Information Theory I: Entropy, Cross-Entropy, KL

Surprisal · Source Coding · Cross-Entropy · KL Divergence

What Is Information Theory?

Information Theory is a field of study built on **probability theory**.

Its foundation was laid by **Claude Shannon** in **1948**
("A Mathematical Theory of Communication").

Since then, applied across many fields:

communication theory · computer science · optimization · cryptography · machine learning · statistical inference

The core goal: quantify the *amount of information* gained — equivalently, the *uncertainty reduced* — when we observe a random variable.

Fun fact: Anthropic's **Claude** is named after Claude Shannon.

Main source: `https://slds-lmu.github.io/i2ml/chapters/13_information_theory/`

Three Sentences, Three Amounts of Information

"The sun rose today."

"It will rain in Yerevan tomorrow."

"Armenia just won the World Cup."

All three are sentences of similar length. Why do they carry such different amounts of **information**?

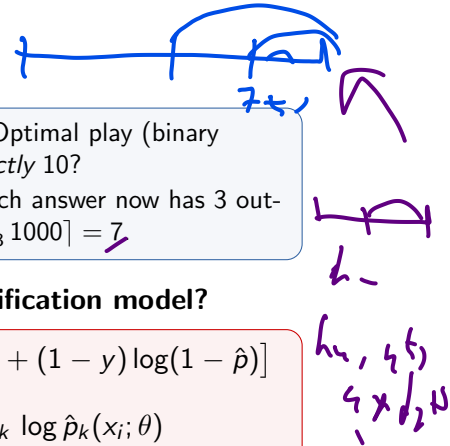
Today's goal: turn this everyday intuition into a number.

Two Concrete Problems We'll Solve

(A) Guess my number between 1 and 1000.

Each yes/no question can at best *halve* the range. Optimal play (binary search) needs $\lceil \log_2 1000 \rceil = 10$ questions. Why *exactly* 10?

What if answers can be yes / no / not sure? Each answer now has 3 outcomes ($\log_2 3 \approx 1.58$ bits) — so you need only $\lceil \log_3 1000 \rceil = 7$



(B) Why this loss function shows up in every classification model?

Binary (you'll recognize this): $L = -\big[y \log \hat{p} + (1-y) \log(1-\hat{p})\big]$

Multiclass: $\mathcal{L}(\theta) = -\sum_{i=1}^{n} \sum_{k} y_{ik} \log \hat{p}_k(x_i; \theta)$

Logistic regression, softmax, neural nets, LLMs

— all minimize this. **info_02** explains why.

Both problems are solved by one quantity: **entropy**.

Roadmap

I. Entropy
Uncertainty in one variable.
Surprisal, $H(X)$.

II. Source Coding
Entropy = the compression limit.
Prefix codes, Shannon's theorem.

III. Cross-Entropy
Cost of using the wrong code.
$H(p\|q)$.

IV. KL Divergence
Extra bits = how far q is from p.
$H(p\|q) = H(p) + D_{\mathrm{KL}}$.

Today: Entropy → Source Coding → Cross-Entropy → KL Divergence.

Entropy

How much **surprise** does a random variable carry?

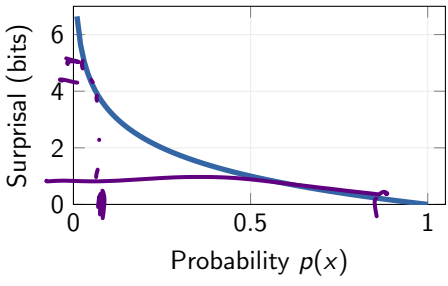
Can we quantify **uncertainty**?

Surprisal: Rare Events Are More Informative

Observing a **rare** event is more "surprising" than observing a common one.

Surprisal (self-information) of outcome x:

$$I(x) = -\log_2 p(x) = \log_2 \frac{1}{p(x)} \qquad \text{(measured in \textbf{bits})}$$



Why this formula?

- **Minus?** $\log_2 p \le 0$ for $p \le 1$, so $-\log_2 p \ge 0$. Equivalently $\log_2(1/p)$.
- **Log?** We want additivity: $I(x, y) = I(x) + I(y)$ for independent events. Only log satisfies $f(pq) = f(p) + f(q)$.
- **Base 2?** Unit choice. $\log_2 \Rightarrow$ **bits**; $\ln \Rightarrow$ nats; $\log_{10} \Rightarrow$ hartleys.

Shannon Entropy = Expected Surprisal

Average the surprisal $-\log_2 p(x)$ over all outcomes:

Shannon Entropy:

$$H(X) \quad := \quad \mathbb{E}[-\log_2 p(X)] \quad = \quad -\sum_x p(x)\,\log_2 p(x)$$

Convention: $0 \cdot \log_2 0 = 0$. **Why "H"?** Shannon borrowed it from Boltzmann's *H-theorem* in physics, where $H = -\sum f \log f$ measures disorder.

Build intuition from the simplest cases:

- **Fair coin** (2 equally likely): $H = -2 \cdot \frac{1}{2}\log_2 \frac{1}{2} = \mathbf{1}$ bit — exactly one yes/no question.
- **Fair die** (6 equally likely): $H = \log_2 6 \approx \mathbf{2.58}$ bits.
- **Loaded 4 outcomes** $p = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$: *predict first — more or less than* $\log_2 4 = 2$*?*

Shannon Entropy = Expected Surprisal

Average the surprisal $-\log_2 p(x)$ over all outcomes:

Shannon Entropy:

$$H(X) \quad := \quad \mathbb{E}[-\log_2 p(X)] \quad = \quad -\sum_x p(x)\,\log_2 p(x)$$

Convention: $0 \cdot \log_2 0 = 0$. **Why "H"?** Shannon borrowed it from Boltzmann's *H-theorem* in physics, where $H = -\sum f \log f$ measures disorder.

Build intuition from the simplest cases:

- **Fair coin** (2 equally likely): $H = -2 \cdot \frac{1}{2}\log_2 \frac{1}{2} = \mathbf{1}$ bit — exactly one yes/no question.
- **Fair die** (6 equally likely): $H = \log_2 6 \approx \mathbf{2.58}$ bits.
- **Loaded 4 outcomes** $p = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$: *predict first — more or less than* $\log_2 4 = 2$*?*

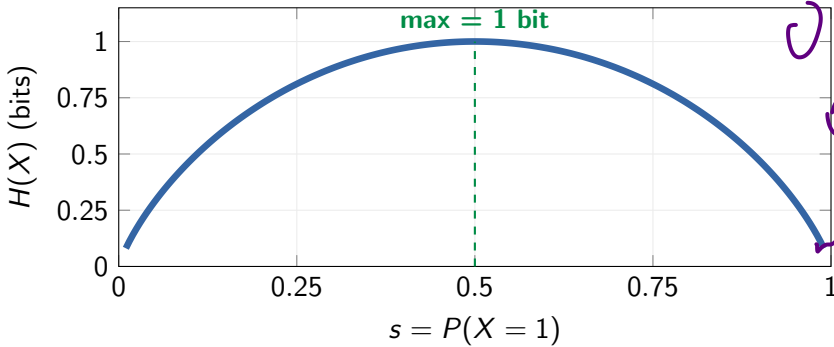
$$H = \tfrac{1}{2}\cdot 1 + \tfrac{1}{4}\cdot 2 + \tfrac{1}{8}\cdot 3 + \tfrac{1}{8}\cdot 3 = \boxed{\tfrac{7}{4} = 1.75 \text{ bits}} \; < \; 2.$$

Uniform maximizes entropy ($H = \log_2 g$); a sure outcome gives $H = 0$. Unequal odds always lower H.

Entropy of the Bernoulli Distribution

For $X \sim \text{Bern}(s)$: $P(X{=}1) = s$, $P(X{=}0) = 1 - s$.

$$H(X) = -s \log_2 s - (1-s) \log_2 (1-s)$$



$H(0) = H(1) = 0$ (deterministic). $H(0.5) = 1$ bit (fair coin: maximum uncertainty).

Properties of Entropy

#	Property	Formula
1	**Non-negative**	$H(X) \geq 0$
2	**Zero for deterministic**	$H(X) = 0$ iff one $p(x) = 1$
3	**Continuous** in probabilities	small $\Delta p \Rightarrow$ small ΔH
4	**Symmetric** in p values	relabeling outcomes doesn't change H
5	**Additive** for independent RVs	$H(X, Y) = H(X) + H(Y)$
6	**Maximal for uniform**	$H(X) \leq \log_2 g$ $(g = \#\text{outcomes})$

Uniqueness (Khinchin, 1957): Shannon entropy is the **only** function (up to a positive multiplicative constant) satisfying continuity, maximality for uniform, and a grouping axiom.

That constant = the choice of log base ($\log_2 \to$ bits, $\ln \to$ nats).

Entropy Is Maximal for Uniform Distributions

Claim: For any distribution on g outcomes, $H(p) \le \log_2 g$, with equality iff $p_i = 1/g$.

Proof (via concavity of $h(t) = -t \log_2 t$): *Jensen reminder: concave $f \Rightarrow \mathbb{E}[f(X)] \le f(\mathbb{E}[X])$.* $h(t) = -t \log_2 t$ is **concave** on $[0,1]$ ($h''(t) = -1/(t \ln 2) < 0$), so Jensen with uniform weights $1/g$:

$$\frac{1}{g}\sum_{i=1}^{g} h(p_i) \;\le\; h\left(\frac{1}{g}\sum_{i=1}^{g} p_i\right) \;=\; h\left(\tfrac{1}{g}\right) = -\tfrac{1}{g}\log_2 \tfrac{1}{g} = \tfrac{\log_2 g}{g}.$$

Entropy Is Maximal for Uniform Distributions

Claim: For any distribution on g outcomes, $H(p) \le \log_2 g$, with equality iff $p_i = 1/g$.

Proof (via concavity of $h(t) = -t \log_2 t$): *Jensen reminder: concave $f \Rightarrow \mathbb{E}[f(X)] \le f(\mathbb{E}[X])$.* $h(t) = -t \log_2 t$ is **concave** on $[0,1]$ ($h''(t) = -1/(t \ln 2) < 0$), so Jensen with uniform weights $1/g$:

$$\frac{1}{g}\sum_{i=1}^{g} h(p_i) \;\le\; h\left(\frac{1}{g}\sum_{i=1}^{g} p_i\right) \;=\; h\left(\tfrac{1}{g}\right) = -\tfrac{1}{g}\log_2 \tfrac{1}{g} = \tfrac{\log_2 g}{g}.$$

$\mathbb{E}[f(X)] \le f(\mathbb{E}[X])$

Multiply both sides by g:

$$H(p) = \sum_{i=1}^{g} h(p_i) \;\le\; \log_2 g.$$

Strict concavity $\Rightarrow$ equality iff all p_i equal, i.e., $p_i = 1/g$. ■

Intuition: the uniform is the "most uncertain" — it assumes nothing about which outcome is more likely.

Alternative proofs: Lagrange multipliers ($\partial \mathcal{L}/\partial p_i = 0$); or Gibbs $D_{\mathrm{KL}}(p \| u) \ge 0$ with u uniform.

Source Coding

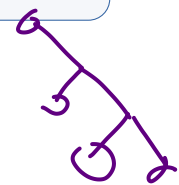
The promise of this section:

The number $H(X) = -\sum p \log_2 p$ is not just a mathematical quantity.

It is *literally* the **minimum number of bits** you need to describe X.

No algorithm, no matter how clever, can do better on average.

We'll **prove** this in the next 5 slides.



The Coding Problem

A shaurma stand's cashier sends order codes to the kitchen. Each order is a symbol from $\mathcal{X} = \{\text{cheese}, \text{shaurma}, \text{potato}, \text{chocko}\}$, encoded as a **binary string**. The probabilities reflect how often each item is ordered.

Fixed-length code: Each symbol gets a codeword of the same length.

Symbol	Probability	Codeword	Length
cheese	1/2	00	2 bits
shaurma	1/4	01	2 bits
potato	1/8	10	2 bits
chocko	1/8	11	2 bits

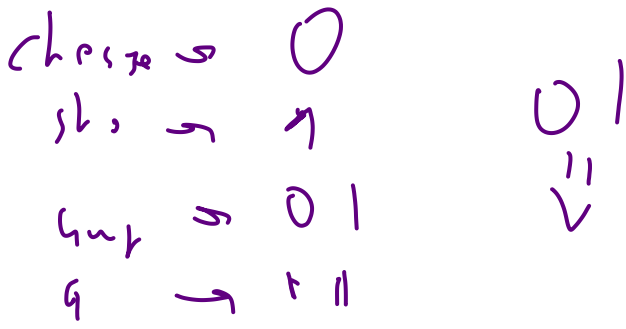
$$\mathbb{E}[L] = \tfrac{1}{2} \cdot 2 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{8} \cdot 2 + \tfrac{1}{8} \cdot 2 = \textbf{2 bits per symbol}$$

Half the orders going through the cashier are cheese — shouldn't it get a **shorter** code than rare items like chocko?

Variable-Length Codes and the Prefix Property

Idea: Shorter codes for more probable symbols, longer for less probable.

Suppose: cheese→0, shaurma→1, potato→01, chocko→11. I send 01. **What did I send?**



Variable-Length Codes and the Prefix Property

Idea: Shorter codes for more probable symbols, longer for less probable.

Suppose: cheese→0, shaurma→1, potato→01, chocko→11. I send 01. **What did I send?**

Either "cheese, shaurma" or "potato" — no way to tell. **Shorter codes create ambiguity.**

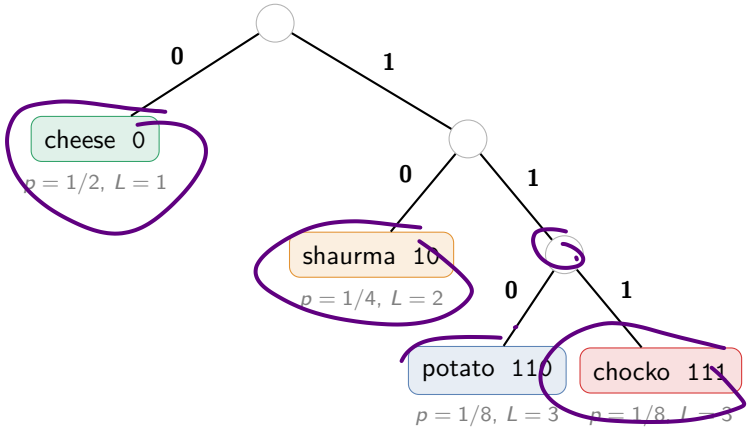
Prefix property: No codeword is a prefix of another codeword.

Guarantees **unambiguous decoding** — read left-to-right, always know where each codeword ends.

Solution — a valid prefix code:

Symbol	**Prob.** $p(x)$	**Codeword**	**Length** $L(x)$	**Surprisal** $-\log_2 p(x)$
cheese	1/2	0	1	1
shaurma	1/4	10	2	2
potato	1/8	110	3	3
chocko	1/8	111	3	3

Prefix Code as a Binary Tree



Shorter paths (fewer bits) for more probable symbols.
Each leaf is a codeword; the prefix property is guaranteed by the tree structure.

What If We Add a Fifth Item?

Suppose the menu gains a 5th item, also ordered $\frac{1}{8}$ of the time. Two things break:

(1) The probabilities no longer sum to 1.

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{9}{8} > 1 \quad \Rightarrow \quad \text{not a valid distribution!}$$

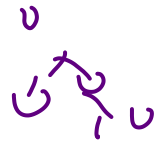
To fit it in, you must take probability mass away from the other items.

What If We Add a Fifth Item?

Suppose the menu gains a 5th item, also ordered $\frac{1}{8}$ of the time. Two things break:

(1) The probabilities no longer sum to 1.

$$\tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{8} + \tfrac{1}{8} = \tfrac{9}{8} > 1 \quad \Rightarrow \quad \text{not a valid distribution!}$$



To fit it in, you must take probability mass away from the other items.

(2) The codeword-length budget is already full. A binary prefix code with lengths L_i exists *iff* it obeys the **Kraft inequality** $\sum_i 2^{-L_i} \leq 1$. Our code:

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{8} = 1 \quad (\textbf{saturated}).$$

No bit-budget left for a 5th codeword — unless we lengthen an existing one.

What If We Add a Fifth Item?

Suppose the menu gains a 5th item, also ordered $\frac{1}{8}$ of the time. Two things break:

(1) The probabilities no longer sum to 1.

$$\tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{8} + \tfrac{1}{8} = \tfrac{9}{8} > 1 \quad \Rightarrow \quad \text{not a valid distribution!}$$

To fit it in, you must take probability mass away from the other items.

(2) The codeword-length budget is already full. A binary prefix code with lengths L_i exists *iff* it obeys the **Kraft inequality** $\sum_i 2^{-L_i} \leq 1$. Our code:

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{8} = 1 \quad (\textbf{saturated}).$$

No bit-budget left for a 5th codeword — unless we lengthen an existing one.

These are the same constraint! The optimal lengths $L_i = -\log_2 p_i$ give $2^{-L_i} = p_i$, so the bit-budget $\sum_i 2^{-L_i}$ *equals* the probability budget $\sum_i p_i = 1$. Running out of probability = running out of code space.

Optimal Code Length Equals Entropy

Key observation: In our prefix code, the code length of each symbol equals its surprisal!

$$L(x) = -\log_2 p(x) \qquad \text{for every symbol } x$$

Optimal Code Length Equals Entropy

Key observation: In our prefix code, the code length of each symbol equals its surprisal!

$$L(x) = -\log_2 p(x) \qquad \text{for every symbol } x$$

The **expected code length**:

$$\begin{aligned}\mathbb{E}[L(X)] &= \sum_x p(x)\, L(x) = \sum_x p(x) \cdot \big(-\log_2 p(x)\big) \\ &= \tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{8} \cdot 3 + \tfrac{1}{8} \cdot 3 = \textbf{1.75 bits} \\ &= H(X) \quad \checkmark\end{aligned}$$

The optimal prefix code assigns $-\log_2 p(x)$ bits to symbol x.

The **average code length** of this optimal code is exactly **the entropy** $H(X)$.

Compared to fixed-length (2 bits): we save $2 - 1.75 = 0.25$ bits per symbol!

Shannon's Source Coding Theorem

Noiseless Coding Theorem (Shannon, 1948):

For any source X with entropy $H(X)$:

(1) No prefix code can achieve $\mathbb{E}[L] < H(X)$.

(2) There exists a prefix code with $\mathbb{E}[L] < H(X) + 1$.

Entropy = the fundamental limit of lossless compression.

If you try to use fewer bits on average, you **must** lose information.

In practice: **Huffman coding** achieves near-optimal code lengths.

This gives entropy a physical meaning: $H(X)$ = minimum average bits needed to describe X.

Cross-Entropy

What happens when we use the **wrong** code?

Using the Wrong Codebook

The true distribution is p, but we **think** it's q and design our code for q.

Symbol	$p(x)$	$q(x)$	$L_p = -\log_2 p$	$L_q = -\log_2 q$	Diff
cheese	1/2	1/4	1	2	+1
shaurma	1/4	1/4	2	2	0
potato	1/8	1/4	3	2	−1
chocko	1/8	1/4	3	2	−1

Wait, what? Potato and chocko use *fewer* bits under the wrong code. So is the wrong code... sometimes better?

Using the Wrong Codebook

The true distribution is p, but we **think** it's q and design our code for q.

Symbol	$p(x)$	$q(x)$	$L_p = -\log_2 p$	$L_q = -\log_2 q$	**Diff**
cheese	$1/2$	$1/4$	1	2	$+1$
shaurma	$1/4$	$1/4$	2	2	0
potato	$1/8$	$1/4$	3	2	-1
chocko	$1/8$	$1/4$	3	2	-1

Wait, what? Potato and chocko use *fewer* bits under the wrong code. So is the wrong code... sometimes better?

Expected length under the **true** p:

$$\mathbb{E}_p[L_q] = \tfrac{1}{2}\cdot 2 + \tfrac{1}{4}\cdot 2 + \tfrac{1}{8}\cdot 2 + \tfrac{1}{8}\cdot 2 = 2 \text{ bits } > \ H(p) = 1.75.$$

Using the Wrong Codebook

The true distribution is p, but we **think** it's q and design our code for q.

Symbol	$p(x)$	$q(x)$	$L_p = -\log_2 p$	$L_q = -\log_2 q$	**Diff**
cheese	$1/2$	$1/4$	1	2	$+1$
shaurma	$1/4$	$1/4$	2	2	0
potato	$1/8$	$1/4$	3	2	-1
chocko	$1/8$	$1/4$	3	2	-1

Wait, what? Potato and chocko use *fewer* bits under the wrong code. So is the wrong code... sometimes better?

Expected length under the **true** p:

$$\mathbb{E}_p[L_q] = \tfrac{1}{2} \cdot 2 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{8} \cdot 2 + \tfrac{1}{8} \cdot 2 = 2 \text{ bits } > \ H(p) = 1.75.$$

Per-symbol diffs can be negative — but they're *weighted by* p. Cheese ($+1$) happens 50% of the time; potato (-1) only 12.5%. Frequent losses always outweigh rare savings.

Cross-Entropy: Definition

Cross-Entropy of p relative to q:

$$H(p\|q) \;=\; -\sum_x p(x)\,\log_2 q(x) \;=\; \mathbb{E}_{X\sim p}[-\log_2 q(X)]$$

Continuous case: $H(p\|q) \;=\; -\int p(x)\log_2 q(x)\,dx.$
Same formula, integral instead of sum.

"Average code length when data comes from p but we use the optimal code for q."

$H(p\|p) = H(p)$ (right code = entropy)

$H(p\|q) \geq H(p)$ (wrong code always wastes bits)

$H(p\|q) \neq H(q\|p)$ (**not symmetric!**)

ML interpretation: p = the true *data-generating process* (DGP); q_θ = your model. Cross-entropy = the cost of **misspecifying the DGP**. Training $= \min_\theta H(p\|q_\theta) = \min_\theta D_{\mathrm{KL}}(p\|q_\theta)$.

KL Divergence

How many bits do we **waste** by using the wrong code?

From Cross-Entropy to KL Divergence

The **gap** between cross-entropy and entropy measures the wasted bits:

$$\underbrace{H(p\|q)}_{\text{wrong code}} - \underbrace{H(p)}_{\text{right code}} = -\sum_x p(x)\log_2 q(x) - \left(-\sum_x p(x)\log_2 p(x)\right)$$

$$= \sum_x p(x)\left[\log_2 p(x) - \log_2 q(x)\right]$$

$$= \sum_x p(x)\,\log_2 \frac{p(x)}{q(x)}$$

From Cross-Entropy to KL Divergence

The **gap** between cross-entropy and entropy measures the wasted bits:

$$\underbrace{H(p\|q)}_{\text{wrong code}} - \underbrace{H(p)}_{\text{right code}} = -\sum_x p(x)\log_2 q(x) - \left(-\sum_x p(x)\log_2 p(x)\right)$$

$$= \sum_x p(x)\left[\log_2 p(x) - \log_2 q(x)\right]$$

$$= \sum_x p(x)\,\log_2 \frac{p(x)}{q(x)}$$

$\frac{p(x)}{q(x)}$

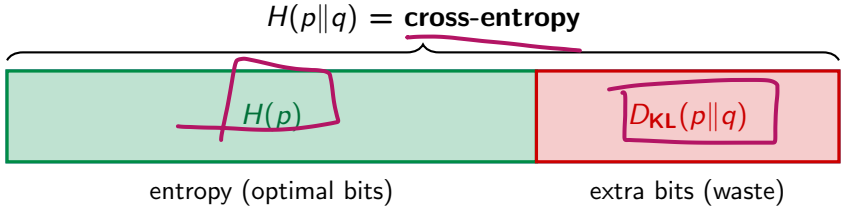
Kullback–Leibler Divergence:

$$D_{\mathrm{KL}}(p\|q) = \sum_x p(x)\,\log_2 \frac{p(x)}{q(x)} = \mathbb{E}_{X\sim p}\left[\log_2 \frac{p(X)}{q(X)}\right]$$

"Average number of **extra bits** when using q instead of p."

The Fundamental Identity

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



Since $D_{\mathrm{KL}}(p\|q) \geq 0$ (we'll prove this next), cross-entropy **always** exceeds entropy:

$H(p\|q) \geq H(p)$, with equality iff $p = q$.

Information Inequality: $D_{\mathrm{KL}} \geq 0$

Gibbs' Inequality: $D_{\mathrm{KL}}(p\|q) \geq 0$, with equality iff $p = q$.

Proof (via Jensen's inequality, since log is concave):

$$
\begin{aligned}
-D_{\mathrm{KL}}(p\|q) &= \sum_x p(x)\, \log \frac{q(x)}{p(x)} && \text{(flip the ratio)} \\
&\leq \log\left(\sum_x p(x) \cdot \frac{q(x)}{p(x)}\right) && \text{(Jensen: } \mathbb{E}[\log Z] \leq \log \mathbb{E}[Z]\text{)} \\
&= \log\left(\sum_x q(x)\right) = \log 1 = 0 \\
&\Rightarrow \quad D_{\mathrm{KL}}(p\|q) \geq 0
\end{aligned}
$$

Information Inequality: $D_{\mathrm{KL}} \geq 0$

Gibbs' Inequality: $D_{\mathrm{KL}}(p\|q) \geq 0$, with equality iff $p = q$.

Proof (via Jensen's inequality, since log is concave):

$$
\begin{aligned}
-D_{\mathrm{KL}}(p\|q) &= \sum_x p(x)\, \log \frac{q(x)}{p(x)} && \text{(flip the ratio)} \\
&\leq \log\left(\sum_x p(x) \cdot \frac{q(x)}{p(x)}\right) && \text{(Jensen: } \mathbb{E}[\log Z] \leq \log \mathbb{E}[Z]\text{)} \\
&= \log\left(\sum_x q(x)\right) = \log 1 = 0 \\
&\Rightarrow \quad D_{\mathrm{KL}}(p\|q) \geq 0
\end{aligned}
$$

Equality iff $q(x)/p(x)$ is constant $\forall\, x$ (strict concavity of log), i.e., $p = q$.

The most fundamental inequality in information theory.
You can never do better than the optimal code. Using any other distribution wastes bits.

KL Divergence Is Not a Distance

Despite measuring "closeness," D_{KL} is **not a distance**:

Property	**True distance?**	**KL?**
Non-negativity: $d(p, q) \geq 0$	✓	✓
Identity: $d(p, q) = 0 \Leftrightarrow p = q$	✓	✓
Symmetry: $d(p, q) = d(q, p)$	✓	×
Triangle inequality	✓	×

Example: $p = \mathrm{Bern}(0.1)$, $q = \mathrm{Bern}(0.5)$.

$D_{\mathrm{KL}}(p\|q) \approx 0.53$ bits $\neq$ $D_{\mathrm{KL}}(q\|p) \approx 0.74$ bits

KL is a **divergence**, not a distance. The asymmetry will matter a lot in ML!

Three Interpretations of KL Divergence

1. Extra bits: Average number of extra bits when coding data from p using the optimal code for q instead of p.

2. Expected log-ratio: $D_{\mathrm{KL}}(p\|q) = \mathbb{E}_{X\sim p}\left[\log \frac{p(X)}{q(X)}\right]$. How "distinguishable" are p and q on average, when data comes from p?

3. Expected log-likelihood ratio: For $H_0 : q$ vs $H_1 : p$, the log-likelihood ratio $\log \frac{p(X)}{q(X)}$ is the per-observation *evidence* for p over q. Its expectation under truth p is exactly $D_{\mathrm{KL}}(p\|q)$: the rate at which evidence accumulates (Stein's lemma).

All three interpretations say the same thing: D_{KL} measures **how different q is from p, as seen by p.**

KL in the Wild: Where It Shows Up in ML

Once you know KL, you start seeing it everywhere:

- **RLHF / GRPO / PPO (LLM training):** the objective adds a penalty $-\beta\, D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$ so the fine-tuned policy π_θ doesn't **drift too far** from the reference model π_{ref}. Without it, the model chases reward and outputs gibberish ("reward hacking").
- **Variational autoencoders:** the loss pulls the latent code toward a prior via $D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, \mathcal{N}(0, I)\big)$.
- **t-SNE / UMAP:** minimize KL between neighbor distributions in high-D vs 2-D — that's literally what draws the plot.
- **Knowledge distillation:** a small "student" matches a big "teacher" by minimizing KL between their output distributions.
- **Drift detection:** $D_{\mathrm{KL}}(\text{today} \,\|\, \text{training})$ flags when production data has shifted.

The **direction** is a deliberate choice: GRPO penalizes $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$ (reverse KL — "stay close to what you knew"). More on forward vs reverse KL in **info_02**.

Differential Entropy: Definition and Examples

For **continuous** X with density $f(x)$, entropy generalizes to:

$$h(X) = -\int f(x)\,\log_2 f(x)\,dx \qquad \text{(same shape, integral instead of sum)}$$

Uniform on $[0, a]$. $f(x) = 1/a$ on $[0, a]$:

$$h(X) = -\int_0^a \tfrac{1}{a}\log_2 \tfrac{1}{a}\,dx = \log_2 a.$$

Gaussian $N(\mu, \sigma^2)$. $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$. Using $\mathbb{E}[(X-\mu)^2] = \sigma^2$:

$$h(X) = -\mathbb{E}[\log_2 f(X)] = -\mathbb{E}\left[-\tfrac{1}{2}\log_2(2\pi\sigma^2) - \tfrac{(X-\mu)^2}{2\sigma^2 \ln 2}\right]$$

$$= \tfrac{1}{2}\log_2(2\pi\sigma^2) + \tfrac{1}{2\ln 2} = \boxed{\tfrac{1}{2}\log_2(2\pi e\,\sigma^2)} \quad \text{(depends only on variance).}$$

How can entropy be negative? A *density* can exceed 1: Uniform$[0, \frac{1}{2}]$ has $f(x) = 2$ on its support. Then the "surprisal" $-\log_2 f(x) = -1 < 0$, so the average $h = -\int f \log_2 f$ comes out **negative** ($h = \log_2 \frac{1}{2} = -1$ bit). Unlike discrete $H(X) \geq 0$ (which counts outcomes), h is measured *relative to a unit-width interval* — a distribution tighter than one unit is "more certain than the baseline," giving negative bits.

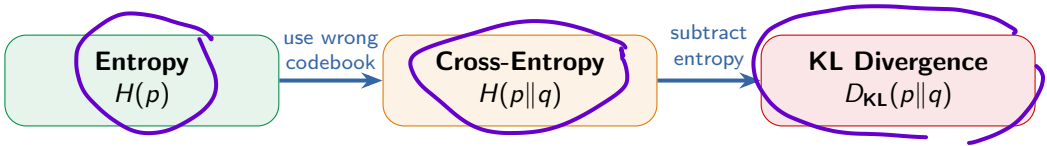
Properties of Differential Entropy

Property	Statement	Why
Translation invariance	$h(X + c) = h(X)$	shift doesn't change density
Scaling	$h(aX) = h(X) + \log_2 \lvert a \rvert$	change of variables
Additive (independence)	$h(X, Y) = h(X) + h(Y)$ if $X \perp Y$	factored density
Chain rule	$h(X, Y) = h(X) + h(Y \mid X)$	always (next lecture)
Can be negative	e.g. $h(\text{Uniform}[0, 1/2]) = -1$	no positivity guarantee
Not coordinate-invariant	$Y = g(X)$ shifts h by $-\log_2 \lvert \det J \rvert$	Jacobian leaks in

Why the scaling weirdness? Yes — it's exactly because finer units need more bits. h counts bits relative to a *unit-width* bin. Switch meters $\to$ centimeters: a unit bin now covers $100\times$ less real length, so there are $100\times$ more bins to distinguish $\Rightarrow$ $+\log_2 100 \approx 6.6$ bits. It's a **resolution artifact**, not extra information — the random variable is "the same."

Good news: $D_{\mathrm{KL}}(p \| q) = \int p \log_2 \frac{p}{q}\, dx$ and mutual information $I(X; Y)$ **are** coordinate-invariant. Use them instead of h for ML.

The Big Picture



$$\underbrace{H(p\|q)}_{\text{cross-entropy}} = \underbrace{H(p)}_{\text{entropy}} + \underbrace{D_{\mathrm{KL}}(p\|q)}_{\text{extra bits}}$$

ML connection: p = true data-generating process (DGP), q_θ = your model. Cross-entropy = the cost of **model misspecification**. Training a classifier $\Leftrightarrow$ minimizing $H(p\|q_\theta)$ $\Leftrightarrow$ minimizing $D_{\mathrm{KL}}(p\|q_\theta)$.

Next lecture (info_02): forward vs reverse KL, mutual information, MLE, decision trees.

Questions?