

Introduction to Data Compression

- Introduction
 - terminology, information theory, codes
- Coding
 - Huffman, arithmetic
- Modeling
 - dictionary, context
- Text Compr'n Systems
 - performance
- Image Compression
 - lossless techniques
- Lossy Compression
 - quantization, coding
- Lossy Image Compr'n
 - JPEG, MPEG
- Audio Compression
 - coding, masking

Introduction

- Terminology
- Performance Measures
- Information Theory
- Codes

Terminology

- **Data Compression**

transforms data to minimize size of its representation

In contrast: **Data Reliability**

is often implemented by adding check and parity bits,
and increases redundancy and size of the data

- **Motivation**

- increase capacity of storage media
- increase communication channel capacity
- achieve faster access to data

Applications

- **Examples**
 - file compression – *Gzip* (Unix), *Compressor* (Mac), *PKZIP* (PC)
 - automatic compression (disk doubler) – *Stacker*
 - facsimile transmission
 - modem compression
 - CD/DVD players
 - WWW – Flash
 - digital camera image storage – JPEG
 - HDTV
 - Teleconferencing

Applications

- **Typical source formats** (format depends on the application)
 - **text** – ASCII or EBCDIC chars (8 bits)
 - **audio** – real-valued fixed precision samples
 - **images** – pixels (picture elements)
 - 1 bit b/w
 - 8 bits grey scale
 - 24 bits color (3 primaries)
 - **video example** – an HDTV video format
 - $1920 \text{ pixels} \times 1080 \text{ lines} \times 30 \text{ fps} \times 8 \text{ bits/color} \times 3 \text{ colors} \rightarrow 1.5 \text{ Gbps}$
 - only $\sim 18 \text{ Mbps}$ available for video
 - 6 MHz/channel supports only 19.2 Mbps,
need some capacity for audio, captioning
 - requires 83:1 video compression

Applications

- What can we expect? Depends on file type, size etc.
 - **Text compression** (Calgary corpus) Pent-200, Win98

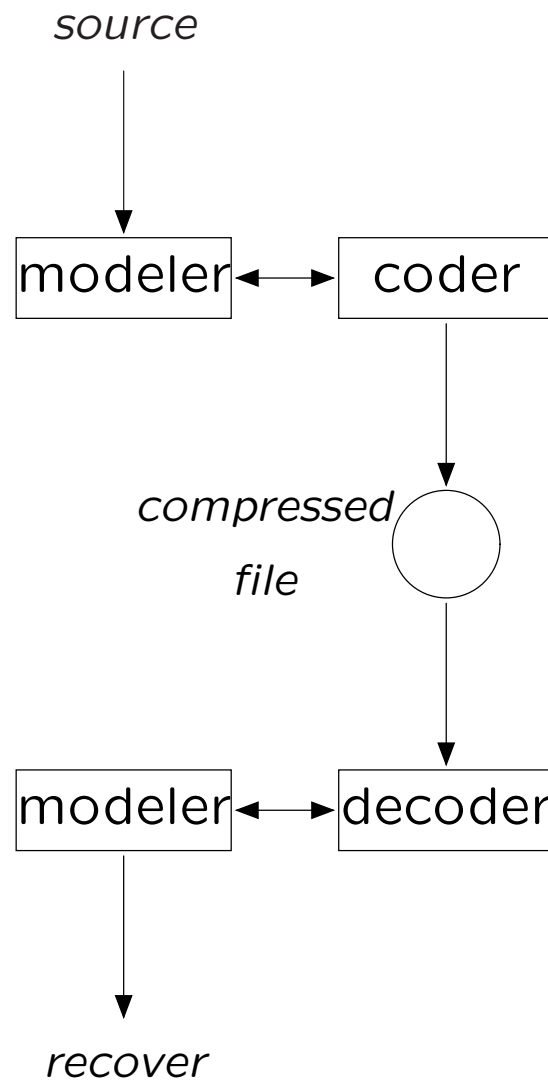
RK	1.81 bpc	29 kcps
LZOP	3.90 bpc	2.98 Mcps
 - **Image compression**

Lossless b/w JBIG	CR = 5%
Lossless grey scale JPEG	5 bpp (CR=65%)
Lossy color JPEG	0.6 bpp (CR=7%)
 - **Sound compression** mp3
CD rates 1.5 Mbps, reduced to 128 Kbps: CR \approx 8%
 - **Video compression** MPEG-1
352 \times 240 color images @ 30 fps = 60 Mbps
reduced to 1.2 Mbps: CR \approx 2%

Terminology

- *Encoding*: compressing, reduce representation
Decoding: recover the original data
- *Lossless*: recover precisely the original data
Lossy: original data not recovered exactly
- *2-pass*: pre-scan of source required
1-pass: no pre-scan required
- Modeling and Coding – components of the compression process
 - **Modeling**: describe form of redundancy
 - build abstract prototype of the source
 - select source elements for focus
 - **Coding**: encode model and description
of how data differs from model, residual
 - construct new representation using model
 - map source elements to produce output

Compression-Decompression Process



Types of Source Element Redundancy

- **distribution**
some elements occur more often than others,
e.g., ‘;’ in C programs
- **repetition**
elements often repeated,
e.g., 1 or 0 in b/w image bit maps
- **patterns**
correlation of symbols occurrence,
e.g., “th, qu” in English
- **positional**
some symbols occur mostly in the same relative position,
e.g., database fields

Methods for Compression

- **pre-filtering** — reduce complexity of data
may remove relevant details
- **eliminate redundancy** — remove any repeated information
- **use human perception models** — remove irrelevant detail
in ways that minimize humans' ability to detect the information loss
- **post-filtering**
attempt to further reduce/mask artifacts
that were introduced by information loss

Performance Measures

Which measures are important depends on the application

- **systemic encode/decode constraints**
 - limited CPU power
 - limited working memory
 - incremental encoding
 - real-time transmittal (or multiple pass permitted)
 - real-time decode
 - random access decode enabled
- speed – chars/sec or pixels/sec
 - **Symmetric** – encode + decode once
videoconferencing, multimedia mail
 - **Asymmetric** – slow encode + fast multiple decode
picture archive, video-on-demand, electronic publishing
 - **Asymmetric** – fast encode + slow rare decode
file system backup, security video tapes

Performance Measures

- **compression effectiveness** (size reduction)
 - compression ratio $CR = \text{new file size as \% of orig size}$
 - compression factor $CF = \text{orig file size} / \text{new file size}$
 - percent savings $PS = \text{amt of reduction as \% of orig size}$
 - bit usage bpc (# bits/char), bpp (# bits/pixel)

- **quality**
 - fidelity – lossless, perceptually lossless, lossy
 - **fidelity criteria**
 - MSE (mean squared error)
 - SNR (signal-to-noise ratio)
 - perceptual quality
 - allow graceful (lossy) degradation
 - allow browsing – rapid recovery of degraded version
 - delay
 - minimize error propagation

Information Theory – Information Content

- more likely events give less information
(learn more from surprising events)
so, measure of information content is inversely related to probability
 - n events equally likely \Rightarrow
to represent each item requires $\log_2 n$ bits
- Information Content of an event having probability p
is $\log(1/p) = -\log p$
 - base 2 logarithm \rightarrow bits
 - base e nats, base 3 trits, base 10 hartleys
- a sequence of independent events has additive information content

Information Theory – Entropy

Shannon Entropy: a discrete memoryless source that emits n chars with probabilities p_1, \dots, p_n has entropy $H = \sum[-p_i \lg p_i]$

- entropy measures the **avg # of bits needed** to encode the output of a source of a random sequence
 - no compression scheme can do better
 - compression methods that approach this limit without knowledge of the pdf are called **universal**
- if sequence el'ts are not indep & ident distr (*iid*) then above formula gives the **first-order entropy**
- in physics, entropy measures **disorder**
 - if all n items equally likely, $H = \lg n$
 - if only 1 item can occur, $H = 0$
- entropy can be thought of as a measure of **uncertainty** as to which character is emitted next

Information Theory – Examples

Example: fair coin

$$\text{Prob}(H) = \text{Prob}(T) = 1/2$$

$$i(H) = i(T) = 1 \text{ bit}$$

$$H = 1$$

Example: biased coin

$$\text{Prob}(H) = 1/8 \quad \text{Prob}(T) = 7/8$$

$$i(H) = 3 \text{ bits} \quad i(T) = 0.193 \text{ bits}$$

$$H = .375 + .169 = 0.544$$

Example: fair and biased dice

- $\text{Prob}(i) = 1/6, (i = 1 \dots 6)$

$$H = 2.585$$

- $\text{Prob}(1) = 3/8, \text{Prob}(i) = 1/8, (i = 2 \dots 6)$

$$H = 2.406$$

- $\text{Prob}(1) = 11/16, \text{Prob}(i) = 1/16, (i = 2 \dots 6)$

$$H = 1.622$$

Information Theory

- **Joint Entropy** of variables X, Y with joint pdf p

$$H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y)$$

- **Conditional Entropy**

$$H(Y|X) = - \sum_{x \in X} p(x) H(Y|X = x) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(y|x)$$

- $H(Y|X) \leq H(Y)$
- knowing something about the context
can reduce the uncertainty (and the entropy)

- chain rule: $H(X, Y) = H(X) + H(Y|X)$

Proof: $p(x, y) = p(x) * p(y|x)$

take logs: $\log p(x, y) = \log p(x) + \log p(y|x)$

take expectations:

$$\sum_{x, y} p(x, y) \log p(x, y) = \sum_x p(x) \log p(x) + \sum_{x, y} p(x, y) \log p(y|x)$$

Information Theory

- **Relative Entropy** between two pdf's, p and q

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

- always non-negative
 - zero only if $p = q$, can be infinite
 - not symmetric and does not satisfy triangle inequality
- **Mutual Information** is the relative entropy between the joint pdf and the product of pdf's

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

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

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

Information Theory – English

- information content of English
 - if 96 printable chars equally likely entropy = $\lg 96 = 6.6$ bpc
need $\lceil \lg 96 \rceil = 7$ bpc
 - using pdf of English text entropy ≈ 4.5 bpc
Huffman code for this pdf = 4.7^+ bpc
 - group text in 8-char blocks: entropy ≈ 2.4 bpc
estimate limit for larger-size blocks = 1.3 bpc
- historical performance of best general-purpose compressors on Calgary corpus

<i>Year</i>	<i>bpc</i>	<i>algorithm</i>
1977	3.94	LZ77
1984	3.32	LZMW
1987	2.71	Gzip
1988	2.48	PPMC
1994	2.33	PPMD
1995	2.29	BWT
1997	1.99	BOA
1999	1.82	RK

Codes

- Types of codes
- Fixed finite codes
- Fixed infinite codes

Types of Codes

classified by **time variance** of codewords

A **code** is a mapping

from source = string over alphabet S

to compressor output = stream of codewords over alphabet C

- **Fixed** code
 - codeword set is time invariant
 - selection is predetermined
- **Static** code
 - codeword set is time invariant
 - selection dictated by model
- **Adaptive** code
 - dynamic codeword set (varies with time)
 - selection/modification dictated by model

Types of Codes

classified by **input-output rates**

(for time-invariant codes, think of parse+codeword lengths)

- **Fixed-to-Fixed** rate code: $S \rightarrow C$
ASCII code
- **Fixed-to-Variable** rate code: $S \rightarrow C^+$
Morse, Huffman codes
- **Variable-to-Fixed** rate code: $S^+ \rightarrow C$
also called *free-parse* methods
Tunstall code, Lempel-Ziv methods
- **Variable-to-Variable** rate code: $S^+ \rightarrow C^+$
Runlength encoding, Arithmetic coding

Types of Codes

classified by **decodability**

- **Ambiguous Code** \exists 2 strings with identical encoding

<i>symbol</i>	a	b	c	d
<i>code</i>	1	01	010	001

- dilemma: 01001 could be bd or cb

- **Complete** every semi-infinite string is decodable

- **Uniquely Decodable** unambiguous and complete

- **Instantaneous Prefix Code** can be decoded as codewords are received
no codeword is a prefix of any other

- C is a prefix code $\Leftrightarrow C$ is instantaneous

Example:

<i>symbol</i>	a	b
<i>code</i>	0	01

- **not** complete, as 110... is undecodable
- unambiguous but not instantaneous

Fixed Codes

- **Advantages** – agreed upon beforehand
 - no need for encoder to transmit code
 - faster because selecting, not computing code
- **Disadvantages** – does not exploit info from model
 - code unrelated to particular text
 - can't remove inherent data redundancy
- **One fixed length** provides no compression
Example: ASCII, EBCDIC
 - modeler does all the work, packing information into 'events'
- **Multiple fixed-length** (e.g., 6 and 12) gain compression by using
 - shorter codes for expected freq chars
 - longer codes for infrequent chars

Example: Morse

Fixed Finite Codes – Enumerative Coding

- for a **known finite set** S of n elements (chars)
map S to the set $\{0, 1, \dots, n - 1\}$ (can refer to elements via indices)
 - problem is reduced to **representing integers** $0, 1, 2, \dots, n - 1$
which normally requires $\lceil \lg n \rceil$ bits
- **to decode** requires inverse map, can implement by
 - maintaining a table, or
 - use algorithm to compute maps
- no compression if all codewords are same length
var-length code gains some compression

Enumerative Coding – Example

$S = \{ \text{length-}m \text{ binary strings having exactly } j \text{ ones} \}$

- $n = |S| = \binom{m}{j} = \frac{m!}{j!(m-j)!}$
- can **compute index** $I \in [0, n - 1]$ for $X = x_{m-1} \dots x_1 x_0$
let positions $\{p_i\}$ in X that contain ones be

$$m - 1 \geq p_1 > p_2 > \dots > p_j \geq 0$$

$$\text{index } I = \binom{p_1}{j} + \binom{p_2}{j-1} + \binom{p_3}{j-2} + \dots + \binom{p_j}{1}$$

- can **compute inverse map**: index $I \rightarrow$ string

$$r \leftarrow I$$

for $i \leftarrow 1$ **to** j

$$p_i \leftarrow \max t \text{ s.t. } \binom{t}{j+1-i} \leq r$$

$$r \leftarrow r - \binom{p_i}{j+1-i}$$

Enumerative Coding – Example

Example: $m = 6, j = 4 \rightarrow n = 15$

- compute **index**(110110)
 - 1-bits in positions 5,4,2,1
 - $\text{index} = \binom{5}{4} + \binom{4}{3} + \binom{2}{2} + \binom{1}{1} = 11$
- compute **inverse** of index 9
 - maximize $\binom{t}{4} \leq 9 \rightarrow p_1 = 5$
 - maximize $\binom{t}{3} \leq 9 - \binom{5}{4} = 4 \rightarrow p_2 = 4$
 - maximize $\binom{t}{2} \leq 4 - \binom{4}{3} = 0 \rightarrow p_3 = 1$
 - maximize $\binom{t}{1} \leq 0 - \binom{1}{2} = 0 \rightarrow p_4 = 0$

⇒ sequence 110011

Fixed Finite Codes – Phasing-In

To represent n codes normally need $B = \lceil \lg n \rceil$ bits

If n not power of 2, can **sometimes** use $B - 1$ bits

- $i < 2^B - n \Rightarrow$ encode i ($B - 1$ bits)
- $i \geq 2^B - n \Rightarrow$ encode $i + 2^B - n$ (B bits)
- save some space
increase encode/decode time

Example: $n = 5 \Rightarrow B = 3$

i	code
0	00
1	01
2	10
3	110
4	111

Fixed Finite Codes – Start-Step-Stop Codes

family based on choice of 3 parameters

- k codeword sets, $k = (stop - start) / step + 1$
- set n has codewords = 111...1 0 xxx...x
prefix: $(n - 1)$ 1's, one 0 (omit when $n = k$)
suffix: $start + (n - 1) * step$ bits

Example: $start = 3, step = 2, stop = 9$

$n=1$	0xxx	{0000, 0001, 0010, 0011, 0100, ... 0111}
$n=2$	10xxxxx	32 codewords
$n=3$	110xxxxxxxx	128 codewords
$n=4$	111xxxxxxxxxx	512 codewords, 680 in all

- **instantaneously decodable:**
read n 1's until either 0 encountered
or $n = (stop - start) / step$
read $start + (n - 1) * step$ more bits, build #

Example: $start = 3, step = 2, stop = 9$

1	1	0	1	1	0	1	0	1	1
8	+32	(n=3)+64	+32		+8		+2	+1	

Fixed Infinite Codes

- for a set of unknown or increasing size
encode 1,2,3,...
with codewords of increasing length
- popular codes
 - Elias
 - Even-Rodeh
 - Zeckendorf (Fibonacci)
 - Golomb and Rice
 - variable-byte

Elias Gamma Code

- instantaneously decodable:
 - read n 0-bits until encounter 1 bit (starts X)
 - read n more bits, computing binary value X

Example:

0	0	0	1	0	0	1
1	2	3= n	8	8	8	9

- to encode j : $\lfloor \lg j \rfloor$ 0's followed by binary value of j
- length of encoding(j) = $2\lfloor \lg j \rfloor + 1$

<i>Gamma Code</i>	<i>Integer</i>	<i>Bits</i>
1	1	1
01x	2 - 3	3
001xx	4 - 7	5
0001xxx	8 - 15	7
00001xxxx	16 - 31	9
000001xxxxx	32 - 63	11

Elias Delta Code

- instantaneously decodable:

read an Elias Gamma Code number V

read $V - 1$ more bits, computing binary value W

$$X = 2^{V-1} + W$$

Example: 0 0 1 0 0 0 0 1
 1 2=n 4 4 4=V 8 8 9

- to encode j : $\lfloor \lg(1 + \lg j) \rfloor$ 0's
 then binary value of $1 + \lfloor \lg j \rfloor$
 then binary value of $j - 2^{\lfloor \lg j \rfloor}$, using $\lfloor \lg j \rfloor - 1$ bits
- length of encoding of $j \approx \lg j + 2 \lg \lg j$

<i>Delta Code</i>	<i>Integer</i>	<i>Bits</i>
1	1	1
010x	2- 3	4
011xx	4- 7	5
00100xxx	8-15	8
00101xxxx	16-31	9
00110xxxxx	32-63	10

Elias Omega Code

- instantaneously decodable:

$j \leftarrow 0$

while peek(next bit)=1 **do**

$j \leftarrow j + 1$

$j \leftarrow$ compute value of next j bits

- to encode j : write 0
while $j \geq 2$
 prepend binary value of j
 $j \leftarrow \lfloor \lg j \rfloor$

- each group encodes $len(\text{next group}) - 1$
the first group has length 2

<i>Omega Code</i>	<i>Integer</i>	<i>Bits</i>
0	1	1
1x0	2- 3	3
101xx0	4- 7	6
111xxx0	8-15	7
101001xxxx0	16-31	11
101011xxxxx0	32-63	12

Even-Rodeh Code

- similar to Elias Omega
- to encode $j \leq 3$: express as 3 bits (has leading 0)
- to encode $j \geq 4$: write 0
 - while $j \geq 8$
 - prepend binary value of j
 - $j \leftarrow \lfloor \lg j \rfloor + 1$
- each group encodes $len(\text{next group})$
the first group has length 3

<i>Even-Rodeh</i>	<i>Integer</i>	<i>Bits</i>
0xx	0- 3	3
1xx0	4- 7	4
1001xxx0	8-15	8
1011xxxx0	16-31	9
1101xxxxx0	32-63	10

- compare

<i>value</i>	1	2	4	8	16	32	64	128	256
Elias ω	1	3	6	7	11	12	13	14	21
Even-Rodeh	3	3	4	8	9	10	11	17	18

Zeckendorf (Fibonacci) Code

- to encode j :
 - express j as sum of Fibonacci #'s
 - represent as bit vector (1,2,3,5,8,13,...)
 - forbidden to have two adjacent 1's
 - append a 1-bit
- instantaneously decodable:
 - 11 delimits end of codeword
 - while parsing, build value based on Fib #'s

Example: 0 1 0 1 0 0 1 1
 1 2 3 5 8 13 21 end

- length of encoding of $j = \lfloor \log_{\phi} j \rfloor + 1$
- Advantages
 - for most pdf's, Fib outperforms Elias
 - robust: 1-bit errors disturb ≤ 3 chars

Compare Elias & Fibonacci Code Lengths

<i>Integer</i>	<i>Gamma</i>	<i>Delta</i>	<i>Fibonacci</i>
1	1	1	2
2	3	4	3
3	3	4	4
4	5	5	4
5-7	5	5	5
8-15	7	8	6-7
16-31	9	9	7-8
32-63	11	10	8-10
64-88	13	11	10
100	13	11	11
1000	19	16	16
10^4	27	20	20
10^5	33	25	25
10^6	39	28	30

Asymptotically, Elias Delta uses fewer bits but, if most numbers are small (< 7) and others not too large then Elias Gamma is best

Golomb Codes

- family of codes, with one parameter (m)
 - start with code for 0
 - designed for coding asymmetric binary events with probs $p \gg (1 - p)$ by encoding runlengths of probable event
 - $p^m = 0.5 \rightarrow \Pr(\text{len } n + m) = \frac{1}{2} \Pr(\text{len } n)$
so codeword($n + m$) should have one more bit than for n
- to encode j
 - transmit unary value of $\lfloor j/m \rfloor$
uses $\lfloor j/m \rfloor$ 1-bits and one 0-bit
 - transmit phased-in binary code for $j \bmod m$
uses $\lfloor \lg m \rfloor$ or $\lceil \lg m \rceil$ bits

Golomb Codes

<i>Integer</i>	$m = 3$	$m = 4$	$m = 5$	$m = 6$
0	00	000	000	000
1	010	001	001	001
2	011	010	010	0100
3	100	011	0110	0101
4	1010	1000	0111	0110
5	1011	1001	1000	0111
6	1100	1010	1001	1000
7	11010	1011	1010	1001
8	11011	11000	10110	10100

Rice Codes

- family of codes, with one parameter (k)
special case of Golomb codes, $m = 2^k$
- to encode j transmit $\lfloor j/2^k \rfloor$ 1-bits, then one 0-bit
transmit k least significant bits of j
- length of encoding of $j = \lfloor j/2^k \rfloor + k + 1$

k = 2:	0xx	0-3	k = 3:	0xxx	0-7
	10xx	4-7		10xxx	8-15
	110xx	8-11		110xxx	16-31
	1111111001	33		11110001	33
- instantaneously decodable
position of 0 gives value prefix
append next k bits

Variable-Byte Code

- simple binary, using minimum required number of bytes
- each byte: value (7 bits), is this last byte? (1 bit)

<i>Integer</i>	<i>Bits</i>
0 - 127	8
128 - 16383	16
16384 - 2097151	24

- integral byte sizes for easy coding
- effective for medium-size numbers
- wasteful for small numbers

Comparing Fixed Codes

