiar distributions when you change the constraints ·

DERIVATION

| SUPPORT | CONSTRAINT | MAX-ENTROPY DISTRIBUTION |
|---|---|---|
| $\{0, 1\}$ | given mean | **Bernoulli** |
| Bounded interval | none beyond support | **Uniform** |
| $[0, \infty)$ | given mean | **Exponential** |
| $\mathbb{R}$ | given mean and variance | **Normal** |

Each "default" distribution in classical statistics is the *least committal* choice given some basic constraint. This is why these distributions show up so much — they are what you get when you assume nothing extra.

# Why Normal · closed under linear operations

Two structural facts make Gaussians uniquely well-behaved under linear maps ·

DERIVATION

**Sum of independent Gaussians is Gaussian.** If $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$ are independent ·

$$X + Y \sim \mathcal{N}\left(\mu_1 + \mu_2,\ \sigma_1^2 + \sigma_2^2\right)$$

**Affine transform of a Gaussian is Gaussian.**

$$aY + b \sim \mathcal{N}\left(a\mu + b,\ a^2\sigma^2\right)$$

**No other common distribution behaves this nicely.** Sum of two Bernoullis isn't Bernoulli; sum of two uniforms isn't uniform. The Gaussian is the *fixed point* of summing.

This is why Gaussians compound cleanly under repeated additive operations.

# Why Normal · conjugacy

A second, equally powerful property ·

DERIVATION

**Conjugacy.** A Gaussian prior $\mathcal{N}(\mu_0, \sigma_0^2)$ on the mean of a Gaussian likelihood $\mathcal{N}(\mu, \sigma^2)$ gives a **Gaussian posterior** with closed-form mean and variance.

No integral, no MCMC. Just two formulas.

This is why classical Bayesian regression with known noise variance is *trivial* — and why we'll have to work much harder for non-Gaussian posteriors (variational inference in L19).

# Why Normal · where this pays off (1/2)

Two consequences you'll see in the next few weeks ·

DERIVATION

**Diffusion (L21)** · the forward process adds Gaussian noise at every step. The closed-form jump

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

exists **exactly because sums of independent Gaussians are Gaussian** — no integral needed.

DERIVATION

**Kalman filter** · linear-Gaussian state-space models have a closed-form posterior at every time step. Used in robotics, control, and signal processing — and a stepping-stone to L19's variational inference.

# Why Normal · where this pays off (2/2)

DERIVATION

**Reparameterization trick (next!)** · sampling from a non-standard Normal is just an affine transform of a standard one ·

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

This is the **single most important sampling trick in deep learning** ·

- VAEs (L19) · sample latent codes through it.
- Diffusion (L21) · every denoising step uses it.
- Bayesian neural nets · weights are sampled this way.

All three rely on Gaussians being closed under affine maps. Each lecture above is a direct consequence of the two structural properties we just stated.

# A small zoo of distributions you'll meet

| NAME | NOTATION | TYPE | USED FOR |
|---|---|---|---|
| Bernoulli | $\text{Bernoulli}(p)$ | discrete, binary | binary classification |
| Categorical | $\text{Categorical}(\boldsymbol{\pi})$ | discrete, $K$-way | multiclass classification |
| Normal | $\mathcal{N}(\mu, \sigma^2)$ | continuous | regression, Gaussian noise |
| Laplace | $\text{Laplace}(\mu, b)$ | continuous, heavy-tail | L1 regularizer prior |
| Beta | $\text{Beta}(\alpha, \beta)$ | continuous on $[0, 1]$ | prior over a probability |
| Multinomial | $\text{Multinomial}(n, \boldsymbol{\pi})$ | discrete | counts in $n$ trials |

You'll need the first three today. **Laplace** comes back when we derive L1; **Beta** when we add a prior to the coin.

# The conditional view · model outputs a distribution

In supervised learning the model **does not output a number**. It outputs the *parameters of a distribution* over $Y$ given the input $\mathbf{x}$.

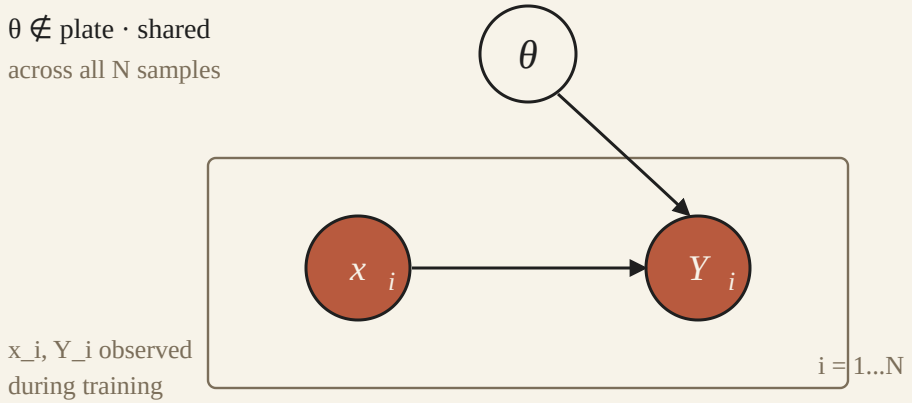
DERIVATION

| TASK | MODEL OUTPUT | CONDITIONAL DISTRIBUTION |
|---|---|---|
| Linear regression | $\hat{\mu}_\theta(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$ | $Y \mid \mathbf{x} \sim \mathcal{N}(\hat{\mu}_\theta(\mathbf{x}),\, \sigma^2)$ |
| Logistic regression | $\hat{p}_\theta(\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$ | $Y \mid \mathbf{x} \sim \text{Bernoulli}(\hat{p}_\theta(\mathbf{x}))$ |
| $K$-class softmax | $\hat{\boldsymbol{\pi}}_\theta(\mathbf{x})$ | $Y \mid \mathbf{x} \sim \text{Categorical}(\hat{\boldsymbol{\pi}}_\theta(\mathbf{x}))$ |

Training asks · *under these conditional distributions, how likely are the labels we actually saw?* Maximize that — the rest follows.

# The supervised graphical model · one picture

Every supervised learning setup we'll see in this course shares the same plate diagram ·




DERIVATION

- $\theta$ — model parameters (we will estimate by MLE / MAP).
- $\mathbf{x}_i$ — input, observed (filled).
- $Y_i$ — output, observed during training (filled), unknown at test time.
- The plate says · "$N$ IID samples, all sharing the same $\theta$."

Whether you are doing logistic regression, an MLP, a Transformer, or a diffusion model — the *outermost* graphical model is **always this**. Only the conditional distribution $p(y \mid \mathbf{x}, \theta)$ inside the plate changes.

PART 1.5

# Sampling

How we draw from distributions — and why every generative model needs it

# Why we sample · two roles

Sampling appears all over deep learning, in two distinct roles ·

DERIVATION

| ROLE | WHAT IT MEANS | EXAMPLES |
|---|---|---|
| **Generation** | Produce new instances from a learned distribution | LLM next-token, VAE images, diffusion, GAN |
| **Monte Carlo estimation** | Approximate an expectation we can't compute analytically | mini-batch SGD, REINFORCE, dropout averaging, evaluating ELBOs |

These two uses are technically the same operation — *draw* $y \sim p$ — but conceptually different. Today we set up the primitives. The advanced uses (reparameterization in VAEs, ancestral sampling in diffusion, nucleus sampling in LLMs) all reduce to combinations of what's on the next four slides.

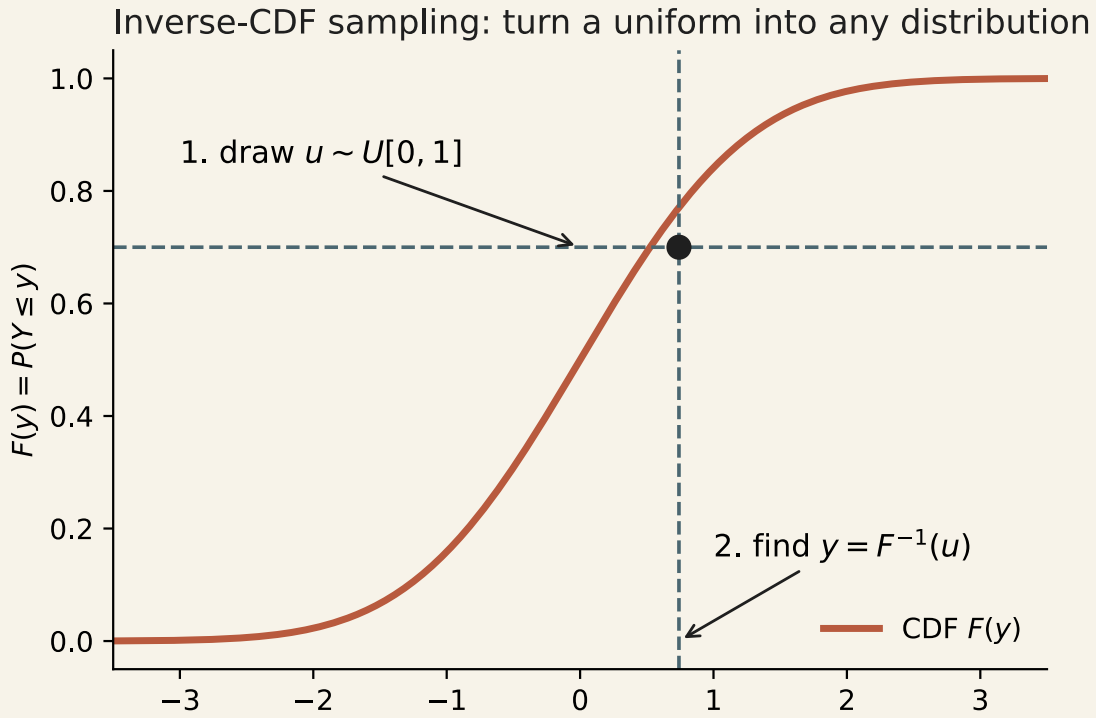
# The master sampling primitive · inverse CDF

For any 1-D distribution with CDF $F(y) = P(Y \leq y)$ ·

DERIVATION

1. Draw $u \sim \text{Uniform}[0, 1]$.
2. Return $y = F^{-1}(u)$.

**Why it works** · $P(Y \leq y) = P(F^{-1}(u) \leq y) = P(u \leq F(y)) = F(y)$. ✓



# Inverse CDF · worked on a Bernoulli

To sample $Y \sim \text{Bernoulli}(p)$ ·

DERIVATION

The CDF jumps from $0$ to $1 - p$ at $y = 0$, then from $1 - p$ to $1$ at $y = 1$.

Inverting ·

```
return 0 if u < 1 - p else 1
```

One uniform draw + one comparison · this is what `torch.bernoulli` does internally. **Same algorithm extends to any 1-D distribution** as long as you can compute the CDF.

# Sampling from a Categorical · the algorithm

Categorical with probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$. The CDF is the **stick-breaking cumulative** ·

DERIVATION

Build $c_k = \sum_{j=1}^{k} \pi_j$ (so $c_K = 1$). Draw $u \sim \mathrm{Uniform}[0, 1]$. Return the smallest $k$ such that $c_k \geq u$.

**Worked** · $\boldsymbol{\pi} = (0.1, 0.4, 0.3, 0.2)$ → cumulative $(0.1, 0.5, 0.8, 1.0)$.

Draw $u = 0.55$ · smallest $k$ with $c_k \geq 0.55$ is $k = 3$. Return class **3**.

# Categorical sampling · why it matters

This is what `torch.multinomial` does · one uniform + one cumulative scan.

KEY IDEA

**Every LLM samples its next token with exactly this algorithm.**

The model produces a vector of $K$ probabilities (where $K \approx$ vocab size, often 50,000+) and we run a single Categorical draw.

You will see this primitive in ·

- L13–L15 · LLM token generation
- L19 · VAE discrete latents
- L21 · diffusion class-conditional sampling

Knowing it once means knowing it everywhere.

# Sampling from a Normal · the affine trick

To sample $Y \sim \mathcal{N}(\mu, \sigma^2)$ ·

DERIVATION

1. Sample $\epsilon \sim \mathcal{N}(0, 1)$ (e.g. via Box-Muller, or just `torch.randn(...)`).
2. Return $y = \mu + \sigma \cdot \epsilon$.

**Why this works** · the affine transform of a Gaussian is a Gaussian (the "closed under linear ops" property from earlier). If $\epsilon \sim \mathcal{N}(0, 1)$, then $\mu + \sigma\epsilon \sim \mathcal{N}(\mu, \sigma^2)$ — exactly what we wanted.

So we only ever need a routine to draw from $\mathcal{N}(0, 1)$. Everything else is multiplication and addition.

# The reparameterization trick · preview of L19

The same affine trick is one of the most important ideas in modern DL.

KEY IDEA

Let the model output $\mu_\theta(\mathbf{x})$ and $\sigma_\theta(\mathbf{x})$ as deterministic functions of an input. To sample a **stochastic** output .

$$y = \mu_\theta(\mathbf{x}) + \sigma_\theta(\mathbf{x}) \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1)$$

The randomness $\epsilon$ is **outside** the gradient path. $\mu_\theta, \sigma_\theta$ are deterministic ⇒ we can **backprop through the sample**.

This is what makes **VAEs trainable** (L19) and powers the entire **diffusion** stack (L21–L22). You'll see this exact form repeatedly — and you now know where it comes from.

# Monte Carlo expectation · the definition

For an integral / sum we can't compute analytically ·

DERIVATION

$$\mathbb{E}_{Y\sim p}[f(Y)] \;=\; \int f(y)\,p(y)\,dy \;\approx\; \frac{1}{N}\sum_{i=1}^{N} f(y_i), \qquad y_i \sim p$$

The Monte Carlo trick · **trade an integral we can't compute for a sample mean we can.** Always works as long as we can sample from $p$.

# Monte Carlo · three properties

DERIVATION

- **Unbiased** · $\mathbb{E}\left[\frac{1}{N}\sum_i f(y_i)\right] = \mathbb{E}_p[f(Y)]$ exactly.
- **Variance scales as** $1/N$ (Law of large numbers).
- **Standard error scales as** $1/\sqrt{N}$ · quadrupling samples *halves* the error.

The $1/\sqrt{N}$ scaling is why Monte Carlo is **slow** in high dimensions but still beats numerical integration whenever $d \gtrsim 4$. Variance reduction (control variates, importance sampling, antithetic variates) is an entire research area built on accelerating this rate.

# Monte Carlo · hidden everywhere in DL

Almost every "loss" you'll write is secretly an expectation being approximated by a single sample ·

DERIVATION

| WHERE | THE EXPECTATION | ESTIMATED BY |
|---|---|---|
| **Mini-batch SGD** | $\mathbb{E}_{(x,y)\sim\text{data}}[\nabla\mathcal{L}(\theta;x,y)]$ | one batch of size $B$ |
| **VAE ELBO** (L19) | $\mathbb{E}_{z\sim q_\phi(z\|x)}[\log p_\theta(x \mid z)]$ | usually **one** $z$ sample |
| **Diffusion loss** (L21) | $\mathbb{E}_{t,\epsilon}[\|\epsilon-\hat{\epsilon}_\theta\|^2]$ | one random $t$ + one $\epsilon$ |
| **REINFORCE / RLHF** (L16) | $\mathbb{E}_{\tau\sim\pi}[R(\tau)\nabla\log\pi(\tau)]$ | a few sampled trajectories |

Each loss above looks deterministic in code — `loss = ...` returns a number. But probabilistically, **it's a sample-mean estimate** of a deeper expectation. Variance reduction (control variates, importance sampling) is a research area for exactly this reason.

# LLM token sampling · preview of L14

You now know the primitive (sampling from a Categorical). LLM generation is just Categorical sampling at every step — but with a few tweaks to control diversity ·

DERIVATION

| STRATEGY | WHAT IT DOES | WHEN TO USE |
|---|---|---|
| **Greedy** ($\arg\max$) | Pick the most likely token | Deterministic; safe but boring |
| **Temperature** $T$ | Sample from $\tilde{\pi}_k \propto \pi_k^{1/T}$ | $T \to 0$ = greedy; $T = 1$ = unchanged; $T > 1$ = more random |
| **Top-$k$** | Keep top $k$ logits, renormalize, sample | Caps diversity at the top |
| **Top-$p$ (nucleus)** | Keep smallest set with cumulative prob $\geq p$ | Adapts to the model's confidence |

L14 covers the full story. The point today · the underlying operation is *sampling from a Categorical* — exactly the inverse-CDF primitive from two slides ago.

PART 2

# Likelihood

The probability of the data, viewed as a function of $\theta$

# Coin toss · setup

You're handed a coin with **unknown** bias $p$. You flip it $N = 10$ times and observe ·

$$\mathcal{D} = \texttt{H}, \ \texttt{H}, \ \texttt{T}, \ \texttt{H}, \ \texttt{T}, \ \texttt{T}, \ \texttt{H}, \ \texttt{H}, \ \texttt{T}, \ \texttt{H}$$

Six heads, four tails.

KEY IDEA

Question · what value of $p$ is *most consistent* with this data?

Intuition says $p \approx 0.6$. We'll make that intuition rigorous and, in doing so, write down the entire framework for everything that follows.

# Likelihood · definition

The **likelihood** of a parameter $\theta$ given data $\mathcal{D}$ is the probability of observing $\mathcal{D}$ under that parameter ·

DERIVATION

$$\mathcal{L}(\theta) := P(\mathcal{D} \mid \theta)$$

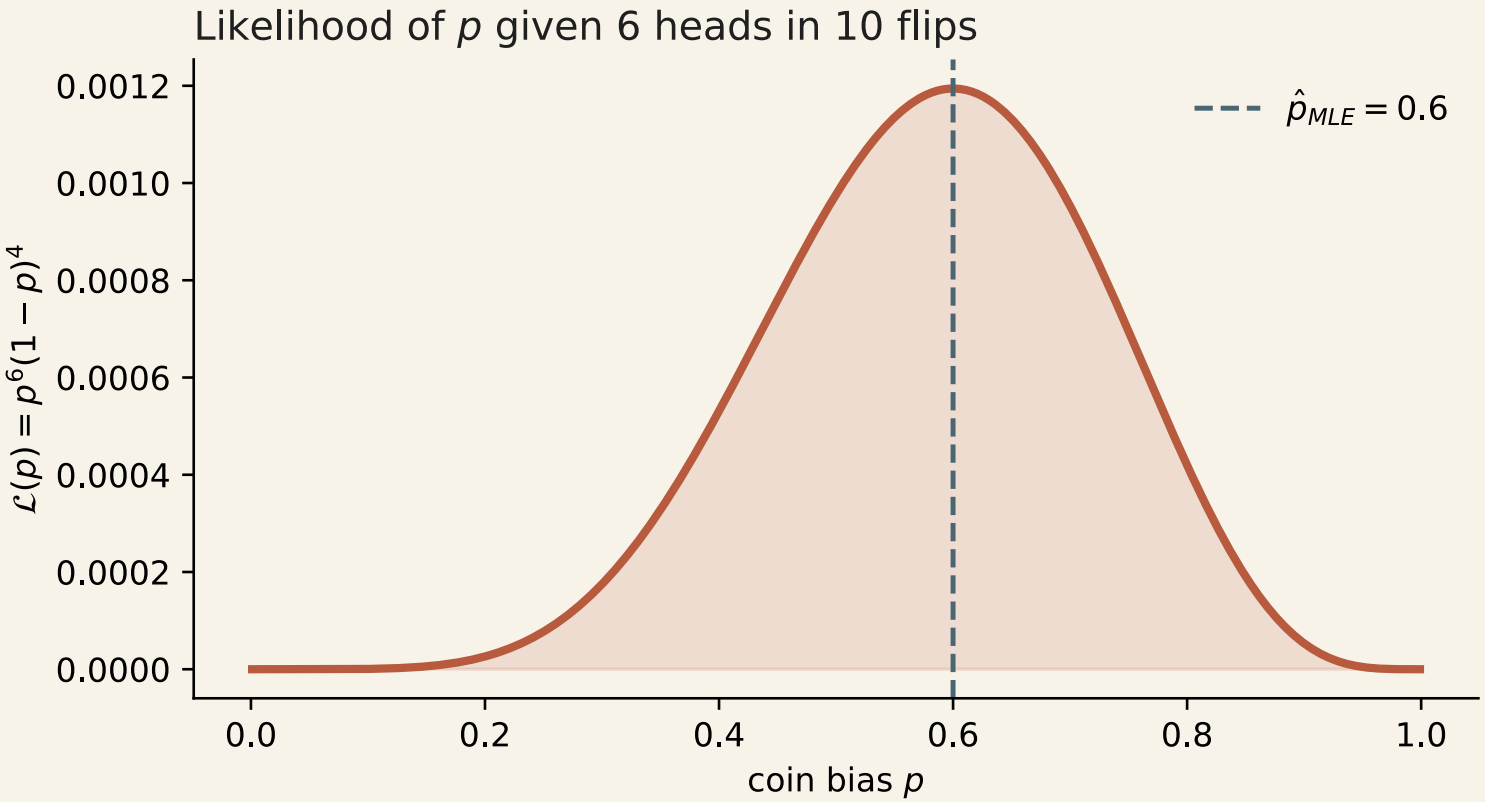
It is **not** a probability over $\theta$ (we'll get that from Bayes' rule next). It is the data's probability, viewed as a function of $\theta$.

For the coin, IID assumption gives ·

$$\mathcal{L}(p) = \prod_{i=1}^{N} P(y_i \mid p) = \prod_{i=1}^{N} p^{y_i}(1-p)^{1-y_i} = p^{\#H}\,(1-p)^{\#T}$$

For our data · $\mathcal{L}(p) = p^6(1-p)^4$.

# Likelihood · plotted




DERIVATION

$\mathcal{L}(p) = p^6(1-p)^4$, plotted on $p \in [0, 1]$.

Maximum at $p = 0.6$ — which matches our intuition (6 heads out of 10).

# Two problems with the raw likelihood

The raw likelihood $\mathcal{L}(\theta) = \prod_{i=1}^{N} P(y_i \mid \theta)$ is a *product*. Two practical issues follow ·

WATCH OUT

**Problem 1 · numerical underflow.** With $N = 1000$ and each factor $\sim 0.5$ ·

$$\mathcal{L} \approx 0.5^{1000} \approx 10^{-301}$$

This is *below* the smallest representable double-precision float ($\approx 10^{-308}$). On a computer, $\mathcal{L}$ becomes literally zero.

**Problem 2 · hard to differentiate.** The product rule on $N$ factors produces $N$ terms — algebraically and computationally messy.

We need the same answer in a form that doesn't underflow and is easy to differentiate.

# The fix · take the log

The logarithm turns products into sums and is monotonic — so maxima are preserved.

DERIVATION

$$\ell(\theta) := \log \mathcal{L}(\theta) = \log \prod_{i=1}^{N} P(y_i \mid \theta) = \sum_{i=1}^{N} \log P(y_i \mid \theta)$$

Now the dataset's "score" is a **sum of $N$ moderate negative numbers** — numerically stable and easy to differentiate term-by-term.

# NLL · the convention we'll always use

We minimize the **negative** log-likelihood (NLL) so "loss" is something we drive *down* with gradient descent ·

DERIVATION

$$\hat{\theta}_{\text{MLE}} = \arg\min_{\theta} \left[-\ell(\theta)\right] = \arg\min_{\theta} \sum_{i} \left[-\log P(y_i \mid \theta)\right]$$

We will *always* work with log-likelihood from this point onward.

KEY IDEA

**Every loss in this course is an NLL.** MSE, BCE, cross-entropy, ELBO, diffusion loss — all are just NLLs of carefully chosen distributions.

PART 3

# Bayes' rule

Inverting conditional probabilities — the foundation of MAP

# Conditional probability · refresher

For two events $A, B$ with $P(B) > 0$ ·

DERIVATION

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

Read · "given that $B$ happened, the probability $A$ also happened."

Cross-multiply and you get the **product rule** ·

$$P(A, B) = P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$$

# Bayes' rule · the formula

From the two ways to factor $P(A, B)$ ·

$$P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$$

Divide by $P(B)$ ·

DERIVATION

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

This **flips the conditional** · if you know $P(B \mid A)$ but want $P(A \mid B)$, Bayes is the bridge.

# Bayes worked example · disease test (setup)

A disease has prevalence 1%. A test has sensitivity 95% and specificity 95%.

**You test positive. How likely are you to have the disease?**

DERIVATION

Let $D$ = "have disease", $+$ = "test positive".

- **Prior** · $P(D) = 0.01$, $P(\bar{D}) = 0.99$
- **Likelihood** · $P(+ \mid D) = 0.95$, $P(+ \mid \bar{D}) = 0.05$

*Stop and guess before the next slide.*

# Bayes worked example · the answer

DERIVATION

**Evidence (total probability of testing positive) ·**

$$P(+) = 0.95 \cdot 0.01 + 0.05 \cdot 0.99 = 0.0095 + 0.0495 = 0.059$$

**Posterior ·**

$$P(D \mid +) = \frac{P(+ \mid D)\, P(D)}{P(+)} = \frac{0.95 \cdot 0.01}{0.059} \approx \mathbf{0.16}$$

Despite a "95% accurate" test, a positive result gives only **16%** chance of disease.

This is the **base-rate fallacy** — and it's the same maths we'll apply to ML when the *prior* on $\theta$ is strong but the *likelihood* of any single data point is weak.

# Bayes for ML · flip onto $\theta$

In ML, $A$ becomes the parameter $\theta$ and $B$ becomes the data $\mathcal{D}$.

DERIVATION

$$\boxed{p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}}$$

This is the **central equation of probabilistic ML**. It tells us how to update our belief about $\theta$ after seeing data $\mathcal{D}$. Each term has a name and a role.

# The four terms · names, roles, colors

DERIVATION

$$\underbrace{p(\theta \mid \mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{likelihood}}\ \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{evidence}}}$$

| TERM | WHAT IT IS | WHERE IT COMES FROM |
|---|---|---|
| **Likelihood** | how plausible is the data under $\theta$ | the model |
| **Prior** | belief about $\theta$ before seeing data | choice / domain knowledge |
| **Posterior** | updated belief about $\theta$ after data | what we compute |
| **Evidence** | $\int p(\mathcal{D} \mid \theta)\, p(\theta)\, d\theta$ — a normalizer | usually intractable, often ignored |

We will keep these colors consistent for the rest of the course.

# Why we usually ignore the evidence

The evidence $p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)\, p(\theta)\, d\theta$ is a constant **with respect to** $\theta$. It does not change as we vary $\theta$.

KEY IDEA

For finding the *single best* $\theta$ (MLE or MAP), the evidence is irrelevant ·

$$\arg\max_{\theta} p(\theta \mid \mathcal{D}) = \arg\max_{\theta} p(\mathcal{D} \mid \theta)\, p(\theta)$$

We only need · **posterior** $\propto$ **likelihood** $\times$ **prior.**

The evidence becomes important only when we want a *full posterior* — Bayesian neural nets, model comparison, ELBO in VAEs (L19). For today, MLE + MAP, we drop it.

# Bayesian updating · the dynamic view

Bayes' rule is not a one-shot operation. As more data arrives, **today's posterior becomes tomorrow's prior** ·

DERIVATION

After dataset $\mathcal{D}_1$ ·

$$p(\theta \mid \mathcal{D}_1) \propto p(\mathcal{D}_1 \mid \theta)\, p(\theta)$$

Now observe more data $\mathcal{D}_2$, independent of $\mathcal{D}_1$ given $\theta$ ·

$$p(\theta \mid \mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2 \mid \theta)\, p(\theta \mid \mathcal{D}_1)$$

The previous posterior $p(\theta \mid \mathcal{D}_1)$ now plays the role of prior. Bayesian inference is **iterative belief updating**.

Practical issue · for general distributions, the posterior may not be in the same family as the prior, so each update changes the *shape* of the formula. **Conjugate priors** (next slide) avoid this · prior and posterior stay in the same family, and updates are just parameter arithmetic.

# Conjugacy · when the math stays clean

DERIVATION

A prior $p(\theta)$ is **conjugate** to a likelihood $p(\mathcal{D} \mid \theta)$ if the posterior $p(\theta \mid \mathcal{D})$ belongs to the **same family** as the prior — only the parameters change.

| LIKELIHOOD | CONJUGATE PRIOR | POSTERIOR |
|---|---|---|
| Bernoulli($\theta$) | Beta($\alpha, \beta$) | Beta($\alpha + h, \beta + t$) |
| Categorical($\boldsymbol{\pi}$) | Dirichlet($\boldsymbol{\alpha}$) | Dirichlet($\boldsymbol{\alpha} + \mathbf{n}$) |
| Normal($\mu, \sigma^2$), $\sigma$ known | Normal($\mu_0, \sigma_0^2$) | Normal (closed-form $\mu, \sigma$) |
| Poisson($\lambda$) | Gamma($\alpha, \beta$) | Gamma($\alpha + n, \beta + N$) |

Conjugacy is **why classical Bayesian statistics looks easy** · all the integrals collapse. Once we go to deep neural nets, conjugacy breaks and we need approximations (variational inference, MCMC) — but for this lecture, conjugacy makes the coin example airtight.

# Beta · prior over a probability

The **Beta distribution** is supported on $[0, 1]$ — perfect for putting a prior on a Bernoulli's $p$.

DERIVATION

$$p \sim \text{Beta}(\alpha, \beta), \qquad p(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha, \beta)}$$

**Mean** · $\mathbb{E}[p] = \dfrac{\alpha}{\alpha + \beta}$

The shape depends on $(\alpha, \beta)$ ·

- $\alpha = \beta = 1$ · **uniform on** $[0, 1]$ — completely flat prior.
- $\alpha = \beta = 2$ · "weakly fair" — peaks at $0.5$, allows variation.
- $\alpha, \beta$ large · sharply concentrated near $\alpha/(\alpha + \beta)$.

So $\alpha, \beta$ jointly control where the prior peaks **and** how concentrated it is.

# Beta · the pseudo-count interpretation

There's a much more intuitive way to read $\alpha, \beta$ ·

KEY IDEA

Think of $\alpha$ and $\beta$ as **"pseudo-counts"** — fake prior heads and tails you've already seen *before* the experiment.

Examples ·

- $\text{Beta}(2, 2)$ ≈ "I've seen 1 fake head and 1 fake tail." Mild belief in fairness.
- $\text{Beta}(50, 50)$ ≈ "I've seen 49 fake heads and 49 fake tails." Strong belief that $p \approx 0.5$ — would take a lot of real data to shift.
- $\text{Beta}(1, 9)$ ≈ "I've seen 0 fake heads and 8 fake tails." Strong belief the coin is biased toward tails.

This pseudo-count framing is what makes the conjugate update so clean — the posterior just **adds the real counts** to the pseudo-counts. We'll see that on the next slide.

# Beta-Binomial · prior and likelihood

The two ingredients ·

DERIVATION

**Prior** · $p \sim \mathrm{Beta}(\alpha, \beta)$

$$p(p) \;\propto\; p^{\alpha-1}(1-p)^{\beta-1}$$

**Likelihood** · $h$ heads and $t$ tails in $N = h + t$ flips

$$p(\mathcal{D} \mid p) \;\propto\; p^{h}(1-p)^{t}$$

(we drop the binomial coefficient — it's constant in $p$).

Both are products of $p$-power times $(1-p)$-power terms. That's the structural reason conjugacy works — multiplying them stays in the same form.

# Beta-Binomial · derive the posterior

By Bayes' rule, posterior $\propto$ likelihood $\times$ prior ·

DERIVATION

$$p(p \mid \mathcal{D}) \;\propto\; \underbrace{p^{h}(1-p)^{t}}_{\text{likelihood}} \cdot \underbrace{p^{\alpha-1}(1-p)^{\beta-1}}_{\text{prior}}$$

Combine exponents ·

$$= p^{h+\alpha-1}\,(1-p)^{t+\beta-1}$$

This is the *kernel* of $\mathrm{Beta}(h+\alpha,\; t+\beta)$.

The posterior is in the **same family** as the prior — that's what "conjugate" means. We started with a Beta, multiplied by a Bernoulli/Binomial likelihood, and got another Beta out.

# Beta-Binomial · the update rule

DERIVATION

$$\text{Beta}(\alpha, \beta) \xrightarrow{h \text{ heads},\ t \text{ tails}} \text{Beta}(\alpha + h,\ \beta + t)$$

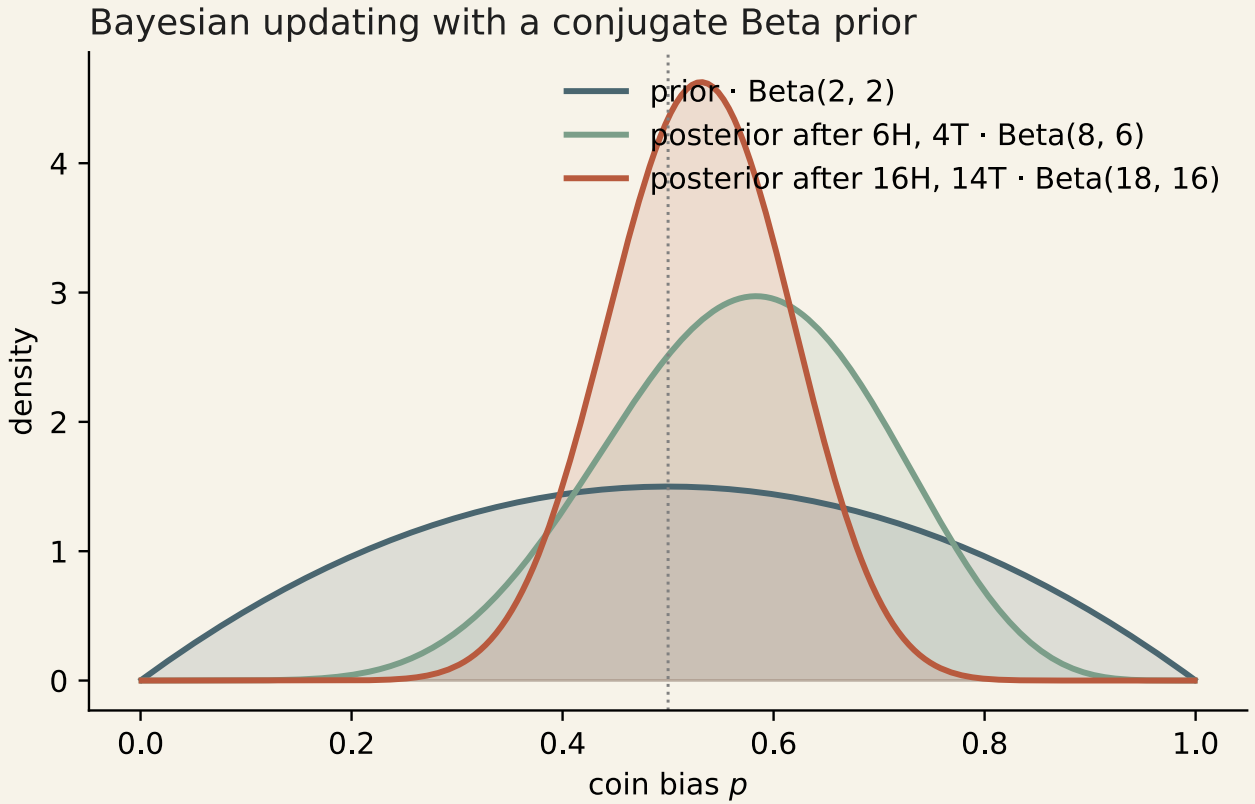
**Interpretation** · just **add the observed counts to the pseudo-counts**. That's the whole update — no integral, no normalization, no MCMC.

This is why we framed $\alpha, \beta$ as "pseudo-counts" earlier. They literally combine with real counts by addition.

**Sequential updating** · because conjugacy preserves the family, you can update *one observation at a time* and the formulas stay the same. Each new flip just adds $1$ to either $\alpha$ or $\beta$.

This is the **cleanest possible Bayesian inference**, and it's what made Bayesian statistics tractable in the pre-MCMC era. Modern DL breaks conjugacy (neural-net likelihoods aren't conjugate to anything) → we need approximations like variational inference (L19) or sampling.

# Beta-Binomial · the picture




DERIVATION

Start with a weakly-fair prior $\mathrm{Beta}(2, 2)$. After 6 heads in 10 flips · posterior is $\mathrm{Beta}(8, 6)$, peaking near $7/12 \approx 0.583$. After another 10 flips with the same rate (16H, 14T total) · $\mathrm{Beta}(18, 16)$, even sharper near $0.529$.

# Beta-Binomial · numeric update (work it out)

Prior · $\mathrm{Beta}(2, 2)$ · pseudo-counts $(1\,H, 1\,T)$, prior mean $= 2/4 = 0.5$.

Observe · $h = 6, t = 4$ in $N = 10$ flips · MLE $= 0.6$.

**Step 1 · update parameters**

$$\alpha' = \alpha + h = 2 + 6 = 8, \quad \beta' = \beta + t = 2 + 4 = 6$$

**Step 2 · posterior summaries**

$$\mathbb{E}[p \mid \mathcal{D}] = \frac{8}{14} \approx 0.571, \quad \text{mode} = \frac{8 - 1}{14 - 2} = \frac{7}{12} \approx 0.583$$

The posterior mean **lives between** the prior mean $(0.5)$ and the MLE $(0.6)$ — exactly what "shrinkage" means. With more data, MAP $\to$ MLE; with less data, the prior pulls the estimate toward $0.5$.

# Estimator #1 · Maximum Likelihood (MLE)

The simplest possible estimator · **ignore the prior**, just maximize the likelihood.

DERIVATION

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} p(\mathcal{D} \mid \theta)$$

Read · *"what value of $\theta$ best explains the data we actually saw?"*

This is what we'll spend Part 4 of this lecture on. Coin → linear regression → logistic regression → multiclass — all derive their loss as the negative log-likelihood under MLE.

# Estimator #2 · Maximum a Posteriori (MAP)

The Bayesian-flavoured estimator · **use the prior**, maximize the full posterior.

DERIVATION

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \; p(\mathcal{D} \mid \theta)\, p(\theta)$$

Read · *"given my prior belief and the data, what's the most probable $\theta$?"*

This is Part 5 (Session 2). MAP = MLE *plus* a regularizer that comes from the log-prior. With a Gaussian prior we'll recover **L2**; with a Laplace prior we'll recover **L1**. Same machinery, different prior.

# MLE vs MAP · the master sentence

KEY IDEA

**MAP = MLE + a prior on $\theta$.**

That single sentence is what makes **regularization fall out of the same machinery as the loss**. There is no separate "loss" and "regularizer" theory — it's all one Bayesian story, and L2/L1 are just choices of prior.

When the data is plentiful, the likelihood dominates and MAP $\rightarrow$ MLE. When data is scarce, the prior matters — and that's exactly when overfitting bites and regularization helps. Bayes' rule formalizes this *automatically*.

PART 4

# Maximum Likelihood Estimation

Concrete derivations · coin · linear regression · logistic regression

# Pop quiz · which course did this student come from?

Three sections of the same exam · grades modelled as Normal ·

| COURSE | MEAN $\mu$ | STD $\sigma$ |
|---|---|---|
| C1 | 80 | 10 |
| C2 | 70 | 10 |
| C3 | 90 | 5 |

A student's marks $y = 82$. Which course is this student most likely from?

*Stop and guess before the next slide.*

# Answer · pick the distribution that best explains the data

Evaluate the density $\mathcal{N}(82 \mid \mu, \sigma^2)$ for each course ·

$$p_1(82) \approx 0.0388, \quad p_2(82) \approx 0.0194, \quad p_3(82) \approx 0.0299$$

Course **C1** wins · its bell is centred close to 82 with reasonable spread.

DERIVATION

**MLE intuition** · among candidate distributions, choose the one under which the **observed data has the highest probability**.

This is the entire idea. Everything below is just doing it carefully.

# MLE for the coin · setup

Data · $\#H = h = 6$ heads out of $N = 10$ flips. Likelihood · $\mathcal{L}(p) = p^h (1-p)^{N-h}$

Take the log ·

DERIVATION

$$\ell(p) = h \log p + (N - h) \log(1 - p)$$

**Step 1 · differentiate.**

$$\frac{d\ell}{dp} = \frac{h}{p} - \frac{N - h}{1 - p}$$

# MLE for the coin · solve

**Step 2 · set derivative to zero.**

$$\frac{h}{p} - \frac{N-h}{1-p} = 0 \implies h(1-p) = (N-h)\,p$$

Expand · $h - hp = Np - hp \Longrightarrow h = Np$.

DERIVATION

$$\boxed{\hat{p}_{\text{MLE}} = \frac{h}{N} = \frac{\#H}{\#H + \#T}}$$

For our data $h = 6, N = 10 \cdot \hat{p} = 0.6$. **Exactly** the empirical frequency.

This is your first MLE derivation. Same recipe applies to everything below.

# The MLE recipe · 3 steps

To find the MLE for any model ·

DERIVATION

1. **Pick a probabilistic model** — choose a distribution $p(y \mid \theta)$ that matches your output type (Bernoulli for binary, Normal for continuous, …).
2. **Write the log-likelihood** of the dataset under that model · sum the per-example $\log p(y_i \mid \theta)$.
3. **Maximize over $\theta$** — by setting derivative to zero analytically, or by gradient ascent (equivalently, gradient descent on the negative log-likelihood).

We will now apply this recipe to **linear regression** (where the answer pops out as MSE) and **logistic regression** (where it pops out as BCE).

# MLE for linear regression · the assumption

Modelling choice · the target is a linear function plus Gaussian noise ·

$$y = \boldsymbol{\theta}^\top \mathbf{x} + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

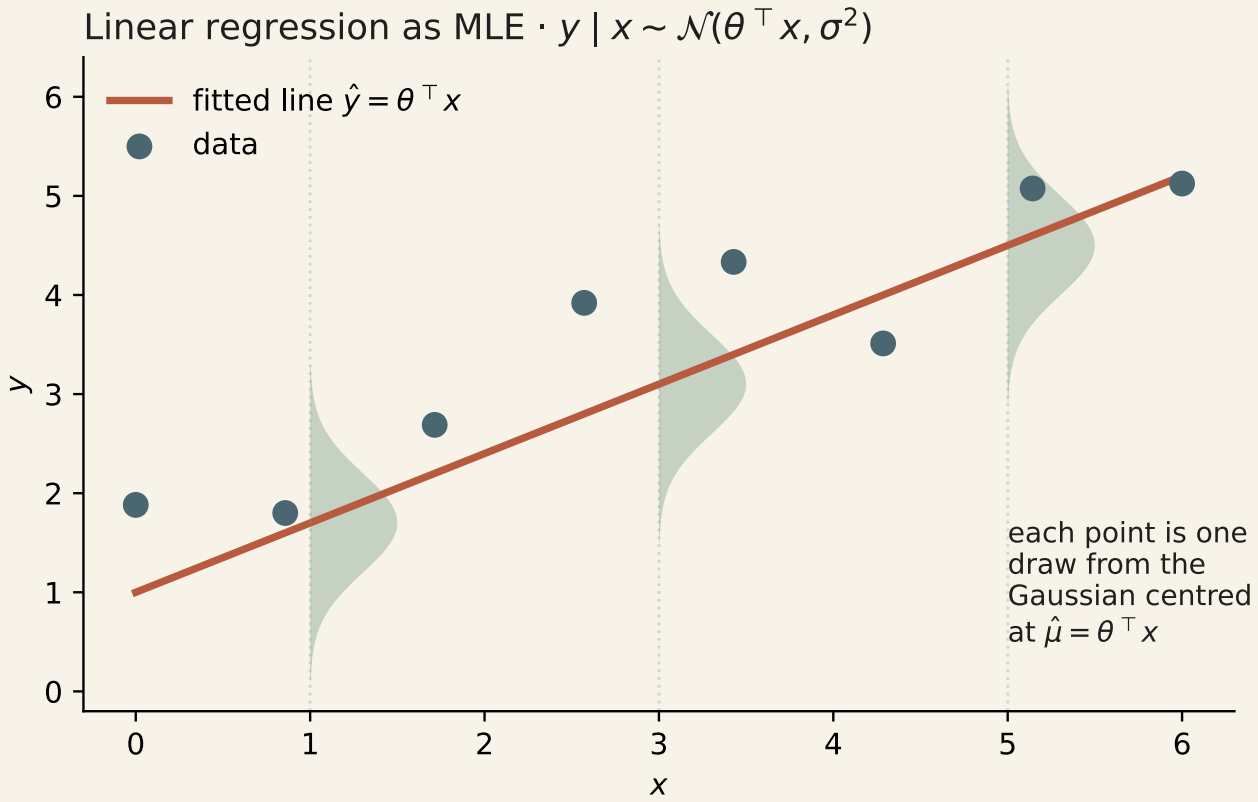
Equivalently · the conditional distribution of $Y$ given $\mathbf{x}$ is

DERIVATION

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y \mid \boldsymbol{\theta}^\top \mathbf{x}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(y - \boldsymbol{\theta}^\top \mathbf{x})^2}{2\sigma^2} \right)$$

The model's prediction $\hat{\mu} = \boldsymbol{\theta}^\top \mathbf{x}$ is the **mean** of the Gaussian. Real $y$ scatters around it with variance $\sigma^2$.

# Linear regression as MLE · the picture




DERIVATION

Each data point is one **draw** from a Gaussian whose mean lies on the regression line. MLE asks · *which line* $\boldsymbol{\theta}^\top \mathbf{x}$ *makes the observed* $\boldsymbol{y}$*'s most probable?*

Equivalently · *which line minimizes the squared distance from each point to the line* — exactly the OLS

# MLE for linear regression · log-likelihood

For a single example,

$$\log p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = -\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2}{2\sigma^2}$$

Sum over the dataset · log of a product of IID terms is a sum.

DERIVATION

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \underbrace{-\frac{N}{2}\log(2\pi\sigma^2)}_{\text{constant in } \theta} - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2$$

Only the second term depends on $\boldsymbol{\theta}$.

# MLE for linear regression · MSE pops out

Maximizing $\ell$ over $\boldsymbol{\theta}$ ·

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\theta} \left[ -\frac{1}{2\sigma^2} \sum_i (y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2 \right]$$

The factor $-1/(2\sigma^2)$ is a negative constant — flip the sign and minimize ·

DERIVATION

$$\boxed{\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2}$$

**This is exactly MSE.** MSE is not a heuristic — it is the **MLE under Gaussian noise**.

If the noise had been Laplace ($\epsilon \sim \text{Laplace}$), the same derivation would give MAE (mean absolute error) instead. Loss design = noise model.

# Closed-form OLS · for completeness

The MSE objective is quadratic in $\boldsymbol{\theta}$, so we can set the gradient to zero analytically.

Stack data · $X \in \mathbb{R}^{N \times d}$, $\mathbf{y} \in \mathbb{R}^N$. Then $\sum (y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2 = \|\mathbf{y} - X\boldsymbol{\theta}\|^2$.

DERIVATION

$$\nabla_\theta \|\mathbf{y} - X\boldsymbol{\theta}\|^2 = -2X^\top(\mathbf{y} - X\boldsymbol{\theta}) = 0$$

$$\boxed{\hat{\boldsymbol{\theta}}_{\text{MLE}} = (X^\top X)^{-1} X^\top \mathbf{y}}$$

The familiar **normal equation** is the closed-form MLE under Gaussian noise. SGD / gradient descent gives the same answer iteratively.

# Worked OLS by hand · setup

Let $x_i \in \mathbb{R}$ (single feature, no bias for clarity). Fit a scalar slope · $\hat{y} = \theta x$.

DERIVATION

**Data ·**

$$(x_1, y_1) = (1, 1.1)$$
$$(x_2, y_2) = (2, 1.9)$$
$$(x_3, y_3) = (3, 3.2)$$

We want the $\theta$ that minimizes $\sum_i (y_i - \theta x_i)^2$. With one parameter, the closed-form OLS becomes a simple ratio · $\hat{\theta} = \sum x_i y_i / \sum x_i^2$.

Let's compute it step by step.

# Worked OLS by hand · solve

DERIVATION

**Step 1 · denominator $X^\top X$ ·**

$$\sum x_i^2 = 1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$$

**Step 2 · numerator $X^\top \mathbf{y}$ ·**

$$\sum x_i y_i = 1 \cdot 1.1 + 2 \cdot 1.9 + 3 \cdot 3.2 = 1.1 + 3.8 + 9.6 = 14.5$$

**Step 3 · solve ·**

$$\hat{\theta} = (X^\top X)^{-1} X^\top \mathbf{y} = \frac{14.5}{14} \approx \mathbf{1.036}$$

That's the OLS estimate by hand. No matrix inversion needed for $d = 1$ — just a ratio.

# Worked OLS by hand · verify

**Predictions** under $\hat{\theta} = 1.036$ ·
$\hat{y}_1 = 1.036, \quad \hat{y}_2 = 2.071, \quad \hat{y}_3 = 3.107$

**Residuals** $r_i = y_i - \hat{y}_i$ ·
$r_1 = 1.1 - 1.036 = 0.064$
$r_2 = 1.9 - 2.071 = -0.171$
$r_3 = 3.2 - 3.107 = 0.093$

**Sum-of-squared residuals** · $r_1^2 + r_2^2 + r_3^2 \approx 0.043.$

KEY IDEA

**Sanity check** · the MSE gradient at $\hat{\theta}$ should be zero.
$\nabla_\theta \mathrm{MSE}(\hat{\theta}) = -2 \cdot 14 \cdot (\hat{\theta} - 14.5/14) = 0$ ✓

This is exactly the answer `torch.linalg.lstsq` returns. We just did it by hand to feel the closed form.

# MLE for logistic regression · the assumption

Now $y \in \{0, 1\}$. Model ·

$$Y \mid \mathbf{x} \sim \text{Bernoulli}\big(\hat{p}_\theta(\mathbf{x})\big), \qquad \hat{p}_\theta(\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$$

Per-example probability ·

DERIVATION

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}) = \hat{p}_\theta(\mathbf{x})^{\,y} \left(1 - \hat{p}_\theta(\mathbf{x})\right)^{1-y}$$

This is the same compact Bernoulli form as the coin — except $\hat{p}$ now depends on $\mathbf{x}$.

# MLE for logistic regression · log-likelihood

Take the log ·
$\log p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)$
where $\hat{p}_i = \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i)$.

Sum over the dataset and **negate** to get NLL ·

DERIVATION

$$L_{\text{NLL}}(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \left[ y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \right]$$

**This is exactly binary cross-entropy.** Same story · BCE is not invented — it's the **MLE under a Bernoulli output**.

# MLE for logistic regression · gradient

Differentiate $L_{\text{NLL}}$ w.r.t. $\boldsymbol{\theta}$. Using $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ and the chain rule (derivation in any ML textbook) ·

DERIVATION

$$\nabla_\theta L_{\text{NLL}} = \sum_{i=1}^{N} (\hat{p}_i - y_i)\, \mathbf{x}_i$$

The gradient is "**prediction minus truth**, weighted by input." This is the exact same form as the linear regression gradient $(\hat{y}_i - y_i)\, \mathbf{x}_i$.

That is **not** a coincidence — it's a feature of *generalized linear models*, all derived from MLE.

# Multiclass · the assumption

Now $y \in \{1, 2, \ldots, K\}$ — one of $K$ mutually exclusive classes. We need the model to output a full **probability distribution** over the $K$ classes. We use the **softmax**.

DERIVATION

For each class $k$, learn a weight vector $\boldsymbol{\theta}_k \in \mathbb{R}^d$. Compute logits $z_k = \boldsymbol{\theta}_k^\top \mathbf{x}$ for $k = 1, \ldots, K$.

**Softmax** turns logits into probabilities ·

$$\hat{\pi}_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})}$$

Two properties · $\hat{\pi}_k > 0$ (because $\exp > 0$) and $\sum_k \hat{\pi}_k = 1$. So $\hat{\boldsymbol{\pi}}$ is a valid distribution.

Modelling assumption · $Y \mid \mathbf{x} \sim \text{Categorical}(\hat{\boldsymbol{\pi}}_\theta(\mathbf{x}))$. Binary logistic is the special case $K = 2$ (with the softmax collapsing to a sigmoid).

# Multiclass · the per-example log-likelihood

If example $i$ has true class $y_i \in \{1, \dots, K\}$, the probability the model assigns to that label is $\hat{\pi}_{i,\, y_i}$ — i.e. the entry of the softmax at position $y_i$.

DERIVATION

$$p(y_i \mid \mathbf{x}_i, \theta) = \hat{\pi}_{i,\, y_i}$$

Equivalently with one-hot encoding $\mathbf{y}_i \in \{0, 1\}^K$ ·

$$p(y_i \mid \mathbf{x}_i, \theta) = \prod_{k=1}^{K} \hat{\pi}_{i,k}^{\, y_{i,k}}$$

Take logs ·

$$\log p(y_i \mid \mathbf{x}_i, \theta) = \sum_{k=1}^{K} y_{i,k} \, \log \hat{\pi}_{i,k} = \log \hat{\pi}_{i,\, y_i}$$

Same compact-Bernoulli trick as before — only one term in the sum survives because the one-hot vector has a single 1.

# Multiclass · NLL = categorical cross-entropy

Sum over the dataset and negate ·

DERIVATION

$$L_{\text{NLL}}(\theta) = -\sum_{i=1}^{N} \log \hat{\pi}_{i,\, y_i} \;=\; -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\, \log \hat{\pi}_{i,k}$$

The right-hand form is the textbook **categorical cross-entropy** — true distribution $\mathbf{y}_i$ (one-hot) against predicted distribution $\hat{\boldsymbol{\pi}}_i$.

This loss powers every multiclass classifier in this course · CIFAR-10 ($K = 10$), ImageNet ($K = 1000$), and the **next-token loss in every LLM** ($K = $ vocab size, often 50,000+) in L13–L15.

# Multiclass · the elegant gradient

Differentiate $L_{\text{NLL}}$ w.r.t. the logit $z_{i,k}$ (derivation uses the softmax-Jacobian trick) ·

DERIVATION

$$\frac{\partial L_{\text{NLL}}}{\partial z_{i,k}} = \hat{\pi}_{i,k} - y_{i,k}$$

**Prediction minus truth, per logit.** Same elegant form as binary logistic ($\hat{p} - y$) and linear regression ($\hat{y} - y$). All three are instances of the same generalized-linear-model identity.

This is why **softmax + cross-entropy** are *always* implemented as one fused op (`F.cross_entropy(logits, target)`) — both for numerical stability (log-sum-exp trick) and because the gradient simplifies dramatically when you do them together.

# Multiclass · worked numeric

3 classes (cat, dog, car). Image with true class **cat** (index 0). Model produces logits $\mathbf{z} = [2.0,\ 1.0,\ 0.1]$.

DERIVATION

**Softmax** (subtract max for stability) · $\mathbf{z} - 2.0 = [0,\ -1,\ -1.9]$.
$\exp = [1.000,\ 0.368,\ 0.150]$ · sum $= 1.518$.
$\hat{\boldsymbol{\pi}} = [\,0.659,\ 0.242,\ 0.099\,]$.

**Per-example loss** · NLL of the true class.
$L = -\log \hat{\pi}_{\text{cat}} = -\log 0.659 = \mathbf{0.418}$

**Gradient on logits** · $\hat{\boldsymbol{\pi}} - \mathbf{y} = [0.659, 0.242, 0.099] - [1, 0, 0] = [-0.341,\ 0.242,\ 0.099]$.

The gradient says · *push the cat-logit up* (negative gradient → the optimizer subtracts it, so cat-logit increases), *push the dog and car logits down*. Exactly what you want.

# Summary so far · the pattern

DERIVATION

| OUTPUT | DISTRIBUTION CHOSEN | NLL TURNS OUT TO BE |
|---|---|---|
| Continuous $y \in \mathbb{R}$ | Normal | **MSE** |
| Binary $y \in \{0, 1\}$ | Bernoulli | **BCE** |
| $K$-class $y \in \{1, \ldots, K\}$ | Categorical | **CE** |

KEY IDEA

Every loss = **NLL under an assumed conditional distribution** $p(y \mid \mathbf{x}, \theta)$.

Pick the distribution to match your data. The loss falls out automatically. No more "memorize MSE for regression and CE for classification" — both come from the same place.

# A worked numeric · BCE on one prediction

Cat-vs-dog classifier · model output $\hat{p} = 0.8$ for one image.

| TRUE CLASS $y$ | LOG-LIKELIHOOD | NLL = LOSS | MODEL "HAPPY"? |
|---|---|---|---|
| 1 (cat) | $\log 0.8 = -0.223$ | $0.223$ | yes — small loss |
| 0 (dog) | $\log 0.2 = -1.609$ | $1.609$ | no — big loss |

KEY IDEA

The loss is small **iff the model assigned high probability to the true class**. That's all cross-entropy is doing — and that's all "maximizing log-likelihood" means once you write it out.

# Session 1 · putting it all together

DERIVATION

1. A model defines a **conditional distribution** $p(y \mid x; \theta)$.
2. **Likelihood** of a dataset $= \prod_i p(y_i \mid x_i; \theta)$ · take $\log$ and sum.
3. **MLE** $= \arg\max_\theta \sum_i \log p(y_i \mid x_i; \theta) = \arg\min_\theta \mathrm{NLL}$.
4. Plug in the right distribution and the right loss falls out automatically ·
   - Normal $\Rightarrow$ MSE
   - Bernoulli $\Rightarrow$ BCE
   - Categorical $\Rightarrow$ Cross-entropy
   - Poisson $\Rightarrow$ exp-link Poisson loss
5. **Bayes' rule** turns this into a *belief-update* story · posterior $\propto$ likelihood $\times$ prior. With conjugate Beta prior, updates are just adding counts.

Session 2 will turn the prior into **regularization** and KL divergence into the *single language* used by every model in the rest of the course.

# Session 1 · practice problems

Try these on paper; answers worked through in the notebook (`lec00-mle-map.ipynb`).

DERIVATION

**P1.** A coin gives 12 heads in 20 flips. Compute the MLE for $p$ and the negative log-likelihood at that estimate.

**P2.** Show that for $Y \mid x \sim \mathcal{N}(\theta x, \sigma^2)$ with **known** $\sigma^2$, the MLE for $\theta$ is $\hat{\theta} = \sum_i x_i y_i / \sum_i x_i^2$. (Hint · single-feature case of OLS.)

**P3.** Write down the conditional distribution and the per-example log-likelihood for **Poisson regression** ($Y \mid x \sim \text{Poisson}(\exp(\theta^\top x))$). What is the resulting NLL loss?

**P4.** For a 3-class softmax with logits $\mathbf{z} = [1, 2, 0]$ and true class $y = 1$, compute (a) the predicted probabilities, (b) the cross-entropy loss, (c) the gradient on each logit.

**P5.** A coin's true bias is $p = 0.3$. You see 0 heads in 5 flips. What is the MLE? Why is the answer absurd, and what would a $\text{Beta}(2, 2)$ prior change?

# End of Session 1

So far · **probabilistic framework + MLE.** Bernoulli, Categorical, Normal · likelihood and log-likelihood · sampling primitives · Bayes' rule with its four named terms · Beta-Binomial conjugate updates · MLE for the coin, linear regression, logistic regression, and multiclass — all derived as NLL of a chosen distribution.

**Session 2 picks up from here** · MAP and regularization (L2 from Gaussian prior, L1 from Laplace), KL divergence and information theory as the unifying lens, and how it all fans out into VAEs, diffusion, RLHF, and the rest of the course.

# Welcome to Session 2

Recap of session 1 · the model defines a distribution, and MLE = NLL minimization. **Today** · what changes when we add a *prior* — and why KL divergence is the right language for everything that comes next.

PART 5

# MAP and the meaning of regularization

L2 from a Gaussian prior · L1 from a Laplace prior

# Why we need a prior · MLE's trap

You flip a coin **3 times** and see **3 heads**. MLE says $\hat{p}_{\text{MLE}} = 3/3 = 1.0$.

WATCH OUT

According to MLE, this coin is **certainly biased to always land heads**. Future tails impossible.

This is absurd. Three flips is not enough evidence to make such an extreme claim. You *know* most coins are roughly fair.

The fix · **encode that prior knowledge** into the inference. That gives MAP.

In ML, the analogous trap is overfitting · with finite data, MLE drives weights to whatever value most exactly fits the training set, even if those values are wildly extreme. A prior pulls them back.

# MAP · maximum a posteriori

By Bayes, $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta)$.

DERIVATION

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

In log space ·

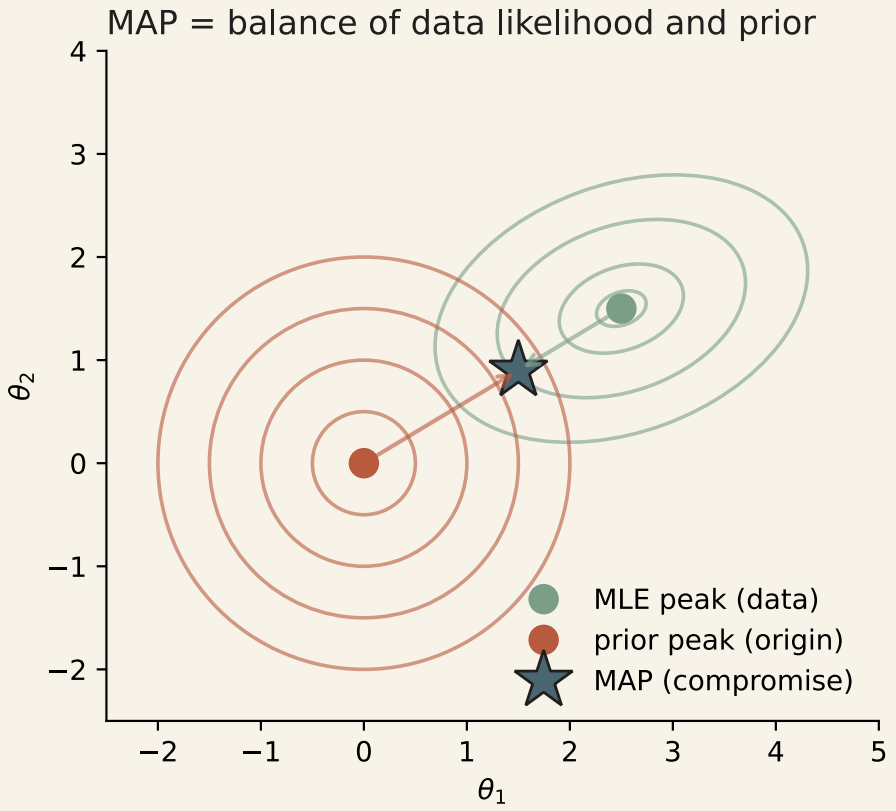
$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \Big[ \underbrace{\log p(\mathcal{D} \mid \theta)}_{\text{log-likelihood}} + \underbrace{\log p(\theta)}_{\text{log-prior}} \Big]$$

Equivalently, **minimize the negative** ·

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta} \big[\, L_{\text{NLL}}(\theta) - \log p(\theta) \,\big]$$

The first term is your usual loss. The second term is whatever the prior gives. **That second term is what we will recognize as L1 or L2.**

# MAP · the geometric picture




DERIVATION

The MAP estimate is the point in $\theta$-space that **best balances** the data's preferences (likelihood) against your prior beliefs (prior).

# Gaussian prior · setup

We choose a prior $p(\theta_j) = \mathcal{N}(0, \sigma_p^2)$ for every weight, independently. In words · *"a priori, weights are small and centred at zero."*

DERIVATION

For a single weight ·

$$\log p(\theta_j) = -\frac{\theta_j^2}{2\sigma_p^2} + \text{const}$$

For the whole vector $\boldsymbol{\theta} \in \mathbb{R}^d$ (independent prior on each component) ·

$$\log p(\boldsymbol{\theta}) = \sum_j \log p(\theta_j) = -\frac{1}{2\sigma_p^2} \sum_j \theta_j^2 + \text{const} = -\frac{1}{2\sigma_p^2} \|\boldsymbol{\theta}\|_2^2 + \text{const}$$

The constant doesn't depend on $\boldsymbol{\theta}$, so it drops out of the optimization.

# Gaussian prior · L2 pops out

Plug into the MAP objective ·

$$L_{\text{MAP}}(\boldsymbol{\theta}) = L_{\text{NLL}}(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$

$$= L_{\text{NLL}}(\boldsymbol{\theta}) + \frac{1}{2\sigma_p^2}\|\boldsymbol{\theta}\|_2^2 + \text{const}$$

DERIVATION

$$\boxed{L_{\text{MAP}}(\boldsymbol{\theta}) = L_{\text{NLL}}(\boldsymbol{\theta}) + \lambda\,\|\boldsymbol{\theta}\|_2^2, \qquad \lambda := \frac{1}{2\sigma_p^2}}$$

**This is exactly L2 regularization (a.k.a. ridge, weight decay).**

- Strong prior (small $\sigma_p^2$) ⇒ large $\lambda$ ⇒ heavy penalty ⇒ weights pulled hard to zero.
- Weak prior (large $\sigma_p^2$) ⇒ small $\lambda$ ⇒ MAP $\to$ MLE.

# Worked numeric · MAP with L2

Linear regression. Suppose at the MLE estimate, the NLL is $L_{\text{NLL}} = 0.50$. Weight vector $\boldsymbol{\theta} = [3, -4]$ — so $\|\boldsymbol{\theta}\|_2^2 = 9 + 16 = 25$.

Pick $\lambda = 0.01$ (corresponds to $\sigma_p^2 = 50$ — a fairly weak prior).

DERIVATION

$$L_{\text{MAP}} = 0.50 + 0.01 \cdot 25 = 0.50 + 0.25 = \mathbf{0.75}$$

The optimizer now pays a price for large weights. Since the gradient of $\|\theta\|^2$ is $2\theta$, every step **shrinks weights by a factor** $(1 - 2\eta\lambda)$ before the data update. This is exactly **weight decay** — and you'll see this exact form again in Adam vs AdamW (L5).

# Laplace prior · setup

Now choose a prior $p(\theta_j) = \text{Laplace}(0, b)$ — same idea (centred at zero) but **heavier tails and a sharper peak at zero**.

DERIVATION

The Laplace density ·

$$p(\theta_j) = \frac{1}{2b} \exp\left(-\frac{|\theta_j|}{b}\right)$$

For one weight ·

$$\log p(\theta_j) = -\frac{|\theta_j|}{b} + \text{const}$$

For the whole vector with independent components ·

$$\log p(\boldsymbol{\theta}) = -\frac{1}{b} \sum_j |\theta_j| + \text{const} = -\frac{1}{b} \|\boldsymbol{\theta}\|_1 + \text{const}$$

Note the **absolute value** in the log — that's the structural difference from Gaussian's squared term.

# Laplace prior · L1 pops out

Plug into MAP ·

$$L_{\text{MAP}}(\boldsymbol{\theta}) = L_{\text{NLL}}(\boldsymbol{\theta}) + \frac{1}{b}\|\boldsymbol{\theta}\|_1 + \text{const}$$

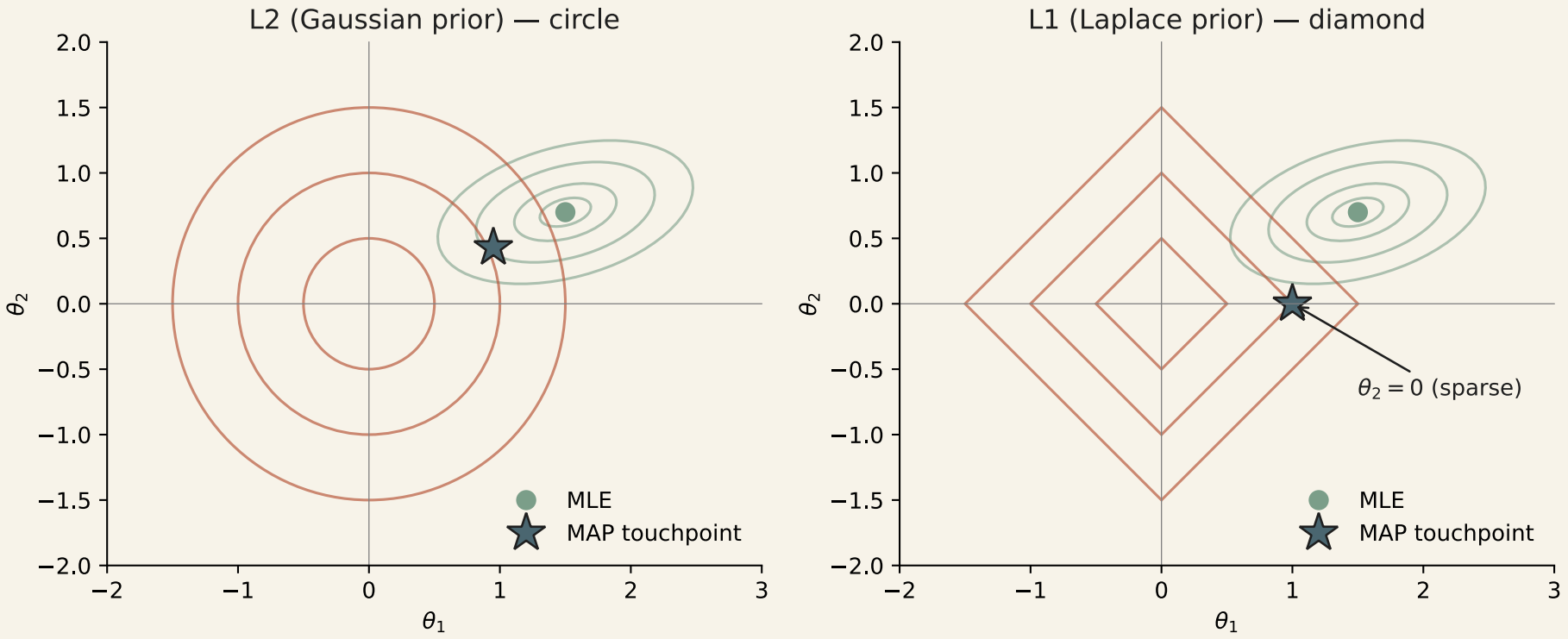
DERIVATION

$$\boxed{L_{\text{MAP}}(\boldsymbol{\theta}) = L_{\text{NLL}}(\boldsymbol{\theta}) + \lambda\,\|\boldsymbol{\theta}\|_1, \qquad \lambda := \frac{1}{b}}$$

**This is exactly L1 regularization (a.k.a. lasso).**

- Same machinery (MAP), different prior, different penalty.
- The Laplace density is "sharply peaked at zero with heavy tails" — and that geometry is what makes L1 produce **sparse solutions**.

# Why L1 produces sparse solutions · the geometry




DERIVATION

Think of MAP as minimizing the loss subject to the prior. In 2D ·

- **L2 ball** — circle. The data-loss contour can touch it anywhere on the circle. Generic touchpoints have *both* coordinates non-zero. Solutions are **shrunk but rarely zero**.
- **L1 ball** — diamond. Corners stick out **on the axes**. Generic data-loss contours hit the diamond *at a corner*,

# L1 vs L2 · summary table

| | L2 (RIDGE) | L1 (LASSO) |
|---|---|---|
| Prior | $\mathcal{N}(0, \sigma_p^2)$ | $\text{Laplace}(0, b)$ |
| Log-prior penalty | $\lambda \sum \theta_j^2$ | $\lambda \sum$ |
| Geometry | Circle | Diamond |
| Solution | small everywhere | many exactly zero |
| Gradient of penalty | $2\lambda\theta_j$ (smooth) | $\lambda \, \text{sign}(\theta_j)$ (kink at 0) |
| When to use | dense problems, "shrink everything" | feature selection, "kill weak features" |

KEY IDEA

L1 and L2 are **the same MAP machinery** — they differ only in the prior over $\theta$. Pick your prior, get your regularizer.

# A coin · MLE vs MAP

Back to the coin · 3 heads, 0 tails. MLE says $\hat{p} = 1.0$ (absurd).

Suppose your prior is $p \sim \text{Beta}(2, 2)$ — *"coin is probably near 0.5"*. MAP combines this with the likelihood ·

DERIVATION

Posterior ∝ Likelihood × Prior · $p^3(1-p)^0 \cdot p^{2-1}(1-p)^{2-1} = p^4(1-p)^1$

Maximize · $\log \text{post} = 4 \log p + \log(1-p)$. Set derivative · $4/p - 1/(1-p) = 0 \Rightarrow p = 4/5 = 0.8$.

MAP says $\hat{p} = 0.8$ — **sensible**. Three heads is *some* evidence the coin is biased, but the prior keeps us from going to 1.0.

This is the same regularization story as L1/L2 on weights — just with a different distribution.

PART 6

# Where this matters · the rest of the course

NLL is the loss everywhere · KL connects · VAE/diffusion/GANs are MAP++

# Every loss is an NLL — the master table

DERIVATION

| OUTPUT | DISTRIBUTION | LOSS = NLL | LECTURE |
|---|---|---|---|
| Real-valued | Normal | MSE | L1 (recap), L19 (VAE recon) |
| Binary | Bernoulli | BCE | L1 (recap) |
| K classes | Categorical | Cross-entropy | L7+ (vision), L13–15 (LLMs) |
| Pixels | per-pixel Normal | per-pixel MSE | L19 (VAE), L21 (diffusion) |
| Tokens | Categorical | next-token CE | L13–L15 (LLMs) |
| Image patch given noise | Normal in pixel/score space | MSE on noise | L21 (diffusion) |
| Latent variable model | Normal + KL prior | ELBO = recon + KL | L19 (VAE) |
| Two distributions match | KL minimization | DPO, distillation | L16, L23 |

The whole course will keep instantiating the same NLL recipe. Each new model just changes **which distribution** is being assumed.

# Information content · the surprise of a single outcome

Before defining entropy, define the **information content** (or "surprise") of one outcome ·

DERIVATION

$$I(y) := -\log p(y)$$

**Why this exact formula?** Three axioms we want $I$ to satisfy ·

1. $I$ depends **only on** $p(y)$ (not on $y$ itself).
2. $I$ is **decreasing** in $p$ — rare events are more informative.
3. $I$ is **additive** for independent events · $I(y_1, y_2) = I(y_1) + I(y_2)$.

   The **only** function satisfying all three is $I(y) = -c \log p(y)$ for $c > 0$ (Shannon 1948). With $c = 1$ and $\log = \log_2$, the unit is **bits**.

# Surprise · "snowing in Kashmir vs Gandhinagar"

Same headline · *"It snowed today."* Two locations ·

- **Kashmir in January** · $p(\text{snow}) \approx 0.5 \cdot I = -\log_2 0.5 = 1$ bit. Mild surprise.
- **Gandhinagar in January** · $p(\text{snow}) \approx 10^{-6} \cdot I = -\log_2 10^{-6} \approx 20$ bits. Front-page news.

KEY IDEA

The **same event** carries different surprise depending on the distribution generating it. That is *exactly* what $I(y) = -\log p(y)$ formalizes.

This is also why log-likelihood works as a model-quality signal · a model that places high probability on the data has *low* per-sample surprise.

# Information content · worked examples

DERIVATION

| EVENT | $p$ | $I = -\log_2 p$ | INTERPRETATION |
|---|---|---|---|
| Fair coin lands heads | $0.5$ | $1$ bit | one yes/no question |
| Roll a 6 on a fair die | $1/6$ | $\approx 2.58$ bits | "less than 3 yes/no questions worth" |
| Win a 1-in-1024 lottery | $1/1024$ | $10$ bits | extremely surprising |
| The sun rises tomorrow | $\approx 1$ | $\approx 0$ bits | no information (already certain) |
| Sample a specific token from 50k vocab | $1/50000$ | $\approx 15.6$ bits | one token = one short word in English |

**Reading** · large $I(y)$ = "I learned a lot from observing $y$." Small $I(y)$ = "I expected this, no information gained."

This is **the per-sample log-loss** in disguise. Cross-entropy is just the *average* of these surprises.

# Entropy · the definition

The **entropy** of a distribution $p$ is the **expected information** of a draw from $p$ ·

DERIVATION

$$H(p) := \mathbb{E}_{Y\sim p}[I(Y)] = -\mathbb{E}_{Y\sim p}[\log p(Y)] = -\sum_{y} p(y)\ \log p(y)$$

In base 2, $H$ is in **bits** · *the average number of bits the optimal code spends per sample from* $p$.

So entropy is what you get when you average the per-sample surprise from the previous slides. Big entropy = on average, draws from $p$ are surprising. Small entropy = mostly predictable.

# Entropy as code length · the Huffman lens

Imagine you must send samples from $p$ over a wire using a **prefix-free binary code**. Shannon's source-coding theorem says ·

DERIVATION

$$\underbrace{H(p)}_{\text{lower bound}} \;\leq\; \mathbb{E}_p[\text{bits per symbol}] \;\leq\; H(p)+1$$

**Concrete** · alphabet $\{A, B, C, D\}$ with $p = (0.5, 0.25, 0.125, 0.125)$ ·

| SYMBOL | $p$ | OPTIMAL CODE | CODE LENGTH |
|---|---|---|---|
| A | $0.5$ | `0` | 1 |
| B | $0.25$ | `10` | 2 |
| C | $0.125$ | `110` | 3 |
| D | $0.125$ | `111` | 3 |

Average length $= 0.5(1) + 0.25(2) + 0.125(3) + 0.125(3) = 1.75$ bits $= H(p)$ exactly.

**Entropy = the cost of describing samples from $p$ optimally.** That's the right physical interpretation.

# Entropy · worked numerics in bits

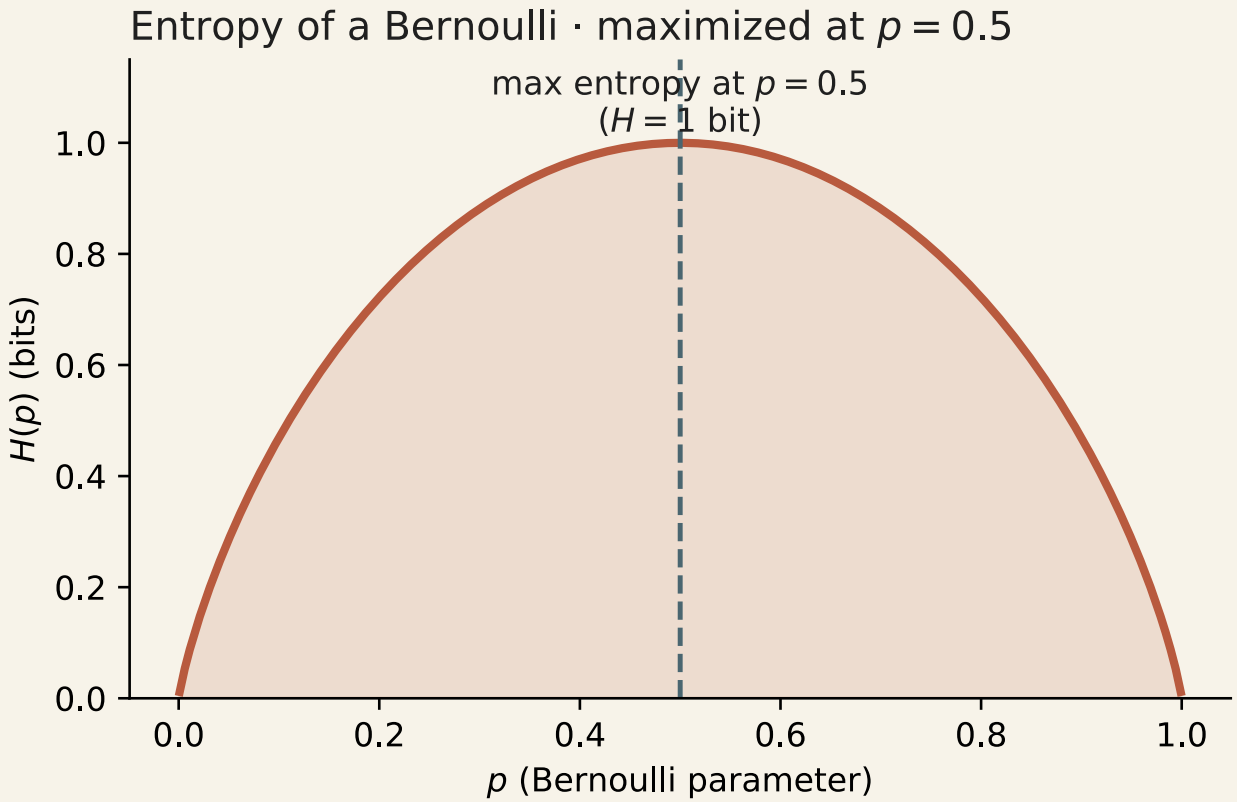
| DISTRIBUTION | $H$ (BITS) | COMPUTATION |
|---|---|---|
| Fair coin | $1$ | $-2 \cdot 0.5 \log_2 0.5 = 1$ |
| Biased coin $p = 0.99$ | $0.08$ | $-0.99 \log_2 0.99 - 0.01 \log_2 0.01$ |
| Uniform over 8 | $3$ | $-8 \cdot \frac{1}{8} \log_2 \frac{1}{8} = \log_2 8$ |
| One-hot (deterministic) | $0$ | nothing to encode |

KEY IDEA

**Entropy peaks for uniform** distributions and is **zero** for deterministic ones. A fair coin needs 1 bit per flip; a near-deterministic coin needs ~0 bits per flip; a uniform-over-K distribution needs $\log_2 K$ bits.

This explains why low-entropy classifier outputs (confident) need fewer bits to encode than high-entropy outputs (uncertain) — and why temperature in LLMs (which controls output entropy) controls "creativity vs determinism."

# Entropy of a Bernoulli · in pictures




DERIVATION

$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$ — concave, symmetric around $p = 0.5$, peaking at $1$ bit.

A *fair* coin needs **one bit** to encode each flip. A coin with $p = 0.99$ is almost deterministic — encoded with arithmetic coding it costs ~$0.08$ bits/flip on average.

# Entropy worked · Categorical (3-class)

Let $\boldsymbol{\pi}$ vary over a 3-class Categorical and compute $H$ in bits ·

DERIVATION

| DISTRIBUTION $\boldsymbol{\pi}$ | $H(\boldsymbol{\pi})$ (BITS) | INTERPRETATION |
|---|---|---|
| $(1, 0, 0)$ — deterministic | $0$ | already certain |
| $(0.7, 0.2, 0.1)$ — peaked | $-0.7 \log_2 0.7 - 0.2 \log_2 0.2 - 0.1 \log_2 0.1 \approx 1.16$ | confident model |
| $(0.5, 0.3, 0.2)$ — moderate | $\approx 1.49$ | mixed evidence |
| $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ — uniform | $\log_2 3 \approx 1.58$ | maximum |

The maximum entropy of a $K$-Categorical is $\log_2 K$, achieved by the uniform. **A confident classifier has low-entropy predictions; a confused one has high-entropy predictions.** This connects directly to the **temperature** in LLM sampling — high $T$ pushes the categorical toward uniform (high entropy), low $T$ sharpens it (low entropy).

# Entropy in the continuous case · differential entropy

For a continuous random variable, replace the sum with an integral ·

DERIVATION

$$h(p) := -\int p(y)\,\log p(y)\,dy$$

(Lower-case $h$ by convention to distinguish from the discrete case. Differential entropy can be **negative** — it's not a "bits per sample" quantity in the same sense.)

For a Normal $\mathcal{N}(\mu, \sigma^2)$ ·

$$h\big(\mathcal{N}(\mu,\sigma^2)\big) = \tfrac{1}{2}\log(2\pi e\,\sigma^2)$$

**Reading** · the differential entropy of a Gaussian only depends on $\sigma^2$ (not on $\mu$). It is **maximum** among all continuous distributions on $\mathbb{R}$ with that fixed variance — recovering the max-entropy reason "Why Normal" from earlier.

**Worked** · $\mathcal{N}(0,1) \to h = \tfrac{1}{2}\log(2\pi e) \approx 1.42$ nats $\approx 2.05$ bits.

# Mutual information · briefly (will return in L17)

For two random variables $X, Y$ with joint $p(x, y)$ ·

DERIVATION

$$I(X;Y) := \mathrm{KL}\big(p(x,y) \,\|\, p(x)\,p(y)\big) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

**Read** · "how much $X$ and $Y$ tell us about each other." Always $\geq 0$, equal to 0 iff $X \perp Y$.

**Why we'll care** ·

- **Self-supervised learning (L17)** · contrastive losses are *lower bounds on mutual information* between augmented views (InfoNCE).
- **Information bottleneck** · representation learning as $I(X;Z)$ minimization subject to $I(Z;Y) \geq \text{target}$.
- **Capacity of a channel** · classical result with the same formula.
  We won't compute $I(X;Y)$ directly today, but you'll see it surface in L17.

# KL divergence · the definition

KL divergence measures how different one distribution $p$ is from another distribution $q$ over the same support.

DERIVATION

$$\mathrm{KL}(p \,\|\, q) := \mathbb{E}_{Y\sim p}\left[\log \frac{p(Y)}{q(Y)}\right] = \sum_{y} p(y)\, \log \frac{p(y)}{q(y)}$$

(Replace $\sum$ with $\int$ for continuous distributions.)

**Read** · *"the average log-ratio when the data really comes from $p$ but you used $q$ to model it."*

It's the **average extra surprise** you experience by encoding samples-from-$p$ using a code optimized for $q$.

# KL · three properties

DERIVATION

1. $\mathrm{KL}(p\|q) \geq 0$ for all $p, q$. (Gibbs' inequality — proof in two slides.)
2. $\mathrm{KL}(p\|q) = 0$ **if and only if** $p = q$ everywhere.
3. **Asymmetric** · in general $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$.

Property 1 is what makes KL a sensible *loss-like* quantity — minimize it and you get to zero only when distributions match.

Property 2 says KL = 0 is the unique minimum. So **minimizing KL is the right objective** for matching distributions.

Property 3 is the surprising one. KL is not a metric — *which direction* you take matters. We'll see this matters a lot in "forward vs reverse KL" later.

# KL worked · two Bernoullis

Let $p = (0.7, 0.3)$ (true) and $q = (0.5, 0.5)$ (model).

DERIVATION

$$\mathrm{KL}(p \,\|\, q) = \sum_y p(y) \log \frac{p(y)}{q(y)} = 0.7 \log \frac{0.7}{0.5} + 0.3 \log \frac{0.3}{0.5}$$

In nats (natural log) · $0.7 \cdot 0.336 + 0.3 \cdot (-0.511) = 0.235 - 0.153 = \mathbf{0.082}$ nats.

In bits · divide by $\ln 2 \approx 0.693$, giving $\approx 0.119$ bits.

**Asymmetry check** · $\mathrm{KL}(q \,\|\, p) = 0.5 \log \frac{0.5}{0.7} + 0.5 \log \frac{0.5}{0.3} = -0.168 + 0.255 = 0.087$ nats — close but **different**. KL is not a metric.

# KL worked · two Categoricals (3-class)

True $p = (0.5, 0.3, 0.2)$. Two candidate models ·

DERIVATION

**Model A** · $q_A = (0.4, 0.4, 0.2)$ — close to truth.

$$\mathrm{KL}(p \,\|\, q_A) = 0.5 \log \tfrac{0.5}{0.4} + 0.3 \log \tfrac{0.3}{0.4} + 0.2 \log \tfrac{0.2}{0.2}$$
$$= 0.5 \cdot 0.223 + 0.3 \cdot (-0.288) + 0 = 0.112 - 0.086 = \mathbf{0.026} \text{ nats.}$$

**Model B** · $q_B = (0.1, 0.1, 0.8)$ — very wrong on class 3.

$$\mathrm{KL}(p \,\|\, q_B) = 0.5 \log \tfrac{0.5}{0.1} + 0.3 \log \tfrac{0.3}{0.1} + 0.2 \log \tfrac{0.2}{0.8}$$
$$= 0.5 \cdot 1.609 + 0.3 \cdot 1.099 + 0.2 \cdot (-1.386)$$
$$= 0.805 + 0.330 - 0.277 = \mathbf{0.858} \text{ nats.}$$

A roughly-aligned model has KL $\approx 0.03$; a wildly mismatched one has KL $\approx 0.86$. **KL ≈ 0 ⇒ the two distributions agree** ; large KL ⇒ they disagree, especially in directions where $p$ has mass but $q$ doesn't (mode-covering penalty).

# KL as wasted code length · the Huffman lens

Same alphabet $\{A, B, C, D\}$. **True** $p = (0.5, 0.25, 0.125, 0.125)$ — Huffman code lengths $(1, 2, 3, 3)$ · optimal cost $H(p) = 1.75$ bits.

Suppose we mistakenly built our code for **wrong** $q = (0.125, 0.125, 0.25, 0.5)$ — Huffman lengths $(3, 3, 2, 1)$.

When we encode samples from $p$ with the code for $q$ ·

$$H(p, q) = 0.5(3) + 0.25(3) + 0.125(2) + 0.125(1) = \mathbf{2.625} \text{ bits}$$

DERIVATION

$$\mathrm{KL}(p \,\|\, q) \;=\; H(p, q) - H(p) \;=\; 2.625 - 1.75 \;=\; \mathbf{0.875} \text{ bits/symbol of waste}$$

**Reading** · KL is the literal *number of extra bits per sample* you pay for using the wrong code. Modelling = compression · this is why "**better model = better compression**" is not a metaphor.

# Why KL ≥ 0 · Jensen's inequality (proof)

**Jensen's inequality** · for any *concave* function $\varphi$ and random variable $X$,

$$\mathbb{E}[\varphi(X)] \leq \varphi(\mathbb{E}[X])$$

Since $\log$ is concave ·

DERIVATION

$$-\mathrm{KL}(p\|q) = \mathbb{E}_p\left[\log \tfrac{q(Y)}{p(Y)}\right] \;\leq\; \log \mathbb{E}_p\left[\tfrac{q(Y)}{p(Y)}\right] = \log \sum_y p(y)\, \tfrac{q(y)}{p(y)} = \log \sum_y q(y) = \log 1 = 0$$

So $-\mathrm{KL} \leq 0$, i.e. $\mathrm{KL}(p\|q) \geq 0$. Equality iff $q(y)/p(y)$ is constant — i.e. $p = q$. ∎

This is the full proof of **Gibbs' inequality**. Jensen's inequality is the same tool used to derive the **ELBO** in VAEs (L19). Same trick, same place — once you see it once, you see it everywhere.

# Cross-entropy = entropy + KL

Algebra ·

$$H(p, q) := -\mathbb{E}_p[\log q(Y)] = -\mathbb{E}_p[\log p(Y)] + \mathbb{E}_p\left[\log \frac{p(Y)}{q(Y)}\right] = H(p) + \mathrm{KL}(p \| q)$$

DERIVATION

$$\boxed{H(p, q) \;=\; H(p) \;+\; \mathrm{KL}(p \,\|\, q)}$$

- **Cross-entropy** $H(p, q)$ · "expected bits to encode samples from $p$ if we used a code optimized for $q$."
- **Entropy** $H(p)$ · "the irreducible cost of encoding $p$." Independent of $q$.
- **KL** · the *extra* bits we waste because $q \neq p$.

  For classification with a **one-hot true label** $p$, $H(p) = 0$ — so cross-entropy *is* KL. That's why we say "the classifier loss minimizes KL to the true label."

# Cross-entropy worked · the one-hot collapse

In classification, the truth $\mathbf{y}$ is **one-hot** and the model output $\hat{\boldsymbol{\pi}}$ is a softmax.

DERIVATION

For any one-hot, $H(\mathbf{y}) = 0$ — there's no uncertainty in the truth. So the cross-entropy formula collapses ·

$$H(\mathbf{y}, \hat{\boldsymbol{\pi}}) = \underbrace{H(\mathbf{y})}_{=0} + \mathrm{KL}(\mathbf{y} \,\|\, \hat{\boldsymbol{\pi}}) = -\log \hat{\pi}_{\,y_{\text{true}}}$$

**Cross-entropy of a one-hot truth against a softmax model is exactly the NLL of the model on the true class.** This is the standard classification loss.

It's also exactly what L1 derived from a Bernoulli/Categorical assumption — same answer through two different lenses (NLL or KL). On the next slide we tabulate it across confidence levels.

# Cross-entropy · table across confidence levels

3 classes, true class is 1 (so $\mathbf{y} = (1, 0, 0)$). Vary the model output and read off the loss ·

DERIVATION

| MODEL $\hat{\boldsymbol{\pi}}$ | $-\log \hat{\pi}_1$ (NATS) | "SENTIMENT" |
|---|---|---|
| $(0.99, 0.005, 0.005)$ | $0.010$ | confidently right · tiny loss |
| $(0.7, 0.2, 0.1)$ | $0.357$ | mostly right |
| $(0.34, 0.33, 0.33)$ | $1.079$ | uncertain (≈ uniform) |
| $(0.05, 0.5, 0.45)$ | $2.996$ | confidently wrong · big loss |

The loss is small **iff the model assigned high probability to the true class** and grows steeply when the model is *confidently wrong*. This asymmetric penalty is what makes cross-entropy a **proper scoring rule** — it rewards calibration, not just accuracy.

The classifier's training loss is just the average of this column over the dataset.

# MLE through the KL lens · setup

Define the **empirical data distribution** as the histogram of the training set, treated as a distribution ·

DERIVATION

$$\hat{p}_{\text{data}}(y) := \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[Y_i = y]$$

This is just a discrete probability mass with weight $1/N$ on each observed point.

Now the average log-likelihood becomes an **expectation under** $\hat{p}_{\text{data}}$ ·

$$\frac{1}{N}\,\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(y_i) = \mathbb{E}_{Y \sim \hat{p}_{\text{data}}}[\log p_\theta(Y)]$$

So the score we already maximize is the *negative cross-entropy* between the empirical distribution and the model.

# MLE through the KL lens · result

Use $H(p, q) = H(p) + \mathrm{KL}(p \| q)$ and drop $H(\hat{p}_{\text{data}})$ as constant in $\theta$ ·

DERIVATION

$$\arg\max_{\theta} \ell(\theta) \;=\; \arg\min_{\theta} H(\hat{p}_{\text{data}}, p_\theta) \;=\; \arg\min_{\theta} \mathrm{KL}(\hat{p}_{\text{data}} \,\|\, p_\theta)$$

**MLE = make the model distribution as KL-close as possible to the empirical distribution.**

Same machinery, two-distribution view. Every classifier you've trained with cross-entropy has been silently minimizing this KL — to the one-hot empirical distribution of class labels.

This is also why MLE is **mode-covering** (forward KL) — covered in the "forward vs reverse KL" slide.

# MAP through the KL lens

Adding the log-prior to the previous slide ·

DERIVATION

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta} \Big[ \underbrace{\text{KL}(\hat{p}_{\text{data}} \,\|\, p_\theta)}_{\text{fit}} \; - \; \underbrace{\tfrac{1}{N} \log p(\theta)}_{\text{prior penalty}} \Big]$$

**L2 regularization** through this lens · "minimize KL-to-empirical, plus pay $\lambda \|\theta\|^2$ for straying from a Gaussian prior."

This is the **fit + don't drift in KL** template that recurs throughout the course.

# KL regularization · everywhere in DL

The same "fit + don't drift in KL" pattern appears in many places ·

DERIVATION

| METHOD | LOSS STRUCTURE | READING |
|---|---|---|
| **L2 regularization** | NLL $+\lambda\|\theta\|^2$ | don't drift from prior |
| **VAE (L19)** | reconstruction $+\mathrm{KL}(q_\phi(z\|x)\|p(z))$ | encoder posterior close to standard normal |
| **DPO / RLHF (L16)** | reward $-\beta\,\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})$ | policy close to base model |
| **Distillation (L23)** | $\mathrm{KL}(p_{\text{teacher}}\|p_{\text{student}})$ | student matches teacher |

Every regularizer in modern DL is some form of **"don't drift too far in KL"** from a reference distribution. Once you see the pattern, you stop memorizing losses and start *deriving* them.

# KL is asymmetric · two directions, two problems

We've already seen that $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$. So *which direction* you minimize is itself a modelling choice — and the two directions optimize for very different things.

KEY IDEA

- **Forward KL** · $\mathrm{KL}(p \,\|\, q_\theta)$ — "average over the truth $p$."
- **Reverse KL** · $\mathrm{KL}(q_\theta \,\|\, p)$ — "average over the model $q_\theta$."

Both are valid. Both are used in DL. Knowing which one your loss optimizes tells you what failure mode to expect. Next two slides unpack each.

# Forward KL · mode-covering

DERIVATION

$$\mathrm{KL}(p \,\|\, q_\theta) = \sum_y p(y) \log \frac{p(y)}{q_\theta(y)}$$

Read · "average the log-ratio over the **truth** $p$."

**Failure mode** · the expectation is taken over $p$, so a region where $p(y) > 0$ but $q_\theta(y) \approx 0$ contributes a *huge* penalty (log of a tiny number).

**Consequence · mode-covering.** $q_\theta$ is *forced* to put mass everywhere $p$ does. If $p$ has two modes, $q_\theta$ has to span both. Single-Gaussian fit to a bimodal target → blurry middle.

**This is what MLE optimizes** — recall MLE = $\arg\min_\theta \mathrm{KL}(\hat{p}_{\text{data}} \| p_\theta)$. So MLE-trained models are *mode-covering* by construction.

# Reverse KL · mode-seeking

DERIVATION

$$\mathrm{KL}(q_\theta \,\|\, p) = \sum_y q_\theta(y) \log \frac{q_\theta(y)}{p(y)}$$

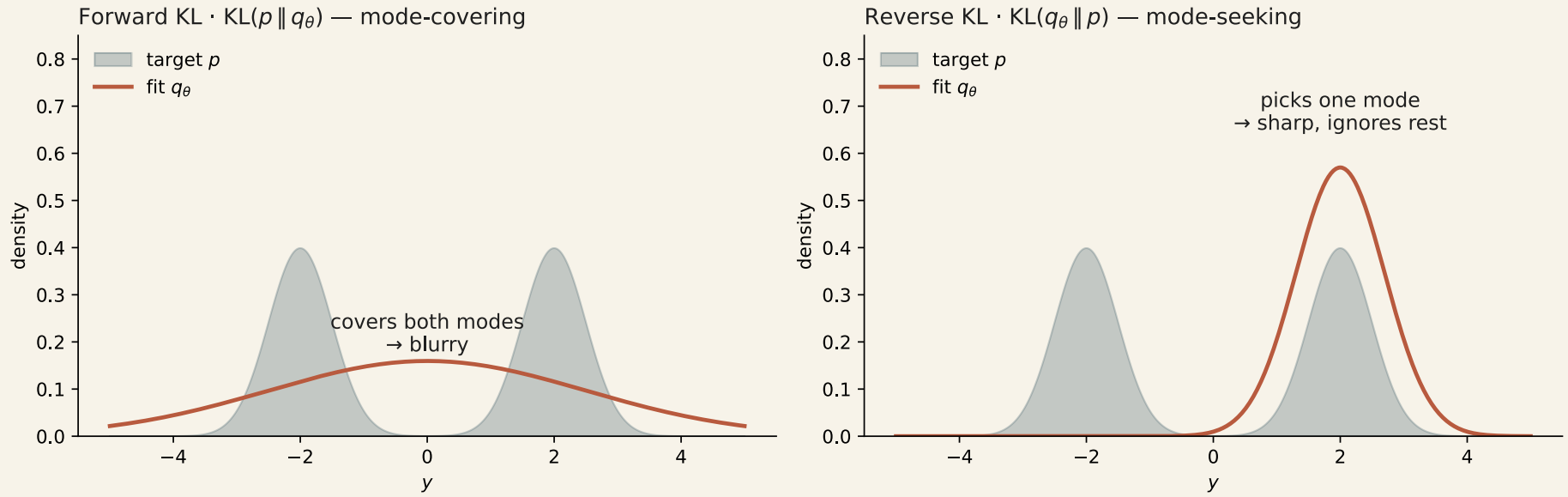
Read · "average the log-ratio over the **model** $q_\theta$."

**Failure mode** · the expectation is taken over $q_\theta$, so a region where $q_\theta(y) > 0$ but $p(y) \approx 0$ contributes a huge penalty. The safest move for $q_\theta$ is to put mass *only* where $p$ is large.

**Consequence · mode-seeking.** $q_\theta$ concentrates on a single high-density region. Two-mode target → $q_\theta$ picks one mode and ignores the other.

**This is approximately what VAE encoders, GAN training, and policy distillation optimize.** Sharper samples, but mode collapse is a real risk.

# Forward vs reverse KL · the picture




DERIVATION

Same bimodal target $p$ (grey shaded). Fit a single Gaussian by minimizing each direction of KL ·

- **Forward KL** spreads the Gaussian across both modes — covers all data, but spends mass on the gap between modes (blurry samples). Why **VAE samples are blurry**.
- **Reverse KL** concentrates on one mode — sharp samples but ignores the other half of the distribution. Why **GANs mode-collapse**.

The two errors are *opposite failure modes*. Knowing which KL direction your loss optimizes tells you which failure mode to expect.

# KL between two Gaussians · the closed form

For $p = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q = \mathcal{N}(\mu_2, \sigma_2^2)$, the KL is in closed form ·

DERIVATION

$$\mathrm{KL}\big(\mathcal{N}(\mu_1, \sigma_1^2) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2)\big) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

For the special case $q = \mathcal{N}(0, 1)$ ·

$$\mathrm{KL}\big(\mathcal{N}(\mu, \sigma^2) \,\|\, \mathcal{N}(0, 1)\big) = \tfrac{1}{2}\big(\mu^2 + \sigma^2 - 1 - \log \sigma^2\big)$$

This is the **exact term that shows up in the VAE loss** (L19) — pulling the encoder's posterior toward a standard-normal prior. Memorize this form; it'll save you 5 minutes of staring when you re-derive it later.

**Worked** · $\mu = 1, \sigma = 0.5 \Rightarrow \mathrm{KL} = \tfrac{1}{2}(1 + 0.25 - 1 + \log 4) \approx 0.82.$

# KL across the rest of the course

Once you see KL as the underlying object, every advanced model becomes a **specific KL minimization** ·

DERIVATION

| MODEL | KL IT MINIMIZES | LECTURE |
|---|---|---|
| Classifier (CE loss) | $\mathrm{KL}(\hat{p}_{\text{data}} \| p_\theta)$ | L1, every classifier |
| **VAE** | recon-NLL $+\mathrm{KL}(q_\phi(z\|x) \,\|\, p(z))$ | L19 |
| **Diffusion (variational view)** | $\sum_t \mathrm{KL}(q(x_{t-1}\|x_t, x_0) \,\|\, p_\theta(x_{t-1}\|x_t))$ | L21 |
| **GAN** | (approximately) Jensen-Shannon — symmetrized KL | L20 |
| **DPO / RLHF KL term** | $\mathrm{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$ | L16 |
| **Knowledge distillation** | $\mathrm{KL}(p_{\text{teacher}} \,\|\, p_{\text{student}})$ | L23 |

You will see KL terms in every generative or alignment loss for the next 24 lectures. Each is an instance of *what we just derived*.

# Session 2 · practice problems

Try these on paper; the notebook has plotting and verification.

DERIVATION

**P6.** For linear regression with NLL $0.4$ and weights $\boldsymbol{\theta} = [-2, 3, 1]$, compute the MAP loss with (a) L2 penalty $\lambda = 0.5$, (b) L1 penalty $\lambda = 0.5$. Which weights does L1 push hardest?

**P7.** Compute $\mathrm{KL}(\mathcal{N}(2, 0.25) \,\|\, \mathcal{N}(0, 1))$ using the closed form. Interpret the answer.

**P8.** Show that for one-hot true label $\mathbf{y}_i$ and softmax prediction $\hat{\boldsymbol{\pi}}_i$, cross-entropy equals $\mathrm{KL}(\mathbf{y}_i \,\|\, \hat{\boldsymbol{\pi}}_i)$. (Hint · entropy of a one-hot is zero.)

**P9.** A bimodal target has modes at $\pm 2$ with equal mass. You fit a single Gaussian $q$. Sketch the optimum under (a) forward KL, (b) reverse KL. Which one is mode-covering?

**P10.** Coin flipped 10 times, observed 8 heads. Starting from $\mathrm{Beta}(2, 2)$ prior, write down the Beta posterior and its mean. How does the posterior mean compare to the MLE $\hat{p} = 0.8$? Why does it differ?

# Foreshadow · how this lecture powers DL

Every advanced model in this course uses MLE (or MAP) under a specific distribution ·

DERIVATION

| MODEL | OUTPUT DISTRIBUTION | LOSS = NLL OF … | LECTURE |
|---|---|---|---|
| LLM (next-token) | Categorical over vocab | $-\log p_\theta(y_{t+1} \mid y_{1:t})$ | L13–L15 |
| VAE | Normal pixel decoder + Gaussian latent | reconstruction NLL + KL to prior | L19 |
| GAN | implicit generator | min–max over $\log D, \log(1-D)$ | L20 |
| Diffusion | Gaussian forward process | MSE on predicted noise | L21–L22 |
| RLHF / DPO | Bradley–Terry preference pair | log-sigmoid of reward gap | L16 |

You now know what *all* of these losses are. They're NLLs.

# Notebook teaser · MLE & MAP in PyTorch

We will pair this lecture with a notebook (`lec00-mle-map.ipynb`) that walks through ·

DERIVATION

1. **MLE for a coin** with `torch.distributions.Bernoulli`.
2. **MLE for linear regression** with `torch.distributions.Normal` — recover OLS.
3. **MLE for logistic regression** with `BCEWithLogitsLoss` — show it equals NLL of Bernoulli.
4. **MAP for linear regression with Gaussian prior** — recover ridge regression.
5. **MAP for linear regression with Laplace prior** — recover lasso, see sparsity emerge as $\lambda$ grows.
6. **Visualize** likelihood and posterior surfaces in 2D for a tiny example.

Same code skeleton, three lines change between MLE and MAP. That's the punchline.

# Common questions · FAQ

**Q.** Do we always have to pick a distribution before training?
**A.** Yes — implicitly or explicitly. When you write MSE, you have *implicitly* assumed Gaussian noise. When you write BCE, Bernoulli. The conscious choice is what we're advocating today.

**Q.** Why minimize NLL instead of maximize LL?
**A.** Pure convention. ML libraries minimize. Negate, minimize, same answer.

**Q.** What if my output isn't Bernoulli/Categorical/Gaussian?
**A.** Pick whatever distribution fits — Poisson for counts, Beta for probabilities, mixture-of-Gaussians for multimodal data. The recipe (NLL = loss) is universal.

**Q.** How does this connect to Bayesian deep learning?
**A.** Bayesian DL keeps the *full posterior* over $\theta$ instead of a single point estimate. We won't go there in this course, but everything we do today is the entry point.

# *Summary* · Lecture 0 — summary

We made the probabilistic framework concrete ·

- **Conditional view** · the model outputs a distribution $p(y \mid \mathbf{x}, \theta)$, not a number.
- **Bayes' rule for parameters** · $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta)$ — four named terms.
- **MLE** · ignore the prior, maximize log-likelihood.
  - MSE = MLE under Gaussian noise.
  - BCE / CE = MLE under Bernoulli / Categorical outputs.
- **MAP** · MLE + a prior on $\theta$, log-prior becomes a regularizer.
  - L2 = MAP with Gaussian prior on weights.
  - L1 = MAP with Laplace prior — sparsity from diamond geometry.
- **NLL is the loss everywhere.** Every advanced model in this course will be an instance of this recipe.

### Read before Lecture 1

Strang Ch 1 (vectors / matrices) · Bishop & Bishop §2.1–2.3 (probability primer) · §4.1–4.3 (linear regression as MLE) · §5.4 (logistic regression).

### Next lecture

**Why deep learning** — why these tools alone aren't enough at scale, and what depth and non-linearity buy us.