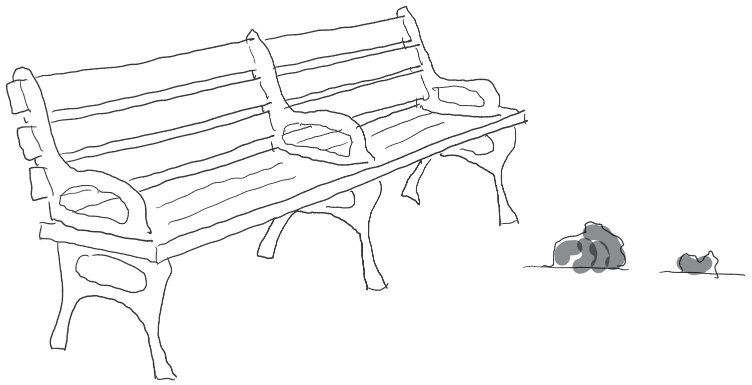
Information theory

A secret message

There are two spies, Alice and Bob. Every morning, Alice finds out a new secret fact: she observes through her telescope, and discovers whether the Prime Minister gets out of bed on the left side or the right side.

Immediately, she takes a walk in the park. She sits on a particular park bench, and arranges the rocks next to it just so: the bigger rock to the left if today was a left-side day, or the bigger rock to the right if today was a right-side day.



A few minutes after she leaves, Bob walks by and quietly observes the rocks. He learns Alice's message, and informs his superiors.

Channel

Alice and Bob have arranged a communication channel: on each time step (i.e., once per day at the assigned time) they can communicate one of two symbols (arrangements of rocks). This is a pretty slow and expensive channel: its rate is one bit per day.

In fact, Alice has plenty of other secret information she wants to send: what the Prime Minister had for breakfast, whether the Parliament is about to declare war, and so forth. So, we can set up a faster channel: Alice and Bob could arrange several meeting spots per day, or agree on a larger number of possible symbols for each meeting. If they agree on 8 different arrangements of the rocks, they can send three bits at a time instead of one. If they then meet twice per day, the rate becomes 6 bits/day.

Efficiency

But let's suppose we've hit the limits of our channel's rate. We're still pretty slow and expensive, so we want to use the channel as efficiently as possible.

Let's say that Alice and Bob each have a copy of a code book: it lists N possible high-level meanings that Alice could want to send, and gives a code word (a sequence of symbols or bits) for each of them. Maybe some of the code words correspond to possible breakfast items, and others correspond to possible war plans. How fast can Alice send what she knows?

For simplicity let's suppose that symbols and bits are the same: each symbol is 0 or 1, instead of having to deal with drawing different arrangements of rocks and flowers.

For example, we could have the following code book:

meaning	code word
pancakes	00
shredded wheat	10
bagels	01
caviar	11

With this code book, every breakfast item takes two bits to send. In general, if we have N different breakfast items, the code words will each have length $\lceil \log_2 N \rceil$ bits. This turns out to be optimal in the worst case: if we are unlucky, we might have to send our longest code word every day, so we want all code words to have as close as possible to equal length.

What if we know ahead of time that some code words are more frequent than others? The minister might have pancakes most mornings, and caviar only rarely. Intuitively, we'd like to send a shorter message for pancakes, at the cost of perhaps having to send a longer one for caviar. If code word i has b_i bits, and if the probability of code word i is p_i, then the expected message length is

$$p_1 b_1 + p_2 b_2 + \ldots + p_N b_N$$

For example, suppose we update our code book and annotate the probabilities:

meaning	code word	probability
pancakes	0	$\frac{1}{2}$
shredded wheat	10	$\frac{1}{4}$
bagels	110	$\frac{1}{8}$
caviar	111	$\frac{1}{8}$

Then the average number of bits in our message is

$$1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} \;=\; \frac{7}{4}$$

This is better! Instead of $\log_2 4 = 2$ bits per code word, we spend on average only 1.75 bits/word. (We'll see below that this code is optimal for this distribution over breakfasts.)

On the other hand, suppose the probability distribution were uniform. The original code book (where all code words are two bits long) would cost us 2 bits per breakfast, both on average and in the worst case. But the revised code book would cost us

$$(1+2+3+3)/4 = 2.25$$

bits per breakfast. There's a cost to choosing the wrong code book.

If we have s different symbols instead of just 0 and 1, it makes sense to report costs using the log base s instead of the log base 2. For example, if we use the numerals $0 \ldots 9$ as our symbols, we have $s = 10$. In this case we report how many decimal digits (or dits) we used, instead of how many bits. It's even possible to report costs with a non-integer base. The only common one is e, the base of the natural logarithm; the units in this case are called nats.

Entropy

Different distributions and different code books lead to different expected bit costs. What is the optimal cost for a given distribution? It turns out that the best code book uses approximately $-\log_2 p_i$ bits to send a code word of probability p_i. (In fact we can hit this value exactly for long sequences of code words, though we may have to be slightly cleverer than a fixed code book.) So, the expected cost is

$$-\sum_{i=1}^{N} p_i \log_2 p_i \quad \text{bits}$$

This quantity is called $H(p)$, the *entropy* of the probability distribution p.

Joint, conditional, and marginal entropy

If we have several random variables, we can ask for the entropy of their joint distribution, or of various conditional or marginal distributions. These are called joint, conditional, or marginal entropies.

For example, suppose that the Prime Minister is more likely to want shredded wheat after getting out of bed on the left side. Call the joint distribution $p_{XY}(X, Y)$:

X	Y	p_{XY}
pancakes	left	$\frac{1}{2}$
shredded wheat	left	$\frac{1}{6}$
pancakes	right	$\frac{1}{6}$
shredded wheat	right	$\frac{1}{6}$

The marginal distribution of breakfast is

X	p_X
pancakes	$\frac{2}{3}$
shredded wheat	$\frac{1}{3}$

On the other hand, if we find out that $Y = \text{left}$, then we have a new conditional distribution over X which we will call $p_{\text{left}}(X) = P(X \mid Y = \text{left})$:

X	p_{left}
pancakes	$\frac{3}{4}$
shredded wheat	$\frac{1}{4}$

Exercise: compute the joint entropy, the marginal entropy of X, and the conditional entropy given $Y =$ left.

Mismatch

So far we've assumed that Alice and Bob both know the distribution of code words. What if Bob only knows an old version of the probability distribution? The possible code words are the same, but the probabilities have changed: the current probabilities (known only to Alice) are p_i for code word i, while the old probabilities (known to both Alice and Bob) are q_i.

In order for Alice and Bob to communicate, they have to use a code based on the old probabilities q_i, since these are the only probabilities that they both know. So, we'll assign a code word of length $\log_2 q_i$ bits to meaning i, in contrast to the optimal scheme which would use $\log_2 p_i$ bits. That means that we will pay a small amount extra: a penalty of

$$\left[-\sum_{i=1}^{N} p_i \log_2 q_i\right] - \left[-\sum_{i=1}^{N} p_i \log_2 p_i\right] \quad \text{bits}$$

This amount is called the *relative entropy* of q with respect to p, or the *KL divergence* from p to q. It is written

$$D(p \,\|\, q) = \sum_{i=1}^{N} p_i \log_2 [p_i / q_i]$$

Note that the KL divergence is always nonnegative, and strictly positive if $p \neq q$: we pay a penalty if we use the wrong distribution to design our codebook.

Relative entropy example

We already did most of the work above to calculate some relative entropies: we gave the optimal codebooks for the two distributions

$$p = \left(\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}\right) \qquad q = \left(\tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \tfrac{1}{8}\right)$$

and looked at the performance of each codebook under each distribution. We found that the codebook for the uniform distribution p always takes 2 bits per code word:

$$\sum_{i=1}^{4} -p_i \log_2 p_i = \sum_{i=1}^{4} -q_i \log_2 p_i = 2 \text{ bits}$$

And we found that the codebook for q does better if the true distribution is q, and worse if it's p instead:

$$\sum_{i=1}^{4} -q_i \log_2 q_i = 1.75 \text{ bits} \qquad \sum_{i=1}^{4} -p_i \log_2 q_i = 2.25 \text{ bits}$$

From the above, we can calculate relative entropies: e.g.,

$$D(p \,\|\, q) = 2.25 - 2 = 0.25 \text{ bits}$$

Exercise: calculate $D(q \,\|\, p)$.

The first example uses the two calculations where the true distribution is p, so it measures the difference between the two codebooks under p: i.e., the penalty when we use the codebook for q under the wrong distribution. The second example uses the two calculations where the true distribution is q, so it measures the penalty when we use the codebook for p under the wrong distribution.

Information gain

The KL divergence can also help measure the amount of information we gain from finding out something new. If Bob thought that the distribution of code words was q, but learns that it is p instead, he gains $b = D(p \,\|\, q)$ bits of information: in his old state of knowledge Alice would have had to pay a penalty of b bits to send him a message, but now she no longer has to pay this penalty.

More generally, if an observation (like some training data) causes us to update our belief about something (like a model's parameters) from a prior distribution q to a posterior distribution p, we say that $D(p \,\|\, q)$ is the information gain from this observation. If we see a lot of highly relevant data, we'll have many bits of information gain; if we see little data or if the data isn't very relevant, we won't have much information gain.

For example, in the joint distribution p_{XY} above (for different breakfasts and different sides of bed), we can calculate the information gain from finding out $Y = \text{left}$: it is

$$D(p_{\text{left}} \,\|\, p_X) = \tfrac{3}{4}(\log_2 \tfrac{3}{4} - \log_2 \tfrac{2}{3}) + \tfrac{1}{4}(log_2 \tfrac{1}{4} - \log_2 \tfrac{1}{3})$$

or about 0.024 bits. (Recall: $p = (\frac{2}{3}, \frac{1}{3})$, $p_{\text{left}} = (\frac{3}{4}, \frac{1}{4})$.)

On the other hand, if we find out $Y = \text{right}$, our new distribution is $p_{\text{right}} = (\frac{1}{2}, \frac{1}{2})$, and our information gain is $D(p_{\text{right}} \parallel p_X)$, or about 0.85 bits.

Exercise: show how to calculate $D(p_{\text{right}} \parallel p_X)$.

What if we don't know Y yet? How much should Alice be willing to sacrifice to obtain her secret observation of side-of-bed? Since the marginal distribution of side of bed is $(\frac{2}{3}, \frac{1}{3})$, we expect to gain:

$$\tfrac{2}{3} D(p_{\text{left}} \parallel p) + \tfrac{1}{3} D(p_{\text{right}} \parallel p)$$

or about 0.044 bits.

Mutual information

This quantity — the expected information gain about X from finding out Y — is also called the *mutual information* between X and Y, written $I(X,Y)$.

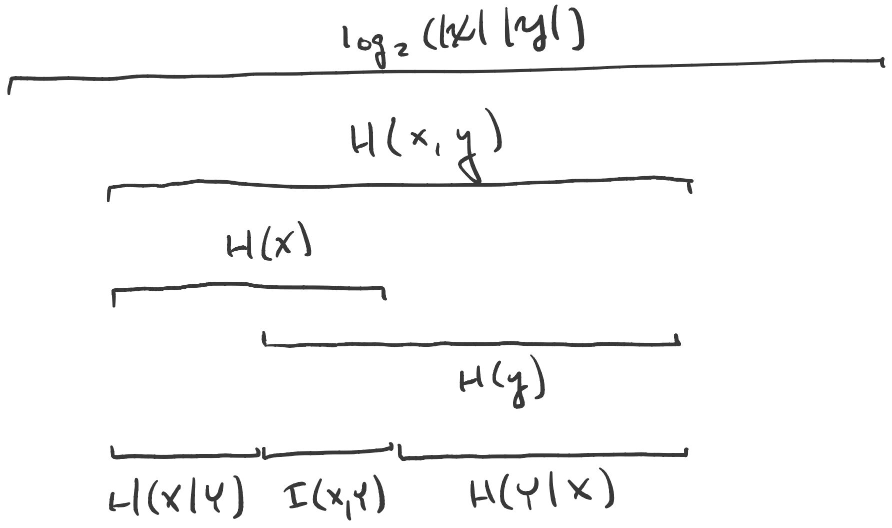
$$I(X,Y) = \mathbb{E}_X[D(P(Y \mid X) \parallel P(Y))]$$

It's not obvious from the definition or the derivations above, but $I(X,Y)$ is symmetric:

$$\begin{aligned}
&I(X,Y) = \mathbb{E}_X[D(P(Y \mid X) \parallel P(Y))] \\
&= \mathbb{E}_X[\textstyle\sum_y P(Y=y \mid X)(\ln P(Y=y \mid X) - \ln P(Y=y))] = \mathbb{E}_{X,Y}[\ln P(Y \mid X) - \ln P(Y)] \\
&= \mathbb{E}_{X,Y}[\ln P(X,Y) - \ln(P(X)P(Y))] = \mathbb{D}(P(X,Y) \parallel P(X)P(Y))
\end{aligned}$$

In addition to showing $I(X,Y)$ is symmetric, this derivation also shows something else particularly interesting: $I(X,Y)$ is the KL divergence between the joint distribution $P(X,Y)$ and the product of marginals $P(X)P(Y)$. The joint and product of marginals are the same if and only if X and Y are independent. So, the mutual information measures how far X and Y are from being independent of one another.

Useful identities



This diagram illustrates some useful relationships among information-related quantities. For all of these identities, assume that we have two random variables X and Y, with domains $\mathcal{X}$ and $\mathcal{Y}$.

- The entropy of the joint distribution is bounded by the entropy of the uniform distribution on $\mathcal{X} \times \mathcal{Y}$.
- The individual entropies of X and Y are bounded by the joint entropy.
- The sum of the individual (marginal) entropies is at least the joint entropy.
- The excess $H(X) + H(Y) - H(X, Y)$ is the mutual information $I(X, Y)$.
- We can split the marginal entropy $H(X)$ into the mutual information $I(X, Y)$ and the conditional entropy $H(X \mid Y)$; similarly for $H(Y)$.
- We can split the joint entropy $H(X, Y)$ into $H(X \mid Y) + I(X, Y) + H(Y \mid X)$.

It's a good exercise to try to derive some of these relationships.

Information in ML

Measures of information turn out to be pretty useful in machine learning. For example, if we are learning how to transform our raw low-level inputs into a new lower-dimensional representation (an unsupervised learning problem), we could ask the low-d representation to be highly informative about the input data. Or, if we're planning to use the learned representation later on in some supervised classification problems, we could ask that it be highly informative about the class labels.

Maxent

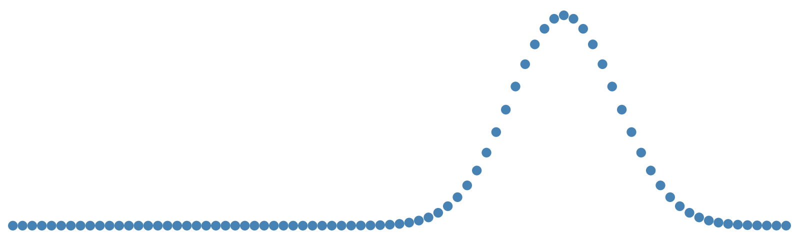
A general and popular learning principle based on information is called *maximum entropy* or *maxent*. This principle says that, if we're trying to learn a probability distribution,

> [Maxent]: We should pick the distribution that has the highest possible entropy (contains the least possible information), subject to the constraint that it agrees with our existing knowledge.

To implement maxent, we have to decide exactly what agreement constraints to impose; different choices will lead to different learning methods. Typically we pick constraints based on a combination of our prior knowledge and any data that we've observed.

We can interpret the maxent principle as saying that we shouldn't draw any conclusions unless they are supported by the data: if we draw an unnecessary conclusion, it will reduce the entropy of our distribution.

For example, suppose we observe a bunch of integers $x_1, x_2, \ldots, x_N$, all in the range of -40 to 40. Suppose that their observed mean is $\frac{1}{N}\sum_{i=1}^{N} x_i = 17$ and their observed variance is $\frac{1}{N}\sum_{i=1}^{N}(x_i - 17)^2 = 121$. We might ask, what distribution generated this data? To answer this question, maxent tells us to find the maximum entropy distribution over $\{-40, \ldots, 40\}$ that agrees with some constraints. Based on the above observations, we can constrain the distribution's mean and variance to match the observed values of 17 and 121. Here's the result:



Above, each dot is one possible random sample: the x position is an integer between -40 and 40, and the y position is proportional to the probability of that integer being sampled.

This plot should look familiar — more on that below.

It's a good exercise to calculate this distribution yourself: write p_{-40} for the probability that x_i is -40, p_{-39} for the probability that x_i is -39, and so forth. We want to maximize the entropy $\sum_{x=-40}^{40} p_x \ln p_x$, subject to the constraints that

- the distribution sums to 1: $\sum_{x=-40}^{40} p_x = 1$
- the mean is correct: $\sum_{x=-40}^{40} xp_x = 17$
- the variance is correct: $\sum_{x=-40}^{40}(x-17)^2 p_x = 121$

To do this maximization, we can use three Lagrange multipliers for the three constraints, and then set the derivative with respect to each p_x to zero. The resulting equations will not have an analytic solution, but we can easily solve them numerically.

A more general version of maxent is *minimum relative entropy*: we specify a *base* distribution, which is the answer that we will conclude in the absence of data. We then require:

> [Minimum relative entropy]: We should pick a distribution that is as close as possible to the base distribution (has the smallest possible relative entropy), subject to the constraint that it agrees with our existing knowledge.

If our base distribution is uniform, then minimum relative entropy coincides with maxent.

Maxent and MLE/MAP

You might notice that the above plot looks a lot like a Gaussian distribution, except that it's restricted to the integers. In fact, if we had done the same exercise with continuous observations, we would have gotten exactly a Gaussian distribution as our answer. This is not an accident: maxent and MLE are exactly equivalent for many common families of distributions. (In fact the equivalence holds for all *exponential families* of distributions, which includes all the textbook examples like Gaussian, Beta, Poisson, etc., though that's beyond our scope.)

If we had used minimum relative entropy instead, we would have gotten MAP inference (under the same assumptions). The base distribution for minimum relative entropy plays the role of the prior for MAP.

Minimum description length

The maximum entropy principle and MLE/MAP are related to another principle called *minimum description length*. MDL connects back to our original view of entropy in terms of cost of communication. It says:

> [Minimum description length]: We should pick the hypothesis that minimizes the combined communication cost (description length) of the hypothesis and data.

Since the communcation cost is a measure of information content, which in turn corresponds to the probability that we assign to each outcome, MDL says that we should pick a hypothesis that

- has high prior probability (so that the hypothesis itself is cheap to communicate)
- gives the data high likelihood (so that the dataset is cheap to communicate given that we've sent they hypothesis)

These criteria make a lot of sense when we compare with previous principles like MAP. But, they bring to the forefront something that might not have been obvious before: every compression algorithm is a learning algorithm. So, for your next ML problem, try gzip as a model!