

Universal Source Coding

Peter Andreasen

December 5, 2001

An overview of the talk

Part I

How to compress data – a brief introduction.

Part II

Universal source coding, definition of problem.

Part III

A universal source code.

How to compress data

Consider some data, for example the sequence of 12 letters from the alphabet $A = \{a, b\}$:

aaaabaaabaaa.

How can we compress the data?

Doing like this,

aaaabaaabaaa,

is considered cheating and does *not* count!

We want something like this:

aaaabaaabaaa \longrightarrow Code \longrightarrow 0110101

0110101 \longrightarrow Decode \longrightarrow aaaabaaabaaa

How to compress data - II

Compression by counting: Let us try something differently. Consider again data from the alphabet $A = \{a, b\}$:

aaaabaaabaaa

We want to compress the data using the alphabet $B = \{0, 1\}$. Note, that $|A| = |B| = 2$.

By using a table like

Data	Encoding
a	0
b	10
aaa	11

we can 'translate' the data like this:

Data	aaa	a	b	aaa	b	aaa
Encoding	11	0	10	11	10	11

into the message 11010111011. Thus we have reduced the data length from 12 to 11 symbols.

Why does it work? Or more importantly, *when* does it work?

How to compress data - III

First, a quote:

I know how to spell 'banana', but I don't know when to stop.

– A little girl

Compression by deja vu: Consider again the sequence

aaaabaaabaaa

If we should read this sequence to another person, we might say “Four a’s followed by baaa repeated two times.”. This can also be written as

aaaabaaa < repeat the last 4 letters 1 time > .

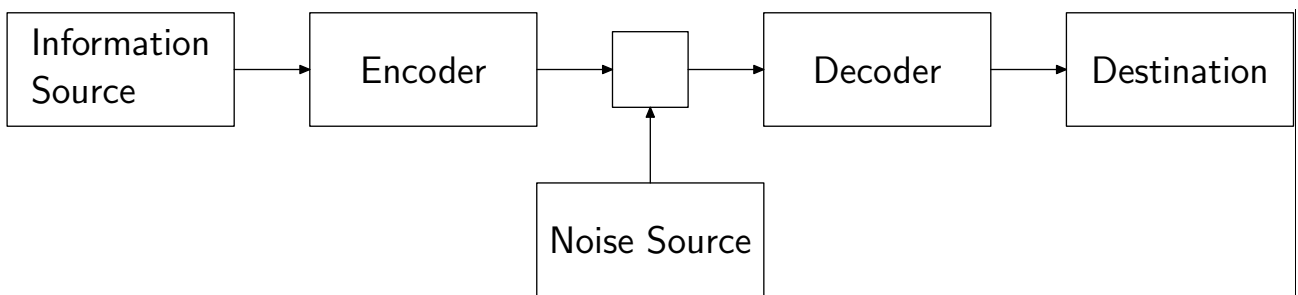
The little girl from above had mentally ‘compressed’ banana in this fashion, but had forgotten how many times to repeat na!

If this idea should work as a true compression method we must, of course, find a way to turn instructions like ‘repeat last 4 letters 1 time’ into sequences of letters.

How to compress data - IV

Compressing data is one of the fundamental problems in information theory.

The Shannon model of a communication system:



We do not consider cases where noise is introduced, hence we focus on the information *source* and the *encoder* with the assumption that the receiver will always be able to decode the message.

Transmission of data across physical distance as well as temporal distance.

Sources and Coding

Definition 1 A code $\varphi_n : A^n \rightarrow \{0, 1\}^*$ is an injective mapping from the set A^n of all sequences of length n over the alphabet A to the set of all finite binary sequences, $\{0, 1\}^*$. We write $|\varphi_n(x_1^n)|$ for the length of the encoding of x_1^n .

Example: Let $A = \{a, b, c\}$, then $\varphi_1 : A \rightarrow \{0, 1\}^*$ given by

x	$\varphi_1(x)$
a	0
b	10
c	11

is a code. And $|\varphi_1(b)| = 2$.

Definition 2 A probabilistic information source is a stationary ergodic probability measure μ on A^∞ . We write $\mu(x_1^n)$ for the probability of a sequence x_1^n of n letters from A .

Example: Let $A = \{0, 1\}$. Tossing an even coin with sides labeled 0 and 1, may be modelled as a probability measure defined by $\mu(0) = \frac{1}{2}$ and $\mu(1) = \frac{1}{2}$. In general, for any sequence x_1^n of 0's and 1's we have $\mu(x_1^n) = (\frac{1}{2})^n$.

Coding and entropy

Definition 3 (§111) *The entropy of a probabilistic source μ , is given by*

$$H(\mu) = \lim_{n \rightarrow \infty} \frac{1}{n} \left(\sum_{x_1^n \in A^n} \mu(x_1^n) \log \frac{1}{\mu(x_1^n)} \right)$$

Note, this is the entropy of the stochastic *model*. Entropy of natural languages or any other *real* data is much harder to compute.

Example: For the coin tossing source we have $\mu(x_1^n) = (\frac{1}{2})^n = 2^{-n}$ and hence the sum above collapses to $2^n 2^{-n} \log 1/2^{-n} = \log 2^n = n$. Therefore $H(\mu) = 1$.

Theorem 1 (Shannon, 1948) *Let μ be a probabilistic source over an alphabet A and let φ_n be a prefix free code, then*

$$\frac{E_{\mu}(|\varphi_n(x_1^n)|)}{n} \geq H(\mu)$$

for any number n . However, there do exist prefix free codes φ_n such that $E_{\mu}(|\varphi_n(x_1^n)|)/n$ is as close to $H(\mu)$ as we like, provided n is big enough.

Source Coding

Codes for which $E_{\mu}(|\varphi_n(x_1^n)|)/n$ approaches the theoretical limit of $H(\mu)$ as n goes towards infinity, are well known. The Shannon code, where $|\varphi_n(x_1^n)| = \lceil -\log \mu(x_1^n) \rceil$ is an example of such a code.

Example: Consider a skewed coin such that flipping it produces 0 with probability $\mu(0) = \frac{1}{4}$ and 1 with probability $\mu(1) = \frac{3}{4}$. The entropy of μ is then $H(\mu) = 0.811$. For a large number, say $n = 2$, the Shannon code would assign a code of length $\lceil -\log \mu(00) \rceil = \lceil -\log \frac{1}{16} \rceil = 4$ to the message 00.

x_1^2	$\mu(x_1^2)$	$\lceil -\log \mu(x_1^2) \rceil$	$\varphi_2(x_1^2)$
00	1/16	4	0000
01	3/16	3	001
10	3/16	3	010
11	9/16	1	1

And the expected codelength divided by n is thus $(4 \times 1/16 + 3 \times 3/16 + 3 \times 3/16 + 1 \times 9/16)/2 = 0.969$. Which is as close to the entropy.

However, we want to ask for more.

Universal Source Coding

In universal source coding we ask for a global code, φ with the following property: For *any* probabilistic source μ the code should perform asymptotically optimal. That is,

$$\lim_{n \rightarrow \infty} \frac{E_{\mu}(|\varphi(x_1^n)|)}{n} - H(\mu) = 0$$

for any μ .

The Shannon code (length of codeword for x_1^n equals $\lceil -\log \mu(x_1^n) \rceil$) is *not* universal. It is tuned for one particular source.

A universal source code should be of *adaptive* nature.

Lempel-Ziv Algorithm

(Compression by déjà vu)

Idea: Parse the string to be encoded according to the following rule

The next phrase is the shortest new phrase

and then encode each phrase.

Example: The sequence

ababbabaaabaaabba

is parsed into

a | b | ab | ba | baa | aba | aa | bb | a

Observation: Each phrase is a previous phrase plus one new symbol. So we encode each phrase by a pair (i, s) where i is an index and $s \in A$:

<i>a</i>	<i>b</i>	<i>ab</i>	<i>ba</i>	<i>baa</i>	<i>aba</i>	<i>aa</i>	<i>bb</i>	<i>a</i>
<i>0, a</i>	<i>0, b</i>	<i>1, b</i>	<i>2, a</i>	<i>4, a</i>	<i>3, a</i>	<i>1, a</i>	<i>2, b</i>	<i>1, λ</i>
00000	00001	00011	00100	01000	00110	00010	00101	0001*

Codelength: We find

$$|\text{codeword}| \simeq c(x_1^n) (\log c(x_1^n) + \log |A|)$$

where $c(x_1^n)$ is the number of phrases of x_1^n .

Conclusion: We have $\varphi_{LZ} : A^* \rightarrow B^*$, a code.

LZ78 is a universal code

Theorem 2 (§183) *Let μ be any probabilistic source, then*

$$\limsup_{n \rightarrow \infty} \frac{E_{\mu} |\varphi_{LZ}(x_1^n)|}{n} = H(\mu),$$

where φ_{LZ} is the Lempel-Ziv 1978 code.

Proof (main idea): LZ78 parses into *distinct* phrases. The *length* of phrases, must grow.

Moreover, of the long phrases of length m , there cannot be too many, if the data is assumed to stem from μ . For example, if we had the maximal number, 2^m , of phrases of each length m , then the data would start to look like the output of a coin-tossing process.

... , aa, aba, ba, ab, bb, ...

One can show, that when parsing x_1^n into *distinct* phrases $x_1^n = w_1 w_2 \cdots w_t$, then

$$|w_i| \geq \frac{\log n}{H(\mu)}$$

in most cases. Long phrases means good compression. Indeed,

$$c(x_1^n) = \frac{nH(\mu)}{\log n}$$

and then

$$|\varphi_n(x_1^n)| \simeq \frac{nH(\mu)}{\log n} (\log n + \log H(\mu) - \log \log n) \simeq nH(\mu),$$

which completes the 'proof'.

This modern proof of the Lempel-Ziv result, carries a distinct flavour of *combinatorics*.

Second order analysis

In 1997 Jacob Ziv gave the Shannon Lecture under the title “Back from Infinity”.

Redundancy: Set

$$R_n = \frac{1}{n}(E(|\varphi_{LZ}(\cdot)|) - H_n(\mu))$$

where E denotes expectation and H_n is the n 'th order entropy.

Second order analysis is study of subclasses of stationary, ergodic class.

In 1992 it was shown that if μ is finite order Markov then

$$R_n = O\left(\frac{\log \log n}{\log n}\right)$$

and in 1997 it was proven that if μ is a Bernoulli source, then

$$R_n = O\left(\frac{1}{\log n}\right)$$

These results are *not* flattering for the Lempel-Ziv code! In both cases a convergence at the speed of $O(\log n/n)$ is possible if one knows the source μ . (Davisson 1973).

Coding of individual sequences

Recall how the Lempel-Ziv parses data:

a, aa, ab, aaa, b, aaa

It is possible to cook up sequences that are really tough for the LZ code:

a, b, aa, ab, ba, bb, aaa, aab, aba, aab, . . .

The Lempel-Ziv algorithm cannot compress this sequence.

The sequence above is known as a Champernowne sequence and is frequency typical for the coin flipping process.

Even though there is no statistical features to exploit, the sequence is very systematic, and hence we feel it should be compressible.

Non-probabilistic 'measures' of complexity of data sources: Hausdorff dimension and Kolmogorov Complexity.