Explaining ZIP Using Information Theory

Frans MJ Willems
f.m.j.willems@tue.nl

Information and Communication Theory Lab,
Signal Processing Systems Group,
Department of Electrical Engineering,
Eindhoven University of Technology, The Netherlands

http://www.tue.nl/ictlab

WinZip and LZ77

WinZip is based on LZ77, a **universal lossless compression method** proposed by Lempel & Ziv [1977].
Compression is achieved by **replacing repeated segments** in the data by **I. a pointer** and **II. the copy-length**. To avoid deadlock **III. the first non-matching symbol is added** to each pointer-length pair.

EXAMPLE LZ77:

search buffer	look-ahead buffer	output
	a b r a c a d a b r a _	(0,-,a)
a	b r a c a d a b r a _	(0,-,b)
a b	r a c a d a b r a _	(0,-,r)
a b r	a c a d a b r a _	(3,1,c)
a b r a c	a d a b r a _	(2,1,d)
a b r a c a d	a b r a _	(7,4,_)
a b r a c a d a b r a _		

QUESTION: Why does LZ77 work? Note that the **statistics of the data are unknown!**

Motivation

Definition

Let $\cdots, X_1, X_0, X_1, X_2, \cdots$ be the output of a **binary stationary** source. Then the **entropy rate** $H_\infty(X)$ of the source is defined as

$$H_\infty(X) \triangleq \lim_{N\to\infty} \frac{1}{N} H(X_1, X_2, \cdots, X_N). \qquad \text{code-bits / source-digit}$$

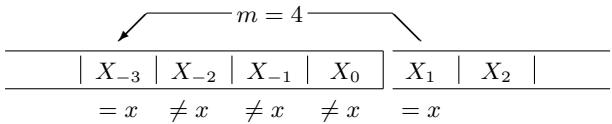
- We will show that a **simple algorithm** achieves rates $R_N = E[L(X^N)/N$ approaching the source's entropy rate $H_\infty(X)$ for $N \to \infty$.
  Here $L(x^N)$ is the length of the codeword for $x^N \in \{0,1\}^N$.
- The algorithm is **universal**. Knowledge of the source statistic is not needed.
- The algorithm that we propose here explains why LZ77 works.

Waiting Times

- Consider the **discrete stationary** process

$$\cdots, X_{-3}, X_{-2}, X_{-1}, X_0, X_1, X_2, X_3, \cdots.$$

- Suppose that $X_1 = x$ for symbol-value $x \in \mathcal{X}$ with $\Pr\{X_1 = x\} > 0$. We say that the **waiting time** of the x that occurred at time $t = 1$ is m if $X_{1-m} = x$ and $X_t \neq x$ for $t = 2 - m, \cdots, 0$.



- Let $Q_x(m)$ be the **conditional probability** that the waiting time of the x occurring at $t = 1$ is m, hence

$$Q_x(m) \triangleq \Pr\{X_{1-m} = x, X_{2-m} \neq x, \cdots, X_0 \neq x | X_1 = x\}.$$

Waiting Times

Then the **average waiting time** for symbol-value x with $\Pr\{X_1 = x\} > 0$ is defined as

$$T(x) \triangleq \sum_{m=1,2,\cdots} mQ_x(m).$$

Kac's Result

Theorem (Kac, 1947)

For stationary and **ergodic** *processes*

$$T(x) = \frac{1}{\Pr\{X_1 = x\}}, \tag{1}$$

for any x *with* $\Pr\{X_1 = x\} > 0$.
In an ergodic process time averages are equal to ensemble averages.

Kac's Result

Example

Consider a **binary IID** process and assume that $\Pr\{X_1 = 0\} = p > 0$. Then

$$\begin{aligned} Q_0(m) &= p(1-p)^{m-1} \text{ and} \\ T(0) &= \sum_{m=1,2,\cdots} mp(1-p)^{m-1} \\ &= p\frac{d}{dp}\sum_{m=1,2,\cdots} -(1-p)^m \\ &= p\frac{d}{dp}\frac{-(1-p)}{1-(1-p)} = p\frac{1}{p^2} \\ &= \frac{1}{p}. \end{aligned}$$

Kac's Result for Sliding Blocks

- Let N be a positive integer. When $\cdots, X_{-1}, X_0, X_1, X_2, \cdots$ is stationary and ergodic, then so is

$$\cdots, \begin{pmatrix} X_{-1} \\ X_0 \\ \vdots \\ X_{N-2} \end{pmatrix}, \begin{pmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{pmatrix}, \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}, \begin{pmatrix} X_2 \\ X_3 \\ \vdots \\ X_{N+1} \end{pmatrix}, \cdots$$

- Therefore Kac's result holds also for "sliding" N-blocks, hence

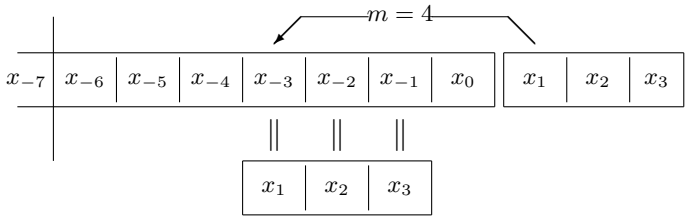
$$T((x_1, x_2, \cdots, x_N)) = \frac{1}{\Pr\{(X_1, X_2, \cdots, X_N) = (x_1, x_2, \cdots, x_N)},$$

  if $\Pr\{(X_1, X_2, \cdots, X_N) = (x_1, x_2, \cdots, x_N)\} > 0$.

- Now the waiting time is m if m is the **smallest positive integer** such that $(x_{1-m}, x_{2-m}, \cdots, x_{N-m}) = (x_1, x_2, \cdots, x_N)$
  or $x_{1-m}^{N-m} = x_1^N$ where $x_a^b \stackrel{\Delta}{=} (x_a, x_{a+1}, \cdots, x_b)$, for $b \geq a$.

Universal Source Coding Algorithm Based on Waiting Times

- Suppose that our source is **binary** i.e. $X_n \in \{0,1\}$ for all integer n. The encoder wants to transmit a source block $x_1^N \stackrel{\Delta}{=} (x_1, x_2, \cdots, x_N)$ to the decoder. Example $N = 3$.



- **BOTH** encoder and decoder **have access to buffers** containing all previous source symbols $\cdots, x_{-2}, x_{-1}, x_0$.
- Using these previous source symbols **in its buffer** the **ENCODER** determines the **waiting time** m of x_1^N. It is the smallest positive integer m that satisfies

$$x_{1-m}^{N-m} = x_1^N.$$

Universal Source Coding Algorithm Based on Waiting Times

- The **waiting time m is encoded and sent to the decoder**.
- The (prefix) code for m consists of a **preamble** $p(m)$ and an **index** $i(m)$ and has length $l(m)$.
  CODE TABLE for the waiting time m for $N = 3$:

m	$p(m)$	$i(m)$	$l(m)$
1	00	-	2+0=2
2	01	0	2+1=3
3	01	1	2+1=3
4	10	00	2+2=4
5	10	01	2+2=4
6	10	10	2+2=4
7	10	11	2+2=4
≥ 8	11	copy of $x_1x_2x_3$	2+3=5

- After decoding m the **DECODER** can reconstruct x_1^N using the previous source symbols **in its buffer**.
- Index lengths are $0, 1, \cdots, N-1$ and the "copy"-code has length N.
- The preamble $p(m)$ of length $\lceil \log_2(N+1) \rceil$ bits specifies one of the $N+1$ alternatives.

Universal Source Coding Algorithm Based on Waiting Times

- The code-table is constructed such that the code-block length $l(m)$ satisfies

$$l(m) = \begin{cases} \lceil \log_2(N+1) \rceil + \lfloor \log_2 m \rfloor & \text{if } m < 2^N, \\ \lceil \log_2(N+1) \rceil + N & \text{if } m \geq 2^N. \end{cases}$$

  This results in the **upper bound**

$$l(m) \leq \lceil \log_2(N+1) \rceil + \log_2 m. \tag{2}$$

- After processing the block x_1^N **both the encoder and decoder can update their buffers**. Then the next block

$$x_{N+1}^{2N} \triangleq x_{N+1}, x_{N+2}, \cdots, x_{2N}$$

  is processed in a similar way, etc.
- **Buffers need only contain** the previous $B = 2^N - 1$ source symbols! If $m \geq 2^N$ then x_1^N is copied and m is not needed anymore.

Analysis of the Waiting-Time Algorithm

- Assume that x_1^N occurred as first block. What is the expected codeword length $L(x_1^N)$ for x_1^N?

$$\begin{aligned} L(x_1^N) &= \sum_{m=1,2,\cdots} Q_{x_1^N}(m) l(m) \\ &\overset{(a)}{\leq} \sum_{m=1,2,\cdots} Q_{x_1^N}(m) \lceil \log_2(N+1) \rceil + \sum_{m=1,2,\cdots} Q_{x_1^N}(m) \log_2 m \\ &\overset{(b)}{\leq} \lceil \log_2(N+1) \rceil + \log_2 \left(\sum_{m=1,2,\cdots} m Q_{x_1^N}(m) \right) \\ &\overset{(c)}{=} \lceil \log_2(N+1) \rceil + \log_2 \frac{1}{\Pr\{X_1^N = x_1^N\}}. \end{aligned}$$

  - (a) follows from the bound (2) on $l(m)$,
  - (b) from Jensen's inequality $E[\log_2 M] \leq \log_2 E[M]$ since the $\log_2$ is a convex-$\cap$.
  - Furthermore (c) follows from Kac's theorem (1).

- Now $L(x_1^N)$ is upper bounded by preamble length $\lceil \log_2(N+1) \rceil$ plus the **ideal codeword length** $\log_2 \frac{1}{\Pr\{X_1^N = x_1^N\}}$.

Analysis of the Waiting-Time Algorithm

- The probability that x_1^N occurred as first block is $\Pr\{X_1^N = x_1^N\}$. For the expected codeword length $E[L(X_1^N)]$ we therefore obtain

$$\begin{aligned} E[L(X_1^N)] &= \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} L(x_1^N) \\ &\leq \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} \left(\lceil \log_2(N+1) \rceil + \log_2 \frac{1}{\Pr\{X_1^N = x_1^N\}} \right) \\ &= \lceil \log_2(N+1) \rceil + H(X_1^N). \end{aligned}$$

- For the rate R_N we now obtain

$$R_N = \frac{E[L(X_1^N)]}{N} \leq \frac{H(X_1^N)}{N} + \frac{\lceil \log_2(N+1) \rceil}{N}.$$

Achieving Entropy

First note that

$$\lim_{N\to\infty} \frac{H(X_1^N)}{N} \triangleq H_\infty(X)$$

and

$$\lim_{N\to\infty} \frac{\lceil \log_2(N+1) \rceil}{N} = 0.$$

Theorem (W., 1986, 1989)

The **Waiting-Time Algorithm** *achieves entropy since*

$$\lim_{N\to\infty} R_N = \lim_{N\to\infty} \left(\frac{H(X_1^N)}{N} + \frac{\lceil \log_2(N+1) \rceil}{N} \right) = H_\infty(X).$$

The encoder and decoder use buffers of length $B = 2^N - 1$ *in which previous symbols are stored.*

NOTE: This algorithm is **universal**. Although the statistics of the source are unknown, entropy is achieved.

Conclusion

- Simple algorithm
- Beautiful analysis

“Algorithms Are Fun”

Prof. PIET SCHALKWIJK,
Chair Information Theory Group, TH Eindhoven
1972 - 1995

Piet's Algorithms: Pascal - Triangle Algorithm [1972]

IEEE TRANSACTIONS ON INFORMATION THEORY, MAY 1972

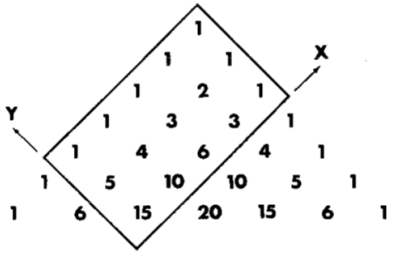


Fig. 1. Pascal's triangle gives an ordering of source sequences.

250 citations, universal, binary IID

Information Theory, Two Flavors

- **FIRST: Capacity Theorems and their Information-theoretic Proofs**
- **THEN: Coding Algorithms and their Information-theoretic Analysis**

Information Theory, Two Flavors

- **Capacity Theorems and their Information-theoretic Proofs**
  Edward van der Meulen (KU Leuven), my PhD thesis supervisor, 1982

- **Coding Algorithms and their Information-theoretic Analysis**
  Piet Schalkwijk (TH Eindhoven), my MSc thesis supervisor, 1979

Information Theory, Two Flavors

- **Capacity Theorems and their Information-theoretic Proofs**
  Edward van der Meulen, and
  Claude E. Shannon approach, Science

- **Coding Algorithms and their Information-theoretic Analysis**
  Piet Schalkwijk, and
  Richard W. Hamming approach, Engineering

Q: Am I an engineer or a scientist?

2025 IEEE Hamming Medal, April 23-24, Tokyo

Recognition of Coding-Algorithms approach to Information Theory