Universal Lossless coding
Lempel-Ziv Coding

Outline

- Shortcomings of Block-to-variable-length codes
- Variable-length-to-block coding
- Lempel-Ziv coding

Shortcomings of Block-to-Variable-length Codes

1. Complexity

   A blocklength K code for a source with M letters has M^K codewords.

   Example: $M = 26$, $K = 5 =$ avg. word length: $M^K \approx 12 \times 10^6$

2. Dependence upon statistical model

   a. What if source probabilities are not known?

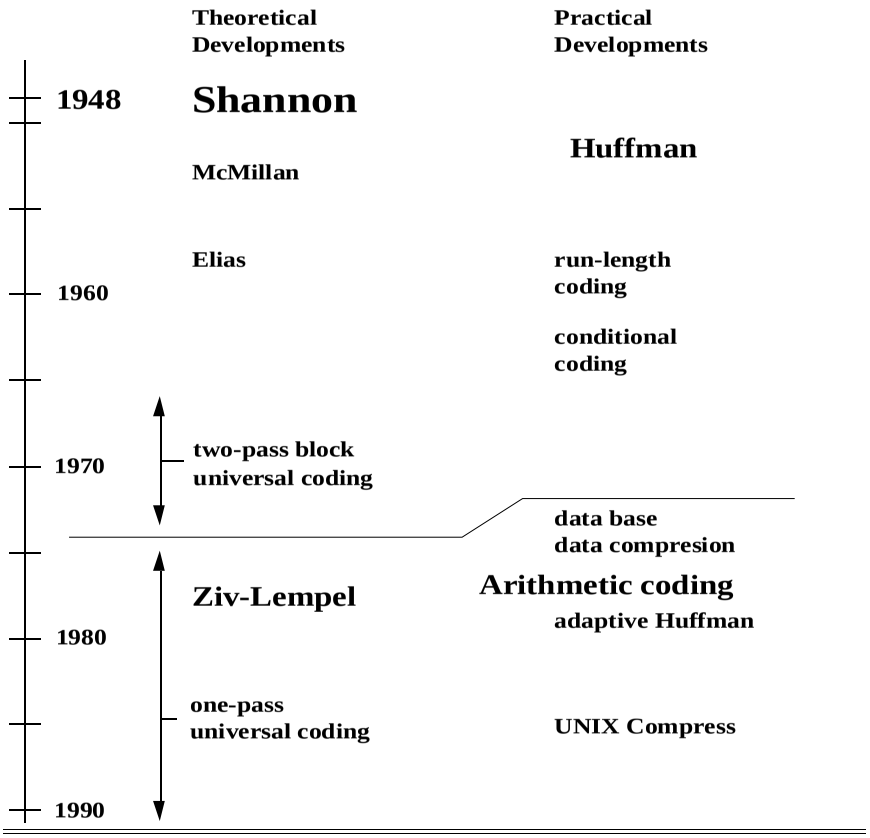
   b. What if source probabilities "change" over time?

   c. What if there are no probabilities?

3. Goals

   Find low complexity codes that are *universal*, meaning that when applied to any stationary ergodic source they should attain rate close to the entropy-rate of that source.

Historical Review



Lempel-Ziv Coding

- In these notes we introduce Lempel-Ziv codes, which is the most widely used family of universal codes.
- There are many variations.
- All are basically a kind of adaptive variable-length-to-block coding. Variable-length to block coding is introduced on the next pages.
- LZ-78 and LZ-77 are the two broad categories of LZ codes.

  LZ-78 refers to codes following the approach described in

  J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding,"*IEEE Trans. Inform. Theory*, vol. 24, pp. 530-536, Sept. 1978.

  LZ-77 refers to codes following the approach described in

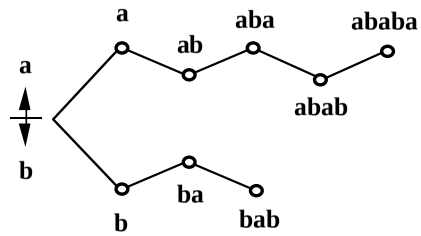
  J. Ziv and A. Lempel, "A universal algorithm for sequential data compression,"*IEEE Trans. Inform. Theory*, vol. 23, pp. 337-343, May. 1977.

Variable-Length-to-Block Coding

aka Dictionary Coding

dictionary of source words	binary codewords $\underline{w}_i$
a	000
b	001
ab	010
aba	011
ba	100
bab	101
abab	110
ababa	111

dictionary tree



"Greedy" Encoding Rule:

Find longest sourceword $\underline{w}$ that prefixes remaining symbols; output codeword.

Example: ababbabbababababab is parsed into <u>abab</u> <u>bab</u> <u>bab</u> <u>ababa</u> <u>bab</u>

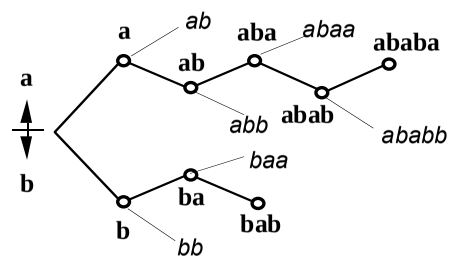
Rate: $R = \dfrac{\text{codeword len.}}{\text{avg. sourceword len.}} = \dfrac{L}{\sum_i p_i n_i} = \dfrac{5\times 3}{18} = \dfrac{5}{6}$ bit/symb. for this example

where p_i = probability that ith source word is chosen, and n_i is its length

Variable-Length-to-Block Coding

dict'ry of source words	stopping words	prob-ability p_i	source length n_i	code-word $\underline{w}_i$
a	aa	.050	1	000
b	bb	.050	1	001
ab	abb	.045	2	010
aba	abaa	.041	3	011
ba	baa	.045	2	100
bab	bab	.405	3	101
abab	ababb	.036	4	110
ababa	ababa	.328	5	111

dictionary tree



For binary alphabet, the probablity of a given source word equals the probability of the "stopping word".

The above probabilities assume greedy encoding and a Markov source with p(a) = p(b) = 0.5, p(b|a) = p(a|b) = 0.9.

Rate: $R = \dfrac{\text{codeword len.}}{\text{avg. sourceword len.}} = \dfrac{L}{\sum_i p_i n_i} = \dfrac{3}{3.4} = 0.88$ bits/symbol

The above probabilities assume greedy encoding and (approximately) a Markov model with p(a) = p(b) = 0.5, p(b|a) = p(a|b) = 0.9 . $H_\infty = 0.47$.

Basic idea: A good code has $p_i \cong 2^{-L}$ for most i.

LZ-78: Lempel-Ziv Coding

An Adaptive/Universal Block to VL Code

example sequence to encode: a b a b a b a b a b a b a b a b a b a b a b a b a b

encoding table

source word	index	code word
a	1	00..000
b	2	00..001
	3	00..010
	4	00..011
	5	00..100
	6	00..101
	7	00..110
	8	00..111
	...	...
	2^L	11..111

Encoding Algorithm

1. Initialize encoding table with each letter of source alphabet as a source word, in alphabetical order, and with 2^L codewords of length L in usual order.
2. Find longest sourceword $\underline{w}$ that prefixes remaining symbols.
3. Send binary codeword for $\underline{w}$ and next symbol α (using some binary code)
4. Mark $\underline{w}\alpha$ as "already encoded".
5. If table not full, add $\underline{w}\alpha$ to encoder table.
6. Go to 2.

Rate: $R = \dfrac{L+1}{\text{length}(\underline{w})+1}$ bits/symbol (drop "+1" when table full)

For the above squence: let n denote the number of source words encoded. Then

$$R \cong \frac{4(L+1)}{n}, \quad \text{when } n \le 2^L, \quad \text{and Rate} \cong \frac{(L+1)}{2^{L-1}}, \quad \text{when } n > 2^L.$$

LZ-78 Decoding

- Assume decoder knows the source alphabet and L.
- Decoding table is initialized with each letter of the alphabet as a source word, in alphabetical order, and with 2^L codewords of length L in usual order.
- As each $\underline{w}$ and α is received, decoder augments the table in the same way as the encoder has done.
- For the example of the previous page, the encoded symbols are:

  1 b 3 a 2 a 5 b 4 b 7 a 6 a .

decoding table		
index	source word	code word
1	a	00..000
2	b	00..001
3		00..010
4		00..011
5		00..100
6		00..101
7		00..110
8		00..111
.		.
2^L		11..111

LZW -- A Variation of LZ-78

T.A. Welch, "A technique for high-performance data compression," *Computer*, pp. 8-19, June 1984.

source sequence: a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b ..

encoding table

index	source word	code word
1	a	00..000
2	b	00..001
3		00..010
4		00..011
5		00..100
6		00..101
7		00..110
8		00..111
.		.
2^L		11..111

Encoding algorithm

1. Initialize encoding table with each letter of source alphabet as a source word, in alphabetical order, and with 2^L codewords of length L in usual order.
2. Find longest sourceword $\underline{w}$ that prefixes remaining symbols.
3. Send binary codeword for $\underline{w}$.
4. Mark $\underline{w}$ as "already encoded".
5. If table not full, add $\underline{w}\alpha$ to encoder table, where α is the symbol that follows $\underline{w}$.
6. Go to 2.

Rate: $R \frac{L}{\text{length of } \underline{w}}$ bits/symbol

For example sequence: $R \cong \frac{4L}{n}$, when $n \le 2^L$; Rate $\cong \frac{L}{2^{L-1}}$, when $n > 2^L$

where n = number of sourcewords encoded

Decoding LZW

- Assume decoder knows the source alphabet and L.
- Decoding table is initialized with each letter of the alphabet as a source word, in alphabetical order, and with 2^L codewords of length L in usual order.
- As each w is received, decoder augments the table in the same way as the encoder has done, except that it leaves a "blank" for α, which it will not know until it receives the next w. Specifically, α is the first letter of the next w.
- For the example of the previous page, the encoded symbols are:

  1 2 3 5 4 7 6 ...

decoding table		
index	source word	code word
1	a	00..000
2	b	00..001
3		00..010
4		00..011
5		00..100
6		00..101
7		00..110
8		00..111
.		.
2^L		11..111

Other variants: do not limit the size of table, nor fix the length of the codewords.

LZ-77

source sequence: a b a b a b a b a b a b a b a b a b a b a b a b a b a b ..

Fix integers B and C. (Typically, $B = C\,2^{\beta C}$, for some $0 < \beta < 1$)

Assume we've already encoded $x_1, \ldots, x_k$.

To encode $x_{k+1}\, x_{k+2} \ldots$, consider $x_{k-B+1} \ldots x_k$ to be stored in a "buffer" of length B .

1. Find pointer I and largest length J ($1 \le I \le B$, $1 \le J \le C$) such that

   $x_{k-J,\ldots,}x_{k-J+I} = x_{k+1}\, x_{k+2\ldots}\, x_{k+J}$

2. Send I and J with $\log_2 B$ bits and $\log_2 C$ bits, respectively

Rate: $R \cong \dfrac{\log B + \log C}{E\,J}$ bits/symbol

For this sequence: $R = \dfrac{\log B + \log C}{C}$

This variation of LZ coding continually adapts.

LZ Optimality

Theorem (LZ '78)

For any stationary ergodic source, if L is chosen large enough,

rate $\cong$ entropy-rate

Theorem (LZ '78)

For any infinite sequence, if L is chosen large enough, the rate is as small as that of any finite-state code designed just for that sequence.

Why? Theorem (Wyner, Ziv '89)

Let $J_B(x)$ length of longest prefix of $x = x_1, x_2, \ldots$ that is matched by some string in the buffer $x_{-B+1}, \ldots, x_0$.

For a stationary ergodic source,

$$\lim_{B\to\infty} \frac{\log B}{J_B(x)} = H_\infty(X) \quad \text{in probability,} \quad \text{i.e. } J_B(x) \cong \frac{\log B}{H_\infty(X)} \text{ with high prob.}$$

Consider Variant LZ 77 with buffer size B and length matching limit $C \cong \dfrac{\log B}{H_\infty(X)}$.

Then J = C with high probability and so

$$\text{rate} = \frac{\log B + \log C}{E\,J} \cong \frac{\log B + \log\log B - \log H_\infty(X)}{(\log B)/H_\infty(X)}$$

$$\cong H_\infty(X)\left(1 + \frac{\log\log B}{\log B} - \frac{\log H_\infty(X)}{\log B}\right) \cong H_\infty(X)$$