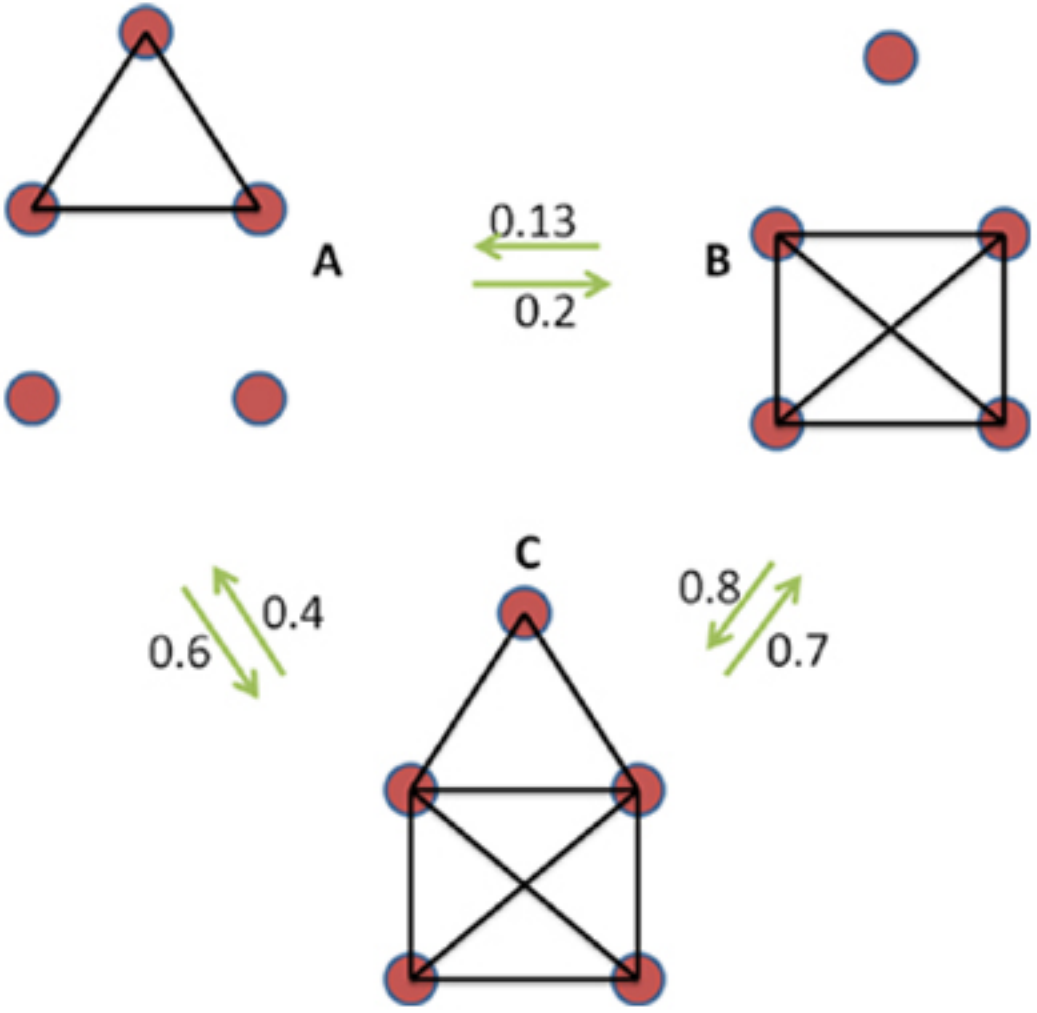
Important Concepts and Subfields of Information Theory



Sheree Tolbert
Hamza Benavides

First Edition, 2012

ISBN 978-81-323-1337-3

Published by:
College Publishing House
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Introduction to Information Theory

Information theory is a branch of applied mathematics and electrical engineering involving the quantification of information. Information theory was developed by Claude E. Shannon to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data. Since its inception it has broadened to find applications in many other areas, including statistical inference, natural language processing, cryptography generally, networks other than communication networks — as in neurobiology, the evolution and function of molecular codes, model selection in ecology, thermal physics, quantum computing, plagiarism detection and other forms of data analysis.

A key measure of information is known as entropy, which is usually expressed by the average number of bits needed for storage or communication. Entropy quantifies the uncertainty involved in predicting the value of a random variable. For example, specifying the outcome of a fair coin flip (two equally likely outcomes) provides less information (lower entropy) than specifying the outcome from a roll of a die (six equally likely outcomes).

Applications of fundamental topics of information theory include lossless data compression (e.g. ZIP files), lossy data compression (e.g. MP3s and JPGs), and channel coding (e.g. for DSL lines). The field is at the intersection of mathematics, statistics, computer science, physics, neurobiology, and electrical engineering. Its impact has been crucial to the success of the Voyager missions to deep space, the invention of the compact disc, the feasibility of mobile phones, the development of the Internet, the study of linguistics and of human perception, the understanding of black holes, and numerous other fields. Important sub-fields of information theory are source coding, channel coding, algorithmic complexity theory, algorithmic information theory, information-theoretic security, and measures of information.

Overview

The main concepts of information theory can be grasped by considering the most widespread means of human communication: language. Two important aspects of a concise language are as follows: First, the most common words (e.g., "a", "the", "I") should be shorter than less common words (e.g., "benefit", "generation", "mediocre"), so that sentences will not be too long. Such a tradeoff in word length is analogous to data

compression and is the essential aspect of source coding. Second, if part of a sentence is unheard or misheard due to noise — e.g., a passing car — the listener should still be able to glean the meaning of the underlying message. Such robustness is as essential for an electronic communication system as it is for a language; properly building such robustness into communications is done by channel coding. Source coding and channel coding are the fundamental concerns of information theory.

Note that these concerns have nothing to do with the *importance* of messages. For example, a platitude such as "Thank you; come again" takes about as long to say or write as the urgent plea, "Call an ambulance!" while the latter may be more important and more meaningful in many contexts. Information theory, however, does not consider message importance or meaning, as these are matters of the quality of data rather than the quantity and readability of data, the latter of which is determined solely by probabilities.

Information theory is generally considered to have been founded in 1948 by Claude Shannon in his seminal work, "A Mathematical Theory of Communication". The central paradigm of classical information theory is the engineering problem of the transmission of information over a noisy channel. The most fundamental results of this theory are Shannon's source coding theorem, which establishes that, on average, the number of *bits* needed to represent the result of an uncertain event is given by its entropy; and Shannon's noisy-channel coding theorem, which states that *reliable* communication is possible over *noisy* channels provided that the rate of communication is below a certain threshold, called the channel capacity. The channel capacity can be approached in practice by using appropriate encoding and decoding systems.

Information theory is closely associated with a collection of pure and applied disciplines that have been investigated and reduced to engineering practice under a variety of rubrics throughout the world over the past half century or more: adaptive systems, anticipatory systems, artificial intelligence, complex systems, complexity science, cybernetics, informatics, machine learning, along with systems sciences of many descriptions. Information theory is a broad and deep mathematical theory, with equally broad and deep applications, amongst which is the vital field of coding theory.

Coding theory is concerned with finding explicit methods, called *codes*, of increasing the efficiency and reducing the net error rate of data communication over a noisy channel to near the limit that Shannon proved is the maximum possible for that channel. These codes can be roughly subdivided into data compression (source coding) and error-correction (channel coding) techniques. In the latter case, it took many years to find the methods Shannon's work proved were possible. A third class of information theory codes are cryptographic algorithms (both codes and ciphers). Concepts, methods and results from coding theory and information theory are widely used in cryptography and cryptanalysis.

Information theory is also used in information retrieval, intelligence gathering, gambling, statistics, and even in musical composition.

Historical background

The landmark event that established the discipline of information theory, and brought it to immediate worldwide attention, was the publication of Claude E. Shannon's classic paper "A Mathematical Theory of Communication" in the *Bell System Technical Journal* in July and October 1948.

Prior to this paper, limited information-theoretic ideas had been developed at Bell Labs, all implicitly assuming events of equal probability. Harry Nyquist's 1924 paper, *Certain Factors Affecting Telegraph Speed,* contains a theoretical section quantifying "intelligence" and the "line speed" at which it can be transmitted by a communication system, giving the relation $W = K\log m$, where W is the speed of transmission of intelligence, m is the number of different voltage levels to choose from at each time step, and K is a constant. Ralph Hartley's 1928 paper, *Transmission of Information,* uses the word *information* as a measurable quantity, reflecting the receiver's ability to distinguish one sequence of symbols from any other, thus quantifying information as $H = \log S^n = n\log S$, where S was the number of possible symbols, and n the number of symbols in a transmission. The natural unit of information was therefore the decimal digit, much later renamed the hartley in his honour as a unit or scale or measure of information. Alan Turing in 1940 used similar ideas as part of the statistical analysis of the breaking of the German second world war Enigma ciphers.

Much of the mathematics behind information theory with events of different probabilities was developed for the field of thermodynamics by Ludwig Boltzmann and J. Willard Gibbs. Connections between information-theoretic entropy and thermodynamic entropy, including the important contributions by Rolf Landauer in the 1960s, are explored in *Entropy in thermodynamics and information theory*.

In Shannon's revolutionary and groundbreaking paper, the work for which had been substantially completed at Bell Labs by the end of 1944, Shannon for the first time introduced the qualitative and quantitative model of communication as a statistical process underlying information theory, opening with the assertion that

> "The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point."

With it came the ideas of

- the information entropy and redundancy of a source, and its relevance through the source coding theorem;
- the mutual information, and the channel capacity of a noisy channel, including the promise of perfect loss-free communication given by the noisy-channel coding theorem;
- the practical result of the Shannon–Hartley law for the channel capacity of a Gaussian channel; as well as
- the bit—a new way of seeing the most fundamental unit of information.

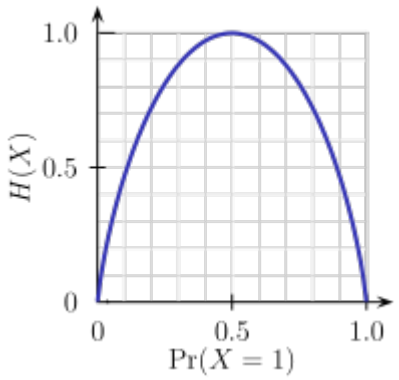
Quantities of information

Information theory is based on probability theory and statistics. The most important quantities of information are entropy, the information in a random variable, and mutual information, the amount of information in common between two random variables. The former quantity indicates how easily message data can be compressed while the latter can be used to find the communication rate across a channel.

The choice of logarithmic base in the following formulae determines the unit of information entropy that is used. The most common unit of information is the bit, based on the binary logarithm. Other units include the nat, which is based on the natural logarithm, and the hartley, which is based on the common logarithm.

In what follows, an expression of the form $p \log p$ is considered by convention to be equal to zero whenever $p = 0$. This is justified because $\lim_{p \to 0+} p \log p = 0$ for any logarithmic base.

Entropy



Entropy of a Bernoulli trial as a function of success probability, often called the **binary entropy function**, $H_b(p)$. The entropy is maximized at 1 bit per trial when the two possible outcomes are equally probable, as in an unbiased coin toss.

The **entropy**, H, of a discrete random variable X is a measure of the amount of *uncertainty* associated with the value of X.

Suppose one transmits 1000 bits (0s and 1s). If these bits are known ahead of transmission (to be a certain value with absolute probability), logic dictates that no information has been transmitted. If, however, each is equally and independently likely to be 0 or 1, 1000 bits (in the information theoretic sense) have been transmitted. Between these two extremes, information can be quantified as follows. If $\mathbb{X}$ is the set of all messages $\{x_1,...,x_n\}$ that X could be, and $p(x)$ is the probability of X given some $x \in \mathbb{X}$, then the entropy of X is defined:

$$H(X) = \mathbb{E}_X[I(x)] = -\sum_{x \in \mathbb{X}} p(x) \log p(x).$$

(Here, $I(x)$ is the self-information, which is the entropy contribution of an individual message, and $\mathbb{E}_X$is the expected value.) An important property of entropy is that it is maximized when all the messages in the message space are equiprobable $p(x) = 1 / n$,—i.e., most unpredictable—in which case $H(X) = \log n$.

The special case of information entropy for a random variable with two outcomes is the **binary entropy function**, usually taken to the logarithmic base 2:

$$H_{\mathrm{b}}(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

Joint entropy

The **joint entropy** of two discrete random variables X and Y is merely the entropy of their pairing: (X,Y). This implies that if X and Y are independent, then their joint entropy is the sum of their individual entropies.

For example, if (X,Y) represents the position of a chess piece — X the row and Y the column, then the joint entropy of the row of the piece and the column of the piece will be the entropy of the position of the piece.

$$H(X, Y) = \mathbb{E}_{X,Y}[-\log p(x, y)] = -\sum_{x,y} p(x, y) \log p(x, y)$$

Despite similar notation, joint entropy should not be confused with **cross entropy**.

Conditional entropy (equivocation)

The **conditional entropy** or **conditional uncertainty** of X given random variable Y (also called the **equivocation** of X about Y) is the average conditional entropy over Y:

$$H(X|Y) = \mathbb{E}_Y[H(X|y)] = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y) = -\sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(y)}.$$

Because entropy can be conditioned on a random variable or on that random variable being a certain value, care should be taken not to confuse these two definitions of conditional entropy, the former of which is in more common use. A basic property of this form of conditional entropy is that:

$$H(X|Y) = H(X, Y) - H(Y).$$

Mutual information (transinformation)

Mutual information measures the amount of information that can be obtained about one random variable by observing another. It is important in communication where it can be used to maximize the amount of information shared between sent and received signals. The mutual information of *X* relative to *Y* is given by:

$$I(X;Y) = \mathbb{E}_{X,Y}[SI(x,y)] = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$

where *SI* (*S*pecific mutual *I*nformation) is the pointwise mutual information.

A basic property of the mutual information is that

$$I(X;Y) = H(X) - H(X|Y).$$

That is, knowing *Y*, we can save an average of *I*(*X*;*Y*) bits in encoding *X* compared to not knowing *Y*.

Mutual information is symmetric:

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,Y).$$

Mutual information can be expressed as the average Kullback–Leibler divergence (information gain) of the posterior probability distribution of *X* given the value of *Y* to the prior distribution on *X*:

$$I(X;Y) = \mathbb{E}_{p(y)}[D_{\mathrm{KL}}(p(X|Y=y)\|p(X))].$$

In other words, this is a measure of how much, on the average, the probability distribution on *X* will change if we are given the value of *Y*. This is often recalculated as the divergence from the product of the marginal distributions to the actual joint distribution:

$$I(X;Y) = D_{\mathrm{KL}}(p(X,Y)\|p(X)p(Y)).$$

Mutual information is closely related to the log-likelihood ratio test in the context of contingency tables and the multinomial distribution and to Pearson's χ^2 test: mutual information can be considered a statistic for assessing independence between a pair of variables, and has a well-specified asymptotic distribution.

Kullback–Leibler divergence (information gain)

The **Kullback–Leibler divergence** (or **information divergence**, **information gain**, or **relative entropy**) is a way of comparing two distributions: a "true" probability distribution *p(X)*, and an arbitrary probability distribution *q(X)*. If we compress data in a manner that assumes *q(X)* is the distribution underlying some data, when, in reality, *p(X)* is the correct distribution, the Kullback–Leibler divergence is the number of average additional bits per datum necessary for compression. It is thus defined

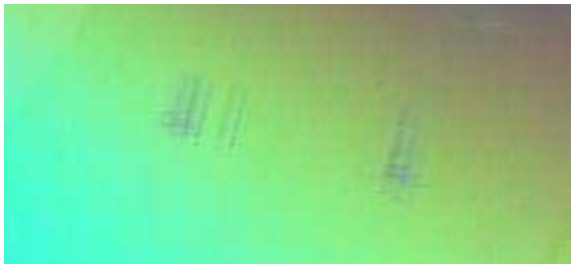
$$D_{\mathrm{KL}}(p(X)\|q(X)) = \sum_{x\in X} -p(x)\log q(x) - (-p(x)\log p(x)) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)}.$$

Although it is sometimes used as a 'distance metric', it is not a true metric since it is not symmetric and does not satisfy the triangle inequality (making it a semi-quasimetric).

Other quantities

Other important information theoretic quantities include Rényi entropy, (a generalization of entropy,) differential entropy, (a generalization of quantities of information to continuous distributions,) and the conditional mutual information.

Coding theory



A picture showing scratches on the readable surface of a CD-R. Music and data CDs are coded using error correcting codes and thus can still be read even if they have minor scratches using error detection and correction.

Coding theory is one of the most important and direct applications of information theory. It can be subdivided into source coding theory and channel coding theory. Using a statistical description for data, information theory quantifies the number of bits needed to describe the data, which is the information entropy of the source.

- Data compression (source coding): There are two formulations for the compression problem:

1. lossless data compression: the data must be reconstructed exactly;

2. lossy data compression: allocates bits needed to reconstruct the data, within a specified fidelity level measured by a distortion function. This subset of Information theory is called rate–distortion theory.

- Error-correcting codes (channel coding): While data compression removes as much redundancy as possible, an error correcting code adds just the right kind of redundancy (i.e., error correction) needed to transmit the data efficiently and faithfully across a noisy channel.

This division of coding theory into compression and transmission is justified by the information transmission theorems, or source–channel separation theorems that justify the use of bits as the universal currency for information in many contexts. However, these theorems only hold in the situation where one transmitting user wishes to communicate to one receiving user. In scenarios with more than one transmitter (the multiple-access channel), more than one receiver (the broadcast channel) or intermediary "helpers" (the relay channel), or more general networks, compression followed by transmission may no longer be optimal. Network information theory refers to these multi-agent communication models.

Source theory

Any process that generates successive messages can be considered a **source** of information. A memoryless source is one in which each message is an independent identically-distributed random variable, whereas the properties of ergodicity and stationarity impose more general constraints. All such sources are stochastic. These terms are well studied in their own right outside information theory.

Rate

Information **rate** is the average entropy per symbol. For memoryless sources, this is merely the entropy of each symbol, while, in the case of a stationary stochastic process, it is

$$r = \lim_{n \to \infty} H(X_n | X_{n-1}, X_{n-2}, X_{n-3}, \ldots);$$

that is, the conditional entropy of a symbol given all the previous symbols generated. For the more general case of a process that is not necessarily stationary, the *average rate* is

$$r = \lim_{n \to \infty} \frac{1}{n} H(X_1, X_2, \ldots X_n);$$

that is, the limit of the joint entropy per symbol. For stationary sources, these two expressions give the same result.

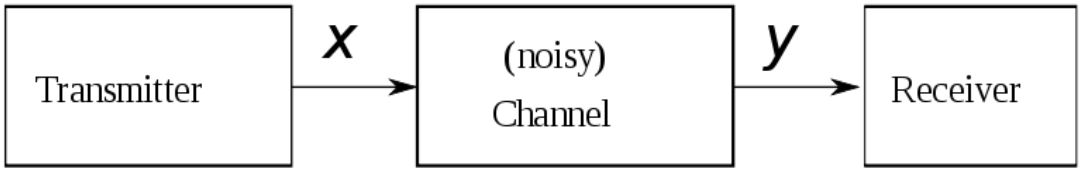
It is common in information theory to speak of the "rate" or "entropy" of a language. This is appropriate, for example, when the source of information is English prose. The rate of

a source of information is related to its redundancy and how well it can be compressed, the subject of **source coding**.

Channel capacity

Communications over a channel—such as an ethernet cable—is the primary motivation of information theory. As anyone who's ever used a telephone (mobile or landline) knows, however, such channels often fail to produce exact reconstruction of a signal; noise, periods of silence, and other forms of signal corruption often degrade quality. How much information can one hope to communicate over a noisy (or otherwise imperfect) channel?

Consider the communications process over a discrete channel. A simple model of the process is shown below:



Here X represents the space of messages transmitted, and Y the space of messages received during a unit time over our channel. Let $p(y \mid x)$ be the conditional probability distribution function of Y given X. We will consider $p(y \mid x)$ to be an inherent fixed property of our communications channel (representing the nature of the **noise** of our channel). Then the joint distribution of X and Y is completely determined by our channel and by our choice of $f(x)$, the marginal distribution of messages we choose to send over the channel. Under these constraints, we would like to maximize the rate of information, or the **signal**, we can communicate over the channel. The appropriate measure for this is the mutual information, and this maximum mutual information is called the **channel capacity** and is given by:

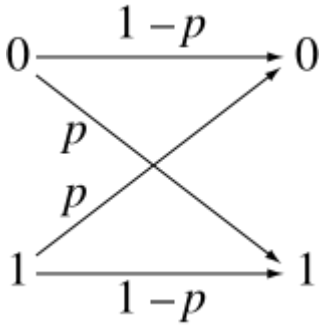
$$C = \max_{f} I(X;Y).$$

This capacity has the following property related to communicating at information rate R (where R is usually bits per symbol). For any information rate $R < C$ and coding error $\varepsilon > 0$, for large enough N, there exists a code of length N and rate $\geq$ R and a decoding algorithm, such that the maximal probability of block error is $\leq \varepsilon$; that is, it is always possible to transmit with arbitrarily small block error. In addition, for any rate $R > C$, it is impossible to transmit with arbitrarily small block error.

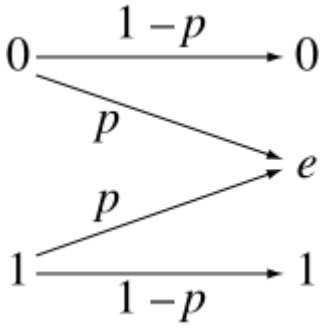
Channel coding is concerned with finding such nearly optimal codes that can be used to transmit data over a noisy channel with a small coding error at a rate near the channel capacity.

Capacity of particular channel models

- A continuous-time analog communications channel subject to Gaussian noise.
- A binary symmetric channel (BSC) with crossover probability p is a binary input, binary output channel that flips the input bit with probability p. The BSC has a capacity of $1 - H_b(p)$ bits per channel use, where H_b is the binary entropy function to the base 2 logarithm:



- A binary erasure channel (BEC) with erasure probability p is a binary input, ternary output channel. The possible channel outputs are *0*, *1*, and a third symbol 'e' called an erasure. The erasure represents complete loss of information about an input bit. The capacity of the BEC is *1* - p bits per channel use.



Applications to other fields

Intelligence uses and secrecy applications

Information theoretic concepts apply to cryptography and cryptanalysis. Turing's information unit, the ban, was used in the Ultra project, breaking the German Enigma machine code and hastening the end of WWII in Europe. Shannon himself defined an important concept now called the unicity distance. Based on the redundancy of the plaintext, it attempts to give a minimum amount of ciphertext necessary to ensure unique decipherability.

Information theory leads us to believe it is much more difficult to keep secrets than it might first appear. A brute force attack can break systems based on asymmetric key algorithms or on most commonly used methods of symmetric key algorithms (sometimes called secret key algorithms), such as block ciphers. The security of all such methods currently comes from the assumption that no known attack can break them in a practical amount of time.

Information theoretic security refers to methods such as the one-time pad that are not vulnerable to such brute force attacks. In such cases, the positive conditional mutual information between the plaintext and ciphertext (conditioned on the key) can ensure proper transmission, while the unconditional mutual information between the plaintext and ciphertext remains zero, resulting in absolutely secure communications. In other words, an eavesdropper would not be able to improve his or her guess of the plaintext by gaining knowledge of the ciphertext but not of the key. However, as in any other cryptographic system, care must be used to correctly apply even information-theoretically secure methods; the Venona project was able to crack the one-time pads of the Soviet Union due to their improper reuse of key material.

Pseudorandom number generation

Pseudorandom number generators are widely available in computer language libraries and application programs. They are, almost universally, unsuited to cryptographic use as they do not evade the deterministic nature of modern computer equipment and software. A class of improved random number generators is termed cryptographically secure pseudorandom number generators, but even they require external to the software random seeds to work as intended. These can be obtained via extractors, if done carefully. The measure of sufficient randomness in extractors is min-entropy, a value related to Shannon entropy through Rényi entropy; Rényi entropy is also used in evaluating randomness in cryptographic systems. Although related, the distinctions among these measures mean that a random variable with high Shannon entropy is not necessarily satisfactory for use in an extractor and so for cryptography uses.

Seismic exploration

One early commercial application of information theory was in the field seismic oil exploration. Work in this field made it possible to strip off and separate the unwanted noise from the desired seismic signal. Information theory and digital signal processing offer a major improvement of resolution and image clarity over previous analog methods.

Chapter 2

Entropy

In information theory, **entropy** is a measure of the uncertainty associated with a random variable. In this context, the term usually refers to the **Shannon entropy**, which quantifies the expected value of the information contained in a message, usually in units such as bits. Equivalently, the Shannon entropy is a measure of the average information content one is missing when one does not know the value of the random variable. The concept was introduced by Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication".

Shannon's entropy represents an absolute limit on the best possible lossless compression of any communication, under certain constraints: treating messages to be encoded as a sequence of independent and identically-distributed random variables, Shannon's source coding theorem shows that, in the limit, the average length of the shortest possible representation to encode the messages in a given alphabet is their entropy divided by the logarithm of the number of symbols in the target alphabet.

A fair coin has an entropy of one bit. However, if the coin is not fair, then the uncertainty is lower (if asked to bet on the next outcome, we would bet preferentially on the most frequent result), and thus the Shannon entropy is lower. Mathematically, a coin flip is an example of a Bernoulli trial, and its entropy is given by the binary entropy function. A long string of repeating characters has an entropy rate of 0, since every character is predictable. The entropy rate of English text is between 1.0 and 1.5 bits per letter, or as low as 0.6 to 1.3 bits per letter, according to estimates by Shannon based on human experiments.

Layman's terms

Entropy is a measure of disorder, or more precisely unpredictability. For example, a series of coin tosses with a fair coin has maximum entropy, since there is no way to predict what will come next. A string of coin tosses with a two-headed coin has zero entropy, since the coin will always come up heads. Most collections of data in the real world lie somewhere in between.

English text has fairly low entropy. In other words, it is fairly predictable. Even if we don't know exactly what is going to come next, we can be fairly certain that, for example, there will be many more e's than z's, or that the combination 'qu' will be much more

common than any other combination with a 'q' in it and the combination 'th' will be more common than any of them. Uncompressed, English text has about one bit of entropy for each byte (eight bits) of message.

If a compression scheme is lossless—that is, you can always recover the entire original message by uncompressing—then a compressed message has the same total entropy as the original, but in fewer bits. That is, it has more entropy per bit. This means a compressed message is more unpredictable, which is why messages are often compressed before being encrypted. Shannon's source coding theorem says (roughly) that a lossless compression scheme cannot compress messages, on average, to have more than one bit of entropy per bit of message. The entropy of a message is in a certain sense a measure of how much information it really contains.

Shannon's theorem also implies that no lossless compression scheme can compress *all* messages. If some messages come out smaller, at least one must come out larger. In the real world, this is not a problem, because we are generally only interested in compressing certain messages, for example English documents as opposed to random bytes, and don't care if our compressor makes random messages larger.

Definition

The entropy H of a discrete random variable X with possible values $\{x_1, ..., x_n\}$ is

$$H(X) = \mathrm{E}(I(X)).$$

Here E is the expected value, and I is the information content of X.

$I(X)$ is itself a random variable. If p denotes the probability mass function of X then the entropy can explicitly be written as

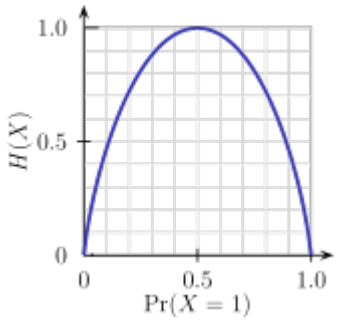
$$H(X) = \sum_{i=1}^{n} p(x_i)\, I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i),$$

where b is the base of the logarithm used. Common values of b are 2, Euler's number e, and 10, and the unit of entropy is bit for $b = 2$, nat for $b = \mathrm{e}$, and dit (or digit) for $b = 10$.

In the case of $p_i = 0$ for some i, the value of the corresponding summand $0 \log_b 0$ is taken to be 0, which is consistent with the limit

$$\lim_{p \to 0+} p \log p = 0.$$

Example



Entropy H(X) (i.e. the expected surprisal) of a coin flip, measured in bits; graphed versus the fairness of the coin Pr(X=1).

Note the maximum of the graph depends on the distribution: Here, at most 1 bit is required to communicate the outcome of a fair coinflip; but the result of a fair die would require at most log2(6) bits.

Consider tossing a coin with known, not necessarily fair, probabilities of coming up heads or tails.

The entropy of the unknown result of the next toss of the coin is maximized if the coin is fair (that is, if heads and tails both have equal probability 1/2). This is the situation of maximum uncertainty as it is most difficult to predict the outcome of the next toss; the result of each toss of the coin delivers a full 1 bit of information.

However, if we know the coin is not fair, but comes up heads or tails with probabilities *p* and *q*, then there is less uncertainty. Every time it is tossed, one side is more likely to come up than the other. The reduced uncertainty is quantified in a lower entropy: on average each toss of the coin delivers less than a full 1 bit of information.

The extreme case is that of a double-headed coin which never comes up tails. Then there is no uncertainty. The entropy is zero: each toss of the coin delivers no information.

Rationale

For a random variable X with n outcomes $\{x_i : i = 1, \ldots, n\}$, the Shannon entropy, a measure of uncertainty (see further below) and denoted by $H(X)$, is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \qquad (1)$$

where $p(x_i)$ is the probability mass function of outcome x_i.

To understand the meaning of Eq. (1), let's first consider a set of n possible outcomes (events) $\{x_i : i = 1, \ldots, n\}$, with equal probability $p(x_i) = 1/n$. An example would be a fair die with n values, from 1 to n. The *uncertainty* for such set of n outcomes is defined by

$$u = \log_b(n). \tag{2}$$

The logarithm is used so to provide the **additivity** characteristic for independent uncertainty. For example, consider appending to each value of the first die the value of a second die, which has m possible outcomes $\{y_j : j = 1, \ldots, m\}$. There are thus mn possible outcomes $\{x_i y_j : i = 1, \ldots, n, j = 1, \ldots, m\}$. The uncertainty for such set of mn outcomes is then

$$u = \log_b(nm) = \log_b(n) + \log_b(m). \tag{3}$$

Thus the uncertainty of playing with two dice is obtained by adding the uncertainty of the second die $\log_b(m)$ to the uncertainty of the first die $\log_b(n)$.

Now return to the case of playing with one die only (the first one). Since the probability of each event is $1/n$, we can write

$$u_i = \log_b\left(\frac{1}{p(x_i)}\right) = -\log_b(p(x_i)),\ \forall i \in \{1, \ldots, n\}.$$

In the case of a non-uniform probability mass function (or density in the case of continuous random variables), we let

$$u_i := -\log_b(p(x_i)) \tag{4}$$

which is also called a surprisal; the lower the probability $p(x_i)$, i.e. $p(x_i) \to 0$, the higher the uncertainty or the surprise, i.e. $u_i \to \infty$, for the outcome x_i.

The average uncertainty $\langle u \rangle$, with $\langle \cdot \rangle$ being the average operator, is obtained by

$$\langle u \rangle = \sum_{i=1}^{n} p(x_i) u_i = -\sum_{i=1}^{n} p(x_i) \log_b(p(x_i)) \tag{5}$$

and is used as the definition of the entropy $H(X)$ in Eq. (1). The above also explained why information *entropy* and information *uncertainty* can be used interchangeably.

One may also define the **conditional entropy** of two events *X* and *Y* taking values x_i and y_j respectively, as

$$H(X|Y) = \sum_{i,j} p(x_i, y_j) \log \frac{p(y_j)}{p(x_i, y_j)}$$

where $p(x_i,y_j)$ is the probability that $X=x_i$ and $Y=_j$. This quantity should be understood as the amount of randomness in the random variable *X* given that you know the value of *Y*. For example, suppose you have a six sided die, the entropy associated to this die is *H(die)*, but supposing that someone told you that it was a rigged die that only landed on 1,2, or 3, then the entropy associated to this die is different, and is equal to *H(die: the die will land on 1,2, or 3)*.

Aspects

Relationship to thermodynamic entropy

The inspiration for adopting the word *entropy* in information theory came from the close resemblance between Shannon's formula and very similar known formulae from thermodynamics.

In statistical thermodynamics the most general formula for the thermodynamic entropy *S* of a thermodynamic system is the Gibbs entropy,

$$S = -k_B \sum p_i \ln p_i$$

where k_B is the Boltzmann constant, and p_i is the probability of a microstate. The Gibbs entropy was defined by J. Willard Gibbs in 1878 after earlier work by Boltzmann (1872).

The Gibbs entropy translates over almost unchanged into the world of quantum physics to give the von Neumann entropy, introduced by John von Neumann in 1927,

$$S = -k_B \ \mathrm{Tr}(\rho \ln \rho)$$

where ρ is the density matrix of the quantum mechanical system and Tr is the trace.

At an everyday practical level the links between information entropy and thermodynamic entropy are not close. Physicists and chemists are apt to be more interested in *changes* in entropy as a system spontaneously evolves away from its initial conditions, in accordance with the second law of thermodynamics, rather than an unchanging probability distribution. And, as the numerical smallness of Boltzmann's constant k_B indicates, the

changes in S / k_B for even minute amounts of substances in chemical and physical processes represent amounts of entropy which are so large as to be right off the scale compared to anything seen in data compression or signal processing.

But, at a multidisciplinary level, connections *can* be made between thermodynamic and informational entropy, although it took many years in the development of the theories of statistical mechanics and information theory to make the relationship fully apparent. In fact, in the view of Jaynes (1957), thermodynamics should be seen as an *application* of Shannon's information theory: the thermodynamic entropy is interpreted as being an estimate of the amount of further Shannon information needed to define the detailed microscopic state of the system, that remains uncommunicated by a description solely in terms of the macroscopic variables of classical thermodynamics. For example, adding heat to a system increases its thermodynamic entropy because it increases the number of possible microscopic states that it could be in, thus making any complete state description longer. Maxwell's demon can (hypothetically) reduce the thermodynamic entropy of a system by using information about the states of individual molecules; but, as Landauer (from 1961) and co-workers have shown, to function the demon himself must increase thermodynamic entropy in the process, by at least the amount of Shannon information he proposes to first acquire and store; and so the total entropy does not decrease (which resolves the paradox).

Entropy as information content

Entropy is defined in the context of a probabilistic model. Independent fair coin flips have an entropy of 1 bit per flip. A source that always generates a long string of B's has an entropy of 0, since the next character will always be a 'B'.

The entropy rate of a data source means the average number of bits per symbol needed to encode it. Shannon's experiments with human predictors show an information rate of between 0.6 and 1.3 bits per character, depending on the experimental setup; the PPM compression algorithm can achieve a compression ratio of 1.5 bits per character in English text.

From the preceding example, note the following points:

1. The amount of entropy is not always an integer number of bits.
2. Many data bits may not convey information. For example, data structures often store information redundantly, or have identical sections regardless of the information in the data structure.

Shannon's definition of entropy, when applied to an information source, can determine the minimum channel capacity required to reliably transmit the source as encoded binary digits (see caveat below in italics). The formula can be derived by calculating the mathematical expectation of the *amount of information* contained in a digit from the information source.

Shannon's entropy measures the information contained in a message as opposed to the portion of the message that is determined (or predictable). *Examples of the latter include redundancy in language structure or statistical properties relating to the occurrence frequencies of letter or word pairs, triplets etc.*

Data compression

Entropy effectively bounds the performance of the strongest lossless (or nearly lossless) compression possible, which can be realized in theory by using the typical set or in practice using Huffman, Lempel-Ziv or arithmetic coding. The performance of existing data compression algorithms is often used as a rough estimate of the entropy of a block of data.

Limitations of entropy as information content

There are a number of entropy-related concepts that mathematically quantify information content in some way:

- the **self-information** of an individual message or symbol taken from a given probability distribution,
- the **entropy** of a given probability distribution of messages or symbols, and
- the **entropy rate** of a stochastic process.

(The "rate of self-information" can also be defined for a particular sequence of messages or symbols generated by a given stochastic process: this will always be equal to the entropy rate in the case of a stationary process.) Other quantities of information are also used to compare or relate different sources of information.

It is important not to confuse the above concepts. Oftentimes it is only clear from context which one is meant. For example, when someone says that the "entropy" of the English language is about 1.5 bits per character, they are actually modeling the English language as a stochastic process and talking about its entropy *rate*.

Although entropy is often used as a characterization of the information content of a data source, this information content is not absolute: it depends crucially on the probabilistic model. A source that always generates the same symbol has an entropy rate of 0, but the definition of what a symbol is depends on the alphabet. Consider a source that produces the string ABABABABAB... in which A is always followed by B and vice versa. If the probabilistic model considers individual letters as independent, the entropy rate of the sequence is 1 bit per character. But if the sequence is considered as "AB AB AB AB AB..." with symbols as two-character blocks, then the entropy rate is 0 bits per character.

However, if we use very large blocks, then the estimate of per-character entropy rate may become artificially low. This is because in reality, the probability distribution of the sequence is not knowable exactly; it is only an estimate. For example, suppose one considers the text of every book ever published as a sequence, with each symbol being

the text of a complete book. If there are N published books, and each book is only published once, the estimate of the probability of each book is $1/N$, and the entropy (in bits) is $-\log_2 1/N$. As a practical code, this corresponds to assigning each book a unique identifier and using it in place of the text of the book whenever one wants to refer to the book. This is enormously useful for talking about books, but it is not so useful for characterizing the information content of an individual book, or of language in general: it is not possible to reconstruct the book from its identifier without knowing the probability distribution, that is, the complete text of all the books. The key idea is that the complexity of the probabilistic model must be considered. Kolmogorov complexity is a theoretical generalization of this idea that allows the consideration of the information content of a sequence independent of any particular probability model; it considers the shortest program for a universal computer that outputs the sequence. A code that achieves the entropy rate of a sequence for a given model, plus the codebook (i.e. the probabilistic model), is one such program, but it may not be the shortest.

For example, the Fibonacci sequence is 1, 1, 2, 3, 5, 8, 13, ... Treating the sequence as a message and each number as a symbol, there are almost as many symbols as there are characters in the message, giving an entropy of approximately $\log_2(n)$. So the first 128 symbols of the Fibonacci sequence has an entropy of approximately 7 bits/symbol. However, the sequence can be expressed using a formula [$F(n) = F(n\text{-}1) + F(n\text{-}2)$ for n={3,4,5,...}, F(1)=1, F(2)=1] and this formula has a much lower entropy and applies to any length of the Fibonacci sequence.

Data as a Markov process

A common way to define entropy for text is based on the Markov model of text. For an order-0 source (each character is selected independent of the last characters), the binary entropy is:

$$H(\mathcal{S}) = -\sum p_i \log_2 p_i,$$

where p_i is the probability of i. For a first-order Markov source (one in which the probability of selecting a character is dependent only on the immediately preceding character), the **entropy rate** is:

$$H(\mathcal{S}) = -\sum_i p_i \sum_j \ p_i(j) \log_2 p_i(j),$$

where i is a **state** (certain preceding characters) and $p_i(j)$ is the probability of j given i as the previous character.

For a second order Markov source, the entropy rate is

$$H(\mathcal{S}) = -\sum_i p_i \sum_j p_i(j) \sum_k p_{i,j}(k) \ \log_2 \ p_{i,j}(k).$$

b-ary entropy

In general the ***b*-ary entropy** of a source $\mathcal{S}$= (*S*,*P*) with source alphabet $S = \{a_1, ..., a_n\}$ and discrete probability distribution $P = \{p_1, ..., p_n\}$ where p_i is the probability of a_i (say $p_i = p(a_i)$) is defined by:

$$H_b(\mathcal{S}) = -\sum_{i=1}^{n} p_i \log_b p_i,$$

Note: the *b* in "*b*-ary entropy" is the number of different symbols of the "ideal alphabet" which is being used as the standard yardstick to measure source alphabets. In information theory, two symbols are necessary and sufficient for an alphabet to be able to encode information, therefore the default is to let *b* = 2 ("binary entropy"). Thus, the entropy of the source alphabet, with its given empiric probability distribution, is a number equal to the number (possibly fractional) of symbols of the "ideal alphabet", with an optimal probability distribution, necessary to encode for each symbol of the source alphabet. Also note that "optimal probability distribution" here means a uniform distribution: a source alphabet with *n* symbols has the highest possible entropy (for an alphabet with *n* symbols) when the probability distribution of the alphabet is uniform. This optimal entropy turns out to be $\log_b n$.

Efficiency

A source alphabet with non-uniform distribution will have less entropy than if those symbols had uniform distribution (i.e. the "optimized alphabet"). This deficiency in entropy can be expressed as a ratio:

$$\text{efficiency}(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) / \log_b(n)$$

Efficiency has utility in quantifying the effective use of a communications channel.

Characterization

Shannon entropy is characterized by a small number of criteria, listed below. Any definition of entropy satisfying these assumptions has the form

$$-K \sum_{i=1}^{n} p_i \log p_i$$

where *K* is a constant corresponding to a choice of measurement units.

In the following, $p_i = \Pr(X = x_i)$ and $H_n(p_1, \ldots, p_n) = H(X)$.

Continuity

The measure should be continuous, so that changing the values of the probabilities by a very small amount should only change the entropy by a small amount.

Symmetry

The measure should be unchanged if the outcomes x_i are re-ordered.

$$H_n\left(p_1, p_2, \ldots\right) = H_n\left(p_2, p_1, \ldots\right) \text{etc.}$$

Maximum

The measure should be maximal if all the outcomes are equally likely (uncertainty is highest when all possible events are equiprobable).

$$H_n(p_1, \ldots, p_n) \leq H_n\left(\frac{1}{n}, \ldots, \frac{1}{n}\right).$$

For equiprobable events the entropy should increase with the number of outcomes.

$$H_n\Big(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n}\Big) < H_{n+1}\Big(\underbrace{\frac{1}{n+1}, \ldots, \frac{1}{n+1}}_{n+1}\Big).$$

Additivity

The amount of entropy should be independent of how the process is regarded as being divided into parts.

This last functional relationship characterizes the entropy of a system with sub-systems. It demands that the entropy of a system can be calculated from the entropies of its sub-systems if the interactions between the sub-systems are known.

Given an ensemble of n uniformly distributed elements that are divided into k boxes (sub-systems) with b_1, b_2, ... b_k elements each, the entropy of the whole ensemble should be equal to the sum of the entropy of the system of boxes and the individual entropies of the boxes, each weighted with the probability of being in that particular box.

For positive integers b_i where $b_1 + \ldots + b_k = n$,

$$H_n\left(\frac{1}{n}, \ldots, \frac{1}{n}\right) = H_k\left(\frac{b_1}{n}, \ldots, \frac{b_k}{n}\right) + \sum_{i=1}^{k} \frac{b_i}{n} H_{b_i}\left(\frac{1}{b_i}, \ldots, \frac{1}{b_i}\right).$$

Choosing $k = n$, $b_1 = \ldots = b_n = 1$ this implies that the entropy of a certain outcome is zero:

$$H_1(1) = 0$$

This implies that the efficiency of a source alphabet with n symbols can be defined simply as being equal to its n-ary entropy.

Further properties

The Shannon entropy satisfies the following properties, for some of which it is useful to interpret entropy as the amount of information learned (or uncertainty eliminated) by revealing the value of a random variable X:

- Adding or removing an event with probability zero does not contribute to the entropy:

$$H_{n+1}(p_1, \ldots, p_n, 0) = H_n(p_1, \ldots, p_n).$$

- It can be confirmed using the Jensen inequality that

$$H(X) = \mathrm{E}\left[\log_b\left(\frac{1}{p(X)}\right)\right] \leq \log_b\left[\mathrm{E}\left(\frac{1}{p(X)}\right)\right] = \log_b(n).$$

This maximal entropy of $\log_b(n)$ is effectively attained by a source alphabet having a uniform probability distribution: uncertainty is maximal when all possible events are equiprobable.

- The entropy or the amount of information revealed by evaluating (X,Y) (that is, evaluating X and Y simultaneously) is equal to the information revealed by conducting two consecutive experiments: first evaluating the value of Y, then revealing the value of X given that you know the value of Y. This may be written as

  $H[(X,Y)] = H(X \mid Y) + H(Y)$.

- If X and Y are two independent experiments, then knowing the value of Y doesn't influence our knowledge of the value of X (since the two don't influence each other by independence):

  $H(X \mid Y) = H(X)$.

- The entropy of two simultaneous events is no more than the sum of the entropies of each individual event, and are equal if the two events are independent. More specifically, if *X* and *Y* are two random variables on the same probability space, and *(X,Y)* denotes their Cartesian product, then

$$H[(X,Y)] \leq H(X) + H(Y).$$

Proving this mathematically follows easily from the previous two properties of entropy.

Extending discrete entropy to the continuous case: differential entropy

The Shannon entropy is restricted to random variables taking discrete values. The formula

$$h[f] = -\int_{-\infty}^{\infty} f(x) \log f(x)\, dx, \quad (1)$$

where *f* denotes a probability density function on the real line, is analogous to the Shannon entropy and could thus be viewed as an extension of the Shannon entropy to the domain of real numbers.

A precursor of the continuous entropy *h*[*f*] given in (1) is the expression for the functional *H* in the H-theorem of Boltzmann.

Formula (1) is usually referred to as the **continuous entropy**, or differential entropy. Although the analogy between both functions is suggestive, the following question must be set: is the differential entropy a valid extension of the Shannon discrete entropy? Differential entropy lacks a number of properties that the Shannon discrete entropy has – it can even be negative – and thus corrections have been suggested, notably limiting density of discrete points.

To answer this question, we must establish a connection between the two functions:

We wish to obtain a generally finite measure as the bin size goes to zero. In the discrete case, the bin size is the (implicit) width of each of the *n* (finite or infinite) bins whose probabilities are denoted by p_n. As we generalize to the continuous domain, we must make this width explicit.

To do this, start with a continuous function *f* discretized as shown in the figure. As the figure indicates, by the mean-value theorem there exists a value x_i in each bin such that

$$f(x_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} f(x)\, dx$$

and thus the integral of the function *f* can be approximated (in the Riemannian sense) by

$$\int_{-\infty}^{\infty} f(x)\,dx = \lim_{\Delta \to 0} \sum_{i=-\infty}^{\infty} f(x_i)\Delta$$

where this limit and "bin size goes to zero" are equivalent.

We will denote

$$H^{\Delta} := -\sum_{i=-\infty}^{\infty} \Delta f(x_i) \log \Delta f(x_i)$$

and expanding the logarithm, we have

$$\begin{aligned} H^{\Delta} &= -\sum_{i=-\infty}^{\infty} \Delta f(x_i) \log \Delta f(x_i) \\ &= -\sum_{i=-\infty}^{\infty} \Delta f(x_i) \log f(x_i) - \sum_{i=-\infty}^{\infty} f(x_i)\Delta \log \Delta. \end{aligned}$$

As $\Delta \to 0$, we have

$$\sum_{i=-\infty}^{\infty} f(x_i)\Delta \to \int f(x)\,dx = 1$$

and also

$$\sum_{i=-\infty}^{\infty} \Delta f(x_i) \log f(x_i) \to \int f(x) \log f(x)\,dx.$$

But note that $\log \Delta \to -\infty$as $\Delta \to 0$, therefore we need a special definition of the differential or continuous entropy:

$$h[f] = \lim_{\Delta \to 0} \left[H^{\Delta} + \log \Delta \right] = -\int_{-\infty}^{\infty} f(x) \log f(x)\,dx,$$

which is, as said before, referred to as the **differential entropy**. This means that the differential entropy *is not* a limit of the Shannon entropy for $n \to \infty$. Rather, if differs from the limit of the Shannon entropy by an infinite offset.

It turns out as a result that, unlike the Shannon entropy, the differential entropy is *not* in general a good measure of uncertainty or information. For example, the differential

entropy can be negative; also it is not invariant under continuous co-ordinate transformations.

Another useful measure of entropy for the continuous case is the **relative entropy** of a distribution, defined as the Kullback-Leibler divergence from the distribution to a reference measure $m(x)$,

$$D_{\mathrm{KL}}(f(x)\|m(x)) = \int f(x) \log \frac{f(x)}{m(x)}\, dx.$$

The relative entropy carries over directly from discrete to continuous distributions, and is invariant under co-ordinate reparameterizations.

Use in combinatorics

Entropy has become a useful quantity in combinatorics. A simple example of this is an alternate proof of the Loomis-Whitney inequality: for every subset $A \subseteq \mathbb{Z}^d$, we have

$$|A|^{d-1} \leq \prod_{i=1}^{d} |P_i(A)|$$

where $P_i(A) = \{(x_1, ..., x_{i-1}, x_{i+1}, ..., x_d) : (x_1, ...x_d) \in A$, that is, P_i is the orthogonal projection in the ith coordinate.

The proof follows as a simple corollary of **Shearer's inequality**: if $X_1,...X_d$ are random variables and $S_1,...,S_n$ are subsets of $\{1,2,...,d\}$ such that every integer between *1* and *d* lie in exactly *r* of these subsets, then

$$H[(X_1, ..., X_d)] \leq \frac{1}{r} \sum_{i=1}^{n} H[(X_j)_{j \in S_i}]$$

where $(X_j)_{j \in S_i}$ is the Cartesian product of random variables X_j with indexes j in S_i (so the dimension of this vector is equal to the size of S_i).

We sketch how Loomis-Whitney follows from this: Indeed, let X be a uniformly distributed random variable with values in A and so that each point in A occurs with equal probability. Then (by the further properties of entropy mentioned above) $H(X) = \log |A|$, where $|A|$ denotes the cardinality of A. Let $S_i = \{1,2,...,i-1,i+1,...,d\}$. The range of $(X_j)_{j \in S_i}$ is contained in $P_i(A)$ and hence $H[(X_j)_{j \in S_i}] \leq \log |P_i(A)|$. Now use this to bound the right side of Shearer's inequality and exponentiate the opposite sides of the resulting inequality you obtain.

Chapter 3

Noisy-Channel Coding Theorem and Mutual Information

Noisy-channel coding theorem

In information theory, the **noisy-channel coding theorem** (sometimes **Shannon's theorem**), establishes that for any given degree of noise contamination of a communication channel, it is possible to communicate discrete data (digital information) nearly error-free up to a computable maximum rate through the channel. This result was presented by Claude Shannon in 1948 and was based in part on earlier work and ideas of Harry Nyquist and Ralph Hartley.

The **Shannon limit** or **Shannon capacity** of a communications channel is the theoretical maximum information transfer rate of the channel, for a particular noise level.

Overview

Stated by Claude Shannon in 1948, the theorem describes the maximum possible efficiency of error-correcting methods versus levels of noise interference and data corruption. The theory doesn't describe *how to construct* the error-correcting method, it only tells us how good the *best possible* method can be. Shannon's theorem has wide-ranging applications in both communications and data storage. This theorem is of foundational importance to the modern field of information theory. Shannon only gave an outline of the proof. The first rigorous proof is due to Amiel Feinstein in 1954.

The Shannon theorem states that given a noisy channel with channel capacity C and information transmitted at a rate R, then if $R < C$ there exist codes that allow the probability of error at the receiver to be made arbitrarily small. This means that, theoretically, it is possible to transmit information nearly without error at any rate below a limiting rate, C.

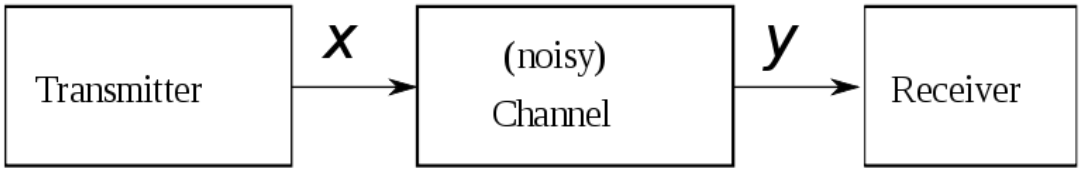
The converse is also important. If $R > C$, an arbitrarily small probability of error is not achievable. All codes will have a probability of error greater than a certain positive minimal level, and this level increases as the rate increases. So, information cannot be guaranteed to be transmitted reliably across a channel at rates beyond the channel

capacity. The theorem does not address the rare situation in which rate and capacity are equal.

The channel capacity C can be calculated from the physical properties of a channel; for a band-limited channel with Gaussian noise, using the Shannon–Hartley theorem.

Simple schemes such as "send the message 3 times and use a best 2 out of 3 voting scheme if the copies differ" are inefficient error-correction methods, unable to asymptotically guarantee that a block of data can be communicated free of error. Advanced techniques such as Reed–Solomon codes and, more recently, turbo codes come much closer to reaching the theoretical Shannon limit, but at a cost of high computational complexity. Using low-density parity-check (LDPC) codes or turbo codes and with the computing power in today's digital signal processors, it is now possible to reach very close to the Shannon limit. In fact, it was shown that LDPC codes can reach within 0.0045 dB of the Shannon limit (for very long block lengths).

Mathematical statement



Theorem (Shannon, 1948):

1. For every discrete memoryless channel, the channel capacity
$$C = \sup_{p_X} I(X;Y)$$
has the following property. For any ε > 0 and $R < C$, for large enough N, there exists a code of length N and rate ≥ R and a decoding algorithm, such that the maximal probability of block error is ≤ ε.
2. If a probability of bit error p_b is acceptable, rates up to $R(p_b)$ are achievable, where
$$R(p_b) = \frac{C}{1 - H_2(p_b)}.$$
and $H_2(p_b)$ is the *binary entropy function*
$$H_2(p_b) = -\left[p_b \log p_b + (1 - p_b) \log(1 - p_b)\right]$$
3. For any p_b, rates greater than $R(p_b)$ are not achievable.

Outline of proof

As with several other major results in information theory, the proof of the noisy channel coding theorem includes an achievability result and a matching converse result. These two components serve to bound, in this case, the set of possible rates at which one can

communicate over a noisy channel, and matching serves to show that these bounds are tight bounds.

The following outlines are only one set of many different styles available for study in information theory texts.

Achievability for discrete memoryless channels

This particular proof of achievability follows the style of proofs that make use of the asymptotic equipartition property (AEP). Another style can be found in information theory texts using error exponents.

Both types of proofs make use of a random coding argument where the codebook used across a channel is randomly constructed - this serves to reduce computational complexity while still proving the existence of a code satisfying a desired low probability of error at any data rate below the channel capacity.

By an AEP-related argument, given a channel, length *n* strings of source symbols X_1^n, and length *n* strings of channel outputs Y_1^n, we can define a *jointly typical set* by the following:

$$\begin{aligned} A_\varepsilon^{(n)} = \{&(x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n \\ &2^{-n(H(X)+\varepsilon)} \leq p(X_1^n) \leq 2^{-n(H(X)-\varepsilon)} \\ &2^{-n(H(Y)+\varepsilon)} \leq p(Y_1^n) \leq 2^{-n(H(Y)-\varepsilon)} \\ &2^{-n(H(X,Y)+\varepsilon)} \leq p(X_1^n, Y_1^n) \leq 2^{-n(H(X,Y)-\varepsilon)}\} \end{aligned}$$

We say that two sequences X_1^n và Y_1^n are *jointly typical* if they lie in the jointly typical set defined above.

Steps

1. In the style of the random coding argument, we randomly generate 2^{nR} codewords of length n from a probability distribution Q.
2. This code is revealed to the sender and receiver. It is also assumed that one knows the transition matrix $p(y \mid x)$ for the channel being used.
3. A message W is chosen according to the uniform distribution on the set of codewords. That is, $Pr(W = w) = 2^{-nR}, w = 1, 2, \ldots, 2^{nR}$.
4. The message W is sent across the channel.
5. The receiver receives a sequence according to
$$P(y^n|x^n(w)) = \prod_{i=1}^{n} p(y_i|x_i(w))$$
6. Sending these codewords across the channel, we receive Y_1^n, and decode to some source sequence if there exists exactly 1 codeword that is jointly typical with Y. If

there are no jointly typical codewords, or if there are more than one, an error is declared. An error also occurs if a decoded codeword doesn't match the original codeword. This is called *typical set decoding*.

The probability of error of this scheme is divided into two parts:

1. First, error can occur if no jointly typical X sequences are found for a received Y sequence
2. Second, error can occur if an incorrect X sequence is jointly typical with a received Y sequence.

- By the randomness of the code construction, we can assume that the average probability of error averaged over all codes does not depend on the index sent. Thus, without loss of generality, we can assume $W = 1$.

- From the joint AEP, we know that the probability that no jointly typical X exists goes to 0 as n grows large. We can bound this error probability by ε.

- Also from the joint AEP, we know the probability that a particular $X_1^n(i)$ and the Y_1^n resulting from $W = 1$ are jointly typical is $\leq 2^{-n(I(X;Y)-3\varepsilon)}$.

Define: $E_i = \{(X_1^n(i), Y_1^n) \in A_\varepsilon^{(n)}\}, i = 1, 2, \ldots, 2^{nR}$

as the event that message i is jointly typical with the sequence received when message 1 is sent.

$$P(\text{error}) = P(\text{error}|W = 1) \leq P(E_1^c) + \sum_{i=2}^{2^{nR}} P(E_i)$$
$$\leq \varepsilon + 2^{-n(I(X;Y)-R-3\varepsilon)}.$$

We can observe that as n goes to infinity, if $R < I(X;Y)$ for the channel, the probability of error will go to 0.

Finally, given that the average codebook is shown to be "good," we know that there exists a codebook whose performance is better than the average, and so satisfies our need for arbitrarily low error probability communicating across the noisy channel.

Weak converse for discrete memoryless channels

Suppose a code of 2^{nR} codewords. Let W be drawn uniformly over this set as an index. Let X^n and Y^n be the codewords and received codewords, respectively.

1. $nR = H(W) = H(W|Y^n) + I(W;Y^n)$ using identities involving entropy and mutual information
2. $\leq H(W|Y^n) + I(X^n(W);Y^n)$ since X is a function of W
3. $\leq 1 + P_e^{(n)} nR + I(X^n(W);Y^n)$ by the use of Fano's Inequality
4. $\leq 1 + P_e^{(n)} nR + nC$ by the fact that capacity is maximized mutual information.

The result of these steps is that $P_e^{(n)} \geq 1 - \frac{1}{nR} - \frac{C}{R}$. As the block length *n* goes to infinity, we obtain $P_e^{(n)}$ is bounded away from 0 if R is greater than C - we can get arbitrarily low rates of error only if R is less than C.

Strong converse for discrete memoryless channels

A strong converse theorem, proven by Wolfowitz in 1957, states that,

$$P_e \geq 1 - \frac{4A}{n(R-C)^2} - e^{-n(R-C)}$$

for some finite positive constant *A*. While the weak converse states that the error probability is bounded away from zero as *n* goes to infinity, the strong converse states that the error goes exponentially to 1. Thus, *C* is a sharp threshold between perfectly reliable and completely unreliable communication.

Channel coding theorem for non-stationary memoryless channels

We assume that the channel is memoryless, but its transition probabilities change with time, in a fashion known at the transmitter as well as the receiver.

Then the channel capacity is given by

$$C = \liminf \max_{p(X_1), p(X_2), \ldots} \frac{1}{n} \sum_{i=1}^{n} I(X_i; Y_i).$$

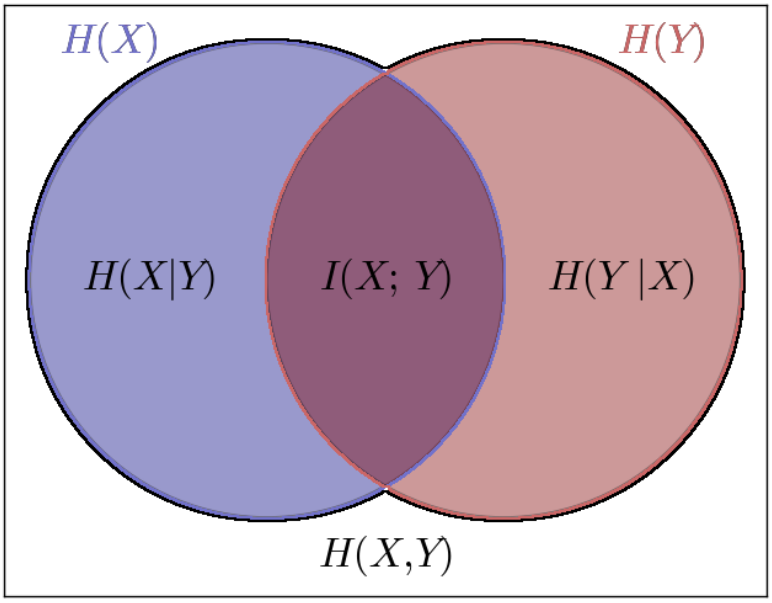
The maximum is attained at the capacity achieving distributions for each respective channel. That is, $C = \liminf \frac{1}{n} \sum_{i=1}^{n} C_i$ where C_i is the capacity of the i*th* channel.

Outline of the proof

The proof runs through in almost the same way as that of channel coding theorem. Achievability follows from random coding with each symbol chosen randomly from the capacity achieving distribution for that particular channel. Typicality arguments use the definition of typical sets for non-stationary sources defined in the asymptotic equipartition property article.

The technicality of lim inf comes into play when $\frac{1}{n}\sum_{i=1}^{n} C_i$ does not converge.

Mutual information



Individual (H(X),H(Y)), joint (H(X,Y)), and conditional entropies for a pair of correlated subsystems X,Y with mutual information I(X; Y).

In probability theory and information theory, the **mutual information** (sometimes known by the archaic term **transinformation**) of two random variables is a quantity that measures the mutual dependence of the two variables. The most common unit of measurement of mutual information is the bit, when logarithms to the base 2 are used.

Definition of mutual information

Formally, the mutual information of two discrete random variables X and Y can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively.

In the continuous case, summation is matched with a definite double integral:

$$I(X;Y) = \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right) \, dx \, dy,$$

where $p(x,y)$ is now the joint probability *density* function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability density functions of X and Y respectively.

These definitions are ambiguous because the base of the log function is not specified. To disambiguate, the function I could be parameterized as $I(X,Y,b)$ where b is the base. Alternatively, since the most common unit of measurement of mutual information is the bit, a base of 2 could be specified.

Intuitively, mutual information measures the information that X and Y share: it measures how much knowing one of these variables reduces our uncertainty about the other. For example, if X and Y are independent, then knowing X does not give any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X and Y are identical then all information conveyed by X is shared with Y: knowing X determines the value of Y and vice versa. As a result, in the case of identity the mutual information is the same as the uncertainty contained in Y (or X) alone, namely the entropy of Y (or X: clearly if X and Y are identical they have equal entropy).

Mutual information quantifies the dependence between the joint distribution of X and Y and what the joint distribution would be if X and Y were independent. Mutual information is a measure of dependence in the following sense: $I(X; Y) = 0$ if and only if X and Y are independent random variables. This is easy to see in one direction: if X and Y are independent, then $p(x,y) = p(x)\, p(y)$, and therefore:

$$\log \left(\frac{p(x,y)}{p(x)\, p(y)} \right) = \log 1 = 0.$$

Moreover, mutual information is nonnegative (i.e. $I(X;Y) \geq 0$; see below) and symmetric (i.e. $I(X;Y) = I(Y;X)$).

Relation to other quantities

Mutual information can be equivalently expressed as

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X,Y) - H(X|Y) - H(Y|X) \end{aligned}$$

where $H(X)$ and $H(Y)$ are the marginal entropies, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y. Since $H(X) \geq H(X|Y)$, this characterization is consistent with the nonnegativity property stated above.

Intuitively, if entropy $H(X)$ is regarded as a measure of uncertainty about a random variable, then $H(X|Y)$ is a measure of what Y does *not* say about X. This is "the amount of uncertainty remaining about X after Y is known", and thus the right side of the first of these equalities can be read as "the amount of uncertainty in X, minus the amount of uncertainty in X which remains after Y is known", which is equivalent to "the amount of uncertainty in X which is removed by knowing Y". This corroborates the intuitive meaning of mutual information as the amount of information (that is, reduction in uncertainty) that knowing either variable provides about the other.

Note that in the discrete case $H(X|X) = 0$ and therefore $H(X) = I(X;X)$. Thus $I(X;X) \geq I(X;Y)$, and one can formulate the basic principle that a variable contains at least as much information about itself as any other variable can provide.

Mutual information can also be expressed as a Kullback-Leibler divergence, of the product $p(x) \times p(y)$ of the marginal distributions of the two random variables X and Y, from $p(x,y)$ the random variables' joint distribution:

$$I(X;Y) = D_{\mathrm{KL}}(p(x,y) \| p(x)p(y)).$$

Furthermore, let $p(x|y) = p(x, y) / p(y)$. Then

$$
\begin{aligned}
I(X;Y) &= \sum_y p(y) \sum_x p(x|y) \log_2 \frac{p(x|y)}{p(x)} \\
&= \sum_y p(y)\ D_{\mathrm{KL}}(p(x|y)\|p(x)) \\
&= \mathbb{E}_Y\{D_{\mathrm{KL}}(p(x|y)\|p(x))\}.
\end{aligned}
$$

Thus mutual information can also be understood as the expectation of the Kullback-Leibler divergence of the univariate distribution $p(x)$ of X from the conditional distribution $p(x|y)$ of X given Y: the more different the distributions $p(x|y)$ and $p(x)$, the greater the information gain.

Variations of mutual information

Several variations on mutual information have been proposed to suit various needs. Among these are normalized variants and generalizations to more than two variables.

Metric

Many applications require a metric, that is, a distance measure between points. The quantity

$$d(X,Y) = H(X,Y) - I(X;Y) = H(X) + H(Y) - 2I(X;Y) = H(X \mid Y) + H(Y \mid X)$$

satisfies the basic properties of a metric; most importantly, the triangle inequality, but also non-negativity, indiscernability and symmetry. This distance metric is also known as the Variation of information.

Since one has $d(X,Y) \leq H(X,Y)$, a natural normalized variant is

$$D(X,Y) = d(X,Y)/H(X,Y) \leq 1.$$

The metric D is a universal metric, in that if any other distance measure places X and Y close-by, then the D will also judge them close.

A set-theoretic interpretation of information shows that

$$D(X,Y) = 1 - I(X;Y)/H(X,Y) = 1 - H(X \cap Y)/H(X \cup Y)$$

which is effectively the Jaccard distance between X and Y.

Conditional mutual information

Sometimes it is useful to express the mutual information of two random variables conditioned on a third.

$$I(X;Y|Z) = \mathbb{E}_Z\big(I(X;Y)|Z\big) = \sum_{z\in Z}\sum_{y\in Y}\sum_{x\in X} p_Z(z)p_{X,Y|Z}(x,y|z)\log\frac{p_{X,Y|Z}(x,y|z)}{p_{X|Z}(x|z)p_{Y|Z}(y|z)},$$

which can be simplified as

$$I(X;Y|Z) = \sum_{z\in Z}\sum_{y\in Y}\sum_{x\in X} p_{X,Y,Z}(x,y,z)\log\frac{p_Z(z)p_{X,Y,Z}(x,y,z)}{p_{X,Z}(x,z)p_{Y,Z}(y,z)}.$$

Conditioning on a third random variable may either increase or decrease the mutual information, but it is always true that

$$I(X;Y|Z) \geq 0$$

for discrete, jointly distributed random variables *X*, *Y*, *Z*. This result has been used as a basic building block for proving other inequalities in information theory.

Multivariate mutual information

Several generalizations of mutual information to more than two random variables have been proposed, such as total correlation and interaction information. If Shannon entropy is viewed as a signed measure in the context of information diagrams, as explained in *Information theory and measure theory*, then the only definition of multivariate mutual information that makes sense is as follows:

$I(X_1;X_1) = H(X_1)$

and for $n > 1$,

$$I(X_1;\,\ldots\,;X_n) = I(X_1;\,\ldots\,;X_{n-1}) - I(X_1;\,\ldots\,;X_{n-1}|X_n),$$

where (as above) we define

$$I(X_1;\,\ldots\,;X_{n-1}|X_n) = \mathbb{E}_{X_n}\big(I(X_1;\,\ldots\,;X_{n-1})|X_n\big).$$

(This definition of multivariate mutual information is identical to that of interaction information except for a change in sign when the number of random variables is odd.)

Applications

Applying information diagrams blindly to derive the above definition has been criticised, and indeed it has found rather limited practical application, since it is difficult to visualize or grasp the significance of this quantity for a large number of random variables. It can be zero, positive, or negative for any $n \geq 3$.

One high-dimensional generalization scheme which maximizes the mutual information between the joint distribution and other target variables is found to be useful in feature selection.

Normalized variants

Normalized variants of the mutual information are provided by the *coefficients of constraint* (Coombs, Dawes & Tversky 1970) or *uncertainty coefficient* (Press & Flannery 1988)

$$C_{XY} = \frac{I(X;Y)}{H(Y)} \quad \text{and} \quad C_{YX} = \frac{I(X;Y)}{H(X)}.$$

The two coefficients are not necessarily equal. A more useful and symmetric scaled information measure is the *redundancy*

$$R = \frac{I(X;Y)}{H(X) + H(Y)}$$

which attains a minimum of zero when the variables are independent and a maximum value of

$$R_{\max} = \frac{\min(H(X), H(Y))}{H(X) + H(Y)}$$

when one variable becomes completely redundant with the knowledge of the other. Another symmetrical measure is the *symmetric uncertainty* (Witten & Frank 2005), given by

$$U(X,Y) = 2R = 2\frac{I(X;Y)}{H(X) + H(Y)}$$

which represents a weighted average of the two uncertainty coefficients (Press & Flannery 1988).

Other normalized versions are provided by the following expressions (Yao 2003, Strehl & Ghosh 2002).

$$\frac{I(X;Y)}{\min(H(X),H(Y))}, \qquad \frac{I(X;Y)}{H(X,Y)}, \qquad \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$

If we consider mutual information as a special case of the total correlation, the normalization is:

$$\frac{I(X;Y)}{\min(H(X),H(Y))}$$

The quantity

$$D'(X,Y) = 1 - \frac{I(X;Y)}{\max(H(X),H(Y))}$$

is a metric, *i.e.* satisfies the triangle inequality, *etc.* The metric D' is also a universal metric.

Weighted variants

In the traditional formulation of the mutual information,

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\frac{p(x,y)}{p(x)\,p(y)},$$

each *event* or *object* specified by (*x*,*y*) is weighted by the corresponding probability *p*(*x*,*y*). This assumes that all objects or events are equivalent *apart from* their probability of occurrence. However, in some applications it may be the case that certain objects or events are more *significant* than others, or that certain patterns of association are more semantically important than others.

For example, the deterministic mapping {(1,1),(2,2),(3,3)} may be viewed as stronger than the deterministic mapping {(1,3),(2,1),(3,2)}, although these relationships would yield the same mutual information. This is because the mutual information is not sensitive at all to any inherent ordering in the variable values (Cronbach 1954, Coombs & Dawes 1970, Lockhead 1970), and is therefore not sensitive at all to the **form** of the relational mapping between the associated variables. If it is desired that the former relation — showing agreement on all variable values — be judged stronger than the later relation, then it is possible to use the following *weighted mutual information* (Guiasu 1977)

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} w(x,y) p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)},$$

which places a weight $w(x,y)$ on the probability of each variable value co-occurrence, $p(x,y)$. This allows that certain probabilities may carry more or less significance than others, thereby allowing the quantification of relevant *holistic* or *prägnanz* factors. In the above example, using larger relative weights for $w(1,1)$, $w(2,2)$, and $w(3,3)$ would have the effect of assessing greater *informativeness* for the relation {(1,1),(2,2),(3,3)} than for the relation {(1,3),(2,1),(3,2)}, which may be desirable in some cases of pattern recognition, and the like. There has been little mathematical work done on the weighted mutual information and its properties, however.

Absolute mutual information

Using the ideas of Kolmogorov complexity, one can consider the mutual information of two sequences independent of any probability distribution:

$$I_K(X;Y) = K(X) - K(X \mid Y).$$

To establish that this quantity is symmetric up to a logarithmic factor ($I_K(X;Y) \approx I_K(Y;X)$) requires the chain rule for Kolmogorov complexity (Li 1997). Approximations of this quantity via compression can be used to define a distance measure to perform a hierarchical clustering of sequences without having any domain knowledge of the sequences (Cilibrasi 2005).

Applications of mutual information

In many applications, one wants to maximize mutual information (thus increasing dependencies), which is often equivalent to minimizing conditional entropy. Examples include:

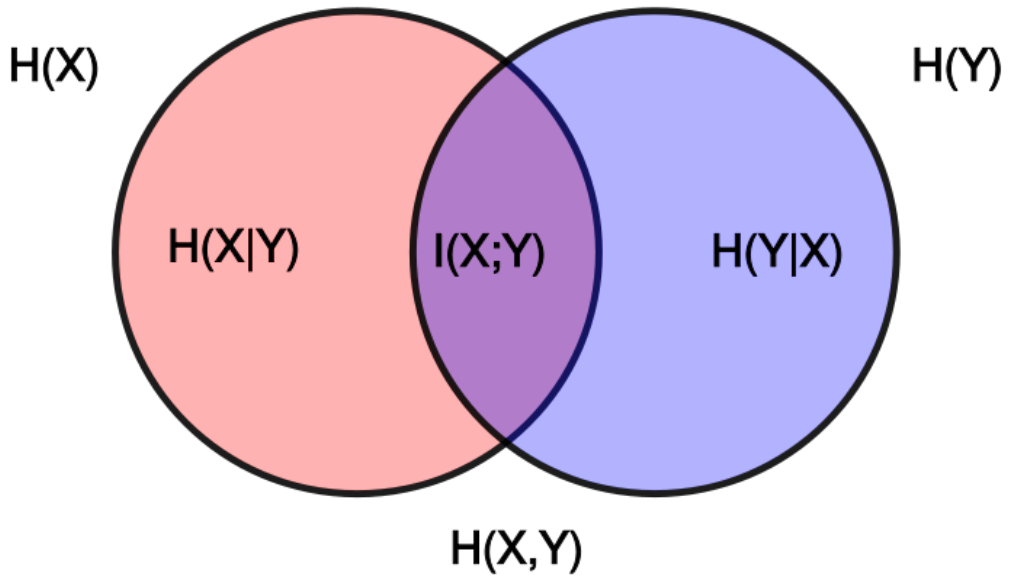
- The channel capacity is equal to the mutual information, maximized over all input distributions.
- Discriminative training procedures for hidden Markov models have been proposed based on the maximum mutual information (MMI) criterion.
- RNA secondary structure prediction from a multiple sequence alignment.
- Phylogenetic profiling prediction from pairwise present and disappearance of functionally link genes.
- Mutual information has been used as a criterion for feature selection and feature transformations in machine learning. It can be used to characterize both the relevance and redundancy of variables, such as the minimum redundancy feature selection.
- Mutual information is often used as a significance function for the computation of collocations in corpus linguistics.

- Mutual information is used in medical imaging for image registration. Given a reference image (for example, a brain scan), and a second image which needs to be put into the same coordinate system as the reference image, this image is deformed until the mutual information between it and the reference image is maximized.
- Detection of phase synchronization in time series analysis
- In the infomax method for neural-net and other machine learning, including the infomax-based Independent component analysis algorithm
- Average mutual information in delay embedding theorem is used for determining the *embedding delay* parameter.
- Mutual information between genes in expression microarray data is used by the ARACNE algorithm for reconstruction of gene networks.
- Mutual information is used as a clusterings comparing measure (Vinh et al., 2009), provided some advantages over other classical measures such as the Rand index and the Adjusted rand index.
- The adjusted-for-chance version of the mutual information is the Adjusted Mutual Information (AMI) (Vinh et al., 2009). It is used for comparing clustering. It corrects the effect of agreement between clusterings solely due to chance, similar to the way the Adjusted rand index corrects the Rand index.

Chapter 4

Quantities of Information and Coding Theory

Quantities of information



A simple information diagram illustrating the relationships among some of Shannon's basic quantities of information.

The mathematical theory of information is based on probability theory and statistics, and measures information with several **quantities of information**. The choice of logarithmic base in the following formulae determines the unit of information entropy that is used. The most common unit of information is the bit, based on the binary logarithm. Other units include the nat, based on the natural logarithm, and the hartley, based on the base 10 or common logarithm.

In what follows, an expression of the form $p \log p$ is considered by convention to be equal to zero whenever p is zero. This is justified because $\lim_{p \to 0+} p \log p = 0$ for any logarithmic base.

Self-information

Shannon derived a measure of information content called the **self-information** or **"surprisal"** of a message m:

$$I(m) = \log \left(\frac{1}{p(m)} \right) = -\log(p(m))$$

where $p(m) = \Pr(M = m)$ is the probability that message m is chosen from all possible choices in the message space M. The base of the logarithm only affects a scaling factor and, consequently, the units in which the measured information content is expressed. If the logarithm is base 2, the measure of information is expressed in units of bits.

Information is transferred from a source to a recipient only if the recipient of the information did not already have the information to begin with. Messages that convey information that is certain to happen and already known by the recipient contain no real information. Infrequently occurring messages contain more information than more frequently occurring messages. This fact is reflected in the above equation - a certain message, i.e. of probability 1, has an information measure of zero. In addition, a compound message of two (or more) unrelated (or mutually independent) messages would have a quantity of information that is the sum of the measures of information of each message individually. That fact is also reflected in the above equation, supporting the validity of its derivation.

An example: The weather forecast broadcast is: "Tonight's forecast: Dark. Continued darkness until widely scattered light in the morning." This message contains no information. However, a forecast of a snowstorm would certainly contain information since such does not happen every evening. There would be an even greater amount of information in an accurate forecast of snow for a warm location, such as Miami.

Entropy

The **entropy** of a discrete message space M is a measure of the amount of **uncertainty** one has about which message will be chosen. It is defined as the average self-information of a message m from that message space:

$$H(M) = \mathbb{E}\{I(M)\} = \sum_{m \in M} p(m) I(m) = -\sum_{m \in M} p(m) \log p(m).$$

where

$\mathbb{E}\{\}$ denotes the expected value operation.

An important property of entropy is that it is maximized when all the messages in the message space are equiprobable (e.g. $p(m) = 1 / M$). In this case $H(M) = \log |M|$.

Sometimes the function H is expressed in terms of the probabilities of the distribution:

$$H(p_1, p_2, \ldots, p_k) = -\sum_{i=1}^{k} p_i \log p_i, \quad \text{where each } p_i \geq 0 \text{ and } \sum_{i=1}^{k} p_i = 1.$$

An important special case of this is the **binary entropy function**:

$$H_{\mathrm{b}}(p) = H(p, 1-p) = -p \log p - (1-p) \log(1-p).$$

Joint entropy

The **joint entropy** of two discrete random variables X and Y is defined as the entropy of the joint distribution of X and Y:

$$H(X, Y) = \mathbb{E}_{X,Y}[-\log p(x, y)] = -\sum_{x,y} p(x, y) \log p(x, y)$$

If X and Y are independent, then the joint entropy is simply the sum of their individual entropies.

(Note: The joint entropy should not be confused with the cross entropy, despite similar notations.)

Conditional entropy (equivocation)

Given a particular value of a random variable Y, the conditional entropy of X given $Y = y$ is defined as:

$$H(X|y) = \mathbb{E}_{X|Y}[-\log p(x|y)] = -\sum_{x \in X} p(x|y) \log p(x|y)$$

where $p(x|y) = \dfrac{p(x, y)}{p(y)}$ is the conditional probability of x given y.

The **conditional entropy** of X given Y, also called the **equivocation** of X about Y is then given by:

$$H(X|Y) = \mathbb{E}_Y\{H(X|y)\} = -\sum_{y\in Y} p(y) \sum_{x\in X} p(x|y) \log p(x|y) = \sum_{x,y} p(x,y) \log \frac{p(y)}{p(x,y)}.$$

A basic property of the conditional entropy is that:

$$H(X|Y) = H(X,Y) - H(Y).$$

Kullback–Leibler divergence (information gain)

The **Kullback–Leibler divergence** (or **information divergence**, **information gain**, or **relative entropy**) is a way of comparing two distributions, a "true" probability distribution p, and an arbitrary probability distribution q. If we compress data in a manner that assumes q is the distribution underlying some data, when, in reality, p is the correct distribution, Kullback–Leibler divergence is the number of average additional bits per datum necessary for compression, or, mathematically,

$$D_{\mathrm{KL}}(p(X)\|q(X)) = \sum_{x\in X} p(x) \log \frac{p(x)}{q(x)}.$$

It is in some sense the "distance" from q to p, although it is not a true metric due to its not being symmetric.

Mutual information (transinformation)

It turns out that one of the most useful and important measures of information is the **mutual information**, or **transinformation**. This is a measure of how much information can be obtained about one random variable by observing another. The mutual information of X relative to Y (which represents conceptually the average amount of information about X that can be gained by observing Y) is given by:

$$I(X;Y) = \sum_{y\in Y} p(y) \sum_{x\in X} p(x|y) \log \frac{p(x|y)}{p(x)} = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}.$$

A basic property of the mutual information is that:

$$I(X;Y) = H(X) - H(X|Y).$$

That is, knowing Y, we can save an average of $I(X;Y)$ bits in encoding X compared to not knowing Y. Mutual information is symmetric:

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,Y).$$

Mutual information can be expressed as the average Kullback–Leibler divergence (information gain) of the posterior probability distribution of *X* given the value of *Y* to the prior distribution on *X*:

$$I(X;Y) = \mathbb{E}_{p(y)}\{D_{\mathrm{KL}}(p(X|Y=y)\|p(X))\}.$$

In other words, this is a measure of how much, on the average, the probability distribution on *X* will change if we are given the value of *Y*. This is often recalculated as the divergence from the product of the marginal distributions to the actual joint distribution:

$$I(X;Y) = D_{\mathrm{KL}}(p(X,Y)\|p(X)p(Y)).$$

Mutual information is closely related to the log-likelihood ratio test in the context of contingency tables and the multinomial distribution and to Pearson's χ^2 test: mutual information can be considered a statistic for assessing independence between a pair of variables, and has a well-specified asymptotic distribution.

Differential entropy

The basic measures of discrete entropy have been extended by analogy to continuous spaces by replacing sums with integrals and probability mass functions with probability density functions. Although, in both cases, mutual information expresses the number of bits of information common to the two sources in question, the analogy does *not* imply identical properties; for example, differential entropy may be negative.

The differential analogies of entropy, joint entropy, conditional entropy, and mutual information are defined as follows:

$$\begin{aligned}
h(X) &= -\int_X f(x)\log f(x)\,dx \\
h(X,Y) &= -\int_Y\int_X f(x,y)\log f(x,y)\,dx\,dy \\
h(X|y) &= -\int_X f(x|y)\log f(x|y)\,dx \\
h(X|Y) &= \int_Y\int_X f(x,y)\log \frac{f(y)}{f(x,y)}\,dx\,dy \\
I(X;Y) &= \int_Y\int_X f(x,y)\log \frac{f(x,y)}{f(x)f(y)}\,dx\,dy
\end{aligned}$$

where $f(x,y)$ is the joint density function, $f(x)$ and $f(y)$ are the marginal distributions, and $f(x \mid y)$ is the conditional distribution.

Coding theory

Coding theory is the study of the properties of codes and their fitness for a specific application. Codes are used for data compression, cryptography, error-correction and more recently also for network coding. Codes are studied by various scientific disciplines—such as information theory, electrical engineering, mathematics, and computer science—for the purpose of designing efficient and reliable data transmission methods. This typically involves the removal of redundancy and the correction (or detection) of errors in the transmitted data.

There are essentially two aspects to Coding theory:

1. Data compression (or, *source coding*)
2. Error correction (or, *channel coding*).

These two aspects may be studied in combination. Source encoding, attempts to compress the data from a source in order to transmit it more efficiently. This practice is found every day on the Internet where the common "Zip" data compression is used to reduce the network load and make files smaller. The second, channel encoding, adds extra data bits to make the transmission of data more robust to disturbances present on the transmission channel. The ordinary user may not be aware of many applications using channel coding. A typical music CD uses the Reed-Solomon code to correct for scratches and dust. In this application the transmission channel is the CD itself. Cell phones also use coding techniques to correct for the fading and noise of high frequency radio transmission. Data modems, telephone transmissions, and NASA all employ channel coding techniques to get the bits through, for example the turbo code and LDPC codes.

Source coding

The aim of source coding is to take the source data and make it smaller.

Principle

Entropy of a source is the measure of information. Basically source codes try to reduce the redundancy present in the source, and represent the source with fewer bits that carry more information.

Data compression which explicitly tries to minimize the average length of messages according to a particular assumed probability model is called entropy encoding.

Various techniques used by source coding schemes try to achieve the limit of Entropy of the source. $C(x) \geq H(x)$, where $H(x)$ is entropy of source (bitrate), and $C(x)$ is the bitrate after compression. In particular, no source coding scheme can be better than the entropy of the source.

Example

Facsimile transmission uses a simple run length code. Source coding includes also removal of all data that superfluous the need of transmitter, this decreases the bandwidth required for the transmission process.

Channel coding

The aim of channel coding theory is to find codes which transmit quickly, contain many valid code words and can correct or at least detect many errors. While not mutually exclusive, performance in these areas is a trade off. So, different codes are optimal for different applications. The needed properties of this code mainly depend on the probability of errors happening during transmission. In a typical CD, the impairment is mainly dust or scratches. Thus codes are used in an interleaved manner. The data is spread out over the disk. Although not a very good code, a simple repeat code can serve as an understandable example. Suppose we take a block of data bits (representing sound) and send it three times. At the receiver we will examine the three repetitions bit by bit and take a majority vote. The twist on this is that we don't merely send the bits in order. We interleave them. The block of data bits is first divided into 4 smaller blocks. Then we cycle through the block and send one bit from the first, then the second, etc. This is done three times to spread the data out over the surface of the disk. In the context of the simple repeat code, this may not appear effective. However, there are more powerful codes known which are very effective at correcting the "burst" error of a scratch or a dust spot when this interleaving technique is used.

Other codes are more appropriate for different applications. Deep space communications are limited by the thermal noise of the receiver which is more of a continuous nature than a bursty nature. Likewise, narrowband modems are limited by the noise, present in the telephone network and also modeled better as a continuous disturbance. Cell phones are subject to rapid fading. The high frequencies used can cause rapid fading of the signal even if the receiver is moved a few inches. Again there are a class of channel codes that are designed to combat fading.

Linear codes

The term **algebraic coding theory** denotes the sub-field of coding theory where the properties of codes are expressed in algebraic terms and then further researched.

Algebraic coding theory is basically divided into two major types of codes:

1. Linear block codes

2. Convolutional codes.

It analyzes the following three properties of a code – mainly:

- code word length
- total number of valid code words
- the minimum distance between two valid code words, using mainly the Hamming distance, sometimes also other distances like the Lee distance.

Linear block codes

Linear block codes have the property of linearity, i.e the sum of any two codewords is also a code word, and they are applied to the source bits in blocks, hence the name linear block codes. There are block codes that are not linear, but it is difficult to prove that a code is a good one without this property.

Linear block codes are summarized by their symbol alphabets (e.g., binary or ternary) and parameters (n,m,d_{min}) where

1. n is the length of the codeword, in symbols,
2. m is the number of source symbols that will be used for encoding at once,
3. d_{min} is the minimum hamming distance for the code.

There are many types of linear block codes, such as

1. Cyclic codes (e.g., Hamming codes)
2. Repetition codes
3. Parity codes
4. Polynomial codes (e.g., BCH codes)
5. Reed–Solomon codes
6. Algebraic geometric codes
7. Reed–Muller codes
8. Perfect codes.

Block codes are tied to the sphere packing problem, which has received some attention over the years. In two dimensions, it is easy to visualize. Take a bunch of pennies flat on the table and push them together. The result is a hexagon pattern like a bee's nest. But block codes rely on more dimensions which cannot easily be visualized. The powerful Golay code used in deep space communications uses 24 dimensions. If used as a binary code (which it usually is) the dimensions refer to the length of the codeword as defined above.

The theory of coding uses the N-dimensional sphere model. For example, how many pennies can be packed into a circle on a tabletop, or in 3 dimensions, how many marbles can be packed into a globe. Other considerations enter the choice of a code. For example, hexagon packing into the constraint of a rectangular box will leave empty space at the

corners. As the dimensions get larger, the percentage of empty space grows smaller. But at certain dimensions, the packing uses all the space and these codes are the so-called "perfect" codes. The only nontrivial and useful perfect codes are the distance-3 Hamming codes with parameters satisfying $(2^r - 1, 2^r - 1 - r, 3)$, and the [23,12,7] binary and [11,6,5] ternary Golay codes.

Another code property is the number of neighbors a single codeword may have. Again, let's use pennies as an example. First we pack the pennies in a rectangular grid. Each penny will have 4 near neighbors (and 4 at the corners which are farther away). In a hexagon, each penny will have 6 near neighbors. When we increase the dimensions, the number of near neighbors increases very rapidly. The result is the number of ways for noise to make the receiver choose a neighbor (hence an error) grows as well. This is a fundamental limitation of block codes, and indeed all codes. It may be harder to cause an error to a single neighbor, but the number of neighbors can be large enough so the total error probability actually suffers.

Properties of linear block codes are used in many applications. for example Syndrome-Coset uniqueness property of linear block codes is used in Trellis shaping, one of the best known shaping codes. This same property is used in Sensor networks for distributed source coding

Convolutional codes

The idea behind a convolutional code is to make every codeword symbol be the weighted sum of the various input message symbols. This is like convolution used in LTI systems to find the output of a system, when you know the input and impulse response.

So we generally find the output of the system convolutional encoder, which is the convolution of the input bit, against the states of the convolution encoder, registers.

Fundamentally, convolutional codes do not offer more protection against noise than an equivalent block code. In many cases, they generally offer greater simplicity of implementation over a block code of equal power. The encoder is usually a simple circuit which has state memory and some feedback logic, normally XOR gates. The decoder can be implemented in software or firmware.

The Viterbi algorithm is the optimum algorithm used to decode convolutional codes. There are simplifications to reduce the computational load. They rely on searching only the most likely paths. Although not optimum, they have generally found to give good results in the lower noise environments.

Convolutional codes are used in voiceband modems (V.32, V.17, V.34) and in GSM mobile phones, as well as satellite and military communication devices.

Other applications of coding theory

Another concern of coding theory is designing codes that help synchronization. A code may be designed so that a phase shift can be easily detected and corrected and that multiple signals can be sent on the same channel.

Another application of codes, used in some mobile phone systems, is code-division multiple access (CDMA). Each phone is assigned a code sequence that is approximately uncorrelated with the codes of other phones. When transmitting, the code word is used to modulate the data bits representing the voice message. At the receiver, a demodulation process is performed to recover the data. The properties of this class of codes allow many users (with different codes) to use the same radio channel at the same time. To the receiver, the signals of other users will appear to the demodulator only as a low-level noise.

Another general class of codes are the automatic repeat-request (ARQ) codes. In these codes the sender adds redundancy to each message for error checking, usually by adding check bits. If the check bits are not consistent with the rest of the message when it arrives, the receiver will ask the sender to retransmit the message. All but the simplest wide area network protocols use ARQ. Common protocols include SDLC (IBM), TCP (Internet), X.25 (International) and many others. There is an extensive field of research on this topic because of the problem of matching a rejected packet against a new packet. Is it a new one or is it a retransmission?

Group Testing

Group testing uses codes in a different way. Consider a large group of items in which a very few are different in a particular way (for eg. Defective products or infected test subjects). The idea of group testing is to determine which items are "different" by using as few tests as possible. The origin of the problem has its roots in the Second World War when the United States Army Air Forces needed to test it's soldiers for Syphilis. It originated from a ground-breaking paper by Robert Dorfman.

Analog coding

Information is encoded analogously in the neural networks of brains, in analog signal processing, and analog electronics. Aspects of analog coding include analog error correction, analog data compression. analog encryption

Neural coding

Neural coding is a neuroscience-related field concerned with how sensory and other information is represented in the brain by networks of neurons. The main goal of studying neural coding is to characterize the relationship between the stimulus and the individual or ensemble neuronal responses and the relationship among electrical activity

of the neurons in the ensemble. It is thought that neurons can encode both digital and analog information, and that neurons follow the principles of information theory and compress information, and detect and correct errors in the signals that are sent throughout the brain and wider nervous system.

Chapter 5

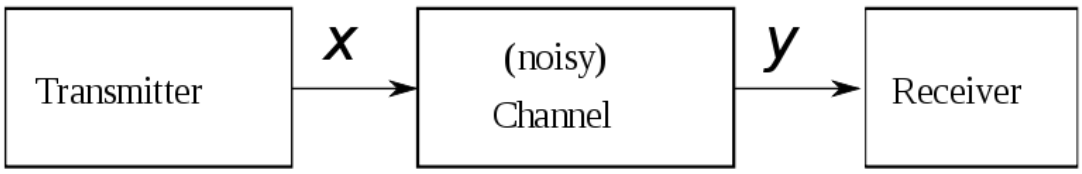
Channel Capacity and Binary Symmetric Channel

Channel capacity

In electrical engineering, computer science and information theory, **channel capacity** is the tightest upper bound on the amount of information that can be reliably transmitted over a communications channel. By the noisy-channel coding theorem, the channel capacity of a given channel is the limiting information rate (in units of information per unit time) that can be achieved with arbitrarily small error probability.

Information theory, developed by Claude E. Shannon during World War II, defines the notion of channel capacity and provides a mathematical model by which one can compute it. The key result states that the capacity of the channel, as defined above, is given by the maximum of the mutual information between the input and output of the channel, where the maximization is with respect to the input distribution.

Formal definition



Let X represent the space of signals that can be transmitted, and Y the space of signals received, during a block of time over the channel. Let

$$p_{Y|X}(y|x)$$

be the conditional distribution function of Y given X. Treating the channel as a known statistic system, $p_{Y|X}(y \mid x)$ is an inherent fixed property of the communications channel (representing the nature of the noise in it). Then the joint distribution

$$p_{X,Y}(x,y)$$

of X and Y is completely determined by the channel and by the choice of

$$p_X(x) = \int_y p_{X,Y}(x,y)\,dy$$

the marginal distribution of signals we choose to send over the channel. The joint distribution can be recovered by using the identity

$$p_{X,Y}(x,y) = p_{Y|X}(y|x)\,p_X(x)$$

Under these constraints, next maximize the amount of information, or the message, that one can communicate over the channel. The appropriate measure for this is the mutual information $I(X;Y)$, and this maximum mutual information is called the **channel capacity** and is given by

$$C = \sup_{p_X} I(X;Y)$$

Noisy-channel coding theorem

The noisy-channel coding theorem states that for any $\varepsilon > 0$ and for any rate R less than the channel capacity C, there is an encoding and decoding scheme that can be used to ensure that the probability of block error is less than ε for a sufficiently long code. Also, for any rate greater than the channel capacity, the probability of block error at the receiver goes to one as the block length goes to infinity.

Example application

An application of the channel capacity concept to an additive white Gaussian noise (AWGN) channel with B Hz bandwidth and signal-to-noise ratio S/N is the Shannon–Hartley theorem:

$$C = B \log\left(1 + \frac{S}{N}\right)$$

C is measured in bits per second if the logarithm is taken in base 2, or nats per second if the natural logarithm is used, assuming B is in hertz; the signal and noise powers S and N are measured in watts or volts^2, so the signal-to-noise ratio here is expressed as a power ratio, *not* in decibels (dB); since figures are often cited in dB, a conversion may be needed. For example, 30 dB is a power ratio of $10^{30\,/\,10} = 10^3 = 1000$.

Channel capacity in wireless communications

This section focuses on the single-antenna, point-to-point scenario.

AWGN channel

If the average received power is $\bar{P}$[W] and the noise power spectral density is N_0 [W/Hz], the AWGN channel capacity is

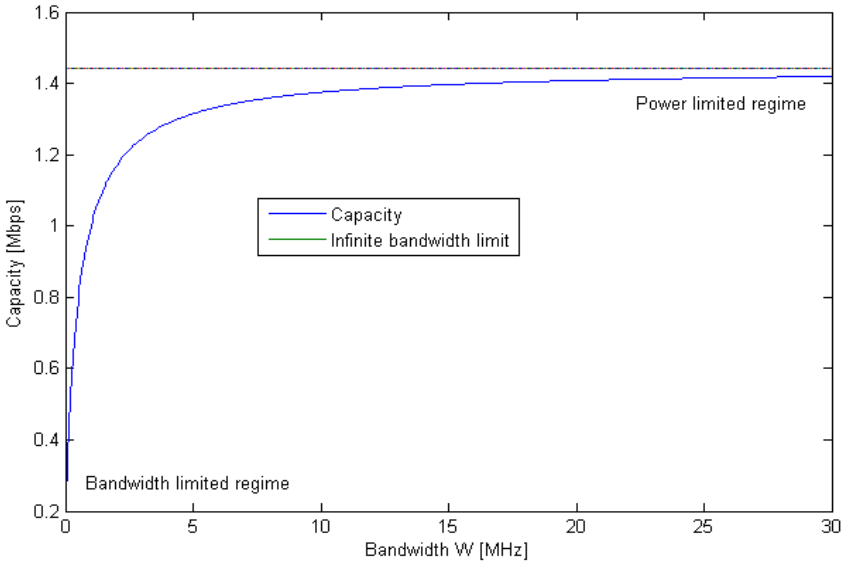
$$C_{awgn} = W \log_2 \left(1 + \frac{\bar{P}}{N_0 W}\right)_{\text{[bits/Hz]}},$$

where $\frac{\bar{P}}{N_0 W}$ is the received signal-to-noise ratio (SNR).

When the SNR is large (SNR >> 0 dB), the capacity $C \approx W \log_2 \frac{\bar{P}}{N_0 W}$ is logarithmic in power and approximately linear in bandwidth. This is called the *bandwidth-limited regime*.

When the SNR is small (SNR << 0 dB), the capacity $C \approx \frac{\bar{P}}{N_0} \log_2 e$ is linear in power but insensitive to bandwidth. This is called the *power-limited regime*.

The bandwidth-limited regime and power-limited regime are illustrated in the figure.



AWGN channel capacity with the power-limited regime and bandwidth-limited regime indicated. Here, $\frac{\bar{P}}{N_o} = 10^6$.

Frequency-selective channel

The capacity of the frequency-selective channel is given by so-called waterfilling power allocation,

$$C_{N_c} = \sum_{n=0}^{N_c-1} \log_2 \left(1 + \frac{P_n^* |\bar{h}_n|^2}{N_0}\right),$$

where $P_n^* = \max\left(\left(\frac{1}{\lambda} - \frac{N_0}{|\bar{h}_n|^2}\right), 0\right)$ and $|\bar{h}_n|^2$ is the gain of subchannel *n*, with λ chosen to meet the power constraint.

Slow-fading channel

In a slow-fading channel, where the coherence time is greater than the latency requirement, there is no definite capacity as the maximum rate of reliable communications supported by the channel, $\log_2(1 + |h|^2 SNR)$, depends on the random channel gain $|h|^2$. If the transmitter encodes data at rate *R* [bits/s/Hz], there is a certain probability that the decoding error probability cannot be made arbitrarily small,

$$p_{out} = \mathbb{P}(\log(1 + |h|^2 SNR) < R),$$

in which case the system is said to be in outage. With a non-zero probability that the channel is in deep fade, the capacity of the slow-fading channel in strict sense is zero. However, it is possible to determine the largest value of *R* such that the outage probability p_{out} is less than ε. This value is known as the ε-outage capacity.

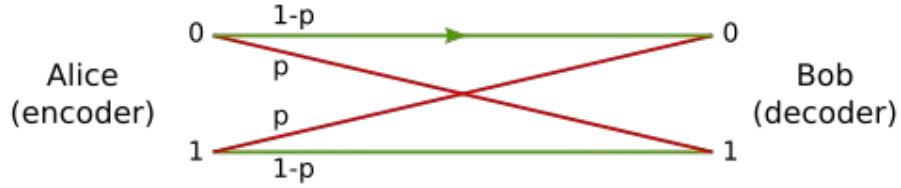
Fast-fading channel

In a fast-fading channel, where the latency requirement is greater than the coherence time and the codeword length spans many coherence periods, one can average over many independent channel fades by coding over a large number of coherence time intervals. Thus, it is possible to achieve a reliable rate of communication of $\mathbb{E}(\log_2(1 + |h|^2 SNR))$ [bits/s/Hz] and it is meaningful to speak of this value as the capacity of the fast-fading channel.

Binary symmetric channel

A **binary symmetric channel** (or BSC) is a common communications channel model used in coding theory and information theory. In this model, a transmitter wishes to send a bit (a zero or a one), and the receiver receives a bit. It is assumed that the bit is *usually* transmitted correctly, but that it will be "flipped" with a small probability (the "crossover probability"). This channel is used frequently in information theory because it is one of the simplest channels to analyze.

Description



The BSC is a *binary channel*; that is, it can transmit only one of two symbols (usually called 0 and 1). (A non-binary channel would be capable of transmitting more than 2 symbols, possibly even an infinite number of choices.) The transmission is not perfect, and occasionally the receiver gets the wrong bit.

This channel is often used by theorists because it is one of the simplest noisy channels to analyze. Many problems in communication theory can be reduced to a BSC. On the other hand, being able to transmit effectively over the BSC can give rise to solutions for more complicated channels.

Definition

A **binary symmetric channel with crossover probability *p*** denoted by BSC_p, is a channel with binary input and binary output and probability of error p; that is, if X is the transmitted random variable and Y the received variable, then the channel is characterized by the conditional probabilities

$$\Pr(Y = 0 \mid X = 0) = 1 - p$$
$$\Pr(Y = 0 \mid X = 1) = p$$
$$\Pr(Y = 1 \mid X = 0) = p$$
$$\Pr(Y = 1 \mid X = 1) = 1 - p$$

It is assumed that $0 \leq p \leq 1/2$. If $p > 1/2$, then the receiver can swap the output (interpret 1 when it sees 0, and vice versa) and obtain an equivalent channel with crossover probability $1 - p \leq 1/2$.

Capacity of BSC_p

The capacity of the channel is $1 - H(p)$, where $H(p)$ is the binary entropy function.

The converse can be shown by a sphere packing argument. Given a codeword, there are roughly $2^{n\,\mathrm{H}(p)}$ typical output sequences. There are 2^n total possible outputs, and the input chooses from a codebook of size 2^{nR}. Therefore, the receiver would choose to partition the space into "spheres" with $2^n / 2^{nR} = 2^{n(1-R)}$ potential outputs each. If $R > 1 - H(p)$, then the spheres will be packed too tightly asymptotically and the receiver will not be able to identify the correct codeword with vanishing probability.

Shannon's channel capacity theorem for BSC_p

Shannon's noisy coding theorem is general for all kinds of channels. We consider a special case of this theorem for a binary symmetric channel with an error probability p.

Noisy coding theorem for BSC_p

We denote (and would continue to denote) noise e from BSC_p as "$e \in BSC_p$". The noise e is essentially a random variable with each of its indexes being a 1 with probability p and a 0 with probability $1 - p$.

Theorem 1 For all $p < \frac{1}{2}$ and all ε such that $0 < \epsilon \leq \frac{1}{2} - p$, there exists a pair of encoding and decoding functions E: $\{0,1\}^k \to \{0,1\}^n$ and D: $\{0,1\}^n \to \{0,1\}^k$ respectively, and $k \leq \lfloor (1 - H(p + \varepsilon)n) \rfloor$ such that every message $m \in \{0,1\}^k$ has the following property: $\Pr_{e \in BSC_p}[D(E(m) + e) \neq m] \leq 2^{-\delta n}$ for a sufficient large integer n.

What this theorem actually implies is, a message when picked from $\{0,1\}^k$, encoded with a random encoding function E, and send across a noisy BSC_p, there is a very high probability of recovering the original message by decoding, if k or in effect the rate of the channel is bounded by the quantity stated in the theorem. The decoding error probability is exponentially small.

We shall now prove Theorem 1.

Proof We shall first describe the encoding function E and the decoding function D used in the theorem. Its good to state here that we would be using the probabilistic method to

prove this theorem. Shannon's theorem was one of the earliest applications of this method.

Encoding function *E*: The encoding function E: $\{0,1\}^k \to \{0,1\}^n$ is selected at random. This means, given any message $m \in \{0,1\}^k$, we choose $E(m) \in \{0,1\}^n$ uniformly and independently at random.

Decoding function *D*: The decoding function D: $\{0,1\}^n \to \{0,1\}^k$ is given any received codeword $y \in \{0,1\}^n$, we find the message $m \in \{0,1\}^k$ such that $\Delta(y,E(m))$ is minimum (with ties broken arbitrarily). This kind of a decoding function is called a *maximum likelihood decoding (MLD)* function.

Now first we show, for a fixed $m \in \{0,1\}^k$ and E chosen randomly, the probability of failure over BSC_p noise is exponentially small. At this point, the proof works for a fixed message m. Next we extend this result to work for *all* m. We achieve this by eliminating half of the codewords from the code with the argument that the proof for the decoding error probability holds for at least half of the codewords. The latter method is called expurgation. This gives the total process the name *random coding with expurgation*.

A high level proof: Given a fixed message $m \in \{0,1\}^k$, we need to estimate the expected value of the probability of the received codeword along with the noise does not give back m on decoding. That is to say, we need to estimate: $\mathbb{E}_E[\Pr_{e \in BSC_p}[D(E(m)+e) \neq m]]$.

Let y be the received codeword. In order for the decoded codeword $D(y)$ not to be equal to the message m, one of the following events must occur:

- y does not lie within the Hamming ball of radius $(p + \varepsilon)$ for any ε greater than 0, centered at $E(m)$. This condition is mainly used to make the calculations easier.
- There is another message $m' \in \{0,1\}^k$ such that $\Delta(y, E(m')) \leq \Delta(y, E(m))$. In other words the errors due to noise take the transmitted codeword closer to another encoded message.

We can apply Chernoff bound to ensure the non occurrence of the first event. By applying Chernoff bound we have, $Pr_{e \in BSC_p}[\Delta(y, E(m)) > (p+\epsilon)] \leq 2^{-\epsilon^2 n}$. Thus, for any $\varepsilon > 0$, we can pick n to be large enough to make the above probability exponentially small.

As for the second event, we note that the probability that $E(m') \in B(y, (p+\epsilon)n)$ is $Vol(B(y,(p + \varepsilon)n) / 2^n$ where $B(x,r)$ is the Hamming ball of radius r centered at vector x

and $Vol(B(x,r))$ is its volume. Using approximation to estimate the number of codewords in the Hamming ball, we have $Vol(B(y,(p+\epsilon)n)/2^n \approx 2^{H(p)n}$. Hence the above probability amounts to $2^{H(p)n} / 2^n = 2^{H(p)n-n}$. Now using union bound, we can upper bound the existence of such an $m' \in \{0,1\}^k$ by $\leq 2^{k+H(p)n-n}$ which is $2^{-\Omega(n)}$, as desired by the choice of k.

A detailed proof: From the above analysis, we calculate the probability of the event that the decoded codeword plus the channel noise is not the same as the original message sent. We shall introduce some symbols here. Let $p(y \mid E(m))$ denote the probability of receiving codeword y given that codeword $E(m)$ was sent. Denote $B(E(m),(p + \varepsilon)n)$ by Ball.

$$\Pr_{e\in BSC_p}[D(E(m)+e) \neq m] = \sum_{y\in\{0,1\}^n} p(y|E(m))\cdot 1_{D(y)\neq m} \leq \sum_{y\notin \text{Ball}} p(y|E(m))\cdot 1_{D(y)\neq m} + \sum_{y\in\text{Ball}} p(y|E(m))\cdot 1_{D(y)\neq m} \leq 2^{-\epsilon^2 n} + \sum_{y\in\text{Ball}} p(y|E(m))\cdot 1_{D(y)\neq m}.$$

We get the last inequality by our analysis using the Chernoff bound above. Now taking expectation on both sides we have, $\mathbb{E}_E[Pr_{e\in BSC_p}[D(E(m) + e) \neq m]] \leq 2^{-\epsilon^2 n} + \sum_{y\in\text{Ball}} p(y \mid E(m)).\mathbb{E}[1_{D(y)\neq m}]$.

$$\sum_{y\in\text{Ball}} p(y|E(m)) \leq 1$$

Now we have . This just says, that the quantity $\mathbb{E}[1_{D(y)\neq m}] \leq 2^{k+H(p+\epsilon)n-n}$, again from the analysis in the higher level proof above. Hence, taking everything together we have $\mathbb{E}_E[\Pr_{e\in BSC_p}[D(E(m)+e) \neq m]] \leq 2^{-\epsilon^2 n} + 2^{k+H(p+\epsilon)n-n} \leq 2^{-\delta n}$, by appropriately choosing the value of δ. Since the above bound holds for **each** message, we have $\mathbb{E}_m[\mathbb{E}_E[\Pr_{e\in BSC_p}[D(E(m)+e)] \neq m]] \leq 2^{-\delta n}$. Now we can change the order of summation in the expectation with respect to the message and the choice of the encoding function E, without loss of generality. Hence we have $\mathbb{E}_E[\mathbb{E}_m[\Pr_{e\in BSC_p}[D(E(m)+e)] \neq m]] \leq 2^{-\delta n}$. Hence in conclusion, by probabilistic method, we have some encoding function E^* and a corresponding decoding function D^* such that $\mathbb{E}_m[\Pr_{e\in BSC_p}[D^*(E^*(m)+e) \neq m]] \leq 2^{-\delta n}$.

At this point, the proof works for a fixed message m. But we need to make sure that the above bound holds for **all** the messages m **simultaneously**. For that, let us sort the 2^k messages by their decoding error probabilities. Now by applying Markov's inequality, we can show the decoding error probability for the first 2^{k-1} messages to be at most $2.2^{-\delta n}$. Thus in order to confirm that the above bound to hold for **every** message m, we could just trim off the last 2^{k-1} messages from the sorted order. This essentially gives us another encoding function E' with a corresponding decoding function D' with a decoding error

probability of at most $2^{-\delta n+1}$ with the same rate. Taking δ' to be equal to $\delta - \frac{1}{n}$ we bound the decoding error probability to $2^{-\delta' n}$. This expurgation process completes the proof of Theorem 1.

Converse of Shannon's capacity theorem

The converse of the capacity theorem essentially states that $1 - H(p)$ is the best rate one can achieve over a binary symmetric channel. Formally the theorem states:

Theorem 2 If $k \geq \lceil (1 - H(p + \varepsilon)n) \rceil$ then the following is true for every encoding and decoding function E: $\{0,1\}^k \longrightarrow \{0,1\}^n$ and D: $\{0,1\}^n \longrightarrow \{0,1\}^k$ respectively:

$Pr_{e \in BSC_p}[D(E(m) + e) \neq m] \geq \frac{1}{2}$.

For a detailed proof of this theorem, the reader is asked to refer to the bibliography. The intuition behind the proof is however showing the number of errors to grow rapidly as the rate grows beyond the channel capacity. The idea is the sender generates messages of dimension k, while the channel BSC_p introduces errors in it while transmission. When the capacity of the channel is $H(p)$, the number of errors is typically $2^{H(p+\varepsilon)n}$ for a code of block length n. The maximum number of messages is 2^k. The output of the channel on the other hand has 2^n possible values. If there is any confusion between any two messages, it is likely that $2^k 2^{H(p+\epsilon)n} \geq 2^n$. Hence we would have $k \geq \lceil (1 - H(p+\epsilon)n) \rceil$, a case we would like to avoid to keep the decoding error probability exponentially small.

Codes for BSC_p

Very recently, a lot of work has been done and is also being done to design explicit error-correcting codes to achieve the capacities of several standard communication channels. The motivation behind designing such codes is to relate the rate of the code with the fraction of errors which it can correct.

The approach behind the design of codes which meet the channel capacities of *BSC*, *BEC* have been to correct a lesser number of errors with a high probability, and to achieve the highest possible rate. Shannon's theorem gives us the best rate which could be achieved over a BSC_p, but it does not give us an idea of any explicit codes which achieve that rate. In fact such codes are typically constructed to correct only a small fraction of errors with a high probability, but achieve a very good rate. The first such code was due to George D. Forney in 1966. The code is a concatenated code by concatenating two different kinds of codes. We shall discuss the construction Forney's code for the Binary Symmetric Channel and analyze its rate and decoding error probability briefly here. Various explicit codes for achieving the capacity of the Binary Erasure Channel have also come up recently.

Forney's code for BSC_p

Forney constructed a concatenated code $C^* = C_{\text{out}} \circ C_{\text{in}}$ to achieve the capacity of Theorem 1 for BSC_p. In his code,

- The outer code C_{out} is a code of block length N and rate $1 - \frac{\epsilon}{2}$ over the field F_{2^k}, and $k = O(logN)$. Additionally, we have a decoding algorithm D_{out} for C_{out} which can correct up to γ fraction of worst case errors and runs in $t_{\text{out}}(N)$ time.
- The inner code C_{in} is a code of block length n, dimension k, and a rate of $1 - H(p) - \frac{\epsilon}{2}$. Additionally, we have a decoding algorithm D_{in} for C_{in} with a decoding error probability of at most $\frac{\gamma}{2}$ over BSC_p and runs in $t_{\text{in}}(N)$ time.

For the outer code C_{out}, a Reed-Solomon code would have been the first code to have come in mind. However, we would see that the construction of such a code cannot be done in polynomial time. This is why a binary linear code is used for C_{out}.

For the inner code code C_{in} we find an linear code by exhaustively searching from the linear code of block length n and dimension k, whose rate meets the capacity of BSC_p, by Theorem 1.

The rate $R(C^*) = R(C_{\text{in}}) \times R(C_{\text{out}}) = (1 - \frac{\epsilon}{2})(1 - H(p) - \frac{\epsilon}{2}) \geq 1$ – $H(p) - \varepsilon$ which almost meets the BSC_p capacity. We further note that the encoding and decoding of C^* can be done in polynomial time with respect to N. As a matter of fact, encoding C^* takes time $O(N^2) + O(Nk^2) = O(N^2)$. Further, the decoding algorithm described takes time $Nt_{\text{in}}(k) + t_{\text{out}}(N) = N^{O(1)}$ as long as $t_{\text{out}}(N) = N^{O(1)}$; and $t_{\text{in}}(k) = 2^{O(k)}$.

Decoding error probability for C^*

A natural decoding algorithm for C^* is to:

- Assume $y'_i = D_{\text{in}}(y_i), \quad i \in (0, N)$
- Execute D_{out} on $y' = (y'_1 \ldots y'_N)$

Note that each block of code for C_{in} is considered a symbol for C_{out}. Now since the probability of error at any index i for D_{in} is at most $\frac{\gamma}{2}$ and the errors in BSC_p are independent, the expected number of errors for D_{in} is at most $\frac{\gamma N}{2}$ by linearity of expectation. Now applying Chernoff bound, we have bound error probability of more

than γN errors occurring to be $e^{\frac{-\gamma N}{6}}$. Since the outer code C_{out} can correct at most γN errors, this is the decoding error probability of C^*. This when expressed in asymptotic terms, gives us an error probability of $2^{-\Omega(\gamma N)}$. Thus the achieved decoding error probability of C^* is exponentially small as Theorem 1.

We have given a general technique to construct C^*. For more detailed descriptions on C_{in} and *Cout* please read the following references. Recently a few other codes have also being constructed for achieving the capacities. Its worth mentioning that LDPC codes are now being considered for this purpose for their faster decoding time. Please refer to the book by Richardson and Urbanke cited in the reference to know more about such codes.

Chapter 6

Kullback–Leibler Divergence

In probability theory and information theory, the **Kullback–Leibler divergence** (also **information divergence**, **information gain**, **relative entropy**, or **KLIC**) is a non-symmetric measure of the difference between two probability distributions *P* and *Q*. KL measures the expected number of extra bits required to code samples from *P* when using a code based on *Q*, rather than using a code based on *P*. Typically *P* represents the "true" distribution of data, observations, or a precise calculated theoretical distribution. The measure *Q* typically represents a theory, model, description, or approximation of *P*.

Although it is often intuited as a distance metric, the KL divergence is not a true metric — for example, it's not symmetric: the KL from *P* to *Q* is not necessarily the same as the KL from *Q* to *P*.

KL divergence is a special case of a broader class of divergences called *f*-divergences. Originally introduced by Solomon Kullback and Richard Leibler in 1951 as the **directed divergence** between two distributions, it is not the same as a divergence in calculus. However, the KL divergence can be derived from the Bregman divergence.

Definition

For probability distributions *P* and *Q* of a discrete random variable their K–L divergence is defined to be

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

In words, it is the average of the logarithmic difference between the probabilities *P* and *Q*, where the average is taken using the probabilities *P*. The K-L divergence is only defined if *P* and *Q* both sum to 1 and if $Q(i) > 0$ for any *i* such that $P(i) > 0$. If the quantity 0log0 appears in the formula, it is interpreted as zero.

For distributions *P* and *Q* of a continuous random variable, KL-divergence is defined to be the integral:

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx,$$

where *p* and *q* denote the densities of *P* and *Q*.

More generally, if *P* and *Q* are probability measures over a set *X*, and *Q* is absolutely continuous with respect to *P*, then the Kullback–Leibler divergence from *P* to *Q* is defined as

$$D_{\mathrm{KL}}(P\|Q) = -\int_X \log \frac{dQ}{dP}\, dP,$$

where $\frac{dQ}{dP}$ is the Radon–Nikodym derivative of *Q* with respect to *P,* and provided the expression on the right-hand side exists. Likewise, if *P* is absolutely continuous with respect to *Q,* then

$$D_{\mathrm{KL}}(P\|Q) = \int_X \log \frac{dP}{dQ}\, dP = \int_X \frac{dP}{dQ} \log \frac{dP}{dQ}\, dQ,$$

which we recognize as the entropy of *P* relative to *Q*. Continuing in this case, if μ is any measure on *X* for which $p = \frac{dP}{d\mu}$ and $q = \frac{dQ}{d\mu}$ exist, then the Kullback–Leibler divergence from *P* to *Q* is given as

$$D_{\mathrm{KL}}(P\|Q) = \int_X p \log \frac{p}{q}\, d\mu.$$

The logarithms in these formulae are taken to base 2 if information is measured in units of bits, or to base *e* if information is measured in nats. Most formulas involving the KL divergence hold irrespective of log base.

Here, this will be referred to as the divergence from P to Q, although some authors call it the divergence "from Q to P" and others call it the divergence "between P and Q" (though note it is not symmetric as this latter terminology implies). Care must be taken due to the lack of standardization in terminology.

Motivation

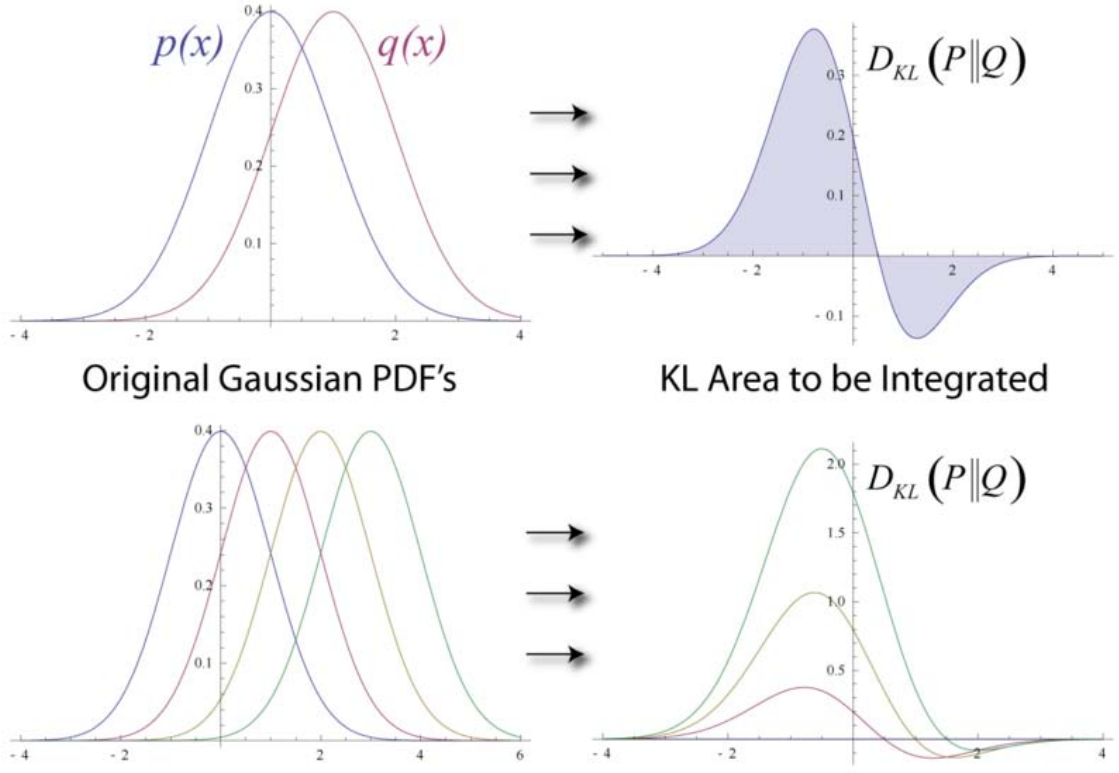


Illustration of the Kullback–Leibler (KL) divergence for two normal Gaussian distributions. Note the typical asymmetry for the KL divergence is clearly visible.

In information theory, the Kraft–McMillan theorem establishes that any directly decodable coding scheme for coding a message to identify one value x_i out of a set of possibilities X can be seen as representing an implicit probability distribution $q(x_i) = 2^{-l_i}$ over X, where l_i is the length of the code for x_i in bits. Therefore, KL divergence can be interpreted as the expected extra message-length per datum that must be communicated if a code that is optimal for a given (wrong) distribution Q is used, compared to using a code based on the true distribution P.

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= -\sum_x p(x)\log q(x) + \sum_x p(x)\log p(x) \\ &= H(P,Q) - H(P) \end{aligned}$$

where $H(P,Q)$ is called the cross entropy of P and Q, and $H(P)$ is the entropy of P.

Properties

The Kullback–Leibler divergence is always non-negative,

$$D_{\mathrm{KL}}(P\|Q) \geq 0,$$

a result known as Gibbs' inequality, with $D_{\mathrm{KL}}(P\|Q)$ zero if and only if $P = Q$. The entropy *H(P)* thus sets a minimum value for the cross-entropy *H(P,Q)*, the expected number of bits required when using a code based on *Q* rather than *P*; and the KL divergence therefore represents the expected number of extra bits that must be transmitted to identify a value *x* drawn from *X*, if a code is used corresponding to the probability distribution *Q*, rather than the "true" distribution *P*.

The Kullback–Leibler divergence remains well-defined for continuous distributions, and furthermore is invariant under parameter transformations. It can therefore be seen as in some ways a more fundamental quantity than some other properties in information theory (such as self-information or Shannon entropy), which can become undefined or negative for non-discrete probabilities.

The Kullback–Leibler divergence is additive for independent distributions in much the same way as Shannon entropy. If P_1,P_2 are independent distributions, with the joint distribution $P(x,y) = P_1(x)P_2(y)$, and Q,Q_1,Q_2 likewise, then

$$D_{\mathrm{KL}}(P\|Q) = D_{\mathrm{KL}}(P_1\|Q_1) + D_{\mathrm{KL}}(P_2\|Q_2).$$

Relation to metrics

One might be tempted to call it a "distance metric" on the space of probability distributions, but this would not be correct as the Kullback–Leibler divergence is not symmetric – that is, $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$, – nor does it satisfy the triangle inequality. Still, being a premetric, it generates a topology on the space of generalized probability distributions, of which probability distributions proper are a special case. More concretely, if $\{P_1, \cdots, P_n\}$ is a sequence of distributions such that

$$\lim_{n\to\infty} D_{\mathrm{KL}}(P_n\|Q) = 0$$

then it is said that $P_n \xrightarrow{D} Q$. Pinsker's inequality entails that $P_n \xrightarrow{\mathrm{D}} P \Rightarrow P_n \xrightarrow{\mathrm{TV}} P$, where the latter stands for the usual convergence in total variation.

Following Renyi (1970, 1961) the term is sometimes also called the **information gain** about *X* achieved if *P* can be used instead of *Q*. It is also called the **relative entropy**, for using *Q* instead of *P*.

Relation to other quantities of information theory

Many of the other quantities of information theory can be interpreted as applications of the KL divergence to specific cases.

The self-information,

$$I(m) = D_{\mathrm{KL}}(\delta_{im} \| \{p_i\}),$$

is the KL divergence of the probability distribution *P(i)* from a Kronecker delta representing certainty that *i=m* — i.e. the number of extra bits that must be transmitted to identify *i* if only the probability distribution *P(i)* is available to the receiver, not the fact that *i=m*.

The mutual information,

$$\begin{aligned} I(X;Y) &= D_{\mathrm{KL}}(P(X,Y) \| P(X)P(Y)) \\ &= \mathbb{E}_X\{D_{\mathrm{KL}}(P(Y|X) \| P(Y))\} \\ &= \mathbb{E}_Y\{D_{\mathrm{KL}}(P(X|Y) \| P(X))\} \end{aligned}$$

is the KL divergence of the product *P(X)P(Y)* of the two marginal probability distributions from the joint probability distribution *P(X,Y)* — i.e. the expected number of extra bits that must be transmitted to identify *X* and *Y* if they are coded using only their marginal distributions instead of the joint distribution. Equivalently, if the joint probability *P(X,Y) is* known, it is the expected number of extra bits that must on average be sent to identify *Y* if the value of *X* is not already known to the receiver.

The Shannon entropy,

$$\begin{aligned} H(X) &= (\mathrm{i})\, \mathbb{E}_x\{I(x)\} \\ &= (\mathrm{ii})\, \log N - D_{\mathrm{KL}}(P(X) \| P_U(X)) \end{aligned}$$

is the number of bits which would have to be transmitted to identify *X* from *N* equally likely possibilities, *less* the KL divergence of the uniform distribution $P_{\mathrm{U}}(X)$ from the true distribution *P(X)* — i.e. *less* the expected number of bits saved, which would have had to be sent if the value of *X* were coded according to the uniform distribution $P_{\mathrm{U}}(X)$ rather than the true distribution *P(X)*.

The conditional entropy,

$$\begin{aligned} H(X|Y) &= \log N - D_{\mathrm{KL}}(P(X,Y) \| P_U(X)P(Y)) \\ &= (\mathrm{i})\ \log N - D_{\mathrm{KL}}(P(X,Y) \| P(X)P(Y)) - D_{\mathrm{KL}}(P(X) \| P_U(X)) \\ &= H(X) - I(X;Y) \\ &= (\mathrm{ii})\ \log N - \mathbb{E}_Y\{D_{\mathrm{KL}}(P(X|Y) \| P_U(X))\} \end{aligned}$$

is the number of bits which would have to be transmitted to identify *X* from *N* equally likely possibilities, *less* the KL divergence of the product distribution $P_{\mathrm{U}}(X)\ P(Y)$ from the true joint distribution *P(X,Y)* — i.e. *less* the expected number of bits saved which

would have had to be sent if the value of X were coded according to the uniform distribution $P_U(X)$ rather than the conditional distribution $P(X|Y)$ of X given Y.

The cross entropy between two probability distributions measures the average number of bits needed to identify an event from a set of possibilities, if a coding scheme is used based on a given probability distribution q, rather than the "true" distribution p. The cross entropy for two distributions p and q over the same probability space is thus defined as follows:

$$\mathrm{H}(p,q) = \mathrm{E}_p[-\log q] = \mathrm{H}(p) + D_{\mathrm{KL}}(p\|q).$$

KL divergence and Bayesian updating

In Bayesian statistics the KL divergence can be used as a measure of the information gain in moving from a prior distribution to a posterior distribution. If some new fact $Y = y$ is discovered, it can be used to update the probability distribution for X from $p(x|I)$ to a new posterior probability distribution $p(x|y,I)$ using Bayes' theorem:

$$p(x|y,I) = \frac{p(y|x,I)p(x|I)}{p(y|I)}$$

This distribution has a new entropy

$$H(p(\cdot|y,I)) = \sum_x p(x|y,I)\log p(x|y,I),$$

which may be less than or greater than the original entropy $H(p(\cdot|I))$. However, from the standpoint of the new probability distribution one can estimate that to have used the original code based on $p(x|I)$ instead of a new code based on $p(x|y,I)$ would have added an expected number of bits

$$D_{\mathrm{KL}}\big(p(\cdot|y,I)\big\|p(\cdot|I)\big) = \sum_x p(x|y,I)\log\frac{p(x|y,I)}{p(x|I)}$$

to the message length. This therefore represents the amount of useful information, or information gain, about X, that we can estimate has been learned by discovering $Y = y$.

If a further piece of data, $Y_2 = y_2$, subsequently comes in, the probability distribution for x can be updated further, to give a new best guess $p(x|y_1,y_2,I)$. If one reinvestigates the information gain for using $p(x|y_1,I)$ rather than $p(x|I)$, it turns out that it may be either greater or less than previously estimated:

$$\sum_x p(x|y_1,y_2,I)\log\frac{p(x|y_1,y_2,I)}{p(x|I)} \quad \text{may be} \leq \text{or} > \text{than}$$
$$\sum_x p(x|y_1,I)\log\frac{p(x|y_1,I)}{p(x|I)}$$

and so the combined information gain does *not* obey the triangle inequality:

$$D_{\mathrm{KL}}\big(p(\cdot|y_1,y_2,I)\big\|p(\cdot|I)\big) \quad \text{may be} <, = \text{or} > \text{than}$$
$$D_{\mathrm{KL}}\big(p(\cdot|y_1,y_2,I)\big\|p(\cdot|y_1,I)\big)+D_{\mathrm{KL}}\big(p(\cdot|y_1,I)\big\|p(x|I)\big)$$

All one can say is that on *average*, averaging using $p(y_2|y_1,x,\mathrm{I})$, the two sides will average out.

Bayesian experimental design

A common goal in Bayesian experimental design is to maximise the expected KL divergence between the prior and the posterior. When posteriors are approximated to be Gaussian distributions, a design maximising the expected KL divergence is called Bayes d-optimal.

Discrimination information

The Kullback–Leibler divergence $D_{\mathrm{KL}}(\, p(x|H_1) \parallel p(x|H_0)\,)$ can also be interpreted as the expected **discrimination information** for H_1 over H_0: the mean information per sample for discriminating in favour of a hypothesis H_1 against a hypothesis H_0, when hypothesis H_1 is true. Another name for this quantity, given to it by I.J. Good, is the expected weight of evidence for H_1 over H_0 to be expected from each sample.

The expected weight of evidence for H_1 over H_0 is **not** the same as the information gain expected per sample about the probability distribution $p(H)$ of the hypotheses,

$$D_{\mathrm{KL}}(\, p(x|H_1) \parallel p(x|H_0)\,) \neq IG = D_{\mathrm{KL}}(\, p(H|\mathrm{x}) \parallel p(H|\mathrm{I})\,).$$

Either of the two quantities can be used as a utility function in Bayesian experimental design, to choose an optimal next question to investigate: but they will in general lead to rather different experimental strategies.

On the entropy scale of *information gain* there is very little difference between near certainty and absolute certainty—coding according to a near certainty requires hardly any more bits than coding according to an absolute certainty. On the other hand, on the logit scale implied by weight of evidence, the difference between the two is enormous – infinite perhaps; this might reflect the difference between being almost sure (on a probabilistic level) that, say, the Riemann hypothesis is correct, compared to being

certain that it is correct because one has a mathematical proof. These two different scales of loss function for uncertainty are *both* useful, according to how well each reflects the particular circumstances of the problem in question.

Principle of minimum discrimination information

The idea of Kullback–Leibler divergence as discrimination information led Kullback to propose the Principle of **Minimum Discrimination Information** (MDI): given new facts, a new distribution *f* should be chosen which is as hard to discriminate from the original distribution f_0 as possible; so that the new data produces as small an information gain $D_{\mathrm{KL}}(f \| f_0)$ as possible.

For example, if one had a prior distribution $p(x,a)$ over x and a, and subsequently learnt the true distribution of a was $u(a)$, the Kullback–Leibler divergence between the new joint distribution for x and a, $q(x|a)\,u(a)$, and the earlier prior distribution would be:

$$D_{\mathrm{KL}}(q(x|a)u(a)||p(x,a)) = \mathbb{E}_{u(a)}\{D_{\mathrm{KL}}(q(x|a)||p(x|a))\} + D_{\mathrm{KL}}(u(a)||p(a)),$$

i.e. the sum of the KL divergence of $p(a)$ the prior distribution for a from the updated distribution $u(a)$, plus the expected value (using the probability distribution $u(a)$) of the KL divergence of the prior conditional distribution $p(x|a)$ from the new conditional distribution $q(x|a)$. This is minimised if $q(x|a) = p(x|a)$ over the whole support of $u(a)$; and we note that this result incorporates Bayes' theorem, if the new distribution $u(a)$ is in fact a δ function representing certainty that a has one particular value.

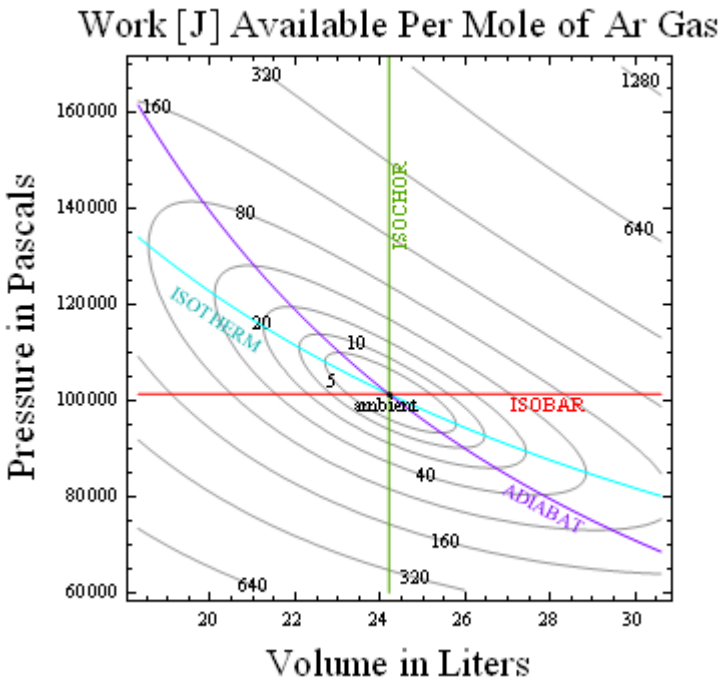
MDI can be seen as an extension of Laplace's Principle of Insufficient Reason, and the Principle of Maximum Entropy of E.T. Jaynes. In particular, it is the natural extension of the principle of maximum entropy from discrete to continuous distributions, for which Shannon entropy ceases to be so useful, but the KL divergence continues to be just as relevant.

In the engineering literature, MDI is sometimes called the **Principle of Minimum Cross-Entropy** (MCE) or **Minxent** for short. This is not entirely helpful. Minimising the KL divergence of m from p with respect to m is equivalent to minimising the cross-entropy of p and m, since

$$H(p,m) = H(p) + D_{\mathrm{KL}}(p\|m),$$

which is appropriate if one is trying to choose a least 'brain-damaged' approximation to p. However, this is just as often *not* the task one is trying to achieve. Instead, just as often it is m that is some fixed prior reference measure, and p that one is attempting to optimise by minimising $D_{\mathrm{KL}}(p||m)$ subject to some constraint. This has led to some ambiguity in the literature, with some authors attempting to resolve the inconsistency by redefining cross-entropy to be $D_{\mathrm{KL}}(p||m)$, rather than $H(p,m)$.

Relationship to available work



Pressure versus volume plot of available work from a mole of Argon gas relative to ambient, calculated as T_o times KL divergence.

Surprisals add where probabilities multiply. The surprisal for an event of probability p is defined as $s \equiv k \ln[1/p]$. If k is $\{1, 1/\ln 2, 1.38\times10^{-23}\}$ then surprisal is in {nats, bits, or J/K} so that, for instance, there are N bits of surprisal for landing all "heads" on a toss of N coins.

Best-guess states (e.g. for atoms in a gas) are inferred by maximizing the *average-surprisal* S (entropy) for a given set of control parameters (like pressure P or volume V). This constrained entropy maximization, both classically and quantum mechanically, minimizes Gibbs availability in entropy units $A \equiv -k \ln Z$ where Z is a constrained multiplicity or partition function.

When temperature T is fixed, free-energy ($T \times A$) is also minimized. Thus if T, V and number of molecules N are constant, the Helmholtz free energy $F \equiv U - TS$ (where U is energy) is minimized as a system "equilibrates." If T and P are held constant (say during processes in your body), the Gibbs free energy $G \equiv U + PV - TS$ is minimized instead. The change in free energy under these conditions is a measure of available work that might be done in the process. Thus available work for an ideal gas at constant temperature T_o and pressure P_o is $W = \Delta G = NkT_o\mathbf{\Theta}[V/V_o]$ where $V_o = NkT_o/P_o$ and $\mathbf{\Theta}[x] \equiv x - 1 - \ln x \geq 0$.

More generally the work available relative to some ambient is obtained by multiplying ambient temperature T_o by KL-divergence or *net-surprisal* $\Delta I \geq 0$, defined as the average value of $k \ln[p/p_o]$ where p_o is the probability of a given state under ambient conditions. For instance, the work available in equilibrating a monatomic ideal gas to ambient values of V_o and T_o is thus $W = T_o \Delta I$, where KL-divergence $\Delta I = Nk(\mathbf{\Theta}[V/V_o] + {}^3\!/_2\mathbf{\Theta}[T/T_o])$. The resulting contours of constant KL-divergence, at right for a mole of Argon at standard temperature and pressure, for example put limits on the conversion of hot to cold as in flame-powered air-conditioning or in the unpowered device to convert boiling-water to ice-water discussed here. Thus KL-divergence measures thermodynamic availability in bits.

Quantum information theory

For density matrices P and Q on a Hilbert space the K–L divergence (or relative entropy as it is often called in this case) from P to Q is defined to be

$$D_{\mathrm{KL}}(P\|Q) = \mathrm{Tr}(P(\log(P) - \log(Q))).$$

In quantum information science it can also be used as a measure of entanglement in a state.

Relationship between models and reality

Just as KL-divergence of "ambient from actual" measures thermodynamic availability, KL-divergence of "model from reality" is also useful even if the only clues we have about reality are some experimental measurements. In the former case KL-divergence describes *distance to equilibrium* or (when multiplied by ambient temperature) the amount of *available work*, while in the latter case it tells you about surprises that reality has up its sleeve or, in other words, *how much the model has yet to learn*.

Although this tool for evaluating models against systems that are accessible experimentally may be applied in any field, its application to models in ecology via Akaike information criterion are particularly well described in papers and a book by Burnham and Anderson. In a nutshell the KL-divergence of a model from reality may be estimated, to within a constant additive term, by a function (like the squares summed) of the deviations observed between data and the model's predictions. Estimates of such divergence for models that share the same additive term can in turn be used to choose between models.

When trying to fit parametrized models to data there are various estimators which attempt to minimize Kullback–Leibler divergence, such as maximum likelihood and maximum spacing estimators.

Symmetrised divergence

Kullback and Leibler themselves actually defined the divergence as:

$$D_{\mathrm{KL}}(P\|Q) + D_{\mathrm{KL}}(Q\|P)$$

which is symmetric and nonnegative. This quantity has sometimes been used for feature selection in classification problems, where *P* and *Q* are the conditional pdfs of a feature under two different classes.

An alternative is given via the λ divergence,

$$D_\lambda(P\|Q) = \lambda D_{\mathrm{KL}}(P\|\lambda P + (1-\lambda)Q) + (1-\lambda) D_{\mathrm{KL}}(Q\|\lambda P + (1-\lambda)Q),$$

which can be interpreted as the expected information gain about *X* from discovering which probability distribution *X* is drawn from, *P* or *Q*, if they currently have probabilities λ and (1 − λ) respectively.

The value λ = 0.5 gives the Jensen–Shannon divergence, defined by

$$D_{\mathrm{JS}} = \tfrac{1}{2} D_{\mathrm{KL}}\left(P\|M\right) + \tfrac{1}{2} D_{\mathrm{KL}}\left(Q\|M\right)$$

where *M* is the average of the two distributions,

$$M = \tfrac{1}{2}(P+Q).$$

D_{JS} can also be interpreted as the capacity of a noisy information channel with two inputs giving the output distributions *p* and *q*. The Jensen–Shannon divergence is the square of a metric that is equivalent to the Hellinger metric, and the Jensen–Shannon divergence is also equal to one-half the so-called *Jeffreys divergence* (Rubner et al., 2000; Jeffreys 1946).

Relationship to Hellinger distance

If *P* and *Q* are two probability measures, then the squared Hellinger distance is the quantity given by

$$H^2(P,Q) = \frac{1}{2}\int \left(\sqrt{\frac{dP}{d\lambda}} - \sqrt{\frac{dQ}{d\lambda}}\right)^2 d\lambda.$$

Noting that $x - 1 \geq \ln x$, so that in particular, $\sqrt{x} - 1 \geq \frac{1}{2}\ln(x)$, we see that

$$\sqrt{\frac{dP}{dQ}} - 1 \geq \frac{1}{2} \ln \frac{dP}{dQ}.$$

Taking expectations with respect to Q, we get

$$2H^2(P,Q) \leq E_Q \ln \frac{dQ}{dP} = \ln 2 \cdot D_{KL}(Q||P).$$

Hence

$$D_{KL}(Q||P) \geq \frac{2}{\ln 2} H^2(P,Q).$$

Other probability-distance measures

Other measures of probability distance are the *histogram intersection*, *Chi-square statistic*, *quadratic form distance*, *match distance*, *Kolmogorov–Smirnov distance*, and *earth mover's distance*.

Data differencing

Just as *absolute* entropy serves as theoretical background for data *compression*, *relative* entropy serves as theoretical background for data *differencing* – the absolute entropy of a set of data in this sense being the data required to reconstruct it (minimum compressed size), while the relative entropy of a target set of data, given a source set of data, is the data required to reconstruct the target *given* the source (minimum size of a patch).

Chapter 7

Rate–Distortion Theory

Rate–distortion theory is a major branch of information theory which provides the theoretical foundations for lossy data compression; it addresses the problem of determining the minimal amount of entropy (or information) *R* that should be communicated over a channel, so that the source (input signal) can be approximately reconstructed at the receiver (output signal) without exceeding a given distortion *D*.

Introduction

Rate–distortion theory gives theoretical bounds for how much compression can be achieved using lossy compression methods. Many of the existing audio, speech, image, and video compression techniques have transforms, quantization, and bit-rate allocation procedures that capitalize on the general shape of rate–distortion functions.

Rate–distortion theory was created by Claude Shannon in his foundational work on information theory.

In rate–distortion theory, the *rate* is usually understood as the number of bits per data sample to be stored or transmitted. The notion of *distortion* is a subject of on-going discussion. In the most simple case (which is actually used in most cases), the distortion is defined as the variance of the difference between input and output signal (i.e., the mean squared error of the difference). However, since we know that most lossy compression techniques operate on data that will be perceived by human consumers (listening to music, watching pictures and video) the distortion measure should preferably be modeled on human perception and perhaps aesthetics: much like the use of probability in lossless compression, distortion measures can ultimately be identified with loss functions as used in Bayesian estimation and decision theory. In audio compression, perceptual models (and therefore perceptual distortion measures) are relatively well developed and routinely used in compression techniques such as MP3 or Vorbis, but are often not easy to include in rate–distortion theory. In image and video compression, the human perception models are less well developed and inclusion is mostly limited to the JPEG and MPEG weighting (quantization, normalization) matrix.

Rate–distortion functions

The functions that relate the rate and distortion are found as the solution of the following minimization problem:

$$\inf_{Q_{Y|X}(y|x)} I_Q(Y;X) \text{ subject to } D_Q \leq D^*.$$

Here $Q_{Y|X}(y \mid x)$, sometimes called a test channel, is the conditional probability density function (PDF) of the communication channel output (compressed signal) Y for a given input (original signal) X, and $I_Q(Y ; X)$ is the **mutual information** between Y and X defined as

$$I(Y;X) = H(Y) - H(Y \mid X)$$

where $H(Y)$ and $H(Y \mid X)$ are the entropy of the output signal Y and the conditional entropy of the output signal given the input signal, respectively:

$$H(Y) = -\int_{-\infty}^{\infty} P_Y(y) \log_2(P_Y(y))\, dy$$
$$H(Y|X) = -\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q_{Y|X}(y|x) P_X(x) \log_2(Q_{Y|X}(y|x))\, dx\, dy.$$

The problem can also be formulated as a distortion–rate function, where we find the infimum over achievable distortions for given rate constraint. The relevant expression is:

$$\inf_{Q_{Y|X}(y|x)} E[D_Q[X,Y]] \text{subject to } I_Q(Y;X) \leq R$$

The two formulations lead to functions which are inverses of each other.

The mutual information can be understood as a measure for *prior* uncertainty the receiver has about the sender's signal (*H(Y)*), diminished by the uncertainty that is left after receiving information about the sender's signal ($H(Y \mid X)$). Of course the decrease in uncertainty is due to the communicated amount of information, which is $I(Y; X)$.

As an example, in case there is *no* communication at all, then $H(Y \mid X) = H(Y)$ and $I(Y; X) = 0$. Alternatively, if the communication channel is perfect and the received signal Y is identical to the signal X at the sender, then $H(Y \mid X) = 0$ and $I(Y; X) = H(Y) = H(X)$.

In the definition of the rate–distortion function, D_Q and D^* are the distortion between X and Y for a given $Q_{Y|X}(y \mid x)$ and the prescribed maximum distortion, respectively. When we use the mean squared error as distortion measure, we have (for amplitude-continuous signals):

$$D_Q = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P_{X,Y}(x,y)(x-y)^2\,dx\,dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q_{Y|X}(y|x)P_X(x)(x-y)^2\,dx\,dy$$

As the above equations show, calculating a rate–distortion function requires the stochastic description of the input X in terms of the PDF $P_X(x)$, and then aims at finding the conditional PDF $Q_{Y|X}(y \mid x)$ that minimize rate for a given distortion D^*. These definitions can be formulated measure-theoretically to account for discrete and mixed random variables as well.

An analytical solution to this minimization problem is often difficult to obtain except in some instances for which we next offer two of the best known examples. The rate–distortion function of any source is known to obey several fundamental properties, the most important ones being that it is a continuous, monotonically decreasing convex (U) function and thus the shape for the function in the examples is typical (even measured rate–distortion functions in real life tend to have very similar forms).

Although analytical solutions to this problem are scarce, there are upper and lower bounds to these functions including the famous Shannon lower bound (SLB), which in the case of squared error and memoryless sources, states that for arbitrary sources with finite differential entropy,

$$R(D) \geq h(X) - h(D)$$

where *h(D)* is the differential entropy of a Gaussian random variable with variance D. This lower bound is extensible to sources with memory and other distortion measures. One important feature of the SLB is that it is asymptotically tight in the low distortion regime for a wide class of sources and in some occasions, it actually coincides with the rate–distortion function. Shannon Lower Bounds can generally be found if the distortion between any two numbers can be expressed as a function of the difference between the value of these two numbers.

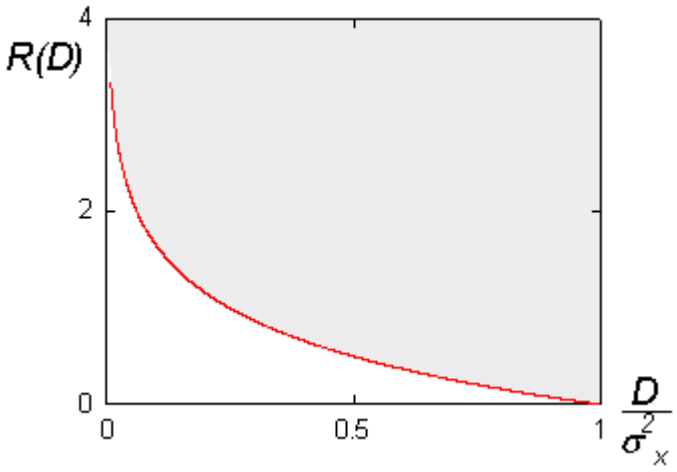
The Blahut-Arimoto algorithm, co-invented by Richard Blahut, is an elegant iterative technique for numerically obtaining rate–distortion functions of arbitrary finite input/output alphabet sources and much work has been done to extend it to more general problem instances.

Memoryless (independent) Gaussian source

If we assume that $P_X(x)$ is Gaussian with variance σ^2, and if we assume that successive samples of the signal X are stochastically independent (or, if you like, the source is *memoryless*, or the signal is *uncorrelated*), we find the following analytical expression for the rate–distortion function:

$$R(D) = \begin{cases} \frac{1}{2}\log_2(\sigma_x^2/D), & \text{if } D \leq \sigma_x^2 \\ \\ 0, & \text{if } D > \sigma_x^2 \end{cases}$$

The following figure shows what this function looks like:



Rate–distortion theory tell us that *no compression system exists that performs outside the gray area*. The closer a practical compression system is to the red (lower) bound, the better it performs. As a general rule, this bound can only be attained by increasing the coding block length parameter. Nevertheless, even at unit blocklengths one can often find good (scalar) quantizers that operate at distances from the rate–distortion function that are practically relevant.

This rate–distortion function holds only for Gaussian memoryless sources. It is known that the Gaussian source is the most "difficult" source to encode: for a given mean square error, it requires the greatest number of bits. The performance of a practical compression system working on—say—images, may well be below the *R(D)* lower bound shown.

Chapter 8

Introduction to Algorithmic Information Theory

Algorithmic information theory is a subfield of information theory and computer science that concerns itself with the relationship between computation and information. According to Gregory Chaitin, it is "the result of putting Shannon's information theory and Turing's computability theory into a cocktail shaker and shaking vigorously."

Overview

Algorithmic information theory principally studies complexity measures on strings (or other data structures). Because most mathematical objects can be described in terms of strings, or as the limit of a sequence of strings, it can be used to study a wide variety of mathematical objects, including integers and real numbers.

This use of the term "information" might be a bit misleading, as it depends upon the concept of compressibility. Informally, from the point of view of algorithmic information theory, the information content of a string is equivalent to the length of the shortest possible self-contained representation of that string. A self-contained representation is essentially a program – in some fixed but otherwise irrelevant universal programming language – that, when run, outputs the original string.

From this point of view, a 3000 page encyclopedia actually contains less information than 3000 pages of completely random letters, despite the fact that the encyclopedia is much more useful. This is because to reconstruct the entire sequence of random letters, one must know, more or less, what every single letter is. On the other hand, if every vowel were removed from the encyclopedia, someone with reasonable knowledge of the English language could reconstruct it, just as one could likely reconstruct the sentence "Ths sntnc hs lw nfrmtn cntnt" from the context and consonants present. For this reason, high-information strings and sequences are sometimes called "random"; people also sometimes attempt to distinguish between "information" and "useful information" and attempt to provide rigorous definitions for the latter, with the idea that the random letters may have more information than the encyclopedia, but the encyclopedia has more "useful" information.

Unlike classical information theory, algorithmic information theory gives formal, rigorous definitions of a random string and a random infinite sequence that do not depend on physical or philosophical intuitions about nondeterminism or likelihood. (The set of random strings depends on the choice of the universal Turing machine used to define Kolmogorov complexity, but any choice gives identical asymptotic results because the Kolmogorov complexity of a string is invariant up to an additive constant depending only on the choice of universal Turing machine. For this reason the set of random infinite sequences is independent of the choice of universal machine.)

Some of the results of algorithmic information theory, such as Chaitin's incompleteness theorem, appear to challenge common mathematical and philosophical intuitions. Most notable among these is the construction of Chaitin's constant **Ω**, a real number which expresses the probability that a self-delimiting universal Turing machine will halt when its input is supplied by flips of a fair coin (sometimes thought of as the probability that a random computer program will eventually halt). Although **Ω** is easily defined, in any consistent axiomatizable theory one can only compute finitely many digits of **Ω**, so it is in some sense *unknowable*, providing an absolute limit on knowledge that is reminiscent of Gödel's Incompleteness Theorem. Although the digits of **Ω** cannot be determined, many properties of **Ω** are known; for example, it is an algorithmically random sequence and thus its binary digits are evenly distributed (in fact it is normal).

History

Algorithmic information theory was founded by Ray Solomonoff, who published the basic ideas on which the field is based as part of his invention of algorithmic probability - a way to overcome serious problems associated with the application of Bayes rules in statistics. He first described his results at a Conference at Caltech in 1960, and in a report, Feb. 1960, "A Preliminary Report on a General Theory of Inductive Inference." Algorithmic information theory was later developed independently by Andrey Kolmogorov, in 1965 and Gregory Chaitin, around 1966.

There are several variants of Kolmogorov complexity or algorithmic information; the most widely used one is based on self-delimiting programs and is mainly due to Leonid Levin (1974). Per Martin-Löf also contributed significantly to the information theory of infinite sequences. An axiomatic approach to algorithmic information theory based on Blum axioms (Blum 1967) was introduced by Mark Burgin in a paper presented for publication by Andrey Kolmogorov (Burgin 1982). The axiomatic approach encompasses other approaches in the algorithmic information theory. It is possible to treat different measures of algorithmic information as particular cases of axiomatically defined measures of algorithmic information. Instead of proving similar theorems, such as the basic invariance theorem, for each particular measure, it is possible to easily deduce all such results from one corresponding theorem proved in the axiomatic setting. This is a general advantage of the axiomatic approach in mathematics. The axiomatic approach to algorithmic information theory was further developed in the book (Burgin 2005) and applied to software metrics (Burgin and Debnath, 2003; Debnath and Burgin, 2003).

Precise Definitions

A binary string is said to be random if the Kolmogorov complexity of the string is at least the length of the string. A simple counting argument shows that some strings of any given length are random, and almost all strings are very close to being random. Since Kolmogorov complexity depends on a fixed choice of universal Turing machine (informally, a fixed "description language" in which the "descriptions" are given), the collection of random strings does depend on the choice of fixed universal machine. Nevertheless, the collection of random strings, as a whole, has similar properties regardless of the fixed machine, so one can (and often does) talk about the properties of random strings as a group without having to first specify a universal machine.

An infinite binary sequence is said to be random if, for some constant *c*, for all n, the Kolmogorov complexity of the initial segment of length *n* of the sequence is at least *n-c*. Importantly, the complexity used here is prefix-free complexity; if plain complexity were used, there would be no random sequences. However, with this definition, it can be shown that almost every sequence (from the point of view of the standard measure - "fair coin" or Lebesgue measure - on the space of infinite binary sequences) is random. Also, since it can be shown that the Kolmogorov complexity relative to two different universal machines differs by at most a constant, the collection of random infinite sequences does not depend on the choice of universal machine (in contrast to finite strings). This definition of randomness is usually called *Martin-Löf* randomness, after Per Martin-Löf, to distinguish it from other similar notions of randomness. It is also sometimes called *1-randomness* to distinguish it from other stronger notions of randomness (2-randomness, 3-randomness, etc.).

(Related definitions can be made for alphabets other than the set {0,1}.)

Specific Sequence

Algorithmic information theory (AIT) is the information theory of individual objects, using computer science, and concerns itself with the relationship between computation, information, and randomness.

The information content or complexity of an object can be measured by the length of its shortest description. For instance the string

```
"0101010101010101010101010101010101010101010101010101010101010101"
```

has the short description "32 repetitions of '01'", while

```
"1100100001100001110111101110110011111010010000100101011110010110"
```

presumably has no simple description other than writing down the string itself.

More formally, the Algorithmic Complexity (AC) of a string x is defined as the length of the shortest program computes or outputs x, where the program is run on some fixed reference universal computer.

A closely related notion is the probability that a universal computer outputs some string x when fed with a program chosen at random. This Algorithmic "Solomon-off" Probability (AP) is key in addressing the old philosophical problem of induction in a formal way.

The major drawback of AC and AP are their incomputability. Time-bounded "Levin" complexity penalizes a slow program by adding the logarithm of its running time to its length. This leads to computable variants of AC and AP, and Universal "Levin" Search (US) solves all inversion problems in optimal (apart from some unrealistically large multiplicative constant) time.

AC and AP also allow a formal and rigorous definition of randomness of individual strings do not depend on physical or philosophical intuitions about non-determinism or likelihood. Roughly, a string is Algorithmic "Martin-Loef" Random (AR) if it is incompressible in the sense that its algorithmic complexity is equal to its length.

AC, AP, and AR are the core sub-disciplines of AIT, but AIT spawns into many other areas. It serves as the foundation of the Minimum Description Length (MDL) principle, can simplify proofs in computational complexity theory, has been used to define a universal similarity metric between objects, solves the Maxwell daemon problem, and many others.

Chapter 9

Chaitin's Constant

In the computer science subfield of algorithmic information theory a **Chaitin constant** or **halting probability** is a real number that informally represents the probability that a randomly-chosen program will halt. These numbers are formed from a construction due to Gregory Chaitin.

Although there are infinitely many halting probabilities, it is common to use the letter Ω to refer to them as if there were only one. Because Ω depends on the program encoding used, it is sometimes called **Chaitin's construction** instead of **Chaitin's constant** when not referring to any specific encoding.

Each halting probability is a normal and transcendental real number which is not computable, which means that there is no halting algorithm that enumerates its digits.

Background

The definition of a halting probability relies on the existence of **prefix-free universal computable functions.** Such a function, intuitively, represents a programming language with the property that no valid program can be obtained as a proper extension of another valid program.

Suppose that F is a partial function that takes one argument, a finite binary string, and possibly returns a single binary string as output. The function F is called **computable** if there is a Turing machine that computes it.

The function F is called **universal** if the following property holds: for every computable function f of a single variable there is a string w such that for all x, $F(w\,x) = f(x)$; here $w\,x$ represents the concatenation of the two strings w and x. This means that F can be used to simulate any computable function of one variable. Informally, w represents a "script" for the computable function f, and F represents an "interpreter" that parses the script as a prefix of its input and then executes it on the remainder of input. Note that for any fixed w the function $f(x) = F(w\,x)$ is computable; thus the universality property states that all computable functions of one variable can be obtained in this fashion.

The **domain** of F is the set of all inputs p on which it is defined. For F that are universal, such a p can generally be seen both as the concatenation of a program part and a data part, and as a single program for the function F.

The function F is called **prefix-free** if there are no two elements p, p' in its domain such that p' is a proper extension of p. This can be rephrased as: the domain of F is a prefix-free code (instantaneous code) on the set of finite binary strings. A simple way to enforce prefix-free-ness is to use machines whose means of input is a binary stream from which bits can be read one at a time. There is no end-of-stream marker; the end of input is determined by when the universal machine decides to stop reading more bits. Here, the difference between the two notions of program mentioned in the last paragraph becomes clear; the one is easily recognized by some grammar, while the other requires arbitrary computation to recognize.

The domain of any universal computable function is a computably enumerable set but never a computable set. The domain is always Turing equivalent to the halting problem.

Definition of halting probabilities

Let P_F be the domain of a prefix-free universal computable function F. The constant Ω_F is then defined as

$$\Omega_F = \sum_{p \in P_F} 2^{-|p|},$$

where $|p|$ denotes the length of a string p. This is an infinite sum which has one summand for every p in the domain of F. The requirement that the domain be prefix-free, together with Kraft's inequality, ensures that this sum converges to a real number between 0 and 1. If F is clear from context then Ω_F may be denoted simply Ω, although different prefix-free universal computable functions lead to different values of Ω.

Use of Chaitin's constant in proving unsolved problems in number theory

Chaitin's constant can be used, in principle, to solve many outstanding problems in number theory, including Goldbach's conjecture and the Riemann hypothesis.

Goldbach's conjecture says every even number greater than 2 is the sum of two primes. For a given even number, let a computer program iterate through the even numbers, beginning with the given one, searching for corresponding two primes in each instance. If the appropriate primes exist, then the program finds them eventually, advances to the next even number and the search is continued. If there are no two primes that add to the even number, then the program exhausts eventually all possible prime pairs in the test, thus it notices that a counterexample has just been found, it halts and Goldbach's

conjecture has been disproved. Thus, this is a counterexample-searching program: it runs forever exactly if Goldbach's conjecture is true; it halts (with a counterexample) exactly if the conjecture is false.

Let this program be N bits long. Given unlimited resources and time, the first N bits of Chaitin's number can be used to prove Goldbach's conjecture as follows. In dovetailing fashion, all programs of all lengths are run, until enough have halted to jointly contribute enough probability to match these first N bits. If the Goldbach program hasn't halted yet, then it never will, since its contribution to the halting probability would affect the first N bits. Thus, the conjecture holds unless we already found a counterexample.

The same Chaitin constant can be used to prove (or disprove) the Riemann hypothesis and many other of the unsolved problems in mathematics.

Interpretation as a probability

The Cantor space is the collection of all infinite sequences of 0s and 1s. A halting probability can be interpreted as the measure of a certain subset of Cantor space under the usual probability measure on Cantor space. It is from this interpretation that halting probabilities take their name.

The probability measure on Cantor space, sometimes called the fair-coin measure, is defined so that for any binary string x the set of sequences that begin with x has measure $2^{-|x|}$. This implies that for each natural number n, the set of sequences f in Cantor space such that $f(n) = 1$ has measure 1/2, and the set of sequences whose nth element is 0 also has measure 1/2.

Let F be a prefix-free universal computable function. The domain P of F consists of an infinite set of binary strings

$$P = \{p_1, p_2, \ldots\}.$$

Each of these strings p_i determines a subset S_i of Cantor space; the set S_i contains all sequences in cantor space that begin with p_i. These sets are disjoint because P is a prefix-free set. The sum

$$\sum_{p \in P} 2^{-|p|}$$

represents the measure of the set

$$\bigcup_{i \in \mathbb{N}} S_i .$$

In this way, Ω_F represents the probability that a randomly selected infinite sequence of 0s and 1s begins with a bit string (of some finite length) that is in the domain of F. It is for this reason that Ω_F is called a halting probability.

Properties

Each Chaitin constant Ω has the following properties:

- It is algorithmically random. This means that the shortest program to output the first n bits of Ω must be of size at least n-O(1). This is because, as in the Goldbach example, those n bits enable us to find out exactly which programs halt among all those of length at most n.
- It is a normal number, which means that its digits are equidistributed as if they were generated by tossing a fair coin.
- It is not a computable number; there is no computable function that enumerates its binary expansion, as discussed below.
- The set of rational numbers q such that $q \leq \Omega$ is computably enumerable; a real number with such a property is called a **left-c.e. real number** in recursion theory.
- It is Turing equivalent to the halting problem and thus at level Δ_2^0 of the arithmetical hierarchy.

Not every set that is Turing equivalent to the halting problem is a halting probability. A finer equivalence relation, **Solovay equivalence**, can be used to characterize the halting probabilities among the left-c.e. reals.

Uncomputability of halting probabilities

A real number is called computable if there is an algorithm which, given n, returns the first n digits of the number. This is equivalent to the existence of a program that enumerates the digits of the real number.

No halting probability is computable. The proof of this fact relies on an algorithm which, given the first n digits of Ω, solves Turing's halting problem for programs of length up to n. Since the halting problem is undecidable, Ω can not be computed.

The algorithm proceeds as follows. Given the first n digits of Ω and a $k \leq n$, the algorithm enumerates the domain of F until enough elements of the domain have been found so that the probability they represent is within $2^{-(k+1)}$ of Ω. After this point, no additional program of length k can be in the domain, because each of these would add 2^{-k} to the measure, which is impossible. Thus the set of strings of length k in the domain is exactly the set of such strings already enumerated.

Incompleteness theorem for halting probabilities

For each specific consistent effectively represented axiomatic system for the natural numbers, such as Peano arithmetic, there exists a constant N such that no bit of Ω after the Nth can be proven to be one or zero within that system. The constant N depends on how the formal system is effectively represented, and thus does not directly reflect the complexity of the axiomatic system. This incompleteness result is similar to Gödel's incompleteness theorem in that it shows that no consistent formal theory for arithmetic can be complete.

Super Omega

As mentioned above, the first n bits of Gregory Chaitin's constant Omega are random or incompressible in the sense that we cannot compute them by a halting algorithm with fewer than n-O(1) bits. However, consider the short but never halting algorithm which systematically lists and runs all possible programs; whenever one of them halts its probability gets added to the output (initialized by zero). After finite time the first n bits of the output will never change any more (it does not matter that this time itself is not computable by a halting program). So there is a short non-halting algorithm whose output converges (after finite time) onto the first n bits of Omega. In other words, the enumerable first n bits of Omega are highly compressible in the sense that they are limit-computable by a very short algorithm; they are not random with respect to the set of enumerating algorithms. Jürgen Schmidhuber (2000) constructed a limit-computable "Super Omega" which in a sense is much more random than the original limit-computable Omega, as one cannot significantly compress the Super Omega by any enumerating non-halting algorithm.

Algorithmically random sequence

Intuitively, an **algorithmically random sequence** (or **random sequence**) is an infinite sequence of binary digits that appears random to any algorithm. The definition applies equally well to sequences on any finite set of characters. Random sequences are key objects of study in algorithmic information theory.

As different types of algorithms are sometimes considered, ranging from algorithms with specific bounds on their running time to algorithms which may ask questions of an oracle, there are different notions of randomness. The most common of these is known as **Martin-Löf randomness** (or **1-randomness**), but stronger and weaker forms of randomness also exist. The term "random" used to refer to a sequence without clarification is usually taken to mean "Martin-Löf random".

Because infinite sequences of binary digits can be identified with real numbers in the unit interval, random binary sequences are often called **random real numbers**. Additionally, infinite binary sequences correspond to characteristic functions of sets of natural numbers; therefore those sequences might be seen as sets of natural numbers.

The class of all Martin-Löf random (binary) sequences is denoted by RAND or MLR.

History

The first suitable definition of a random sequence was given by Per Martin-Löf in 1966. Earlier researchers such as Richard von Mises had attempted to formalize the notion of a test for randomness in order to define a random sequence as one that passed all tests for randomness; however, the precise notion of a randomness test was left vague. Martin-Löf's key insight was to use the theory of computation to formally define the notion of a test for randomness. This contrasts with the idea of randomness in probability; in that theory, no particular element of a sample space can be said to be random.

Martin-Löf randomness has since been shown to admit many equivalent characterizations — in terms of compression, randomness tests, and gambling — that bear little outward resemblance to the original definition, but each of which satisfy our intuitive notion of properties that random sequences ought to have: random sequences should be incompressible, they should pass statistical tests for randomness, and it should be difficult to make money betting on them. The existence of these multiple definitions of Martin-Löf randomness, and the stability of these definitions under different models of computation, give evidence that Martin-Löf randomness is a fundamental property of mathematics and not an accident of Martin-Löf's particular model. The thesis that the definition of Martin-Löf randomness "correctly" captures the intuitive notion of randomness has been called the **Martin-Löf–Chaitin Thesis**; it is somewhat similar to the Church–Turing thesis.

Three equivalent definitions

Martin-Löf's original definition of a random sequence was in terms of constructive null covers; he defined a sequence to be random if it is not contained in any such cover. Leonid Levin and Claus-Peter Schnorr proved a characterization in terms of Kolmogorov complexity: a sequence is random if there is a uniform bound on the compressibility of its initial segments. Schnorr gave a third equivalent definition in terms of martingales (a type of betting strategy). Li and Vitanyi's book *An Introduction to Kolmogorov Complexity and Its Applications* is an excellent introduction to these ideas.

- **Kolmogorov complexity** (Schnorr 1973, Levin 1973): Kolmogorov complexity can be thought of as a lower bound on the algorithmic compressibility of a finite sequence (of characters or binary digits). It assigns to each such sequence *w* a natural number *K(w)* that, intuitively, measures the minimum length of a computer program (written in some fixed programming language) that takes no input and will output *w* when run. Given a natural number *c* and a sequence *w*, we say that *w* is ***c*-incompressible** if $K(w) \geq |w| - c$.

  An infinite sequence S *is Martin-Löf random if and only if there is a constant* c *such that all of* S*'s finite prefixes are* c*-incompressible.*

- **Constructive null covers** (Martin-Löf 1966): This is Martin-Löf's original definition. For a finite binary string *w* we let C_w denote the **cylinder generated by** *w*. This is the set of all infinite sequences beginning with *w*, which is a basic open set in Cantor space. The **product measure** $\mu(C_w)$ of the cylinder generated by *w* is defined to be $2^{-|w|}$. Every open subset of Cantor space is the union of a countable sequence of disjoint basic open sets, and the measure of an open set is the sum of the measures of any such sequence. An *effective open set* is an open set that is the union of the sequence of basic open sets determined by a recursively enumerable sequence of binary strings. A *constructive null cover* or *effective measure 0 set* is a recursively enumerable sequence U_i of effective open sets such that $U_{i+1} \subseteq U_i$ and $\mu U_i \leq 2^{-i}$ for each natural number *i*. Every effective null cover determines a G_δ set of measure 0, namely the intersection of the sets U_i.

  A sequence is defined to be Martin-Löf random if it is not contained in any G_δ set determined by a constructive null cover.

- **Constructive martingales** (Schnorr 1971): A martingale is a function $d : \{0,1\}^* \to [0,\infty)$ such that, for all finite strings *w*, $d(w) = (d(w^\frown 0) + d(w^\frown 1))/2$, where $a^\frown b$ is the concatenation of the strings *a* and *b*. This is called the "fairness condition"; a martingale is viewed as a betting strategy, and the above condition requires that the better plays against fair odds. A martingale *d* is said to **succeed** on a sequence *S* if $\limsup_{n\to\infty} d(S \upharpoonright n) = \infty,$ where $S \upharpoonright n$ is the first *n* bits of *S*. A martingale *d* is **constructive** (also known as **weakly computable**, **lower semi-computable**, **subcomputable**) if there exists a computable function $\widehat{d} : \{0,1\}^* \times \mathbb{N} \to \mathbb{Q}$ such that, for all finite binary strings *w*

1. $\widehat{d}(w,t) \leq \widehat{d}(w,t+1) < d(w),$ for all positive integers *t*,
2. $\lim_{t\to\infty} \widehat{d}(w,t) = d(w).$

  A sequence is Martin-Löf random if and only if no constructive martingale succeeds on it.
  (Note that the definition of martingale used here differs slightly from the one used in probability theory. That definition of martingale has a similar fairness condition, which also states that the expected value after some observation is the same as the value before the observation, given the prior history of observations. The difference is that in probability theory, the prior history of observations just refers to the capital history, whereas here the history refers to the exact sequence of 0s and 1s in the string.)

Interpretations of the definitions

The Kolmogorov complexity characterization conveys the intuition that a random sequence is incompressible: no prefix can be produced by a program much shorter than the prefix.

The null cover characterization conveys the intuition that a random real number should not have any property that is "uncommon". Each measure 0 set can be thought of as an uncommon property. It is not possible for a sequence to lie in no measure 0 sets, because each one-point set has measure 0. Martin-Löf's idea was to limit the definition to measure 0 sets that are effectively describable; the definition of an effective null cover determines a countable collection of effectively describable measure 0 sets and defines a sequence to be random if it does not lie in any of these particular measure 0 sets. Since the union of a countable collection of measure 0 sets has measure 0, this definition immediately leads to the theorem that there is a measure 1 set of random sequences. Note that if we identify the Cantor space of binary sequences with the interval [0,1] of real numbers, the measure on Cantor space agrees with Lebesgue measure.

The martingale characterization conveys the intuition that no effective procedure should be able to make money betting against a random sequence. A martingale d is a betting strategy. d reads a finite string w and bets money on the next bit. It bets some fraction of its money that the next bit will be 0, and then remainder of its money that the next bit will be 1. d doubles the money it placed on the bit that actually occurred, and it loses the rest. $d(w)$ is the amount of money it has after seeing the string w. Since the bet placed after seeing the string w can be calculated from the values $d(w)$, $d(w0)$, and $d(w1)$, calculating the amount of money it has is equivalent to calculating the bet. The martingale characterization says that no betting strategy implementable by any computer (even in the weak sense of constructive strategies, which are not necessarily computable) can make money betting on a random sequence.

Properties and examples of Martin-Löf random sequences

- Chaitin's halting probability Ω is an example of a random sequence.
- RAND^c (the complement of RAND) is a measure 0 subset of the set of all infinite sequences. This is implied by the fact that each constructive null cover covers a measure 0 set, there are only countably many constructive null covers, and a countable union of measure 0 sets has measure 0. This implies that RAND is a measure 1 subset of the set of all infinite sequences.
- Every random sequence is normal.
- There is a constructive null cover of RAND^c. This means that all effective tests for randomness (that is, constructive null covers) are, in a sense, subsumed by this

universal test for randomness, since any sequence that passes this single test for randomness will pass all tests for randomness. (Martin-Löf 1966)

- There is a *universal* constructive martingale **d**. This martingale is universal in the sense that, given any constructive martingale *d*, if *d* succeeds on a sequence, then **d** succeeds on that sequence as well. Thus, **d** succeeds on every sequence in RAND^{c} (but, since **d** is constructive, it succeeds on no sequence in RAND). (Schnorr 1971)

- The class RAND is a Σ^0_2subset of Cantor space, where Σ^0_2refers to the second level of the arithmetical hierarchy. This is because a sequence *S* is in RAND if and only if there is some open set in the universal effective null cover that does not contain *S*; this property can be seen to be definable by a Σ^0_2formula.

- There is a random sequence which is Δ^0_2, that is, computable relative to an oracle for the Halting problem. (Schnorr 1971) Chaitin's Ω is an example of such a sequence.

- No random sequence is decidable, computably enumerable, or co-computably-enumerable. Since these correspond to the Δ^0_1, Σ^0_1, and Π^0_1levels of the arithmetical hierarchy, this means that $\Delta^0_2$is the lowest level in the arithmetical hierarchy where random sequences can be found.

- Every sequence is Turing reducible to some random sequence. (Kučera 1985/1989, Gács 1986). Thus there are random sequences of arbitrarily high Turing degree.

Relative randomness

As each of the equivalent definitions of a Martin-Löf random sequence is based on what is computable by some Turing machine, one can naturally ask what is computable by a Turing oracle machine. For a fixed oracle *A*, a sequence *B* which is not only random but in fact satisfies the equivalent definitions for computability relative to *A* (e.g., no martingale which is constructive relative to the oracle *A* succeeds on *B*) is said to be random relative to *A*. Two sequences, while themselves random, may contain very similar information, and therefore neither will be random relative to the other. Any time there is a Turing reduction from one sequence to another, the second sequence cannot be random relative to the first, just as computable sequences are themselves nonrandom; in particular, this means that Chaitin's Ω is not random relative to the halting problem.

An important result relating to relative randomness is van Lambalgen's theorem, which states that if *C* is the sequence composed from *A* and *B* by interleaving the first bit of *A*, the first bit of *B*, the second bit of *A*, the second bit of *B*, and so on, then *C* is algorithmically random if and only if *A* is algorithmically random, and *B* is

algorithmically random relative to A. A closely related consequence is that if A and B are both random themselves, then A is random relative to B if and only if B is random relative to A.

Stronger than Martin-Löf randomness

Relative randomness gives us the first notion which is stronger than Martin-Löf randomness, which is randomness relative to some fixed oracle A. For any oracle, this is at least as strong, and for most oracles, it is strictly stronger, since there will be Martin-Löf random sequences which are not random relative to the oracle A. Important oracles often considered are the halting problem, $\emptyset'$, and the nth jump oracle, $\emptyset^{(n)}$, as these oracles are able to answer specific questions which naturally arise. A sequence which is random relative to the oracle $\emptyset^{(n-1)}$ is called n-random; a sequence is 1-random, therefore, if and only if it is Martin-Löf random. A sequence which is n-random for every n is called arithmetically random. The n-random sequences sometimes arise when considering more complicated properties. For example, there are only countably many Δ^0_2 sets, so one might think that these should be non-random. However, the halting probability Ω is Δ^0_2 and 1-random; it is only after 2-randomness is reached that it is impossible for a random set to be Δ^0_2.

Weaker than Martin-Löf randomness

Additionally, there are several notions of randomness which are weaker than Martin-Löf randomness. Some of these are weak 1-randomness, Schnorr randomness, computable randomness, partial computable randomness. Additionally, Kolmogorov-Loveland randomness is known to be no stronger than Martin-Löf randomness, but it is not known whether it is actually weaker.

Normal number

In mathematics, a **normal number** is a real number whose infinite sequence of digits in every base b is distributed uniformly in the sense that each of the b digit values has the same natural density $1/b$, also all possible b^2 pairs of digits are equally likely with density b^{-2}, all b^3 triplets of digits equally likely with density b^{-3}, etc.

While a general proof can be given that almost all numbers are normal, this proof is not constructive and only very few concrete numbers have been shown to be normal. For example, it is widely believed that the numbers $\sqrt{2}$, π, and e are normal, but a proof remains elusive.

Definitions

Let Σ be a finite alphabet of b digits, and Σ^∞ the set of all sequences that may be drawn from that alphabet. Let S, $w \in \Sigma^\infty$ be two such sequences, of which the latter is finite string drawn from the alphabet Σ. Let n be a positive integer. Define $N_S(w, n)$ to be the number of times the string w appears as a substring in the first n digits of the sequence S. (For instance, if S = 01010101..., then $N_S(010, 8) = 3$.) S is **normal** if, for all finite strings $w \in \Sigma^*$,

$$\lim_{n\to\infty} \frac{N_S(w,n)}{n} = \frac{1}{b^{|w|}}$$

(where $|\ w\ |$ denotes the length of the string w) In other words, S is normal if all strings of equal length occur with equal asymptotic frequency. For example, in a normal binary sequence (a sequence over the alphabet {0,1}), 0 and 1 each occur with frequency ½; 00, 01, 10, and 11 each occur with frequency ¼; 000, 001, 010, 011, 100, 101, 110, and 111 each occur with frequency ⅛, etc. Roughly speaking, the probability of finding the string w in any given position in S is precisely that expected if the sequence had been produced at random.

Suppose now that b is an integer greater than 1 and x is a real number. Consider the infinite digit sequence expansion $S_{x,\,b}$ of x in the base b positional number system (we ignore the decimal point). We say x is **normal in base *b*** if the sequence $S_{x,\,b}$ is normal. The number x is called a **normal number** (or sometimes an **absolutely normal number**) if it is normal in base b for every integer b greater than 1.

A given infinite sequence is either normal or not normal, whereas a real number, having a different base-b expansion for each integer $b \geq 2$, may be normal in one base but not in another (Cassels 1959 and Schmidt 1960). All normal numbers in base r are normal in base s if and only if $\log r / \log s$ is a rational number (Schmidt 1960).

A disjunctive sequence is a sequence in which every finite string appears. A normal sequence is disjunctive, but a disjunctive sequence need not be normal. A number that is disjunctive to every base is called *absolutely disjunctive* or is said to be a *lexicon* (Calude and Zamfirescu, 1999). A lexicon contains all writings, which have been or will be ever written, in any possible language. Every normal number is b-dense, but not necessarily vice versa. A set is called "residual" if it contains the intersection of a countable family of open dense sets. The set of absolutely disjunctive reals (lexicons) is a residual (Calude and Zamfirescu, 1999).

Another weaker property than normality is simple normality. A number is **simply normal in base *b*** if each individual digit appears with frequency $1/b$. For a given base b, a number can be simply normal (but not normal or b-dense), b-dense (but not simply normal or normal), normal (and thus simply normal and b-dense), or none of these.

Properties and examples

The concept of a normal number was introduced by Émile Borel in 1909. Using the Borel-Cantelli lemma, he proved the normal number theorem: almost all real numbers are normal, in the sense that the set of non-normal numbers has Lebesgue measure zero (Borel 1909). This theorem established the existence of normal numbers. In 1917, Waclaw Sierpinski showed that it is possible to specify a particular such number. Becker and Figueira proved in 2002 that there is a computable normal number.

The set of non-normal numbers, though "small" in the sense of being a null set, is "large" in the sense of being uncountable. For instance, there are uncountably many numbers whose decimal expansion does not contain the digit 5, and none of these are normal.

Champernowne's number

0.12345678910111213141516 17...,

obtained by concatenating the decimal representations of the natural numbers in order, is normal in base 10, but it might not be normal in some other bases. The Copeland–Erdős constant

0.23571113171923293137414 3...,

obtained by concatenating the prime numbers in base 10, is also known to be normal in base 10.

Every Chaitin's constant Ω is a normal number (Calude, 1994). A computable normal number was constructed in (Becher 2002). Although these constructions do not directly give the digits of the numbers constructed, the second shows that it is possible in principle to enumerate all the digits of a particular normal number.

No rational number is normal to any base, since the digit sequences of rational numbers are eventually periodic.

Bailey and Crandall show an explicit uncountably infinite class of *b*-normal numbers by perturbing Stoneham numbers.

It has been an elusive goal to prove the normality of numbers which were not explicitly constructed for the purpose. It is for instance unknown whether $\sqrt{2}$, π, ln(2) or e is normal (but all of them are strongly conjectured to be normal, because of some empirical evidence). It is not even known which digits occur infinitely often in the decimal expansions of those constants. It has been conjectured that every irrational algebraic number is normal; while no counterexamples are known, there also exists no algebraic number that has been proven to be normal in any base.

Additional properties of normal numbers include:

- Every positive number x is the product of two normal numbers. For instance if y is chosen uniformly at random from the interval (0,1) then almost surely y and x/y are both normal, and their product is x.

- If x is normal in base b and $q \neq 0$ is a rational number, then $x \cdot q$ is normal in base b. (Wall 1949)

- If $A \subseteq \mathbb{N}$ is *dense* (for every $\alpha < 1$ and for all sufficiently large n, $|A \cap \{1, \ldots, n\}| \geq n^{\alpha}$) and $a_1, a_2, a_3, \cdots$ are the base-b expansions of the elements of A, then the number $0.a_1a_2a_3\cdots$, formed by concatenating the elements of A, is normal in base b (Copeland and Erdős 1946). From this it follows that Champernowne's number is normal in base 10 (since the set of all positive integers is obviously dense) and that the Copeland-Erdős constant is normal in base 10 (since the prime number theorem implies that the set of primes is dense).

- A sequence is normal if and only if every *block* of equal length appears with equal frequency. (A block of length k is a substring of length k appearing at a position in the sequence that is a multiple of k: e.g. the first length-k block in S is $S[1..k]$, the second length-k block is $S[k+1..2k]$, etc.) This was implicit in the work of Ziv and Lempel (1978) and made explicit in the work of Bourke, Hitchcock, and Vinodchandran (2005).

- A number is normal in base b if and only if it is simply normal in base b^k for every integer $k \geq 1$. This follows from the previous block characterization of normality: Since the n^{th} block of length k in its base b expansion corresponds to the n^{th} digit in its base b^k expansion, a number is simply normal in base b^k if and only if blocks of length k appear in its base b expansion with equal frequency.

- A number is normal if and only if it is simply normal in every base. This follows from the previous characterization of base b normality.

- A number is b-normal if and only if there exists a set of positive integers $m_1 < m_2 < m_3 < \cdots$ where the number is simply normal to bases b^m for all $m \in \{m_1, m_2, \ldots\}$. No finite set suffices to show that the number is b-normal.

- The set of normal sequences is **closed under finite variations**: adding, removing, or changing a finite number of digits in any normal sequence leaves it normal.

Connection to finite-state machines

Agafonov showed an early connection between finite-state machines and normal sequences: every subsequence selected from a normal sequence by a regular language is

also normal. In other words, if one runs a finite-state machine on a normal sequence, where each of the finite-state machine's states are labeled either "output" or "no output", and the machine outputs the digit it reads next after entering an "output" state, but does not output the next digit after entering a "no output state", then the sequence it outputs will be normal (Agafonov 1968).

A deeper connection exists with finite-state gamblers (FSGs) and information lossless finite-state compressors (ILFSCs).

- A **finite-state gambler** (a.k.a. **finite-state martingale**) is a finite-state machine over a finite alphabet Σ, each of whose states is labelled with percentages of money to bet on each digit in Σ. For instance, for an FSG over the binary alphabet Σ = {0,1}, the current state *q* bets some percentage $q_0 \in [0, 1]$ of the gambler's money on the bit 0, and the remaining $q_1 = 1 - q_0$ fraction of the gambler's money on the bit 1. The money bet on the digit that comes next in the input (total money times percent bet) is multiplied by | Σ |, and the rest of the money is lost. After the bit is read, the FSG transitions to the next state according to the input it received. A FSG *d* **succeeds** on an infinite sequence *S* if, starting from \$1, it makes unbounded money betting on the sequence; i.e., if

  $$\limsup_{n\to\infty} d(S \upharpoonright n) = \infty,$$

  where $d(S \upharpoonright n)$ is the amount of money the gambler *d* has after reading the first *n* digits of *S*.

- A **finite-state compressor** is a finite-state machine with output strings labelling its state transitions, including possibly the empty string. (Since one digit is read from the input sequence for each state transition, it is necessary to be able to output the empty string in order to achieve any compression at all). An information lossless finite-state compressor is a finite-state compressor whose input can be uniquely recovered from its output and final state. In other words, for a finite-state compressor *C* with state set *Q*, *C* is information lossless if the function $f : \Sigma^* \to \Sigma^* \times Q$, mapping the input string of *C* to the output string and final state of *C*, is 1-1. Compression techniques such as Huffman coding or Shannon-Fano coding can be implemented with ILFSCs. An ILFSC *C* **compresses** an infinite sequence *S* if

  $$\liminf_{n\to\infty} \frac{|C(S \upharpoonright n)|}{n} < 1,$$

  where $|C(S \upharpoonright n)|$ is the number of digits output by *C* after reading the first *n* digits of *S*. Note that the compression ratio (the limit inferior above) can always be made to equal 1 by the 1-state ILFSC that simply copies its input to the output.

Schnorr and Stimm showed that no FSG can succeed on any normal sequence, and Bourke, Hitchcock and Vinodchandran showed the converse. Therefore:

A sequence is normal if and only if there is no finite-state gambler that succeeds on it.

Ziv and Lempel showed:

A sequence is normal if and only if it is incompressible by any information lossless finite-state compressor

(they actually showed that the sequence's optimal compression ratio over all ILFSCs is exactly its *entropy rate*, a quantitative measure of its deviation from normality, which is 1 exactly when the sequence is normal). Since the LZ compression algorithm compresses asymptotically as well as any ILFSC, this means that the LZ compression algorithm can compress any non-normal sequence. (Ziv Lempel 1978)

These characterizations of normal sequences can be interpreted to mean that "normal" = "finite-state random"; i.e., the normal sequences are precisely those that appear random to any finite-state machine. Compare this with the algorithmically random sequences, which are those infinite sequences that appear random to any algorithm (and in fact have similar gambling and compression characterizations with Turing machines replacing finite-state machines).

Connection to equidistributed sequences

A number x is normal in base b if and only if the sequence $\left(b^k x\right)_{k=0}^{\infty}$ is equidistributed modulo 1, or equivalently, using Weyl's criterion, if and only if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} e^{2\pi i m b^k x} = 0 \quad \text{for all integers } m \geq 1.$$

Arithmetical hierarchy

In mathematical logic, the **arithmetical hierarchy**, **arithmetic hierarchy** or **Kleene hierarchy** classifies certain sets based on the complexity of formulas that define them. Any set that receives a classification is called **arithmetical**.

The arithmetical hierarchy is important in recursion theory, effective descriptive set theory, and the study of formal theories such as Peano arithmetic.

The Tarski-Kuratowski algorithm provides an easy way to get an upper bound on the classifications assigned to a formula and the set it defines.

The hyperarithmetical hierarchy and the analytical hierarchy extend the arithmetical hierarchy to classify additional formulas and sets.

The arithmetical hierarchy of formulas

The arithmetical hierarchy assigns classifications to the formulas in the language of first-order arithmetic. The classifications are denoted Σ^0_n and Π^0_n for natural numbers *n* (including 0). The Greek letters here are lightface symbols, which indicates that the formulas do not contain set parameters.

If a formula φ is logically equivalent to a formula with only bounded quantifiers then φ is assigned the classifications Σ^0_0 and Π^0_0.

The classifications Σ^0_n and Π^0_n are defined inductively for every natural number *n* using the following rules:

- If φ is logically equivalent to a formula of the form $\exists n_1 \exists n_2 \cdots \exists n_k \psi$, where ψ is Π^0_n, then φ is assigned the classification Σ^0_{n+1}.
- If φ is logically equivalent to a formula of the form $\forall n_1 \forall n_2 \cdots \forall n_k \psi$, where ψ is Σ^0_n, then φ is assigned the classification Π^0_{n+1}.

In other words, a Σ^0_n formula is equivalent to a formula that begins with some existential quantifiers and alternates $n - 1$ times between series of existential and universal quantifiers; while a Π^0_n formula is equivalent to a formula that begins with some universal quantifiers and alternates similarly.

Because every formula is equivalent to a formula in prenex normal form, every formula with no set quantifiers is assigned at least one classification. Because meaningless quantifiers can be added to any formula, once a formula is assigned the classification Σ^0_n or Π^0_n it will be assigned the classifications Σ^0_m and Π^0_m for every *m* greater than *n*. The most important classification assigned to a formula is thus the one with the least *n*, because this is enough to determine all the other classifications.

The arithmetical hierarchy of sets of natural numbers

A set *X* of natural numbers is defined by formula φ in the language of Peano arithmetic if the elements of *X* are exactly the numbers that satisfy φ. That is, for all natural numbers *n*,

$$n \in X \Leftrightarrow \mathbb{N} \models \phi(\underline{n}),$$

where $\underline{n}$ is the numeral in the language of arithmetic corresponding to *n*. A set is definable in first order arithmetic if it is defined by some formula in the language of Peano arithmetic.

Each set *X* of natural numbers that is definable in first order arithmetic is assigned classifications of the form Σ^0_n, Π^0_n, and Δ^0_n, where *n* is a natural number, as follows. If *X* is definable by a Σ^0_n formula then *X* is assigned the classification Σ^0_n. If *X* is definable by a Π^0_n formula then *X* is assigned the classification Π^0_n. If *X* is both Σ^0_n and Π^0_n then *X* is assigned the additional classification Δ^0_n.

Note that it rarely makes sense to speak of Δ^0_n formulas; the first quantifier of a formula is either existential or universal. So a Δ^0_n set is not defined by a Δ^0_n formula; rather, there are both Σ^0_n and Π^0_n formulas that define the set.

A parallel definition is used to define the arithmetical hierarchy on finite Cartesian powers of the natural numbers. Instead of formulas with one free variable, formulas with *k* free number variables are used to define the arithmetical hierarchy on sets of *k*-tuples of natural numbers.

Relativized arithmetical hierarchies

Just as we can define what it means for a set *X* to be recursive relative to another set *Y* by allowing the computation defining *X* to consult *Y* as an oracle we can extend this notion to the whole arithmetic hierarchy and define what it means for *X* to be Σ^0_n, Δ^0_n or Π^0_n in *Y*, denoted respectively $\Sigma^{0,Y}_n$ $\Delta^{0,Y}_n$ and $\Pi^{0,Y}_n$. To do so, fix a set of integers *Y* and add a predicate for membership in *Y* to the language of Peano arithmetic. We then say that *X* is in $\Sigma^{0,Y}_n$ if it is defined by a Σ^0_n formula in this expanded language. In other words *X* is $\Sigma^{0,Y}_n$ if it is defined by a Σ^0_n formula allowed to ask questions about membership in *Y*. Alternatively one can view the $\Sigma^{0,Y}_n$ sets as those sets that can be built starting with sets recursive in *Y* and alternatively projecting and taking the complements of these sets up to *n* times.

For example let *Y* be a set of integers. Let *X* be the set of numbers divisible by an element of Y. Then *X* is defined by the formula $\phi(n) = \exists m \exists t(Y(m) \wedge m \times t = n)$ so *X* is in $\Sigma^{0,Y}_1$ (actually it is in $\Delta^{0,Y}_0$ as well since we could bound both quantifiers by n).

Arithmetic reducibility and degrees

Arithmetical reducibility is an intermediate notion between Turing reducibility and hyperarithmetic reducibility.

A set is **arithmetical** (also **arithmetic** and **arithmetically definable**) if it is defined by some formula in the language of Peano arithmetic. Equivalently X is arithmetical if X is Σ^0_n or Π^0_n for some integer n. A set X **is arithmetical in** a set Y, denoted $X \leq_A Y$, if X is definable a some formula in the language of Peano arithmetic extended by a predicate for membership in Y. Equivalently, X is arithmetical in Y if X is in $\Sigma^{0,Y}_n$ or $\Pi^{0,Y}_n$ for some integer n. A synonym for $X \leq_A Y$ is: X is **arithmetically reducible** to Y

The relation $X \leq_A Y$ is reflexive and transitive, and thus the relation $\equiv_A$ defined by the rule

$$X \equiv_A Y \Leftrightarrow X \leq_A Y \wedge Y \leq_A X$$

is an equivalence relation. The equivalence classes of this relation are called the **arithmetic degrees**; they are partially ordered under $\leq_A$.

The arithmetical hierarchy of subsets of Cantor and Baire space

The Cantor space, denoted 2^ω, is the set of all infinite sequences of 0s and 1s; the Baire space, denoted ω^ω or $\mathcal{N}$, is the set of all infinite sequences of natural numbers. Note that elements of the Cantor space can be identified with sets of integers and elements of the Baire space with functions from integers to integers.

The ordinary axiomatization of second-order arithmetic uses a set-based language in which the set quantifiers can naturally be viewed as quantifying over Cantor space. A subset of Cantor space is assigned the classification Σ^0_n if it is definable by a Σ^0_n formula. The set is assigned the classification Π^0_n if it is definable by a Π^0_n formula. If the set is both Σ^0_n and Π^0_n then it is given the additional classification Δ^0_n. For example let $O \subset 2^\omega$ be the set of all infinite binary strings which aren't all 0 (or equivalently the set of all non-empty sets of integers). As $O = \{X \in 2^\omega | \exists n(X(n) = 1)\}$ we see that O is defined by a Σ^0_1 formula and hence is a Σ^0_1 set.

Note that while both the elements of the Cantor space (regarded as sets of integers) and subsets of the Cantor space are classified in arithmetic hierarchies but these are not the same hierarchy. In fact the relationship between the two hierarchies is interesting and non-trivial. For instance the Π^0_n elements of the Cantor space are not (in general) the

same as the elements X of the Cantor space so that $\{X\}$ is a Π^0_nsubset of the Cantor space. However, many interesting results relate the two hierarchies.

There are two ways that a subset of Baire space can be classified in the arithmetical hierarchy.

- A subset of Baire space has a corresponding subset of Cantor space under the map that takes each function from ω to ω to the characteristic function of its graph. A subset of Baire space is given the classification Σ^1_n, Π^1_n, or Δ^1_nif and only if the corresponding subset of Cantor space has the same classification.
- An equivalent definition of the analytical hierarchy on Baire space is given by defining the analytical hierarchy of formulas using a functional version of second-order arithmetic; then the analytical hierarchy on subsets of Cantor space can be defined from the hierarchy on Baire space. This alternate definition gives exactly the same classifications as the first definition.

A parallel definition is used to define the arithmetical hierarchy on finite Cartesian powers of Baire space or Cantor space, using formulas with several free variables. The arithmetical hierarchy can be defined on any effective Polish space; the definition is particularly simple for Cantor space and Baire space because they fit with the language of ordinary second-order arithmetic.

Note that we can also define the arithmetic hierarchy of subsets of the Cantor and Baire spaces relative to some set of integers. In fact boldface $\mathbf{\Sigma}^0_n$is just the union of $\Sigma^{0,Y}_n$ for all sets of integers Y. Note that the boldface hierarchy is just the standard hierarchy of Borel sets.

Extensions and variations

It is possible to define the arithmetical hierarchy of formulas using a language extended with a function symbol for each primitive recursive function. This variation slightly changes the classification of some sets.

A more semantic variation of the hierarchy can be defined on all finitary relations on the natural numbers; the following definition is used. Every computable relation is defined to be $\Sigma^0_0$and Π^0_0. The classifications $\Sigma^0_n$and $\Pi^0_n$are defined inductively with the following rules.

- If the relation $R(n_1, \ldots, n_l, m_1, \ldots, m_k)$is Σ^0_nthen the relation $S(n_1, \ldots, n_l) = \forall m_1 \cdots \forall m_k R(n_1, \ldots, n_l, m_1, \ldots, m_k)$is defined to be Π^0_{n+1}

- If the relation $R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is Π^0_n then the relation $S(n_1, \ldots, n_l) = \exists m_1 \cdots \exists m_k R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is defined to be Σ^0_{n+1}

This variation slightly changes the classification of some sets. It can be extended to cover finitary relations on the natural numbers, Baire space, and Cantor space.

Meaning of the notation

The following meanings can be attached to the notation for the arithmetical hierarchy on formulas.

The subscript *n* in the symbols Σ^0_n and Π^0_n indicates the number of alternations of blocks of universal and existential number quantifiers that are used in a formula. Moreover, the outermost block is existential in Σ^0_n formulas and universal in Π^0_n formulas.

The superscript 0 in the symbols Σ^0_n, Π^0_n, and Δ^0_n indicates the type of the objects being quantified over. Type 0 objects are natural numbers, and objects of type *i* + 1 are functions that map the set of objects of type *i* to the natural numbers. Quantification over higher type objects, such as functions from natural numbers to natural numbers, is described by a superscript greater than 0, as in the analytical hierarchy. The superscript 0 indicates quantifiers over numbers, the superscript 1 would indicate quantification over functions from numbers to numbers (type 1 objects), the superscript 2 would correspond to quantification over functions that take a type 1 object and return a number, and so on.

Examples

- The Σ^0_1 sets of numbers are those definable by a formula of the form $\exists n_1 \cdots \exists n_k \psi(n_1, \ldots, n_k, m)$ where ψ has only bounded quantifiers. These are exactly the recursively enumerable sets.

- The set of natural numbers that are indices for Turing machines that compute total functions is Π^0_2. Intuitively, an index *e* falls into this set if and only if for every *m* "there is an *s* such that the Turing machine with index *e* halts on input *m* after *s* steps". A complete proof would show that the property displayed in quotes in the previous sentence is definable in the language of Peano arithmetic by a Σ^0_1 formula.

- Every Σ^0_1 subset of Baire space or Cantor space is an open set in the usual topology on the space. Moreover, for any such set there is a computable

enumeration of Gödel numbers of basic open sets whose union is the original set. For this reason, Σ^0_1sets are sometimes called *effectively open*. Similarly, every Π^0_1set is closed and the Π^0_1sets are sometimes called *effectively closed*.

- Every arithmetical subset of Cantor space of Baire space is a Borel set. The lightface Borel hierarchy extends the arithmetical hierarchy to include additional Borel sets. For example, every Π^0_2subset of Cantor or Baire space is a G_δ set (that is, a set which equals the intersection of countably many open sets). Moreover, each of these open sets is $\Sigma^0_1$and the list of Gödel numbers of these open sets has a computable enumeration. If φ(*X*,*n*,*m*) is a Σ^0_0formula with a free set variable *X* and free number variables *n*,*m* then the Π^0_2set $\{X \mid \forall n \exists m \phi(X, n, m)\}$ is the intersection of the Σ^0_1sets of the form $\{X \mid \exists m \phi(X, n, m)\}$ as *n* ranges over the set of natural numbers.

Properties

The following properties hold for the arithmetical hierarchy of sets of natural numbers and the arithmetical hierarchy of subsets of Cantor or Baire space.

- The collections $\Pi^0_n$and $\Sigma^0_n$are closed under finite unions and finite intersections of their respective elements.
- A set is Σ^0_nif and only if its complement is Π^0_n. A set is Δ^0_nif and only if the set is both $\Sigma^0_n$and Π^0_n, in which case its complement will also be Δ^0_n.
- The inclusions $\Delta^0_n \subsetneq \Pi^0_n$and $\Delta^0_n \subsetneq \Sigma^0_n$hold for $n \geq 1$.
- The inclusions $\Pi^0_n \subsetneq \Pi^0_{n+1}$and $\Sigma^0_n \subsetneq \Sigma^0_{n+1}$hold for all *n* and the inclusion $\Sigma^0_n \cup \Pi^0_n \subsetneq \Delta^0_{n+1}$holds for $n \geq 1$. Thus the hierarchy does not collapse.

Incompleteness theorem for halting probabilities

For each specific consistent effectively represented axiomatic system for the natural numbers, such as Peano arithmetic, there exists a constant *N* such that no bit of Ω after the *N*th can be proven to be one or zero within that system. The constant *N* depends on how the formal system is effectively represented, and thus does not directly reflect the complexity of the axiomatic system. This incompleteness result is similar to Gödel's incompleteness theorem in that it shows that no consistent formal theory for arithmetic can be complete.

Super Omega

As mentioned above, the first n bits of Gregory Chaitin's constant Omega are random or incompressible in the sense that we cannot compute them by a halting algorithm with fewer than n-O(1) bits. However, consider the short but never halting algorithm which systematically lists and runs all possible programs; whenever one of them halts its probability gets added to the output (initialized by zero). After finite time the first n bits of the output will never change any more (it does not matter that this time itself is not computable by a halting program). So there is a short non-halting algorithm whose output converges (after finite time) onto the first n bits of Omega, for any n. In other words, the enumerable first n bits of Omega are highly compressible in the sense that they are limit-computable by a very short algorithm; they are not random with respect to the set of enumerating algorithms. Jürgen Schmidhuber (2000) constructed a limit-computable "Super Omega" which in a sense is much more random than the original limit-computable Omega, as one cannot significantly compress the Super Omega by any enumerating non-halting algorithm.

Chapter 10

Kolmogorov Complexity

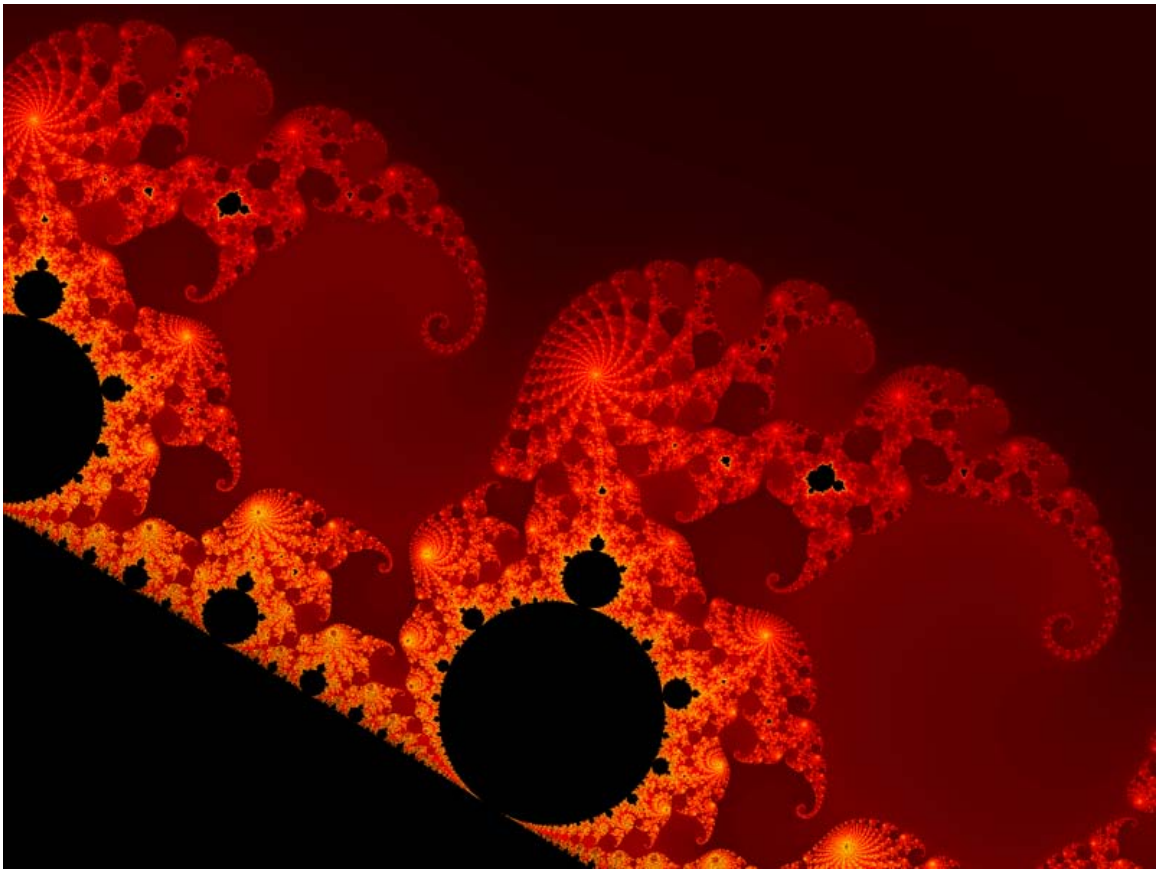
In algorithmic information theory (a subfield of computer science), the **Kolmogorov complexity** of an object, such as a piece of text, is a measure of the computational resources needed to specify the object.

Kolmogorov complexity is also known as **descriptive complexity**, **Kolmogorov-Chaitin complexity**, **stochastic complexity**, **algorithmic entropy**, or **program-size complexity**.

For example, consider the following two strings of length 64, each containing only lowercase letters, numbers, and spaces:

```
abababababababababababababababababababababababababababababababab
4c1j5b2p0cv4w1x8rx2y39umgw5q85s7uraqbjfdppa0q7nieieqe9noc4cvafzf
```

The first string has a short English-language description, namely "ab 32 times", which consists of 11 characters. The second one has no obvious simple description (using the same character set) other than writing down the string itself, which has 64 characters.



This image illustrates part of the Mandelbrot set fractal. Simply storing the 24-bit color of each pixel in this image would require 1.62 million bits; but a small computer program can reproduce these 1.62 million bits using the definition of the Mandelbrot set. Thus, the Kolmogorov complexity of the raw file encoding this bitmap is much less than 1.62 million.

More formally, the complexity of a string is the length of the string's shortest description in some fixed universal description language. The sensitivity of complexity relative to the choice of description language is discussed below. It can be shown that the Kolmogorov complexity of any string cannot be more than a few bytes larger than the length of the string itself. Strings whose Kolmogorov complexity is small relative to the string's size are not considered to be complex. The notion of Kolmogorov complexity can be used to state and prove impossibility results akin to Gödel's incompleteness theorem and Turing's halting problem.

Definition

To define Kolmogorov complexity, we must first specify a description language for strings. Such a description language can be based on any programming language, such as Lisp, Pascal, or Java Virtual Machine bytecode. If **P** is a program which outputs a string *x*, then **P** is a description of *x*. The length of the description is just the length of **P** as a

character string. In determining the length of **P**, the lengths of any subroutines used in **P** must be accounted for. The length of any integer constant *n* which occurs in the program **P** is the number of bits required to represent *n*, that is (roughly) $\log_2 n$.

We could alternatively choose an encoding for Turing machines, where an *encoding* is a function which associates to each Turing Machine **M** a bitstring <**M**>. If **M** is a Turing Machine which on input *w* outputs string *x*, then the concatenated string <**M**> *w* is a description of *x*. For theoretical analysis, this approach is more suited for constructing detailed formal proofs and is generally preferred in the research literature. The binary lambda calculus may provide the simplest definition of complexity yet.

Any string *s* has at least one description, namely the program

```
function GenerateFixedString()
   return s
```

If a description of *s*, *d*(*s*), is of minimal length—i.e. it uses the fewest number of characters—it is called a **minimal description** of *s*. Then the length of *d*(*s*)—i.e. the number of characters in the description—is the **Kolmogorov complexity** of *s*, written *K*(*s*). Symbolically,

$$K(s) = |d(s)|.$$

We now consider how the choice of description language affects the value of *K* and show that the effect of changing the description language is bounded.

Theorem. If K_1 and K_2 are the complexity functions relative to description languages L_1 and L_2, then there is a constant *c* (which depends only on the languages L_1 and L_2) such that

$$\forall s \ |K_1(s) - K_2(s)| \leq c.$$

Proof. By symmetry, it suffices to prove that there is some constant *c* such that for all bitstrings *s*,

$$K_1(s) \leq K_2(s) + c.$$

Now, suppose there is a program in the language L_1 which acts as an interpreter for L_2:

```
function InterpretLanguage(string p)
```

where *p* is a program in L_2. The interpreter is characterized by the following property:

> Running InterpretLanguage on input *p* returns the result of running *p*.

Thus if **P** is a program in L_2 which is a minimal description of s, then InterpretLanguage(**P**) returns the string s. The length of this description of s is the sum of

1. The length of the program InterpretLanguage, which we can take to be the constant c.
2. The length of **P** which by definition is $K_2(s)$.

This proves the desired upper bound.

History and context

Algorithmic information theory is the area of computer science that studies Kolmogorov complexity and other complexity measures on strings (or other data structures).

The concept and theory of Kolmogorov Complexity is based on a crucial theorem first discovered by Ray Solomonoff who published it in 1960, describing it in "A Preliminary Report on a General Theory of Inductive Inference" as part of his invention of Algorithmic Probability. He gave a more complete description in his 1964 publications, "A Formal Theory of Inductive Inference," Part 1 and Part 2 in *Information and Control*.

Andrey Kolmogorov later independently published this theorem in *Problems Inform. Transmission*, 1, (1965), 1-7. Gregory Chaitin also presents this theorem in J. ACM, 16 (1969). Chaitin's paper was submitted October 1966, revised in December 1968 and cites both Solomonoff's and Kolmogorov's papers.

The theorem says that among algorithms that decode strings from their descriptions (codes) there exists an optimal one. This algorithm, for all strings, allows codes as short as allowed by any other algorithm up to an additive constant that depends on the algorithms, but not on the strings themselves. Solomonoff used this algorithm, and the code lengths it allows, to define a string's `universal probability' on which inductive inference of a string's subsequent digits can be based. Kolmogorov used this theorem to define several functions of strings: complexity, randomness, and information.

When Kolmogorov became aware of Solomonoff's work, he acknowledged Solomonoff's priority (IEEE Trans. Inform Theory, 14:5(1968), 662-664). For several years, Solomonoff's work was better known in the Soviet Union than in the Western World. The general consensus in the scientific community, however, was to associate this type of complexity with Kolmogorov, who was concerned with randomness of a sequence while Algorithmic Probability became associated with Solomonoff, who focused on prediction using his invention of the universal a priori probability distribution.

There are several other variants of Kolmogorov complexity or algorithmic information. The most widely used one is based on self-delimiting programs and is mainly due to Leonid Levin (1974).

An axiomatic approach to Kolmogorov complexity based on Blum axioms (Blum 1967) was introduced by Mark Burgin in the paper presented for publication by Andrey Kolmogorov (Burgin 1982). This approach was further developed in the book (Burgin 2005) and applied to software metrics (Burgin and Debnath, 2003; Debnath and Burgin, 2003).

Some consider that naming the concept "Kolmogorov complexity" is an example of the Matthew effect.

Basic results

In the following discussion let $K(s)$ be the complexity of the string s.

It is not hard to see that the minimal description of a string cannot be too much larger than the string itself: the program GenerateFixedString above that outputs s is a fixed amount larger than s.

Theorem. There is a constant c such that

$$\forall s \; K(s) \leq |s| + c.$$

Incomputability of Kolmogorov complexity

The first result is that there is no way to effectively compute K.

Theorem. K is not a computable function.

In other words, there is no program which takes a string s as input and produces the integer $K(s)$ as output. We show this by contradiction by making a program that creates a string that should only be able to be created by a longer program. Suppose there is a program

```
function KolmogorovComplexity(string s)
```

that takes as input a string s and returns $K(s)$. Now consider the program

```
function GenerateComplexString(int n)
   for i = 1 to infinity:
      for each string s of length exactly i
         if KolmogorovComplexity(s) >= n
            return s
            quit
```

This program calls KolmogorovComplexity as a subroutine. This program tries every string, starting with the shortest, until it finds a string with complexity at least n, then returns that string. Therefore, given any positive integer n, it produces a string with Kolmogorov complexity at least as great as n. The program itself has a fixed length U.

The input to the program GenerateComplexString is an integer n; here, the size of n is measured by the number of bits required to represent n which is $\log_2(n)$. Now consider the following program:

```
function GenerateParadoxicalString()
      return GenerateComplexString(n0)
```

This program calls GenerateComplexString as a subroutine and also has a free parameter n_0. This program outputs a string s whose complexity is at least n_0. By an auspicious choice of the parameter n_0 we will arrive at a contradiction. To choose this value, note s is described by the program GenerateParadoxicalString whose length is at most

$$U + \log_2(n_0) + C$$

where C is the "overhead" added by the program GenerateParadoxicalString. Since n grows faster than $\log_2(n)$, there exists a value n_0 such that

$$U + \log_2(n_0) + C < n_0.$$

But this contradicts the definition of having a complexity at least n_0. That is, by the definition of $K(s)$, the string s returned by GenerateParadoxicalString is only supposed to be able to be generated by a program of length n_0 or longer, but GenerateParadoxicalString is shorter than n_0. Thus the program named "KolmogorovComplexity" cannot actually computably find the complexity of arbitrary strings.

This is proof by contradiction where the contradiction is similar to the Berry paradox: "Let n be the smallest positive integer that cannot be defined in fewer than twenty English words." It is also possible to show the uncomputability of K by reduction from the uncomputability of the halting problem H, since K and H are Turing-equivalent.

In the programming languages community there is a corollary known as the Full employment theorem, stating there is no perfect size-optimizing compiler.

Chain rule for Kolmogorov complexity

The chain rule for Kolmogorov complexity states that

$$K(X, Y) = K(X) + K(Y|X) + O(\log(K(X, Y))).$$

It states that the shortest program that reproduces X and Y is no more than a logarithmic term larger than a program to reproduce X and a program to reproduce Y given X. Using this statement one can define an analogue of mutual information for Kolmogorov complexity.

Compression

It is straightforward to compute upper bounds for $K(s)$: simply compress the string s with some method, implement the corresponding decompressor in the chosen language, concatenate the decompressor to the compressed string, and measure the resulting string's length.

A string s is compressible by a number c if it has a description whose length does not exceed $|s| - c$. This is equivalent to saying $K(s) \leq |s| - c$. Otherwise s is incompressible by c. A string incompressible by 1 is said to be simply *incompressible*; by the pigeonhole principle, incompressible strings must exist, since there are 2^n bit strings of length n but only $2^n - 1$ shorter strings, that is strings of length $n - 1$ or less.

For the same reason, most strings are complex in the sense that they cannot be significantly compressed: $K(s)$ is not much smaller than $|s|$, the length of s in bits. To make this precise, fix a value of n. There are 2^n bitstrings of length n. The uniform probability distribution on the space of these bitstrings assigns exactly equal weight 2^{-n} to each string of length n.

Theorem. With the uniform probability distribution on the space of bitstrings of length n, the probability that a string is incompressible by c is at least $1 - 2^{-c+1} + 2^{-n}$.

To prove the theorem, note that the number of descriptions of length not exceeding $n - c$ is given by the geometric series:

$$1 + 2 + 2^2 + \cdots + 2^{n-c} = 2^{n-c+1} - 1.$$

There remain at least

$$2^n - 2^{n-c+1} + 1$$

many bitstrings of length n that are incompressible by c. To determine the probability divide by 2^n.

This theorem is the justification for various challenges in comp.compression FAQ. Despite this result, it is sometimes claimed by certain individuals (considered cranks) that they have produced algorithms which uniformly compress data without loss.

Chaitin's incompleteness theorem

We know that, in the set of all possible strings, most strings are complex in the sense that they cannot be described in any significantly "compressed" way. However, it turns out that the fact that a specific string is complex cannot be formally proved, if the string's complexity is above a certain threshold. The precise formalization is as follows. First fix a particular axiomatic system **S** for the natural numbers. The axiomatic system has to be

powerful enough so that to certain assertions **A** about complexity of strings one can associate a formula $\mathbf{F_A}$ in **S**. This association must have the following property: if $\mathbf{F_A}$ is provable from the axioms of **S**, then the corresponding assertion **A** is true. This "formalization" can be achieved either by an artificial encoding such as a Gödel numbering or by a formalization which more clearly respects the intended interpretation of **S**.

Theorem. There exists a constant L (which only depends on the particular axiomatic system and the choice of description language) such that there does not exist a string s for which the statement

$$K(s) \geq L$$

(as formalized in **S**) can be proven within the axiomatic system **S**.

Note that by the abundance of nearly incompressible strings, the vast majority of those statements must be true.

The proof of this result is modeled on a self-referential construction used in Berry's paradox. The proof is by contradiction. If the theorem were false, then

Assumption (X): For any integer n there exists a string s for which there is a proof in **S** of the formula "$K(s) \geq n$" (which we assume can be formalized in **S**).

We can find an effective enumeration of all the formal proofs in **S** by some procedure

```
function NthProof(int n)
```

which takes as input n and outputs some proof. This function enumerates all proofs. Some of these are proofs for formulas we do not care about here (examples of proofs which will be listed by the procedure NthProof are the various known proofs of the law of quadratic reciprocity, those of Fermat's little theorem or the proof of Fermat's last theorem all translated into the formal language of **S**). Some of these are complexity formulas of the form $K(s) \geq n$ where s and n are constants in the language of **S**. There is a program

```
function NthProofProvesComplexityFormula(int n)
```

which determines whether the n^{th} proof actually proves a complexity formula $K(s) \geq L$. The strings s and the integer L in turn are computable by programs:

```
function StringNthProof(int n)
function ComplexityLowerBoundNthProof(int n)
```

Consider the following program

```
function GenerateProvablyComplexString(int n)
```

```
     for i = 1 to infinity:
        if  NthProofProvesComplexityFormula(i) and
ComplexityLowerBoundNthProof(i) ≥ n
           return StringNthProof(i)
           quit
```

Given an n, this program tries every proof until it finds a string and a proof in the formal system **S** of the formula $K(s) \geq L$ for some $L \geq n$. The program terminates by our **Assumption (X)**. Now this program has a length U. There is an integer n_0 such that $U + \log_2(n_0) + C < n_0$, where C is the overhead cost of

```
   function GenerateProvablyParadoxicalString()
      return GenerateProvablyComplexString(n0)
      quit
```

The program GenerateProvablyParadoxicalString outputs a string s for which there exists an L such that $K(s) \geq L$ can be formally proved in **S** with $L \geq n_0$. In particular $K(s) \geq n_0$ is true. However, s is also described by a program of length $U + \log_2(n_0) + C$ so its complexity is less than n_0. This contradiction proves **Assumption (X)** cannot hold.

Similar ideas are used to prove the properties of Chaitin's constant.

Kolmogorov randomness

Kolmogorov randomness (also called *algorithmic randomness*) defines a string (usually of bits) as being random if and only if it is shorter than any computer program that can produce that string. This definition of randomness is critically dependent on the definition of Kolmogorov complexity. To make this definition complete, a computer has to be specified, usually a Turing machine. According to the above definition of randomness, a random string is also an "incompressible" string, in the sense that it is impossible to give a representation of the string using a program whose length is shorter than the length of the string itself. However, according to this definition, most strings shorter than a certain length end up to be (Chaitin-Kolmogorovically) random because the best one can do with very small strings is to write a program that simply prints these strings.

Chapter 11

Binary Lambda Calculus and Linear Partial Information

Binary lambda calculus

Binary lambda calculus (**BLC**) is a technique for using the lambda calculus to study Kolmogorov complexity, by working with a standard binary encoding of lambda terms, and a designated universal machine. Binary lambda calculus is a new idea introduced by John Tromp in 2008.

Background

BLC is designed to provide a very simple and elegant concrete definition of descriptional complexity (Kolmogorov complexity).

Roughly speaking, the complexity of an object is the length of its shortest description.

To make this precise, we take descriptions to be bitstrings, and identify a description method with some computational device, or machine, that transforms descriptions into objects. Objects are usually also just bitstrings, but can have additional structure as well, e.g., pairs of strings.

Originally, Turing machines were used for this purpose, being perhaps the most well known formalism for computation. But they are somewhat lacking in ease of construction and composability. Another classical computational formalism, the Lambda calculus, offers distinct advantages in ease of use. The lambda calculus doesn't incorporate any notion of I/O though, so one needs to be designed.

Binary I/O

Adding a readbit primitive function to lambda calculus, as Chaitin did for LISP, would destroy its referential transparency, unless one distinguishes between an I/O action and its result, as Haskell does with its monadic I/O. But that requires a type system, an unnecessary complication.

Instead, BLC requires translating bitstrings into lambda terms, to which the machine (itself a lambda term) can be readily applied.

Bits 0 and 1 are translated into the standard lambda booleans B_0 = True and B_1 = False:

$$\text{True} = \lambda x\ \lambda y.\ x$$
$$\text{False} = \lambda x\ \lambda y.\ y$$

which can be seen to directly implement the if-then-else operator.

Now consider the pairing function

$$\langle , \rangle = \lambda x\ \lambda y\ \lambda z.\ zxy$$

applied to two terms M and N

$$\langle M, N \rangle = \lambda z.\ zMN.$$

Applying this to a boolean yields the desired component of choice.

A string $s = b_0 b_1 \ldots b_{n-1}$ is represented by repeated pairing as

$$\langle B_0, \langle B_1 \ldots \langle B_{n-1}, z \rangle \ldots \rangle \rangle$$ which is denoted as $s : z$.

The z appearing in the above expression requires some further explanation.

Delimited versus undelimited

Descriptional complexity actually comes in two distinct flavors, depending on whether the input is considered to be delimited.

Knowing the end of your input makes it easier to describe objects. For instance, you can just copy the whole input to output. This flavor is called *plain* or *simple* complexity.

But in a sense it is additional information. A file system for instance needs to separately store the length of files. The C language uses the null character to denote the end of a string, but this comes at the cost of not having that character available within strings.

The other flavor is called *prefix* complexity, named after prefix codes, where the machine needs to figure out, from the input read so far, whether it needs to read more bits. We say that the input is self-delimiting. This works better for communication channels, since one can send multiple descriptions, one after the other, and still tell them apart.

In the I/O model of BLC, the flavor is dictated by the choice of z. If we keep it as a free variable, and demand that it appears as part of the output, then the machine must be

working in a self-delimiting manner. If on the other hand we use a lambda term specifically designed to be easy to distinguish from any pairing, then the input becomes delimited. BLC chooses *False* for this purpose but gives it the more descriptive alternative name of *Nil*. Dealing with lists that may be Nil is straightforward: since

$$\langle x, y\rangle\ M\ N = M\ x\ y\ N$$, and
$$Nil\ M\ N = N$$

one can write function *M* and *N* to deal with the two cases, the only caveat being that *N* will be passed to *M* as its third argument.

Universality

We can find a description method *U* such that for any other description method *D*, there is a constant *c* (depending only on *D*) such that no object takes more than *c* extra bits to describe with method *U* than with method *D*. And BLC is designed to make these constants relatively small. In fact the constant will be the length of a binary encoding of a *D*-interpreter written in BLC, and *U* will be a lambda term that parses this encoding and runs this decoded interpreter on the rest of the input. *U* won't even have to know whether the description are delimited or not; it works the same either way.

Having already solved the problem of translating bitstring into lambda calculus, we now face the opposite problem: how to encode lambda terms into bitstrings?

Lambda encoding

First we need to write our lambda terms in a particular notation using what is known as De Bruijn indices. Our encoding is then defined recursively as follows

$$\widehat{\lambda M} = 00\widehat{M}$$
$$\widehat{M\ N} = 01\widehat{M}\widehat{N}$$
$$\widehat{i} = 1^i0$$

For instance, the pairing function λ*x*λ*y*λ*z*.*zxy* is written λλλ.132 in De Bruijn format, which has encoding $00\ 00\ 00\ 01\ 01\ 10\ 1110\ 110$.

A **closed** lambda term is one in which all variables are bound, i.e. without any free variables. In De Bruijn format, this means that an index *i* can only appear within at least *i* nested lambdas. The number of closed terms of size *n* bits is given by sequence A114852 of the On-Line Encyclopedia of Integer Sequences.

The shortest possible closed term is the identity function $\widehat{\lambda 1} = 0010$. In delimited mode, this machine just copies its input to its output.

So, what is this universal machine *U*?

Here it is, in De Bruijn format (all indices are single digit):

(λ11)(λ(λλλ1(λλ1(λ3(6(λ2(6(λλ3(λ123)))(7(λ7(λ31(21))))))(1(5(λ12))(λ7(λ7(λ2(14)))3)))))(11))(λ1((λ11)(λ11)))

This is in binary:

01010001101000010000000110000001100001011110011111110000101
11001111111000000111100001011011011100111111110000111111110
00010111101001110100101100111111000011011000010111111110000
11111111000011100110111101110011010000110010001101000011010

A detailed analysis of machine *U* may be found in.

Complexity, concretely

In general, we can make complexity of an object conditional on several other objects that are provided as additional argument to the universal machine. Plain (or simple) complexity *KS* and prefix complexity *KP* are defined by

$$\begin{aligned} KS(x|y_1,\ldots,y_k) &= \min\{\ell(p) \mid U\ (p : Nil)\ y_1\ \ldots\ y_k = \ x\ \ \} \\ KP(x|y_1,\ldots,y_k) &= \min\{\ell(p) \mid U\ (p :\ z\ )\ y_1\ \ldots\ y_k = \langle x,z\rangle\} \end{aligned}$$

Theorems, concretely

The identity program λ1 proves that

$$KS(x) \leq \ell(x) + 4$$

The program
λλ(λ11)(λ(λλλ1(λλλλ1(λ8(λ8(λ132))((λ11)(λ(λλ1(λλ2(1(λλλλ1(λλ1)(85))1)(λ1(λλ2)2)))(11))6)))(21))(11))(λλλ132)12 proves that

$$KP(x|\ell(x)) \leq \ell(x) + 239$$

The program
(λ11)(λλλ1(λ1(3(λλ1))(44((λ11)(λ(λλλ1(λλλλ1(λ8(λ8(λ132))((λ11)(λ(λλ1(λλ2(1(λλλλ1(λλ1)(85))1)
(λ1(λλ2)2)))(11))6)))(21))(11))(λ4((λ11)(λλλ2(λλλ4(λλ512)(3(λ1)(66)2))(1(λλ2)2))1))))))
(λλλ132) proves that

$$KP(x) \leq \ell(\overline{x}) + 401$$

where $\overline{x}$ is a self-delimiting code for x defined by

$$\begin{aligned}\overline{0} &= 0 \\ \overline{n+1} &= 1\ \overline{\ell(n)}\ n\end{aligned}$$

in which we identify numbers and bitstrings according to lexicographic-in-reverse order. This code has the nice property that for all k,

$$\ell(\overline{n}) \leq \ell(n) + \ell(\ell(n)) + \cdots + \ell^{k-1}(n) + O(\ell^k(n))$$

Number	String	Delimited
0		0
1	0	10
2	1	110 0
3	00	110 1
4	10	1110 0 00
5	01	1110 0 10
6	11	1110 0 01
7	000	1110 0 11
8	100	1110 1 000
9	010	1110 1 100

Symmetry of information

The program

(λ(λ(λλ(λ12(λλ31(54)(λλλ1(λ164)2)))((λ11)(λ(λλλ1(λλ1(λ3(6(λ2(6(λλ3(λ123)))(7(λ7(λ31(21))))))(1(5(λ12))(λ7(λ7(λ2(14)))3)))))(11))(λ11)))((λ11)(λ(λλλ1(λλ1(λ(λ4(7(λλ4(8(λ4(31))(λλλ34))(9(λλ10(λ6(5(31)))(λλλ264)))))(λ(λ11)(λλλ1(λ1(3(λ1(λ1)))(44(λ4(161)))))(λ8(λ4(2(λλ2)31))(λλλ14))4))(λλ15(λ143)))))(11))(λλλ3((λ11)(λλλλλ(λ11)(λ(λλλ2(λλλλ3(λ85(λ127)))(λλ523(λ1(624))(λλλ1)(λλλ1))(λ1(λλ1(λλ2))2))(11))(λλλ2(λλλ210(9(λλλ111)))(12(λλ1)))(λλλλ2)22(1(λ665(λ5(λλλ3(λλλ2(λλλ2(λλλ1(λ1(λ1)))(2011))7)))(113))))2(λ1)(λ1(λ1))1(λλ2)1(λλ1)))))(λλλλ3(λλλ3(λλλ1(1011(λ1)(λ1(λ1)))))))(λλλ1(λ4(λ4(λλ14(32)))))

proves that

$$KP(x, y) \leq KP(x) + KP(y|x^*) + 1388$$

where x^* is a shortest program for *x*.

This inequality is the easy half of an equality (up to a constant) known as **Symmetry of information**. Proving the converse

$$KP(y|x^*) \leq KP(x,y) - KP(x) + O(1)$$

involves simulating infinitely many programs in dovetailing fashion, seeing which ones halt and output the pair of *x* (for which a shortest program is given) and any *y*, and for each such program *p*, requesting a new codeword for *y* of length $\ell(p) - KP(x)$. The Kraft inequality ensures that this infinite enumeration of requests can be fulfilled, and in fact codewords for *y* can be decoded on the fly, in tandem with the above enumeration. Details of this fundamental result by Chaitin can be found in.

A quine

The term Q = λ(λ11)(λλ1(λλλλ14(6632)))11 concatenates two copies of its input, proving that

$$KS(xx) \leq \ell(x) + 74$$

Applying it to its own encoding gives a 148 bit quine:

$$U(\hat{Q}\hat{Q} : Nil) = \hat{Q}\hat{Q}$$

Compression

So far, we've seen surprisingly little in the way of actually compressing an object into a shorter description; in the last example, we were only breaking even. For $\ell(x) > 74$ though, we do actually compress *xx* by $\ell(x) - 74$ bits.

What could be the shortest program that produces a larger output? The following is a good candidate: the program (λ1111(λλ1(λλ1)2))(λλ2(21)), of size 55 bits, uses Church numerals to output exactly $2^{2^{2^2}} = 65536$ ones. That beats both gzip and bzip2, compressors that need 360 and 352 bits respectively, to describe the same output (as a 8192 byte file with a single letter name).

Halting probability

The halting probability of the prefix universal machine is defined as the probability it will output any term that has a normal form (this includes all translated strings):

$$\Omega_\lambda = \sum_{U(p:z)=\langle x,z\rangle,\, x\in NF} 2^{-\ell(p)}$$

With some effort, we can determine the first 4 bits of this particular number of wisdom:

$$\Omega_\lambda = .0001\ldots_2$$

where probability $.0001_2 = 2^{-4}$ is already contributed by programs *00100* and *00101* for terms True and False.

BLC8: byte sized I/O

While bit streams are nice in theory, they fare poorly in interfacing with the real world. The language BLC8 is a more practical variation on BLC in which programs operate on a stream of bytes, where each byte is represented as a delimited list of 8 bits in little-endian order.

BLC8 requires a more complicated universal machine:

*U*8 = (λ11)(λ(λλλ1(λλλ2(λλλ(λ7(10(λ5(2(λλ3(λ123)))(11(λ3(λ31(21)))))3)
(4(1(λ15)3)(10(λ2(λ2(16)))6)))8)(λ1(λ87(λ162))))(λ1(43)))(11))(λλ2((λ11)(λ11)))(λλ1)

The parser in U8 keeps track of both remaining bytes, and remaining bits in the current byte, discarding the latter when parsing is completed. Thus the size of U8, which is 357 bits as a BLC program, is 45 bytes in BLC8.

Brainfuck

The following BLC8 program

```
(λ(λ(λ11)(λ(λλλ1(λ1(λλ1(λλ1(λλ1)(λλ1)(λλ(λ3(1(7(5(λλλλλ1(λ6(λ1)143))(λλ2(λλ
132)1))(λλ2(λ1)((λ11)
(λλ(λλ1(λλλλ4(1(λλ1)(76(λ1)3))(1(λλ2)(7(λ1)63)))1)(221))71))))(7(1(5(λλλλ2(λλ
6(λ1)21(λ164)))(λλλλ1(λ5(λ1)1
(λ154)))))(5(10(λ1(λ1)))(11(λ2((λ11)(λλλλ(λ11)(λλ1(λλλ3(552)(λλ2))1)1(3(λ554(
λ4(λ3(21)))))(2(λ1))1)1))))))
(λ11(λ11(λ3(λ3(λ3(λ3(21)))))))))))(11))(λ1(λλλλ4)322))((λ11)(λλλ12(332))1))((λ
λ2(2(21)))(λλ2(21))(λλ1(λλ2)2)(λλ1))
```

is an unbounded tape Brainfuck interpreter in 936 bits (or, equivalently, 117 bytes). The formatting obscures the fact that line 3 has two double digit indices (10 and 11), while line 4 starts with another 2 occurrences of 11.

This provides a nice example of the property that universal description methods differ by at most a constant in complexity. Writing a BLC8 interpreter in Brainfuck, which would

provide a matching upper bound in the other direction, is left as an exercise for die-hard Brainfuck programmers.

The interpreter expects its input to consist of a Brainfuck program followed by a `]` followed by the input for the Brainfuck program. The interpreter only looks at bits 0,1,4 of each character to determine which of `,-.+<>][` it is, so any characters other than those 8 should be stripped from a Brainfuck program. Any unconsumed input is appended to the output of the Brainfuck program. Consuming more input than is available results in an error (the non-list result $\lambda x.x$).

To conform to the more standard behaviour of not changing the current cell on reaching EOF, replace the fragment 1(λ6(λ1)143) on line 1 with (λ21(1521))(λλλ8(λ1)3652) at a cost of 45 more bits.

Linear partial information

Linear partial information (LPI) is a method of making decisions based on insufficient or fuzzy information. LPI was introduced in 1970 by Polish - Swiss mathematician Edward Kofler (1911–2007) to simplify decision processes. Comparing to other methods the LPI-fuzziness is algorithmically simple and particularly in decision making, more practically oriented. Instead of an indicator function the decision maker linearizes any fuzziness by establishing of linear restrictions for fuzzy probability distributions or normalized weights. In the LPI-procedure the decision maker linearizes any fuzziness instead of applying a membership function. This can be done by establishing stochastic and non-stochastic LPI-relations. A mixed stochastic and non-stochastic fuzzification is often a basis for the LPI-procedure. By using the LPI-methods any fuzziness in any decision situation can be considered on the base of the linear fuzzy logic.

Definition

Any Stochastic Partial Information **SPI(p)**, which can be considered as a solution of a linear inequality system, is called Linear Partial Information **LPI(p)** about probability **p**. It can be considered as an LPI-fuzzification of the probability **p** corresponding to the concepts of linear fuzzy logic.

Applications

a) The MaxEmin Principle
To obtain the maximally warranted expected value, the decision maker has to choose the strategy which maximizes the minimal expected value. This procedure leads to the MaxEmin - Principle and is an extension of the Bernoulli's principle.

b) The MaxWmin Principle
This principle leads to the maximal guaranteed weight function, regarding the extreme weights.
c) The Prognostic Decision Principle (PDP)
This principle is based on the prognosis interpretation of strategies under fuzziness.

Fuzzy equilibrium and stability

Despite the fuzziness of information, it is often necessary to choose the optimal, most cautious strategy, for example in economic planning, in conflict situations or in daily decisions. This is impossible without the concept of fuzzy equilibrium. The concept of fuzzy stability is considered as an extension into a time interval, taking into account the corresponding stability area of the decision maker. The more complex is the model, the softer a choice has to be considered. The idea of fuzzy equilibrium is based on the optimization principles. Therefore the MaxEmin-, MaxGmin- and PDP-stability have to be analyzed. The violation of these principles leads often to wrong predictions and decisions.

LPI equilibrium point

Considering a given LPI-decision model, as a convolution of the corresponding fuzzy states or a disturbance set, the fuzzy equilibrium strategy remains the most cautious one, despite of the presence of the fuzziness. Any deviation from this strategy can cause a loss for the decision maker.

Chapter 12

Minimum Description Length

The **minimum description length principle** is a formalization of Occam's Razor in which the best hypothesis for a given set of data is the one that leads to the best compression of the data. MDL was introduced by Jorma Rissanen in 1978. It is an important concept in information theory and learning theory.

Overview

Any set of data can be represented by a string of symbols from a finite (say, binary) alphabet.

"The fundamental idea behind the MDL Principle is that any regularity in a given set of data can be used to compress the data, i.e. to describe it using fewer symbols than needed to describe the data literally." (Grünwald, 1998)

To select the hypothesis that captures the most regularity in the data, scientists look for the hypothesis with which the best compression can be achieved. In order to do this, a code is fixed to compress the data, most generally with a (Turing-complete) computer language. A program is written in that language that outputs the data, the program thus representing the data. The length of the shortest program that outputs the data is called the Kolmogorov complexity of the data. This is the central idea of Ray Solomonoff's idealized theory of inductive inference.

Inference

However, this mathematical theory does not provide a practical way of doing inference. The most important reasons for this are:

- Kolmogorov complexity is uncomputable; there exists no algorithm that, when input an arbitrary sequence of data, outputs the shortest program that produces the data.
- The Kolmogorov complexity depends on what computer language is used to describe programs. This is an arbitrary choice, but it does influence the complexity up to some constant additive term. For that reason, constant terms tend to be disregarded in Kolmogorov complexity theory. But in practice, where often only a small amount of data is available, such constants may have a very

large influence on the inference results: good results cannot be guaranteed when one is working with limited data.

MDL attempts to remedy these, by:

- Restricting the set of allowed codes in such a way that it becomes possible (computable) to find the shortest codelength of the data, relative to the allowed codes, and
- Choosing a code that is reasonably efficient whatever the data at hand. This point is somewhat elusive and much research is still going on in this area.

Rather than "programs", in MDL theory one usually speaks of candidate hypotheses, models or codes. The set of allowed codes is then called the model class. (To confuse matters, some authors refer to the model class as the model.) The code is then selected for which the sum of the description of the code and the description of the data with the help of the code is minimal.

One of the important properties of MDL methods is that they provide a natural safeguard against overfitting, because it implements a tradeoff between the complexity of the hypothesis (model class) and the complexity of the data given the hypothesis.

Example of MDL

A coin is flipped 1,000 times and the numbers of heads and tails are recorded.

- Consider two model classes: the first is a code that represents outcomes with a 0 for heads or a 1 for tails. This code represents the hypothesis that the coin is fair. The code length according to this code is always exactly 1,000 bits.
- The second model class consists of all codes that are efficient for a coin with some specific bias, representing the hypothesis that the coin is not fair. Say that we observe 510 heads and 490 tails. Then the code length according to the best code in the second model class is shorter than 1,000 bits.

For this reason a naive statistical method might choose the second model as a better explanation for the data. However, an MDL approach would construct a *single* code based on the hypothesis, instead of just using the best one. To do this, it's simplest to use a two-part code in which the element of the model class with the best performance is specified. Then the data is specified using that code. A lot of bits are needed to specify which code to use; thus the total codelength based on the second model class would be larger than 1,000 bits.

Conclusion

If you follow an MDL approach the conclusion is inevitably that there is not enough evidence to support the hypothesis of the 'biased coin', even though the best element of the second model class provides better fit to the data.

MDL Notation

Central to MDL theory is the one-to-one correspondence between code length functions and probability distributions. (The lemma involved is the Kraft-McMillan inequality.) For any probability distribution P, it is possible to construct a code C such that the length (in bits) of $C(x)$ is equal to $-\log_2 P(x)$; this code minimizes the expected code length. Vice versa, given a code C, one can construct a probability distribution P such that the same holds. (Rounding issues are ignored here.) In other words, searching for an efficient code reduces to searching for a good probability distribution, and vice versa.

Related concepts

MDL is very strongly connected to probability theory and statistics through the correspondence between codes and probability distributions mentioned above. This has led some researchers to view MDL as equivalent to Bayesian inference. Code length of the model and code length of model and data together in MDL correspond to prior probability and marginal likelihood respectively in the Bayesian framework. This point of view is expressed for example in David MacKay's *Information Theory, Inference, and Learning Algorithms*.

While Bayesian machinery is often useful in constructing efficient MDL codes, the MDL framework also accommodates other codes that are not Bayesian. An example is the Shtarkov 'normalized maximum likelihood code', which plays a central role in current MDL theory, but has no equivalent in Bayesian inference. Furthermore, Rissanen stresses that we should *make no assumptions* about the *true* data generating process: in practice, a model class is typically a simplification of reality and thus does not contain any code or probability distribution that is true in any objective sense.

According to the MDL philosophy, Bayesian methods should be dismissed if they are based on 'unsafe' priors that would lead to poor results. The priors that are acceptable from an MDL point of view also tend to be favored in so-called *objective Bayesian* analysis; however, there the motivation is usually different.

Other Systems

MDL was not the first information-theoretic approach to learning; as early as 1968 Wallace and Boulton pioneered a related concept called Minimum Message Length (MML). The difference between MDL and MML is a source of ongoing confusion. Superficially, the methods appear mostly equivalent, but there are some significant differences, especially in interpretation:

- MML is a fully subjective Bayesian approach: it starts from the idea that one represents one's beliefs about the data generating process in the form of a prior distribution. MDL avoids assumptions about the data generating process.

- Both methods make use of two part codes: the first part always represents the information that one is trying to learn such as the index of a model class (model selection), or parameter values (parameter estimation). The second part is an encoding of the data given the information in the first part. The difference is that in the MDL literature, it is advocated that unwanted parameters should be moved to the second part of the code, where they can be represented with the data by using a so-called one-part code. This is often more efficient than a two-part code. In the original description of MML, all parameters are encoded in the first part so all parameters are learned.