Information Theory and Networks

Lecture 7: Communications and storage – simple coding

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

September 18, 2013

Part I

Communications and storage – simple coding

Part I

Communications and storage – simple coding

Motivating problem

> Work expands so as to fill the time available for its completion.
> *Parkinson's law*

> Data expands to fill the space available for storage.
> *Parkinson's law of disk space*

Compression

We want to “compress”

- Text
- Audio
- Pictures
- Video
- ...

Often we want lossless compression (i.e., data can be decompressed with no loss).

Compression
We want to “compress”
- Text
- Audio
- Pictures
- Video
- ...
Often we want lossless compression (i.e., data can be decompressed with no loss).

Many modern compression algorithms are lossy. They allow loss of “perceptually irrelevant data”, i.e., data that doesn’t affect our perception (hopefully) of the media. Examples include:

- JPEG image compression
- MP3 Audio compression
- Various video codecs

On the first day of his class in Information Theory, a new student was confused by the Professor, who would call out a number, thence followed by enraptured laughter.
The student asked what was going on, and eventually was told “We’ve assigned each joke a number, and so instead of wasting time telling the joke, we just give the number.”
The student, catching on, called out 42, but received only a polite smile from a few of the others. He was again confused, and asked why they didn’t laugh. One of the others said, with embarrassment, “Its all in the way you tell it”.

On the first day of his class in Information Theory, a new student was confused by the Professor, who would call out a number, thence followed by enraptured laughter.
The student asked what was going on, and eventually was told “We’ve assigned each joke a number, and so instead of wasting time telling the joke, we just give the number.”
The student, catching on, called out 42, but received only a polite smile from a few of the others. He was again confused, and asked why they didn’t laugh. One of the others said, with embarrassment, “Its all in the way you tell it”.

Stupid compression

- Give each document a (short) number
  - to decompress, just give the document corresponding to the number back
- Has some problems
  - doesn't generalise (only works for pre-described documents)
  - algorithm effectively stores the content

Stupid compression
- Give each document a (short) number
  - to decompress, just give the document corresponding to the number back
- Has some problems
  - doesn't generalise (only works for pre-described documents)
  - algorithm effectively stores the content

Universal compression

Ideally, we would have a universal compression algorithm such that

$$\frac{\text{bytes after compression}}{\text{bytes before compression}} \leq \alpha$$

for some $0 < \alpha < 1$ (and as small as possible) for every possible file.

Universal compression
Ideally, we would have a universal compression algorithm such that
$$\frac{\text{bytes after compression}}{\text{bytes before compression}} \leq \alpha$$
for some $0 < \alpha < 1$ (and as small as possible) for every possible file.

Universal compression

Theorem

There is no lossless compression algorithm that strictly reduces every file.

Proof.

Consider there are 2^N files with N bits.
There are $2^0 + 2^1 + \cdots + 2^{N-1} = 2^N - 1$ files with less than N bits.
By the pigeonhole principle we don't have enough shorter files to represent all the files of length N, so it isn't possible to compress all of them using the one algorithm. □

Universal compression
Theorem
There is no lossless compression algorithm that strictly reduces every file.
Proof.
Consider there are 2^N files with N bits.
There are $2^0 + 2^1 + \cdots + 2^{N-1} = 2^N - 1$ files with less than N bits.
By the pigeonhole principle we don't have enough shorter files to represent all the files of length N, so it isn't possible to compress all of them using the one algorithm.

Universal compression

Theorem

There is no lossless compression algorithm that strictly reduces every file.

What about common cases?

- gzip, bzip, zip, ...
- maybe they don't work on all files
- but work pretty well most of the time

They exploit the structure of typical text/image/...

Universal compression
Theorem
There is no lossless compression algorithm that strictly reduces every file.
What about common cases?
gzip, bzip, zip, ...
maybe they don't work on all files
but work pretty well most of the time
They exploit the structure of typical text/image/...

Text Compression

- Language is made up of words
  - words are repeated
  - words have different frequencies
- Top ten (in TV): you, i, I, to, the, a, and, that, it, of, me
- Varies by *corpus*
  - genre
  - writer
- We could give a shorter code to common words (or patterns)

Text Compression
- Language is made up of words
  - words are repeated
  - words have different frequencies
- Top ten (in TV): you, i, I, to, the, a, and, that, it, of, me
- Varies by corpus
  - genre
  - writer
- We could give a shorter code to common words (or patterns)

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists
http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/TV/2006/1-1000

Simple problem

- We have a "text" made up of a series of messages, or symbols

$$a, b, c, d$$

- We know the PMF of the messages

$$P(a), P(b), P(c), P(d)$$

- How could we code the message to compress it?
  - lets write our code in binary

Simple problem
- We have a "text" made up of a series of messages, or symbols
a, b, c, d
- We know the PMF of the messages
$P(a), P(b), P(c), P(d)$
- How could we code the message to compress it?
  - lets write our code in binary

Example

Code

$$\begin{aligned} a &\leftrightarrow 00 \\ b &\leftrightarrow 01 \\ c &\leftrightarrow 10 \\ d &\leftrightarrow 11 \end{aligned}$$

Average number of bits per word is 2
Can you do better if you know the PMF?

Example
Code
$a \leftrightarrow 00$
$b \leftrightarrow 01$
$c \leftrightarrow 10$
$d \leftrightarrow 11$
Average number of bits per word is 2
Can you do better if you know the PMF?

Example

$$\begin{aligned} P(a) &= 1/2 \\ P(b) &= 1/4 \\ P(c) &= 1/8 \\ P(d) &= 1/8 \end{aligned}$$

Example
$P(a) = 1/2$
$P(b) = 1/4$
$P(c) = 1/8$
$P(d) = 1/8$

Decodability

We need to be able to decompress the data

- Obviously we need 1:1 mapping from words to codes
  - but really we need 1:1 mapping from messages to coded version
- Remember that "word-break" would have to be either
  - an extra symbol
  - implicit in the code

Decodable code

End of codes indicated by a 1

$$
\begin{aligned}
a &\leftrightarrow 01 \\
b &\leftrightarrow 001 \\
c &\leftrightarrow 0001 \\
d &\leftrightarrow 00001
\end{aligned}
$$

Average message length

$$\text{bits per word } = 2\frac{1}{2} + 3\frac{1}{4} + 4\frac{1}{8} + 5\frac{1}{8} = \frac{23}{8}$$

We know we can do better, but how much?

Example

$$
\begin{aligned}
a &\leftrightarrow 0\\
b &\leftrightarrow 10\\
c &\leftrightarrow 110\\
d &\leftrightarrow 111
\end{aligned}
$$

- Decodable, because any sequence we convert back to its message
- Compresses, because more common words have shorter codes

$$\text{bits per word } = 1\frac{1}{2} + 2\frac{1}{4} + 3\frac{1}{8} + 3\frac{1}{8} = \frac{7}{4}$$

  - average compression from $2 \to 7/4$

Simple question

Is there a fundamental limit to lossless compression?

What was the entropy here?

More complex question

How would you come up with a code

- given the PMF?
- just given the data?

How do ensure the code is decodable?
Are there other redundancies in data that we can exploit and how would we find them efficiently?

More complex question
How would you come up with a code
- given the PMF?
- just given the data?
How do ensure the code is decodable?
Are there other redundancies in data that we can exploit and how would we find them efficiently?

Further reading I

Gjerrit Meinsma, *Data compression & information theory*, Mathematisch cafe, 2003, wwwhome.math.utwente.nl/~meinsmag/onzin/shannon.pdf.