

information theory; source coding

Linda Doyle

we want to encode a source

- so we want to take the output of a source (text/voice/image etc.) and encode it as a string of 1s and 0s
- we learned the last day that there are different kinds of sources, discrete memoryless source and sources which have memory

source coding

we are going to look how to code sources, mainly focusing on memoryless for the moment

source coding/compression is a very big area - should get sense of it all even if we study a small part of it.

LOSSLESS
COMPRESSOR

LOSSY
COMPRESSOR

lossless compressor

- A lossless compressor maps all files to different encodings; if it shortens some files, it necessarily makes others longer.
- We try to design the compressor so that the probability that a file is lengthened is very small and the probability that it is shortened is large.

lossless continued

- In lossless data compression, the integrity of the data is preserved.
- The original data and the data after compression and decompression are **exactly the same** because the compression and decompression algorithms are **exactly the inverse of each other**.

lossy compressor

- A lossy compressor compresses some files, but maps some files to the same encoding.
- The occurrence of one of these confusable files leads to a failure
- We'll denote by δ the probability that the source string is one of the confusable files, so a lossy compressor has a probability δ of failure.
- If δ can be made very small then a lossy compressor may be practically useful.

Why bother with lossy?

- Loss of information is acceptable in a picture or video.
- The reason is that our eyes and ears cannot distinguish subtle changes.
- Loss of information is not acceptable in a text file or a program file.
- For examples:
 - Joint photographic experts group (JPEG)
 - Motion picture experts group (MPEG)



100%



50%



10%



5%

Both types exploit redundancy

Lossless - Lossless techniques enable exact reconstruction of the original document from the compressed information

Exploits redundancy in data

Lossy compression - reduces a file by permanently eliminating certain redundant information

Exploits redundancy and human perception

LOSSY USUALLY ACHIEVE HIGHER COMPRESSION RATES

let's formally define the generation of codewords

Definition A source code C for a random variable X is a mapping from \mathcal{X} , the range of X , to \mathcal{D}^* , the set of finite-length strings of symbols from a D -ary alphabet. Let $C(x)$ denote the codeword corresponding to x and let $l(x)$ denote the length of $C(x)$.

For example, $C(\text{red}) = 00$, $C(\text{blue}) = 11$ is a source code for $\mathcal{X} = \{\text{red, blue}\}$ with alphabet $\mathcal{D} = \{0, 1\}$.

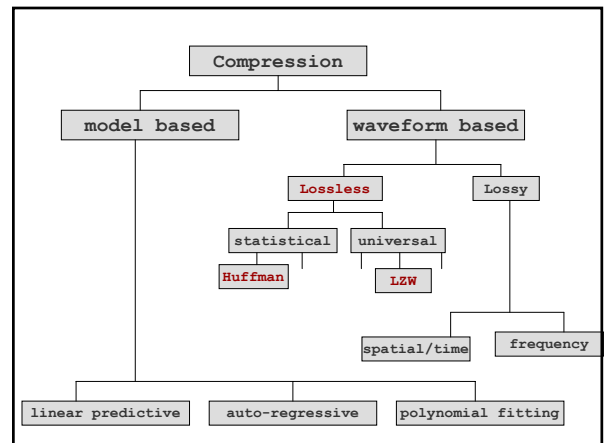
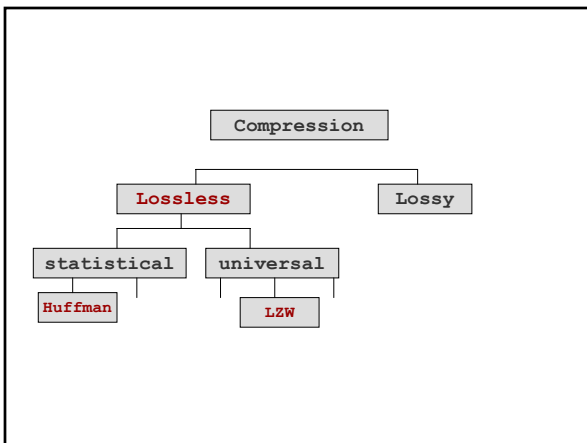
$C(\text{red})$ and $C(\text{blue})$ both have length 2

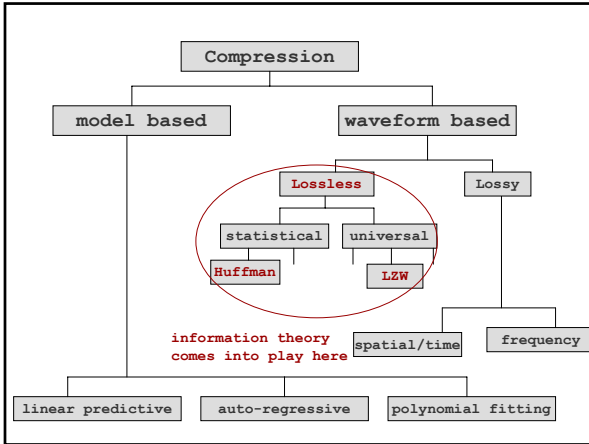
static and dynamic lossless compressors

- In a *static* method the mapping from the set of messages to the set of codewords is fixed before transmission begins, so that a given message is represented by the same codeword every time it appears in the message being encoded.
- **Static coding** requires two passes: one pass to compute probabilities (or frequencies) and determine the mapping, and a second pass to encode.
- Examples: **Static Huffman Coding**

static and dynamic lossless compressors

- In an *adaptive* method the mapping from the set of messages to the set of codewords changes over time.
- All of the adaptive methods are *one-pass* methods; only one scan of the message is required.
- Examples: **LZW plus others (can get adaptive forms of Huffman)**





we want encoding rules

1. to encode a memoryless source
2. so that the source sequence can be retrieved from the encoded sequence, at least with a high probability
3. so that the number of code letters is as small as possible

so what we want to do now

- is finding ways of determining what those codes are
- we need to look a little more into different types of codes

expected code length

Definition The *expected length* $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by

$$L(C) = \sum_{x \in X} p(x)l(x), \quad (5.1)$$

where $l(x)$ is the length of the codeword associated with x .

Without loss of generality, we can assume that the D -ary alphabet is $D = \{0, 1, \dots, D - 1\}$.

example 1

Example 5.1.1 Let X be a random variable with the following distribution and codeword assignment:

$$\begin{aligned} \Pr(X = 1) &= \frac{1}{2}, & \text{codeword } C(1) &= 0 \\ \Pr(X = 2) &= \frac{1}{4}, & \text{codeword } C(2) &= 10 \\ \Pr(X = 3) &= \frac{1}{8}, & \text{codeword } C(3) &= 110 \\ \Pr(X = 4) &= \frac{1}{8}, & \text{codeword } C(4) &= 111. \end{aligned} \quad (5.2)$$

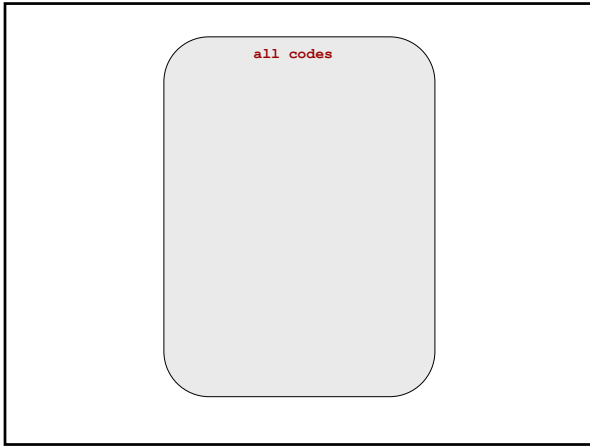
The entropy $H(X)$ of X is 1.75 bits, and the expected length $L(C) = E[l(X)]$ of this code is also 1.75 bits. Here we have a code that has the same average length as the entropy. We note that any sequence of bits can be uniquely decoded into a sequence of symbols of X . For example, the bit string 0110111100110 is decoded as 134213.

example 2

Example 5.1.2 Consider another simple example of a code for a random variable:

$$\begin{aligned} \Pr(X = 1) &= \frac{1}{3}, & \text{codeword } C(1) &= 0 \\ \Pr(X = 2) &= \frac{1}{3}, & \text{codeword } C(2) &= 10 \\ \Pr(X = 3) &= \frac{1}{3}, & \text{codeword } C(3) &= 11. \end{aligned} \quad (5.3)$$

Just as in Example 5.1.1, the code is uniquely decodable. However, in this case the entropy is $\log 3 = 1.58$ bits and the average length of the encoding is 1.66 bits. Here $E[l(X)] > H(X)$.

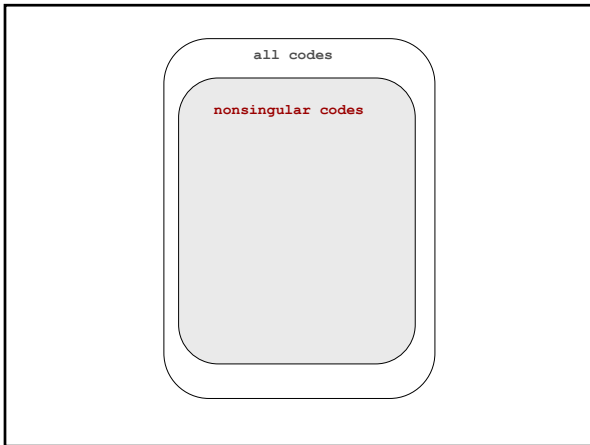


examples

TABLE 5.1 Classes of Codes

X	Singular
1	0
2	0
3	0
4	0

all have same code



nonsingular codes

Definition A code is said to be *nonsingular* if every element of the range of X maps into a different string in \mathcal{D}^* ; that is,

$$x \neq x' \Rightarrow C(x) \neq C(x'). \quad (5.4)$$

nonsingular codes

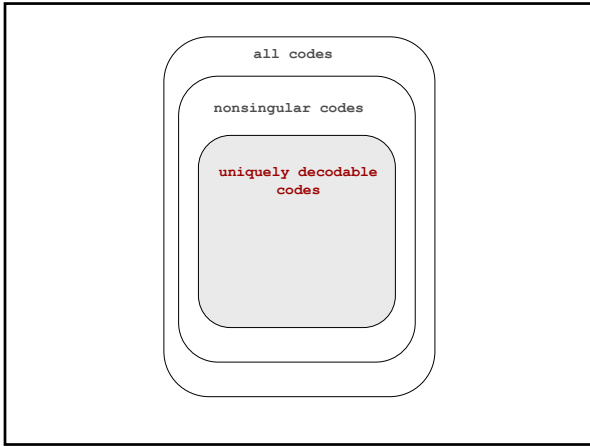
- Nonsingularity suffices for an unambiguous description of a single value of X .
- But we usually wish to send a sequence of values of X .
- Nonsingular codes can then get confusing when everything runs together

examples

TABLE 5.1 Classes of Codes

X	Singular	Non-singular, but Not Uniquely Decodable
1	0	0
2	0	010
3	0	01
4	0	10

you can mix things up here
for example 010 can be 2, 14 or 31



extension of a code

Definition The *extension* C^* of a code C is the mapping from finite-length strings of \mathcal{X} to finite-length strings of \mathcal{D} , defined by

$$C(x_1x_2 \cdots x_n) = C(x_1)C(x_2) \cdots C(x_n), \quad (5.5)$$

where $C(x_1)C(x_2) \cdots C(x_n)$ indicates concatenation of the corresponding codewords.

Example 5.1.4 If $C(x_1) = 00$ and $C(x_2) = 11$, then $C(x_1x_2) = 0011$.

a code is uniquely decodable if its extension is nonsingular!

uniquely decodable codes

- so in this case there is no mistaking a code as being belonging to a different random variable
- i.e. the extended version of the code is not the same as anything else

uniquely decodable codes

- they obviously offer an improvement
- BUT you typically have to look forward through the coded sequence to figure out how to decode - it is not instantaneous

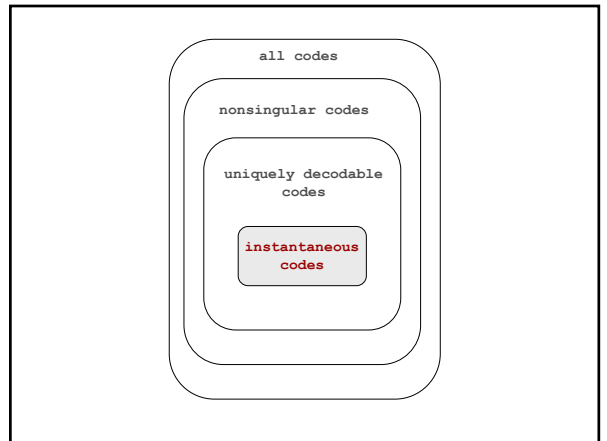
examples

TABLE 5.1 Classes of Codes

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous
1	0	0	10
2	0	010	00
3	0	01	11
4	0	10	110

If the first two bits are 00 or 10, they can be decoded immediately.

If the first two bits are 11 you have to look to see if it is followed by a 0 or a 00 or a 1 to determine that it is 3 for example



instantaneous code

Definition A code is called a *prefix code* or an *instantaneous code* if no codeword is a prefix of any other codeword.

some comments

- An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable.
- Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it. We need not wait to see the codewords that come later.
- An instantaneous code is a *selfpunctuating code*; we can look down the sequence of code symbols and add the commas to separate the codewords without looking at later symbols.

examples

TABLE 5.1 Classes of Codes

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous	Instantaneous
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

no need to look ahead – it just drops out! no need for prefixes

So

- instantaneous or prefix codes seem like the best
- so how do we find them easily?

Finding an instantaneous code

- We wish to construct instantaneous codes of minimum expected length to describe a given source. It is clear that we cannot assign short codewords to all source symbols and still be instantaneous.
- The set of codeword lengths possible for instantaneous codes is limited by what is known as the **Kraft inequality**.

Kraft Inequality

Theorem 5.2.1 (Kraft inequality) For any instantaneous code (prefix code) over an alphabet of size D , the codeword lengths l_1, l_2, \dots, l_m must satisfy the inequality

$$\sum_i D^{-l_i} \leq 1. \quad (5.6)$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

example

C_3 :

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	1/2	1.0	1
b	10	1/4	2.0	2
c	110	1/8	3.0	3
d	111	1/8	3.0	3

- this code satisfies the Kraft inequality
- code lengths are 1,2,3 and 3 and substitution in the previous equation shows this to be true

Note

- the theorem does not say that any codes whose lengths satisfies the Kraft inequality is a prefix code - e.g. look at the third set of codes on our table - other example 0,00 and 11
- It says that some instantaneous code exists with those lengths - e.g. fourth set of codes on table or 0,10,11

TABLE 5.1 Classes of Codes

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous	
			Instantaneous	Instantaneous
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

a more graphical way of thinking

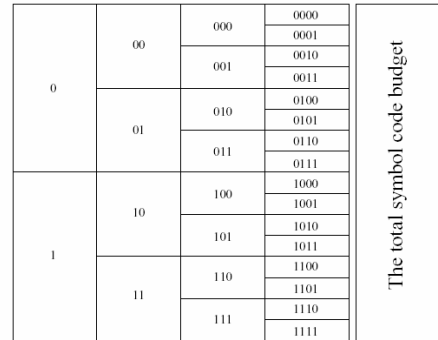
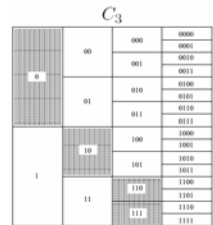


Figure 5.1. The symbol coding budget. The 'cost' 2^{-l} of each codeword (with length l) is indicated by the size of the box it is written in. The total budget available when making a uniquely decodable code is 1. You can think of this diagram as showing a *codeword supermarket*, with the codewords arranged in aisles by their length, and the cost of each codeword indicated by the size of its box on the shelf. If the cost of the codewords that you take exceeds the budget then your code will not be uniquely decodable.

Code Word Budget is taken from David McKay - it is a useful way of thinking

C_3 :

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	1/2	1.0	1
b	10	1/4	2.0	2
c	110	1/8	3.0	3
d	111	1/8	3.0	3



From visual inspection it is clear that this code does not exceed the budget and hence can be uniquely decoded!

C_6 :

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	1/2	1.0	1
b	01	1/4	2.0	2
c	011	1/8	3.0	3
d	111	1/8	3.0	3

C_6

From visual inspection it is clear that this code does not exceed the budget and hence can be uniquely decoded!

FIGURE 5.2. Code tree for the Kraft inequality.

The instantaneous condition on the codewords implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords.

C_3 :

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	1/2	1.0	1
b	10	1/4	2.0	2
c	110	1/8	3.0	3
d	111	1/8	3.0	3

C_3

From visual inspection it is clear that this code is instantaneous!

C_6 :

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	1/2	1.0	1
b	01	1/4	2.0	2
c	011	1/8	3.0	3
d	111	1/8	3.0	3

C_6

from visual inspection this is not!

so

- If I have a bunch of codes I can check if they are instantaneous.

in class exercise

Create three different sets of codewords to represent A,B,C,D,E and F

1. one set should be nonsingular but not uniquely decodable
2. one set should be uniquely decodable but not instantaneous
3. one set should be instantaneous