

Lossless Source Coding

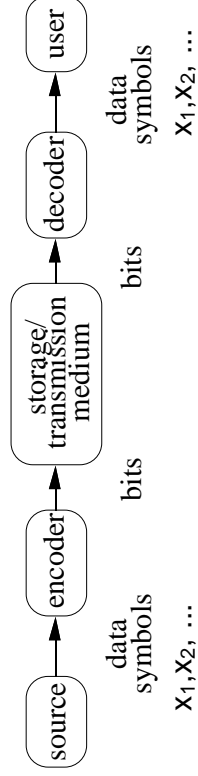
Formerly called noiseless coding, or coding for a noiseless channel.

References: Sayood Section 2.4, 3.1, 3.2, Neuhoff & Forney (to be posted)

Sections 9.1-9.5, Gersho&Gray.

Topic is also in many information and communication theory books.

Block diagram of source coding system:



Goal:

encode data symbols into bits so decoder can recover data symbols perfectly

Performance:

accuracy: in lossless coding, we demand perfect reproduction;
i.e. decoder produces same data symbols as source

efficiency: rate = R = avg. no. of bits per source symbol
should be as small as possible

Optimal Performance:

R_X^* = least rate of any lossless source code for source X

January 10, 2007

Lossless-1

Types of Lossless Source Codes

Block codes

- fixed-length to fixed-length
- fixed-to-variable length
- variable-to-fixed length
- variable-to-variable length

Non-block codes:

- conditional arithmetic
- Lempel-Ziv (dictionary)

We first consider

Fixed-length to variable-length block codes, where the input blocklength is 1.
Such codes encode one source symbol at a time, independently of other symbols. We call them "first-order" codes.

Then we consider a generalization: block-to-variable-length codes, which encode k source symbols at a time, for some $k \geq 1$.

Our goal at this time is not so much to learn the best practical codes, but to learn the fundamental limits of lossless source coding. Discussion of efficient practical techniques like arithmetic and Lempel-Ziv coding will be postponed.

January 10, 2007

Lossless-2

Examples of First-Order Codes

Source alphabet: $A = \{1,2,3,4\}$, Probabilities: $P = \{.5, .25, .125, .125\}$

Several codes: codebooks and encoding rules

p(x)	x	code code code code code code							
		a	b	c	d	e	f		
.5	1	$\underline{y}_1 =$	00	0	10	0	0	0	0
.25	2	$\underline{y}_2 =$	01	10	110	10	10	1	01
.125	3	$\underline{y}_3 =$	10	110	111	10	11	01	011
.125	4	$\underline{y}_4 =$	11	111	0	11	111	10	0111
rate			2	1.75	2.25	1.5	1.625	1.25	1.875
			FL	VL		not UD	not UD	not UD	UD, not prefix

code a: fixed-length

code b: variable-length; smaller rate than code a

code c: same codewords as code b; different assignment; larger rate.

code d: better rate than b; but doesn't distinguish 2 and 3

code e: good rate, but can't distinguish 32 from 41. 32 \rightarrow 1110, 41 \rightarrow 1110

code f: good rate, can't distinguish 3 \rightarrow 01 from 1 2 \rightarrow 01.

code g: simple to encode and decode, not as good as b.

How do we know codes a, b and c are uniquely decodable (UD)?

a and b have the property that no codeword is prefix of another. They are UD.

January 10, 2007

Lossless-3

Formal Introduction

Source

- Modeled as a discrete-time, discrete-valued random process, i.e. a sequence of random variables X_1, X_2, \dots , where each X_i is a discrete random variable representing i th source symbol.
- Unless otherwise stated we assume the random process is stationary and ergodic. This implies the random variables are identically distributed. That is, each has the same distribution as some generic discrete random variable X .
- The performance, i.e. rate, of first-order lossless codes depends only on the probability distribution of X , i.e. we don't need to know joint distributions.
- Let A denote the possible outcomes of X , i.e. the possible data symbols. Let M denote the cardinality of A , i.e. the number of distinct data symbols. There are two cases:
 - (a) $M < \infty$, i.e. A is a finite alphabet. We write $A = \{a_1, \dots, a_M\}$, $M < \infty$; or $w \log A = \{1, 2, \dots, M\}$.
 - (b) $M = \infty$, i.e. A is a countably infinite alphabet. We write $A = \{a_1, a_2, \dots\}$ or $w \log A = \{1, 2, 3, \dots\}$
- Let $P_i = \Pr(X=a_i)$ or $\Pr(X=i)$.
- Let $\underline{P} = (P_1, P_2, \dots)$ denote the ordered set of probabilities.
- Note: Lossless source codes do not exist for continuous-time or continuous-valued sources.

January 10, 2007

Lossless-4

First-order, variable-length codes (aka variable-length codes)

Described by a *binary codebook* $C = \{\underline{v}_1, \underline{v}_2, \dots\}$ of binary sequences, called *codewords*, of varying lengths.

Encoding rule: $e(a_i) = \underline{v}_i$ or $e(i) = \underline{v}_i$

That is, we assume the ordering of the codewords determines the assignment of codewords to symbols in the source alphabet.

Decoding rule: specific form depends on encoding rule

Performance, i.e. Rate:

$$R = \text{average codeword length} = \bar{L} = \sum_{i=1}^M l_i P_i \quad \text{bits/symbol}$$

where $l_i = l(\underline{v}_i)$ = length of \underline{v}_i

Visualize codebook with a binary tree

A tree has *nodes*, *branches*, a *root node*, *terminal nodes*, *interior nodes*, and *paths*.

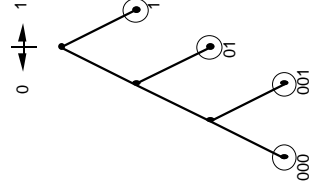
Each node is associated with the binary path leading to it.

To visualize a code we draw a tree with just enough branches so there is a path to a node corresponding to each codeword.

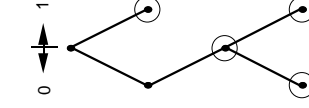
We mark the nodes corresponding to codewords. (Circles are used below.)

Examples:

$$C_1 = \{1, 01, 000, 0001\}$$



$$C_2 = \{1, 01, 011, 010\}$$



It's easy to see if one codeword is the prefix of another.

Can use an implementation of the tree (e.g. as linked list) for decoding C_1 in a linked list and use the tree to guide decoding.

Unique Decodability (UD) (codes need to have this property)

Defn: A code is uniquely decodable (UD) if any finite sequence of its codewords corresponds to only one message sequence.
Equivalently, the concatenation of any finite set of codewords is different than the concatenation of some other finite set of codewords.
Equivalently, the concatenation of any given sequence of codewords cannot be parsed in a different way into codewords.

Codes d, e, f are not UD

UD codes also called *separable* or *uniquely decipherable*.

Tests for UD

- In general it can be hard to tell if a code is UD.
- Examples: {00, 001, 1010, 0101}
- Example :
 {010, 0001, 0110, 1100, 00011, 00110, 11110, 101011}
- There is a systematic test due to Sardinas and Patterson for determining if a code is UD in finitely many steps. See, Sayood pp. 30,31. See *Information Theory*, R. Ash, for a proof that this method works.

January 10, 2007

Lossless-7

Defn: A code is **prefix** (prefix-free) if no codeword is the prefix of another.

- Prefix codes are UD, because the concatenation of any sequence of codewords can be parsed in only one way: If \underline{z} is such a sequence of codewords, then it must be prefixed by some codeword, say \underline{v} . It can't be prefixed by another codeword, say \underline{w} , because then either \underline{v} would prefix \underline{w} or vice versa, both of which are impossible. Therefore, \underline{v} is the only codeword that prefixes \underline{z} . Now remove \underline{v} from the beginning of \underline{z} and repeat the argument. Continuing in this way will produce the one and only parsing of \underline{z} into codewords.
- Codes a and b are prefix. The rest are not.
- Prefix codes are *instantaneously decodable*, meaning that as soon as one receives the last bit of a codeword one can decode it.
(They are sometimes called instantaneous codes.)
- There are UD codes that are not prefix, for example, Code g. Notice that it is not instantaneously decodable.

January 10, 2007

Lossless-8

- Some UD codes that are not prefix cannot be decoded until we stop using them.

Example: {00, 001, 1010, 0101}.

This is UD since it is suffix free. However, there can be problems with UD codes that are not prefix. For example, how to decode the infinite sequence

0 0 1 0 1 0 1 0 1 0 1 0 1 0 ...

From now on, unless otherwise stated, we restrict attention to prefix codes, because they are easy to recognize, easy to design, easy to decode, instantaneously decodable, and because there is no loss of performance, as can be shown.

January 10, 2007

Lossless-9

Questions of interest for first-order variable-length lossless codes

- Given a finite set of probabilities $\{P_1, \dots, P_M\}$, how to design a prefix code to have smallest rate, i.e. average length.

- What is the resulting smallest possible average length, i.e. rate?

Let L^* and R^* denote these quantities?

Basic idea: short codewords for large P_i 's

- But, cannot make all lengths small: (illustrate with tree)
- What sets of lengths can a prefix code have??
- Answer given by the Kraft Inequality Theorem.

January 10, 2007

Lossless-10

Kraft Inequality Theorem

(Sec. 2.4.3, Sayood; Ch 2, Sec. 2.7, Neuhoff & Forney)

Positive statement:

If integers l_1, \dots, l_M satisfy $\sum_{i=1}^M 2^{-l_i} \leq 1$, which is called the Kraft inequality, then there exists a prefix code with these lengths.

Negative statement: The lengths l_1, \dots, l_M of any UD code (prefix or not) satisfy the Kraft inequality, i.e. $\sum_{i=1}^M 2^{-l_i} \leq 1$

Combined statement: There is a UD code with lengths l_1, \dots, l_M

$$\text{iff } \sum_{i=1}^M 2^{-l_i} \leq 1.$$

Narrower statement: There is a prefix code with lengths l_1, \dots, l_M iff $\sum_{i=1}^M 2^{-l_i} \leq 1$.

Corollary: For any UD code, there is a prefix code with the same lengths. Thus restricting to prefix codes entails no loss in performance.

Infinite alphabets: The Kraft inequality theorem also holds when $|A| = M = \infty$

January 10, 2007

Lossless-11

Code Design Algorithms

Given a list of probabilities $\underline{P} = \{P_1, \dots, P_2\}$, we want to find a prefix code with small average length. Equivalently we want to find a set of lengths that satisfy the Kraft inequality and have small average length. The basic idea is that highest probability symbols should have the shortest codewords. But how short should they be?

Shannon Codes (aka Shannon-Fano codes)

Basic idea: use lengths

$$l_i \cong -\log_2 P_i$$

Is this possible? Lengths need to be integers; try

$$l_i = \lceil -\log_2 P_i \rceil \quad (\text{round up to be conservative})$$

and check Kraft inequality:

$$\sum_{i=1}^n 2^{-l_i} = \sum_{i=1}^n 2^{\lceil -\log_2 P_i \rceil} \leq \sum_{i=1}^n 2^{-\lfloor \log_2 P_i \rfloor} = \sum_{i=1}^n P_i = 1$$

Since Kraft inequality is satisfied, there are codes with these lengths.

These codes are called Shannon codes. The rate is

$$R_{\text{Sh}} = \sum_{i=1}^n P_i \lceil -\log_2 P_i \rceil$$

January 10, 2007

Lossless-12

How good are Shannon codes?

A. Upper bound

$$R_{Sh} = \sum_i P_i \lceil -\log_2 P_i \rceil < -\sum_i P_i \log_2 P_i + 1, \quad \text{because } \lceil -\log_2 P_i \rceil < -\log_2 P_i + 1$$

B. Lower bound: For any prefix or UD code whatsoever

$$R \geq -\sum_i P_i \log_2 P_i$$

Derivation: show $R - (-\sum_i P_i \log_2 P_i) \geq 0$:

$$\begin{aligned} R - (-\sum_i P_i \log_2 P_i) &= \sum_i P_i l_i + \sum_i P_i \log_2 P_i = \sum_i P_i (l_i + \log_2 P_i) \\ &= -\sum_i P_i \log_2 \frac{2^{-l_i}}{P_i} \\ &\geq -\sum_i P_i \left(\frac{2^{-l_i}}{P_i} - 1 \right) \frac{1}{\ln 2} \quad \text{because } \ln x \leq x-1 \Rightarrow \log_2 x = \frac{\ln x}{\ln 2} \leq \frac{x-1}{\ln 2} \\ &= -\left(\sum_i 2^{-l_i} - \sum_i P_i \right) \frac{1}{\ln 2} \\ &\geq -\left(1 - 1 \right) \frac{1}{\ln 2} = 0 \quad \text{because UD nature of code implies that its lengths satisfy the Kraft inequality} \end{aligned}$$

C. A and B show that no code could have rate less than $R_{Sh}-1$

January 10, 2007

Lossless-13

Definitions:

$$H = H(X) = H(\underline{P}) = -\sum_i P_i \log_2 P_i = \text{entropy of } X \text{ and } \underline{P}$$

$R^* = L^*$ = least rate and average length of any prefix (or UD) code.

In summary, we've established

Lossless Source Coding Theorem for first-order variable-length codes: For any set of probabilities,

$$H \leq R^* \leq R_{Sh} < H + 1,$$

That is, there exists a code (e.g. a Shannon code) with $R < H+1$, and every prefix code has $R \geq H$.

Notes:

- Entropy is obviously an important quantity. It can be interpreted as the amount of randomness in X or the uncertainty in X .
- Notice the three alternative notations for entropy.
- The theorem doesn't actually find the best code or R^* . But we've sandwiched R^* between H and $H+1$.
- The "Huffman" design algorithm will enable us to find R^* , but it won't indicate how R^* depends on the probabilities. This Theorem does.
- The rate of a Shannon code is always within 1 bit of R^* .

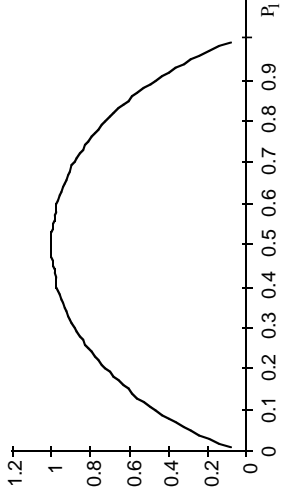
January 10, 2007

Lossless-14

Examples:

- Binary random variable: $P = \{P_1, 1-P_1\}$,

$$H = -P_1 \log_2 P_1 - (1-P_1) \log_2 (1-P_1) \\ \triangleq H(P_1)$$



- 4-ary random variable: $P = \{.5, .25, .125, .125\}$

$$H = 1.75, R_{Sh} = 1.75.$$

- More generally, whenever all probabilities are powers of 2,

$$R_{Sh} = H \quad (\text{an exercise})$$

- English text: $M = 27, \log_2 27 = 4.75, H \cong 4.12.$

Table of English letter probabilities given in Chap 2, N&F.

Properties of Entropy: $H = -\sum_i P_i \log_2 P_i$

0. $0 \log_2 0 \triangleq 0$, because $\lim_{p \rightarrow 0} p \log_2 p = 0$
1. "Fundamental inequality" of information theory:
 $\ln p \leq p-1$ and $\log_2 p = \frac{\ln p}{\ln 2} \leq \frac{p-1}{\ln 2}$
 Equality holds iff $p = 1$.
2. $H \geq 0$. $H = 0$ iff $P_i = 1$ for some i .
- Proof: $H \geq 0$ because $-P_i \log_2 P_i \geq 0$ each i .
 $(H = 0) \Leftrightarrow (-P_i \log_2 P_i = 0 \text{ each } i) \Leftrightarrow (P_i = 0 \text{ or } 1 \text{ each } i) \Leftrightarrow P_i = 1 \text{ some } i$.

3. If $|A| = M < \infty$, then $H \leq \log_2 M$. $H = \log_2 M$ iff $P_i = \frac{1}{M}$ all i
 Proof: $\log_2 M - H = \sum_i P_i (\log_2 M + \log_2 P_i) = -\sum_i P_i \log_2 \frac{1}{MP_i}$
 $\geq -\sum_i P_i \left(\frac{1}{MP_i} - 1\right) \frac{1}{\ln 2} = \left(-\sum_i \frac{1}{M} + \sum_i P_i\right) \frac{1}{\ln 2} = 0$
 with equality iff $\frac{1}{MP_i} = 1$ for each i

4. If $|A| = \infty$, then H can be finite or infinite.

5. Suppose one replaces P_1, P_2 by Q_1 and Q_2 where $Q_1+Q_2 = P_1+P_2$ and Q_1, Q_2 are more nearly equiprobable than P_1, P_2 . Then

$$H(P_1, P_2, P_3, \dots) < H(Q_1, Q_2, P_3, \dots)$$

This shows that entropy increases as the probabilities become more equal and decreases as they become more dissimilar.

Proof: WLOG assume $P_1 > Q_1 \geq Q_2 > P_2$. Let $\delta = P_1 - Q_1 = Q_2 - P_2 > 0$. Then

$$\begin{aligned} & H(Q_1, Q_2, P_3, \dots) - H(P_1, P_2, P_3, \dots) \\ &= -Q_1 \log_2 Q_1 - Q_2 \log_2 Q_2 + P_1 \log_2 P_1 + P_2 \log_2 P_2 \\ &= - (P_1 - \delta) \log_2 (P_1 - \delta) - (P_2 + \delta) \log_2 (P_2 + \delta) + P_1 \log_2 P_1 + P_2 \log_2 P_2 \\ &= -P_1 \log_2 \frac{P_1 - \delta}{P_1} - P_2 \log_2 \frac{P_2 + \delta}{P_2} + \delta \log_2 \frac{P_1 - \delta}{P_2 + \delta} \\ &> -P_1 \left(\frac{P_1 - \delta}{P_1} - 1 \right) \frac{1}{\ln 2} - P_2 \left(\frac{P_2 + \delta}{P_2} - 1 \right) \frac{1}{\ln 2} + \delta \log_2 \frac{P_1 - \delta}{P_2 + \delta}, \quad (\log x < \frac{x-1}{\ln 2}, x < 1) \\ &= \delta \log_2 \frac{P_1 - \delta}{P_2 + \delta} \geq \delta \log_2 1 = 0. \end{aligned}$$

6. Properties 2,3,5 suggest that H can be interpreted as measure of "uncertainty" or "randomness" inherent in X or P .
But its real quantitative significance comes from coding theorems, like the one stated previously that say that the least rate of any variable-length code is approximately H .

Huffman's Code Design Algorithm -- Huffman Codes

Given a finite set of probabilities $\{P_1, \dots, P_M\}$, Huffman's algorithm finds an optimal prefix code, i.e. one with smallest average length, i.e. one whose average length is R . The resulting codes are called Huffman codes.

There are usually many optimal codes. They can even have different sets of lengths.

See Section 3.1, 3.2 Sayood or Chap. 2 of Neuhoff and Forney for a description of Huffman's algorithm. The latter contains a proof that it generates optimal codes.

Other Code Designs

Fixed-length binary code:

For $n \in A = \{0, \dots, 2^m - 1\}$,

$b_m(n)$ = m bit binary representation of n,

$l(n) = m$

Codes for infinite alphabets:

$M = \infty$, $A = \{1, 2, 3, \dots\}$, $P_1 > P_2 > \dots$

Can't store an infinite number of codewords,
so need a systematic encoding rule.

January 10, 2007

Lossless-19

Examples of Systematic Codes for an Infinite Alphabet

	unary	gamma	delta	golomb b = 3	golomb b=4
1	0	0*	0	0 0	0 00
2	10	10 0	100 0	0 10	0 01
3	110	10 1	100 1	0 11	0 10
4	1110	110 00	101 00	10 0	0 11
5	11110	110 01	101 01	10 10	10 00
6	111110	110 10	101 10	10 11	10 01
7	1111110	110 11	101 11	110 0	10 10
8	11111110	1110 000	11000 000	110 10	10 11
9	111111110	1110 001	11000 001	110 11	110 00
10	1111111110	1110 010	11000 010	1110 0	110 01

* denotes special case

Reference: Witten, Moffat, Bell, *Managing Gigabytes*

January 10, 2007

Lossless-20

unary code: $u(n) = (1^{n-1} 0)$ (n-1 ones followed by one zero), $l(n) = n$

gamma code: $\gamma(n) = (u(m+1) b_m(r))$, $l(n) = m+1 + m = 2\lfloor \log_2 n \rfloor + 1$

where $m = \lfloor \log_2 n \rfloor$ and $r = n - 2^m$, equivalently, $n = 2^m + r$, $0 \leq r < 2^m$

Unary code specifies how many bits are used by binary code

Groups of 2^m integers have same prefix. This is similar to code used in JPEG, except that JPEG uses a better prefix than unary.

delta code: $\delta(n) = (\gamma(m+1) b_m(r))$, $l(n) = 2\lfloor \log_2 (\lfloor \log_2 n \rfloor + 1) \rfloor + 1 + \lfloor \log_2 n \rfloor$

where $m = \lfloor \log_2 n \rfloor$ and $r = n - 2^m$

Gamma code specifies how many bits are used by binary code.

Golomb code: Fix b , e.g. $b = 3$,

$G_b(n) = (u(m+1) b_{\log_b}^*(r))$, $l(n) = m+2 + \log_2 b$

where $m = \lfloor \frac{n-1}{b} \rfloor$ and $r = n - 1 - mb$, equivalently, $n - 1 = mb + r$, $0 \leq r < b$,

and $b_{\log_b}^*(r)$ is modified binary code, using $\lfloor \log_2 b \rfloor$ or $\lceil \log_2 b \rceil$ bits

Groups of b symbols have same prefix, followed by binary code.

Rice code: A Golomb code with b a power of 2.

Golomb, Rice are good for geometric (exponential) probabilities: $P_n = c Q^n$, $n=1,2, \dots$