

Information Theory for Model Selection

Overview

1. Introduction
2. Entropy
3. Joint entropy and divergence
4. Coding
5. Example: Coding and hypothesis tests
6. Minimum description length (MDL)

Introduction

Models and data compression Think of a statistical model $M(Y)$ model as a means to compress data Y_1, \dots, Y_n prior to digital transmission from one computer (“sender”) to another (“receiver”). We want to send this “message” as 0/1’s using as few bits as possible (as little bandwidth as possible). One approach is to use a compression algorithm that encodes the bits in a standard manner such that the receiver can expand the compressed file (Unix offers, for example, the tools `compress` and `gzip` for this purpose). Alternatively, might we be able to use a model for the same goal? (As it turns out, methods like `compress` are based on universal models, models that adapt to the data at hand.)

Model complexity Standardized tools like `compress` shrink a file in a way such that the receiver can uniquely decode the file (lossless coding). If we use a model as part of our compression, we have to (a) agree with the receiver what type of model is being used and (b) transmit the *parameters* of that model to the receiver. The more complex the parametrization, the more parameters we need to send, and thus the more bits that get involved.

Entropy How well can we compress a file? As we shall see, the entropy of the source of the data defines limits for how much a file may be compressed.

Binary data The discussion in this lecture is confined to discrete random variables, particularly binary random variables for which $Y_i \in \{0, 1\}$. It all generalizes in a somewhat natural way to the continuous case (basically, just round things). A good reference is the book *Elements of Information Theory* by Cover and Thomas. The next lecture considers regular data; all of the theory, including how entropy defines limits to data compression, extends nicely to that setting.

Entropy

Definition The entropy of a discrete random variable $X \sim p(x)$ is defined

$$H(X) = - \sum_x p(x) \log_2 p(x) = \mathbb{E} \log_2 \frac{1}{p(x)} \geq 0, \quad (1)$$

where the sum is over x s.t. $p(x) > 0$ (the support of X). The base 2 log is in keeping with our focus on binary data and the number of bits required to transmit a signal; three bits allow us to transmit up to $2^3 = 8$ different messages.

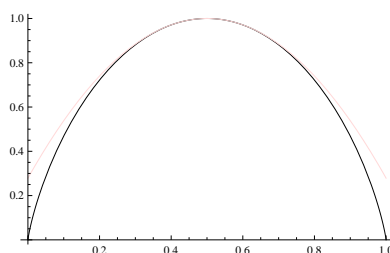
Coin-toss example Define the Boolean random variable

$$X = \begin{cases} 0, & w.p. 1-p \\ 1, & w.p. p \end{cases} \quad 0 \leq p \leq 1. \quad (2)$$

Then the entropy of X when $p = \frac{1}{2}$ is

$$\begin{aligned} H(X) &= -p \log_2 p - (1-p) \log_2 (1-p) \\ &\leq -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \\ &= -\log_2 \frac{1}{2} = 1 \end{aligned}$$

Thus, the entropy of a fair coin tosses is 1 (bit), and this is the maximum entropy for a Boolean random variable. This plot graphs $H(X) = H(p)$ as a function of p associated with the random variable X ; $H(p) = 0$ is zero when $p = 0, 1$ and concave.



The lightly shaded curve is a quadratic approximation to the entropy,

$$H(p) \leq 1 - \frac{2(p - \frac{1}{2})^2}{\log(2)} \tag{3}$$

Four-values example Define the random variable Y taking values

$$Y = \begin{cases} 0, & w.p. 1/2 \\ 1, & 1/4 \\ 2, & 1/8 \\ 3, & 1/8 \end{cases} \tag{4}$$

Then the entropy of Y is

$$\begin{aligned} H(Y) &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - 2 \frac{1}{8} \log_2 \frac{1}{8} \\ &= 1/2 + 2/4 + 6/8 \\ &= 1.75 \text{ bits} \end{aligned}$$

Entropy and data compression How many bits does it take to encode the value of Y ? In other words, how many “yes/no” questions does it take to determine the value of Y , and what ought these questions be? The answer on average is that the best sequence of questions takes $H(Y) = 1.75$ questions. What sequence of questions would you ask? (A method of coding known as Huffman coding produces a binary tree that goes with this sequence.)

Not like variance $H(X)$ is defined by the values of the density, and does not depend on where you locate the x s. Thus the random variables X defined in (2) and $100X$ have the same entropy. You don’t even need for X to be numerical; the values of X might be letters of the alphabet instead.

Joint Entropy and Divergence

Definition The joint entropy of two random variables X and Y is

$$H(X, Y) = \mathbb{E} \frac{1}{p(x, y)} = - \sum_{x, y} p(x, y) \log_2 p(x, y), \quad (5)$$

where the sum is over the support of the joint distribution.

Conditional entropy The conditional entropy is what a statistician might call instead the average conditional entropy,

$$\begin{aligned} H(Y|X) &= \mathbb{E} H(Y|X = x) \\ &= \sum_x p(x) \left(- \sum_y p(y|x) \log_2 p(y|x) \right) \\ &= -\mathbb{E}_{X, Y} \log_2 p(y|x) \end{aligned}$$

Picture Venn diagrams work very nicely for joint/conditional entropy. Let two sets represent $H(X)$ and $H(Y)$. Then $H(X) - H(Y) = H(X|Y)$, the randomness in X given Y is known. The intersection has two representations,

$$H(X) \cap H(Y) = H(X) - H(X|Y) = H(Y) - H(Y|X), \quad (6)$$

but what is this intersection?

Mutual information The “information” in one random variable X which is “shared” by another Y (the intersection in equation 6) is called the mutual information,

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) \\ &= - \sum_x p(x) \log_2 p(x) + \sum_{x, y} p(x, y) \log_2 p(x|y) \\ &= - \sum_{x, y} p(x, y) \log_2 p(x) + \sum_{x, y} p(x, y) \log_2 p(x|y) \\ &= \sum_{x, y} p(x, y) \log_2 \left(\frac{p(x|y) p(y)}{p(x) p(y)} \right) \\ &= \sum_{x, y} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right) \end{aligned}$$

$$= \mathbb{E}_{X,Y} \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right),$$

which is the expected likelihood ratio of the joint distribution to the product of the marginals.

Kullback-Leibler divergence The Kullback-Liebler divergence of some density q from the “true” density p is defined as

$$D(p \parallel q) = \sum_x p(x) \log_2 \left(\frac{p(x)}{q(x)} \right) = \mathbb{E} \log_2 \left(\frac{p(x)}{q(x)} \right). \quad (7)$$

D is *not* symmetric and hence is not a distance function as required in a metric space. It is, however, always nonnegative,

$$\begin{aligned} D(p \parallel q) &= \sum_x p(x) \log_2 \left(\frac{p(x)}{q(x)} \right) \\ &= \sum_x p(x) \left(-\log_2 \left(\frac{q(x)}{p(x)} \right) \right) \\ &\geq -\log_2 \left(p(x) \left(\frac{q(x)}{p(x)} \right) \right) \\ &= 0, \end{aligned}$$

since the function $-\log_2$ is convex.

Mutual information = divergence from independence Returning to $I(X|Y)$, we find that

$$I(X; Y) = D(p(x, y) \parallel p(x)p(y)),$$

so that the mutual information can be thought of as the distance of the joint distribution from the product of the marginals. *i.e.*, how “far” is the pair (X, Y) from independence.

Coding

Prefix codes Suppose we assign a codeword (a sequence of bits that identify an item) with length $L(y)$ to each value y of the random variable Y defined in (4):

Prob	Value	Codeword	$L(y)$	$\mathbb{E} L(Y)$
1/2	0	0	1	1/2
1/4	1	10	2	1/2
1/8	2	110	3	3/8
1/8	3	111	3	3/8
Total			1.75 = $H(Y)$	

The code lengths are $L(x) = -\log_2 p(x)$. For example, the sequence

$$023102 \Rightarrow 0 \ 110 \ 111 \ 10 \ 0 \ 110$$

Notice that the spaces separating the codewords are not needed. The code is “self-punctuating” in that we can uniquely parse the result without the separators,

$$0110111100110 \Rightarrow 023102$$

Such a self-punctuating code is known as a prefix code. (It’s really a “prefix-free” code. No codeword is a prefix of any other codeword; the codes are instantaneous.)

Goal of compression The holy grail of compression is to find the shortest prefix code. One does this by assigning short codewords to items of high probability. The most common values get the short codewords. One can show (see Cover and Thomas, Chapter 5) that the expected code length is at least the entropy,

$$\mathbb{E} L(X) \geq H(X) ,$$

with equality in special cases (such as Y above) in which $p(x) = 2^{-L(x)}$. For such codes the expected code length is the entropy,

$$\mathbb{E} L(X) = \sum_x p(x)(-\log_2 p(x)) = H(X) .$$

Block coding, arithmetic coding In general, probabilities associated with random variables do not have the special form as powers of two, $p(x) = 2^{-L(x)}$ so that one cannot achieve the entropy bound. (Think of a Boolean with $p \approx 1$ for which $H \approx 0$. If we have to encode value of

the random variable one-at-a-time, the compression is lousy. However, if we instead encode long blocks of length n values together, we can get quite close to this limit.

Alternatively, one can use a technique known as arithmetic coding to produce a code that comes close to the entropy limit without imposing blocks. Arithmetic coding uses the cumulative probability distribution to map the input symbols into a list of bits.

Effect of wrong probability distribution Suppose that we think that the random variable to be compressed has density q , but the density is really p . What's the impact of using the wrong codewords (*i.e.*, using $L(x) = -\log_2 q(x)$ bits)? The expected length of the incorrect code is:

$$\begin{aligned} \mathbb{E} L(X) &= \sum_x p(x) \left\lceil \log_2 \frac{1}{q(x)} \right\rceil \\ &< \sum_x p(x) \left(1 + \log_2 \frac{1}{q(x)} \right) \\ &= \sum_x p(x) \left(1 + \log_2 \frac{p(x)}{q(x)} \frac{1}{p(x)} \right) \\ &= 1 + H(X) + D(p \parallel q) . \end{aligned}$$

Thus, the effect of assuming the wrong “model” (here, a probability density) is the Kullback-Leibler divergence between the distributions, the expected value of the log of the likelihood ratio.

Example: Transmitting a Boolean String

Problem Consider the problem of sending a string of n binary digits using a few bits as possible, where each digit in the string is a realized value of a sample of iid random variables $X_i \stackrel{\text{iid}}{\sim} \text{Bool}(p)$, $i = 1, \dots, n$.

Ideal code The ideal code for this compression uses approximately

$$n H(p) \leq n$$

bits to transmit the string. If $p = \frac{1}{2}$, the compressed string is simply the original data; no compression occurs. This table shows how much

compression occurs for various values of p : (*i.e.*, its a table of the entropy function graphed previously)

p	$H(p)$
0.0001	0.0015
0.0010	0.0114
0.0039	0.0369
0.01	0.0808
0.1	0.4690
0.2	0.7219
0.3	0.8813
0.4	0.9710
0.5	1.

Thus, if $n = 100$ and $p = 0.1$, we save over half of the original length, needing only 47 bits to send the sequence (on average).

A tangent... Approximating the ideal code The optimal code when $p \approx 1/n$ or $(n-1)/n$ roughly sends the index of the “rare” symbol. That is, tell the receiver where the exception occurs. For example, with $p = \frac{1}{256} \approx 0.0039$ and $n = 256$ (8 bits long), we expect to see one 1. Assuming the receiver knows our coding scheme and n , we transmit the locations of any 1s that occur. If one 1 shows up, then it only takes 9 bits to send the whole message: 8 to send the index and 1 to say there’s no more. Notice that $256 H(1/256) \approx 9.4$.

Unknown parameters Suppose we don’t know p . We can estimate p in the probability distribution using the MLE \hat{p} ... but should we bother? We have two choices. If \hat{p} is near 0.5, then why not just send the string as is, treating them as fair coin tosses; there’s not much gain by compression anyway. If \hat{p} is “far” from 0.5, then we could find the optimal code and obtain expected length approximately $nH(\hat{p})$.

However, if we use \hat{p} to encode the data, then we have to tell the receiver which value we’ve used so that the receiver can decode the sequence. (That is, the optimal code is uniquely determined using a standard algorithm from the value of \hat{p} .) Our choices are:

$$\begin{aligned} \hat{p} \text{ near } 0.5 &\implies \text{Treat as iid} &\implies n \text{ bits} \\ \hat{p} \text{ far from } 0.5 &\implies \text{Use associated code} &\implies nH(\hat{p}) + \text{code for } \hat{p} \end{aligned}$$

What's close and what's far? How far need \hat{p} lie from 0.5 in order to warrant our attention? Recall that the standard error of \hat{p} is no more than

$$SE(\hat{p}) \leq \frac{1}{2\sqrt{n}}.$$

Looking back at our approximation (3) to the entropy of a Boolean, we see that we save b bits in transmitting x_1, \dots, x_n if (write $\hat{p} = 0.5 + \epsilon$)

$$nH(\hat{p}) \approx n \left(1 - \frac{2}{\log 2} \epsilon^2 \right) \leq n - b, \tag{8}$$

To save b bits when compressing the data, $|\hat{p} - \frac{1}{2}|$ needs to be more than

$$\epsilon \geq \sqrt{\frac{b \log 2}{2n}} = \sqrt{2b \log 2} SE(\hat{p})$$

(using $SE(\hat{p}) = 1/(2\sqrt{n})$). Here is a table for the constant term:

bits saved	SEs
b	$\sqrt{2b \log 2}$
1	1.18
2	1.67
3	2.04
4	2.35
8	3.33
16	4.71

Implications for model selection To save one bit in encoding x_1, \dots, x_n , \hat{p} has to be more than one standard error away from $\frac{1}{2}$. That means that we only get to use one bit to encode \hat{p} or else we have lost by encoding the data. That's only enough to tell you the sign of $\hat{p} - \frac{1}{2}$. In order to have 16 bits to encode the parameter estimate, \hat{p} needs to be almost 5 standard errors away from $\frac{1}{2}$.

Minimum Description Length Criterion — MDL

MDL for two models Which model should we use in the previous example? The model of fair coin tosses or a model which says p differs from $\frac{1}{2}$? This amounts to a simple hypothesis test of $H_0 : p = \frac{1}{2}$ versus $H_1 : p \neq \frac{1}{2}$. The MDL rule for choosing between these is to pick by minimizing the following:

Hypothesis	Bits to Transmit
H_0	n
H_1	$L(\hat{p}) + nH(\hat{p})$

where $L(p)$ denotes the number of bits used to encode the parameter value.

Parameter coding In his definition of the MDL principle, Rissanen (1989,2008, *Stochastic Complexity in Statistical Inquiry*), Rissanen encodes p assuming in effect that all values are equally likely (a uniform prior).

For example, suppose we decide to encode \hat{p} to within $\frac{1}{n}$ of the observed value (*i.e.*, tell the receiver exactly how many 1's and 0's lie in the data sequence). Then the net message length is

$$nH(\hat{p}) + L(\hat{p}, 1/n) = nH(\hat{p}) + \log_2 n$$

For us to choose to encode the parameter value (choose hypothesis H_1), \hat{p} must be quite far from $\frac{1}{2}$. With $n = 256$, for example, we need to save 8 bits, implying that \hat{p} needs to be more than 3.3 standard errors away from $\frac{1}{2}$. The penalty becomes quite steep as n increases, and we seldom choose H_1 . (This phenomenon is known as the Lindley paradox in Bayesian inference. Bayesians behave like MDL when “testing” a hypothesis, making it very hard to reject the null for large n .)

Alternatively, we can encode a rounded parameter estimate. Suppose we round \hat{p} to within $\frac{1}{\sqrt{n}}$ — the scale of its standard error. In effect we are transmitting a **z-score** for the parameter estimate. Now the encoded length is

$$n(H(\hat{p}) + D(\hat{p}||\bar{p})) + L_R(\hat{p}, 1/\sqrt{n}) = n(H(\hat{p}) + D(\hat{p}||\bar{p})) + \underbrace{\frac{1}{2} \log_2 n}$$

where \bar{p} denotes the rounded parameter estimate. The term $D(\hat{p} \parallel \bar{p})$ arises since we use the rounded value of \hat{p} to compress the sequence. This divergence is quite small:

$$\begin{aligned}
 D(\hat{p} \parallel \bar{p}) &= \hat{p} \log_2 \frac{\hat{p}}{\bar{p}} + (1 - \hat{p}) \log_2 \frac{1 - \hat{p}}{1 - \bar{p}} \\
 &= \hat{p} \log_2 \frac{\hat{p}}{\hat{p} + \delta} + (1 - \hat{p}) \log_2 \frac{1 - \hat{p}}{1 - \hat{p} - \delta} \quad \text{for } \delta = \bar{p} - \hat{p} \leq O(1/\sqrt{4n}) \\
 &= \hat{p} \log_2 \frac{1}{1 + \delta/\hat{p}} + (1 - \hat{p}) \log_2 \frac{1}{1 - \delta/(1 - \hat{p})} \\
 &\approx \frac{\delta^2/\hat{p} + \delta^2/(1 - \hat{p})}{\log 2} \\
 &= \frac{\delta^2}{\hat{p}(1 - \hat{p}) \log 2} \\
 &\leq \frac{1}{4n\hat{p}(1 - \hat{p}) \log 2} \quad \text{which is } \left(\text{var} \left(\sum_i X_i \right)^2 \log 2 \right)^{-1} \\
 &\approx \frac{1}{n \log 2}
 \end{aligned} \tag{9}$$

This contributes about 1 bit to the total length of the code. Now to choose H_1 with $n = 256$ we need only observe \hat{p} more than 2.35 SE's away from $\frac{1}{2}$ (check with D). In either case, we put a steep penalty on the transmission of the parameter \hat{p} . Coding with \hat{p} rounded to accuracy $1/\sqrt{n}$ is part of Rissanen's motivation for MDL.

Local coding A different parameter coding strategy reconsiders the problem. Since we seek to distinguish $p = \frac{1}{2}$ versus $p \neq \frac{1}{2}$, we need to be most careful near $p = \frac{1}{2}$. Implicitly if we are really trying to separate H_0 from H_1 , then we must consider $p = \frac{1}{2}$ to be quite plausible (starting to sound Bayesian?). Since $p = \frac{1}{2}$ is special, let's devote the short codes to the values near $\frac{1}{2}$.

Consider a z -score scaling. In order to concentrate the probabilities near $\frac{1}{2}$, consider a "prior" on z -scores rounded to integers which resembles a Cauchy density (watch for the double use of π)

$$\pi(z) = \pi \left(\frac{\hat{p} - \frac{1}{2}}{\text{SE}(\hat{p})} \right) = \frac{3}{\pi^2} \frac{1}{z^2}$$

where the normalizing constant is determined by knowing that $\sum_{i=1}^{\infty} 1/i^2 =$

$\pi^2/6$. Since the sum is only over $z = \pm 1, \pm 2, \dots, \pm\sqrt{n}/2$, this expression is really an approximation, but it's good enough since

$$\begin{aligned} \sum_{i=1}^{\sqrt{n}/2} \frac{1}{i^2} &= \frac{\pi^2}{6} - \sum_{i=\sqrt{n}/2+1}^{\infty} \frac{1}{i^2} \\ &= \frac{\pi^2}{6} - \int_{i=\sqrt{n}/2+1}^{\infty} \frac{1}{i^2} di \\ &= \frac{\pi^2}{6} + \frac{1}{i} \Big|_{i=\sqrt{n}/2+1}^{\infty} \\ &\approx \frac{\pi^2}{6} - \frac{2}{\sqrt{n}} \end{aligned}$$

The entropy of this discrete density (again, the z-score is the number of SE's away from $\frac{1}{2}$ and we sum to infinity) is

$$\begin{aligned} H(\pi) &= - \sum_{z \neq 0} p(z) \log_2 p(z) \\ &= \frac{3}{\pi^2} \left(\sum_{z \neq 0} \frac{1}{z^2} \log_2 \left(\frac{3z^2}{\pi^2} \right) \right) \end{aligned} \tag{10}$$

$$= \frac{6}{\pi^2} \sum_{z \neq 0} \frac{\log_2 |z|}{z^2} + \log_2 \frac{3}{\pi^2} \tag{11}$$

$$\begin{aligned} &= \frac{12}{\pi^2 \log 2} \int_1^{\infty} \frac{\log z}{z^2} dz + \log_2 \frac{3}{\pi^2} \\ &= \frac{12}{\pi^2 \log 2} + \log_2 \frac{3}{\pi^2} \end{aligned} \tag{12}$$

Most importantly, the entropy is **not a function of n** .

Comparison to uniform code length Both implementations of *MDL* lose the same amount $D(\hat{p} \parallel \bar{p})$ due to rounding, so this part is identical. The difference lies in the “penalty” for coding the parameter. For example, here are examples of the two supposing \hat{p} takes on different values with $n = 256$. $SE(\hat{p}) \approx 1/32 = 0.03$ for $p \approx \frac{1}{2}$.

z	Uniform $\frac{1}{2}\log_2 n$	Local $-\log_2 \pi(z)$
1	7	1.718
2	7	3.718
3	7	4.888
4	7	5.718
5	7	6.362
6	7	6.888
7	7	7.333
8	7	7.718
9	7	8.058
10	7	8.362

Thus, local coding is shorter unless \hat{p} is about 7 standard errors away from $\frac{1}{2}$. Thus, local coding rejects H_0 for values of \hat{p} closer to $\frac{1}{2}$.

MDL reference For an introduction to the topic, see the first chapter of the book *Minimum Description Length* by Grunwald, Myung, and Pitt.