Information theory

How can we quantify the **value of data**? Is there a way to compare the "information content" in a book, a video, a strand of DNA, etc.?

Claude Shannon pioneered the field of information theory to answer this and related questions.

Consider a discrete random variable X with distribution P(X) and support Ax

$$X \sim p(x), \quad \sum_{x \in \mathcal{A}_x} p(x) = 1, \quad \text{supp}(x) = \mathcal{A}_x$$

The **Shannon information content** of the event X=x is defined as

$$h(X=x) = \log_2\left[\frac{1}{p(x)}\right] \text{ bits}$$

where the units are of **bits**. (For continuous random variables, we usually use the natural log and the units are **nats**, but for this lecture everything will be discrete.)

Why does this make sense? The event X=x contains more information the less probable it was in the first place. For example, should I have emailed everyone today to say that class was **not** canceled? Let's say that, in the past, class was only ever canceled once out of 1024 previous lectures. From your perspective, let's say that class being canceled today is a Bernoulli random variable with probability:

$$p(X=1) = 2^{-10} = 1/1024$$

The information of class being canceled (X=1) would be worth 10 bits to you:

$$h(X=1) = \log_2\left[\frac{1}{p(x=1)}\right] = 10 \text{ bits}$$

Meanwhile the information of class **not** being canceled (X=0) would only be worth about 0.0015 bits to you:

$$h(X=0) = \log_2\left[\frac{1}{p(x=0)}\right] \approx 0.0015 \text{ bits}$$

It's worth so little because it's so predictable; I don't need to send you an email to say that X=0 (and doing so would probably annoy you).

Related to this: should **you** send me an email asking whether class is canceled? This is a question about how much information you **expect to gain** the answer. The expected value of the information content is:

$$\mathbb{E}_{x \sim p(x)}[h(X=x)] = \sum_{x \in \mathcal{A}_x} p(x) \log_2\left[\frac{1}{p(x)}\right] \equiv H(X) \text{ bits}$$

Which is the definition of **Shannon entropy**, the central concept in information theory.

Entropy is a **functional** of the probability distribution P(X). We sometimes right it in terms of the random variable X and sometimes in terms of the probability distribution:

$$H(X) \equiv H(p(x)) \equiv H(p)$$

In this particular case, the information you expect to gain from observing X (i.e., the entropy of X) is:

$$\frac{1}{1024} \cdot 10 + \frac{1023}{1024} \cdot 0.0015 \approx 0.01 \text{ bits}$$

Which doesn't seem like much (you probably shouldn't email me to ask).

That said, what is the maximum amount of information you could expect to gain? In the case of binary random variables, the entropy is sometimes written as the **binary entropy function** of p=P(X=1):

$$H_2(p) = p \cdot \log_2 \frac{1}{p} + (1-p) \cdot \log_2 \frac{1}{1-p}$$

So when would the entropy be maximal?

$$0 = \frac{\partial}{\partial p} H_2(p) \longrightarrow p = 1-p = 0.5$$

So in the binary case, entropy is maximized when P(X=1)=0.5. In other words, when we are **maximally uncertain.**

And what is the entropy of a uniform Bernoulli random variable?

$$H_2(0.5) = 2 \cdot 0.5 \cdot \log_2 2 = 1 \text{ bit}$$

So a binary random variable is worth **1 bit** of information maximum.

It turns out that for any discrete distribution with finite support, the uniform distribution is the maximum entropy distribution. For simplicity, say that X takes on K categories and its distribution is represented by p1 ...pK:

$$\mathcal{A}_X = \{1 \dots K\}, \quad p = (p_1 \cdots p_K)$$

$$\underset{p}{\operatorname{argmax}} \; H(p) = \left(\frac{1}{K} \cdots \frac{1}{K}\right)$$

And the maximum entropy of a discrete X that takes K categories is:

$$\max_p H(p) = \max_p \sum_j p_j \log_2 \left[\frac{1}{p_j}\right] = K \cdot \frac{1}{K} \log_2 K = \log_2 K \text{ bits}$$

This quantity is known as the **raw bit content** of X and is often denoted:

$$H_0(X) = \log_2 |\mathcal{A}_X|$$

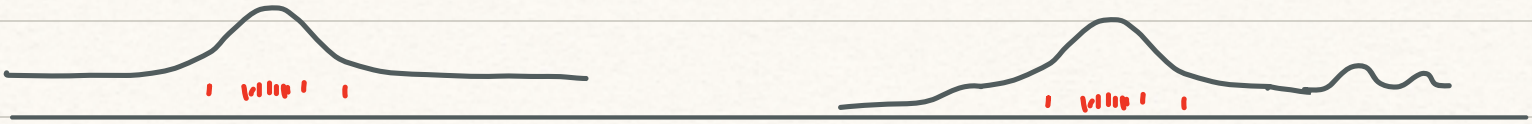
Entropy as uncertainty

We've seen entropy before but didn't name it as such. Consider the ELBO in EM:

$$\mathbb{E}_q\left[\log \frac{p(x,z)}{q(z)}\right] = \mathbb{E}_q\left[\log p(x,z)\right] + \underbrace{\mathbb{E}_q\left[\log \frac{1}{q(z)}\right]}_{= H(q)}$$

So we can think about the ELBO as a sum of (the expected) complete-data likelihood plus the **entropy of the surrogate distribution q**. The ELBO is an objective function we seek to maximize. Maximizing a likelihood makes sense. Why does maximizing the entropy of the q-distribution also make sense?

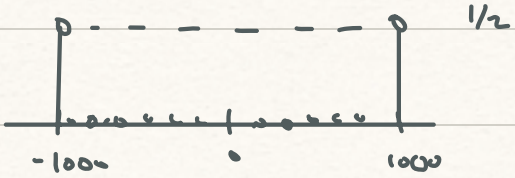
As an illustrative example, consider the following two densities for a dataset (the red points):



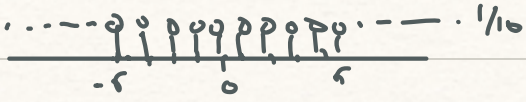
The two assign the exact same density to the data points, but the one on the right does weird stuff in the tails while the one on the left is uniform away from the data. Which of these two would you prefer as a fit to the data (assuming we don't have any additional prior information about the areas away from the data points)?

It makes sense to prefer the density that **maximizes entropy subject to fitting the data well**. This is the density that will be uniform (loosely speaking) in areas with no data. This is informal and philosophical but still commonsense.

Entropy != variance: Many are tempted to equate the variance of a distribution with "**uncertainty**". But the entropy and variance of a random variable can be very different, and equating them is often a source of confusion and bad methodology.



High variance, low entropy



Low variance, high entropy

An important thing to understand is that "**entropy does not care about the labels on the dice**"; it is only a functional of the **probabilities of the sides of the dice**. This is not true for variance, which is a functional of both the probabilities and the values themselves:

$$\mathrm{Var}(X) = \sum_{x \in A_X} p(x)\left(x - \mathbb{E}[X]\right)^2 \quad \text{vs.} \quad H(X) = \sum_{x \in A_X} p(x) \log_2\left[\frac{1}{p(x)}\right]$$

This is exactly what makes **information** a universal language for talking about the value of data. It doesn't matter whether X is a video, a book, the event of cancelling class, etc.; the only thing that matters is P(X).

Exponential families as the unique maximum entropy distributions

With all that in mind, let's revisit exponential families, and see another reason why they are special.

Any exponential family can be written as:

$$p_\eta(x) = \mathbb{1}(x \in A_x)\, h(x) \exp(\eta^T t(x) - a(\eta))$$

Define the set of all distributions over the support Ax which also satisfy the following **moment constraint**:

$$\mathcal{S}_\alpha \triangleq \left\{ p : \mathbb{E}_{x \sim p(x)}[t(x)] = \alpha,\ x \in A_x \right\}$$

The exponential family distribution above is a member of this set. It is more than just a mere member though...

> **Theorem:** if $p_\eta \in \mathcal{S}_\alpha$ then $H(p_\eta) > H(p')\ \forall\ p' \in \mathcal{S}_\alpha$.

The distribution in this set that has **maximum entropy** is **unique** and is an **exponential family.**

We will prove this in two steps: 1) first, by showing that the entropy of this expfam is **at least as large** as any other member of the set and then 2) by showing that it is unique.

proof: ① $H(p_\eta) \geq H(p')\ \forall\ p' \in \mathcal{S}_\alpha$ ② uniqueness

① show $H(p_\eta) \geq H(p')\ \forall\ p' \in \mathcal{S}_\alpha$.

For simplicity, assume the distribution is discrete (this can be generalized):

- $p(x = k) = p_k, \quad \sum_k p_k = 1$

We will introduce a Lagrangian to enforce constraints when maximizing the entropy:

- $\Lambda(p, \eta, \eta_0) = -H(p) + \eta^T(\alpha - \mathbb{E}[t(x)]) + \eta_0\left(\sum_{k \in S} p_k - 1\right)$

  Lagrangian term to enforce the **moment constraint**

  Lagrangian term to enforce the **normalization constraint**

Now find the minimizer:

- $\operatorname{argmin}_p \Lambda(p, \eta, \eta_0)$

We will do this by solving for each coordinate at a time, setting its partial derivative to 0:

- $\frac{\partial}{\partial p_j}\Lambda(p, \eta, \eta_0) = \frac{\partial}{\partial p_j}\sum_k p_k \log p_k + \frac{\partial}{\partial p_j}\eta^T(\alpha - \sum_k p_k t(x)) + \frac{\partial}{\partial p_j}\eta_0\left(\sum_k p_k - 1\right)$

  $= \log p_j + 1 \quad - \eta^T t(j) \quad + \eta_0$

Setting this to zero:

- $0 = \;''\qquad''\;\Rightarrow \log p_j = \eta^T t(j) - \eta_0 - 1$

$$\Rightarrow \quad p_j = \exp(\eta^T t(j) - \eta_0 - 1)$$

We can always replace the Lagrangian parameters with a form that ensures the constraints they are meant to enforce. In this case, we can enforce that the pj terms sum to 1 (by setting:)

- $\eta_0 = a(\eta) - 1, \qquad a(\eta) = \log \sum_{x \in A_x} \exp(\eta^T t(x))$

Notice that we could also introduce a base measure term h(x) in this step:

$$p_j = h(j) \exp(\eta^T t(j) - \eta_0 - 1)$$

$$\eta_0 = \log \sum_{x \in A_x} h(x) \exp(\eta^T t(x)) + 1$$

We've gotten rid of one set of Lagrangian parameters, but not the other set, which enforces the **moment constraint**. We can view these as parameters of the probability distribution:

- $p_j \equiv p_\eta(j)$

It's not necessarily true that there exists a setting of these parameters that enforces the moment constraint. However IF there does exist such a setting, then the resulting distribution is a maximum entropy member of the set, and is an exponential family (as is clear from its form):

- if $\exists \eta$ s.t. $\mathbb{E}_{x \sim p_\eta}[t(x)] = \alpha$ then

$$p_\eta \in \mathcal{S}_\alpha \quad \text{and} \quad H(p_\eta) \geq H(p') \;\; \forall \, p' \in \mathcal{S}_\alpha$$

(2) show uniqueness.

Now consider any other distribution in the set: any $p \in \mathcal{S}_\alpha$

We can write its entropy as:

$$H(p) = -\sum_k p(k) \log p(k) = -\sum_k p(k) \log \left[p(k) \frac{p_\eta(k)}{p_\eta(k)} \right]$$

$$= -\sum_k p(k) \log \frac{p(k)}{p_\eta(k)} - \sum_k p(k) \log p_\eta(k)$$

The first term is just the KL between p and the max-ent expfam. The second term we plug-in the expfam form:

$$= -KL(p \,\|\, p_\eta) - \sum_k p(k) \left[\eta^T t(k) - a(\eta) \right]$$

The expectation pushes in, and can be replaced by definition, with the moment constraint:

$$= -KL(p \| p_\eta) - \left[\eta^T \underbrace{\mathop{\mathbb{E}}_{x\sim p}[t(x)]}_{=\alpha = \mathop{\mathbb{E}}_{x\sim p_\eta}[t(x)]} - a(\eta)\right]$$

Since both distributions are a member of this set, expectations under both will equal alpha. Therefore, we can replace the expectation WRT p with an expectation WRT to the max-ent expfam:

$$= -KL(p \| p_\eta) \underbrace{- \sum_k p_\eta(k) \log p_\eta(k)}_{= H(p_\eta)}$$

After doing so, we recognize this as a KL-divergence minus the entropy of the max-ent expfam. Therefore:

$$H(p) = H(p_\eta) - KL(p \| p_\eta)$$

Since the entropy of the expfam is maximal, and since the KL is non-negative, then H(p) must equal the entropy of the expfam if p is also a max-ent distribution. However this can only happen if the KL is 0, and that can only occur if the two distributions are equal. Therefore the expfam is the **unique max-ent distribution**:

$$H(p) = H(p_\eta) \quad \text{IFF} \quad p = p_\eta.$$

So besides all their other convenient properties (e.g., conjugacy), why should we like building models out of exponential family distributions? Because, subject to their moment constraints (i.e., "fitting the data") they are are uniquely uncertain!

Game of submarine

The following is from MacKay chapter 4.1. There is a missing submarine somewhere in a 64-cell grid. We need to guess where it is. Every time we choose a cell, we learn whether or not it is there. The probability of the submarine being in any given cell is uniform 1/64:

$$P(Z = k) = \frac{1}{64}, \quad k \in \{1 \dots 64\}$$

This will be an iterated guess-and-check game. Define the probability we guess correctly on the first attempt to be:

$$p(\text{correct on 1st try}) \equiv p(x_1 = 1) = \frac{1}{64}$$

The information gained from this event is:

$$h(x_1 = 1) = \log_2 64 = 6 \text{ bits}$$

The information gained from the event that we do not find the sub on the first try:

$$h(x_1 = 0) = \log_2 \frac{64}{63} \approx 0.023 \text{ bits}$$

Now on the second attempt (assuming we missed on the first), the probabilities change because we can rule out one of the cells. So the probability of guessing correctly on the second attempt, given that we missed on the first is:

$$p(x_2 = 1 \mid x_1 = 0) = 1/63$$

What is the total information gained after 32 misses?

$$h(x_1 = 0, \dots x_{32} = 0) = \log_2 \frac{64}{63} + \log_2 \frac{63}{62} + \dots + \log_2 \frac{33}{32}$$

$$= \log_2 \frac{64}{32} = \log_2 2 = 1 \text{ bit}$$

What about the total information gained after hitting on the 33rd attempt:

$$h(x_1 = 0 \dots x_{32} = 0, x_{33} = 1) = 1 + \log_2 32 = 6 \text{ bits}$$

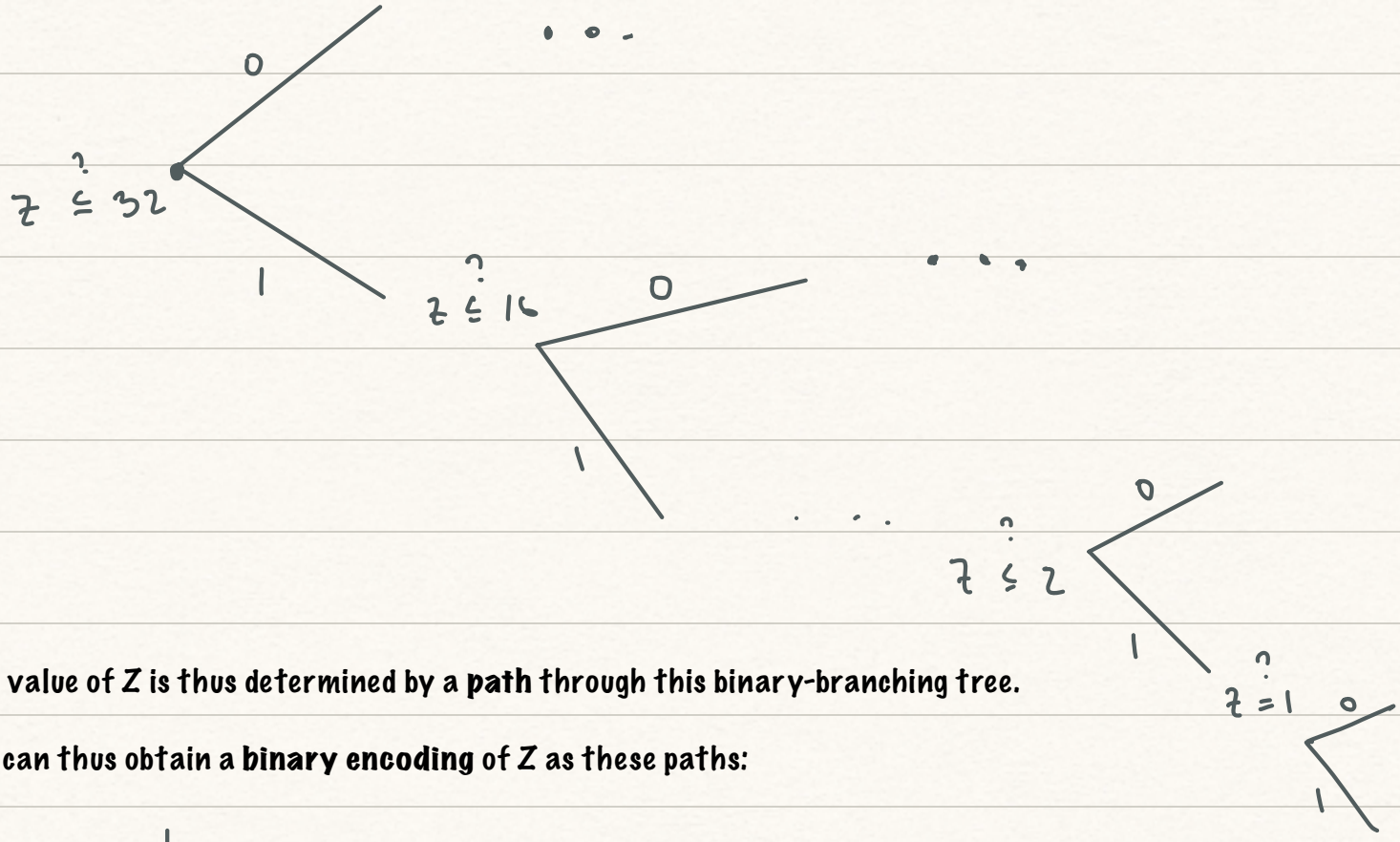
What about hitting after 48 misses?

$$h(x_1 = 0 \dots x_{48} = 0, x_{49} = 1) = \log \frac{64}{63} + \dots + \log_2 \frac{17}{16} + \log_2 16$$

$$= \log_2 \frac{64}{16} + \log_2 16 = 6 \text{ bits}$$

So finding the sub is **always worth 6 bits** no matter when you find it.

Why 6 bits? One interpretation: Z is worth the answer to 6 yes-no questions.



The value of Z is thus determined by a **path** through this binary-branching tree.

We can thus obtain a **binary encoding** of Z as these paths:

z	$C(z)$
1	111111
2	111110
⋮	⋮
64	000000

This is a 6-bit encoding of Z (since it uses 6 0s-or-1s).

It should not be too surprising that it takes 6-bits to encode Z since the **raw bit content of Z** is 6 bits:

$$H_0(Z) = \log_2 |\mathcal{A}_Z| = \log_2 64 = 6 \text{ bits}$$

In this particular case, we were maximally uncertain about where the sub was. What if Z were more predictable? i.e., $H(z) < H0(Z)$? Can we do better in terms of **encoding Z more efficiently** (i.e., using fewer bits)? This was the problem that Claude Shannon and others at Bell Labs solved. The context was WW2, and transmitting/receiving messages (e.g., via Morse code) efficiently was extremely important. We will see in the next section how knowing $P(Z)$ let's us encode (or "compress") Z optimally, subject to some error tolerance.

Lossy compression of fixed-length symbol codes

Consider the following example from MacKay (chapter 4.3, page 75).
The alphabet is letters a-h. The first four letters account for probability mass 15/16.

$\mathcal{A}_x$ =	a	b	c	d	e	f	g	h
$p(x)$ =	1/4	1/4	1/4	3/16	1/64	...		1/64

$$p(x \in \{a,b,c,d\}) = \frac{15}{16}$$

The **raw bit content** of this ensemble is **3 bits.**

$$H_0(x) = \log_2 8 = 3 \text{ bits}$$

This means a fixed-length symbol code would have 3-bits naively:

$c(x)$ =	000	001	...	111

Let's say we now have some **error tolerance.** We are okay if some symbols fail to be communicated some (small) fraction of the time. Say the error tolerance is 1/16:

$$\delta = 1/16$$

To reduce the number of bits used to transmit messages, subject to this error tolerance, we could assign fixed-length symbol codes to **only** the first four letters. The remaining letters will have no symbol; thus we cannot transmit or receive them (they are always errors).

$\mathcal{A}_x$ =	a	b	c	d	e	f	g	h
$p(x)$ =	1/4	1/4	1/4	3/16	1/64	...		1/64
$c_\delta(x)$ =	00	01	10	11	N/A	...		N/A

We say that {a,b,c,d} is the **smallest delta-sufficient subset** in this case, because it is the smallest subset of the support of X satisfying:

$$p(x \in S_\delta) \geq 1 - \delta$$

The **essential bit content** of X is then:

$$H_\delta(x) = \log_2 |S_\delta|$$

Notice that when the error tolerance is zero, the essential bit content equals the raw bit content.

Lossy compression of blocks

Can we do better? What if we instead received/transmitted **blocks** of symbols? Define an N-length block as:

$$X^N = (X_1 \cdots X_N), \qquad X_i \overset{iid}{\sim} p(X) \quad i = 1 \ldots N$$

What is the entropy of such a sequence?

$$H(X^N) = -\sum_{x_1} \cdots \sum_{x_N} p(x_1 \cdots x_N) \log_2 p(x_1 \cdots x_N)$$

$$= -\sum_{x_1} \cdots \sum_{x_N} p(x_1 \cdots x_N) \sum_{i=1}^{N} \log_2 p(x_i)$$

$$= -\sum_{i=1}^{N} \sum_{x_i} p(x_i) \log_2 p(x_i) = \sum_{i=1}^{N} H(X) = NH(X)$$

For iid Xi's, it is just the sum of the individual entropies. (Historical note: Shannon chose log as the transformation of p(x) because "information should be additive".)

Therefore a sequence can be encoded with no loss with NH0(X) bits (i.e., its raw bit content).

A seminal result in info theory is the **source coding theorem**, which states:

Theorem: X^N can be encoded in $NH(X)$ bits with negligible loss as $N \to \infty$.

For any given sequence of length N, define the number of symbols taking value k to be:

$$N_k \triangleq \sum_{i=1}^{N} 1(x_i = k)$$

In a "typical" sequence, this number will be close to the mean under multinomial sampling:

$$N_k \simeq N p_k$$

By extension, the probability of a "typical" sequence will roughly be:

$$p(x^N) \simeq \prod_{k \in A_x} p_k^{N p_k}$$

And therefore the information content of a "typical" sequence will roughly be:

$$h(x^N) \simeq N \sum_k p_k \log_2 \left[\frac{1}{p_k}\right] = NH(X)$$

More formally we say that a sequence is **epsilon-typical** if:

$$X^N \text{ is } \varepsilon\text{-typical if } \left| \frac{1}{N} \sum_{i=1}^{N} h(x_i) - H(X) \right| < \varepsilon$$

This then defines the **(epsilon, N)-typical set:**

$$A_{\epsilon,N} = \{ x^N : x^N \text{ is } \epsilon\text{-typical} \}$$

What is the probability that a given sequence is in the typical set?

$$p(x^N \in A_{\epsilon,N}) = ?$$

First define as a random variable the information content of a single symbol in the sequence:

$$R_i \triangleq h(x_i) = \log_2\left[\frac{1}{p(x_i)}\right]$$

Since all the xi's are iid, the Ri's are also iid, and they have mean equal to the entropy of x:

$$R_i \overset{iid}{\sim} p(R), \quad \mathbb{E}[R] = \mathbb{E}_{x\sim p(x)}[h(x)] = H(x)$$

We can therefore rewrite the epsilon-typical condition in terms of a sample and population mean:

$$x^N \text{ is } \epsilon\text{-typical if } \left|\frac{1}{N}\sum_{i=1}^{N} R_i - \mathbb{E}[R]\right| < \epsilon$$

By the LLN we can therefore say that:

$$\lim_{N\to\infty} p\left(\left|\frac{1}{N}\sum_{i=1}^{N} R_i - \mathbb{E}[R]\right| < \epsilon\right) \to 1$$

For large enough N, every generated sequence is almost surely in the typical set!

How big is the typical set? First rewrite the definition of epsilon-typical as

$$H(x) - \epsilon \le \frac{1}{N}\log_2\frac{1}{p(x^N)} \le H(x) + \epsilon$$

$$2^{-N(H(x)+\epsilon)} \le p(x^N) \le 2^{-N(H(x)-\epsilon)}$$

So the probability of a typical sequence is very close:

$$p(x^N) \simeq 2^{-NH(x)}$$

If all generated sequences are in the typical set, and if all of them have uniform-ish probability, then the size of the typical set must be the inverse of that probability:

if all x^N s.t. $x^N \sim p(x^N)$ are in $A_{\epsilon,N}$

and if $p(x^N) \simeq 2^{-NH(x)}$ for all $x^N \in A_{\epsilon,N}$

then $|A_{\epsilon,N}| \simeq 2^{NH(x)}$

This is the **asymptotic equipartition principle (AEP)**.

The typical set is **much smaller** than the set of all possible sequences.

For example, consider just binary symbols, with slightly uneven probabilities:

$$|\mathcal{A}_X| = 2, \quad p(x_i = 1) = 0.4$$

Now compare the **raw bit content** of a length-N sequence to the size of the typical set:

$$\frac{|\mathcal{A}_{\epsilon,N}|}{|\mathcal{A}_X|^N} \simeq 2^{N(H(x) - H_0(x))} = 2^{N(0.97 - 1.0)} = 2^{-0.03N}$$

Say N=3000. Then only 2^{-30} proportion of binary sequences are typical, a vanishing fraction.

Lossless compression with variable-length symbol codes

A similar intuition is obtained by allowing for variable-length codes. Consider the following setup:

$\mathcal{A}_X$	=	a	b	c	d
$c(x)$	=	00	01	10	11 ← prefix codes
$l(x)$	=	2	2	2	2
$p(x)$	=	1/2	1/4	1/8	1/8

The expected length of any given message is:

$$\mathbb{E}[l(x)] = 2 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot 2 \cdot \frac{1}{8} = 2 \text{ bits}$$

A version of the source coding theorem tells us that the expected code length is bounded by the entropy:

Theorem: $\mathbb{E}[l(x)] \geq H(x)$.

$\mathbb{E}[l(x)] = H(x)$ only if $l(x) = h(x) = \log_2\left[\frac{1}{p(x)}\right]$.

Now allow for **variable-length** codes, e.g.:

$c(x)$	=	0	10	110	111

The expected length of a given message is now:

$$\mathbb{E}[l(x)] = \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot 3 \cdot \frac{1}{8} = 1 + \frac{6}{8} = 1.75 \text{ bits}$$

The takeaway is that it is still NH(X) bits (on average) to encode a sequence X^N.

Modeling as compression

The source coding theorem tells us that an optimal coding scheme for X exists and is based on P(X). In practice we do not know P(X). Modeling P(X) and learning to optimally compress X are in this sense the same problem. It is not a coincidence that many modeling approaches for P(X) have a flavor of compression or dimensionality reduction. We will explore this connection more formally, particularly when we study autoencoders.