Lossless Source Coding - Contents

- Classification of Lossless Source Codes
- Variable-Length Coding for Scalars
  - Unique Decodability
  - Entropy
  - The Huffman Algorithm
  - Conditional Huffman Codes
  - Adaptive Huffman Codes
- Variable-Length Coding for Vectors
  - Huffman Codes for Fixed-Length Vectors
  - Huffman Codes for Variable-Length Vectors
- Elias Coding and Arithmetic Coding
  - Elias Coding
  - Arithmetic Coding
- Probability Interval Partitioning Entropy Coding
- Comparison of Lossless Coding Techniques
- Adaptive Coding

Lossless Source Coding - Overview

- Reversible mapping of sequence of discrete source symbols into sequences of codewords
- Other names: noiseless coding, entropy coding
- Original source sequence can be exactly reconstructed - not the case in lossy coding
- Bit rate reduction possible, if source data contain statistical properties that are exploitable for data compression

Lossless Source Coding - Terminology

- Message $\boldsymbol{s}^{(L)} = \{s_0, \cdots, s_{L-1}\}$ drawn from stochastic process $\boldsymbol{S} = \{S_n\}$
- Sequence $\boldsymbol{b}^{(K)} = \{b_0, \cdots, b_{K-1}\}$ of K *bits* ($b_k \in \mathcal{B} = \{0, 1\}$)
- Process of lossless coding: Message $\boldsymbol{s}^{(L)}$ is converted to $\boldsymbol{b}^{(K)}$
- Assume:
  - Subsequence $\boldsymbol{s}^{(N)} = \{s_n, \cdots, s_{n+N-1}\}$ with $1 \leq N \leq L$ and
  - Bits $\boldsymbol{b}^{(\ell)}(\boldsymbol{s}^{(N)}) = \{b_0, \cdots, b_{\ell-1}\}$ assigned to it
- *Lossless source code*
  - Encoder mapping:

$$\boldsymbol{b}^{(\ell)} = \gamma\big(\boldsymbol{s}^{(N)}\big) \tag{1}$$

  - Decoder mapping:

$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\boldsymbol{b}^{(\ell)}\big) = \gamma^{-1}\big(\gamma\big(\boldsymbol{s}^{(N)}\big)\big) \tag{2}$$

Classification of Lossless Source Codes

- *Lossless source code*
  - Encoder mapping:

$$\boldsymbol{b}^{(\ell)} = \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \tag{3}$$

  - Decoder mapping:

$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\, \boldsymbol{b}^{(\ell)} \,\big) = \gamma^{-1}\big(\, \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \,\big) \tag{4}$$

- *Fixed-to-fixed mapping*: N and ℓ are both fixed (discussed as special case of fixed-to-variable)
- *Fixed-to-variable mapping*: N fixed and ℓ variable - Huffman algorithm for scalars and vectors (discussed in lecture)
- *Variable-to-fixed mapping*: N variable and ℓ fixed - Tunstall codes [Tunstall, 1967, Savari and Gallager, 1997] (not discussed in lecture)
- *Variable-to-variable mapping*: ℓ and N are both variable - arithmetic codes and PIPE codes (discussed in lecture)

Variable-Length Coding for Scalars

- Assign a separate codeword to each scalar symbol s_n of a message $\boldsymbol{s}^{(L)}$
- Assume: message $\boldsymbol{s}^{(L)}$ generated by stationary discrete random process $\boldsymbol{S} = \{S_n\}$
- Random variables $S_n = S$ with symbol alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$ and marginal pmf $p(a) = P(S = a)$
- Lossless source code: Assign each a_i to binary codeword $\boldsymbol{b}_i = \{b_0^i, \cdots, b_{\ell(a_i)-1}^i\}$, length $\ell(a_i) \geq 1$
- Average codeword length

$$\boxed{\bar{\ell} = E\{\ell(S)\} = \sum_{i=0}^{M-1} p(a_i)\, \ell(a_i)} \tag{5}$$

Optimization Problem

- Average code word length is given as

$$\bar{\ell} = \sum_{k=0}^{K-1} p(a_k) \cdot \ell(a_k) \tag{6}$$

- *The goal of the entropy code design problem is to minimize $\bar{\ell}$ while being able to uniquely decode*

a_i	$p(a_i)$	code A	code B	code C	code D	code E
a_0	0.5	0	0	0	00	0
a_1	0.25	10	01	01	01	10
a_2	0.125	11	010	011	10	110
a_3	0.125	11	011	111	110	111
$\bar{\ell}$		1.5	1.75	1.75	2.125	1.75

Unique Decodability and Prefix Codes

- For unique decodability, we need to generate a code $\gamma : a_i \rightarrow \boldsymbol{b}_i$ such that

$$\text{if} \qquad a_k \neq a_j \qquad \text{then} \qquad \boldsymbol{b}_k \neq \boldsymbol{b}_j \tag{7}$$

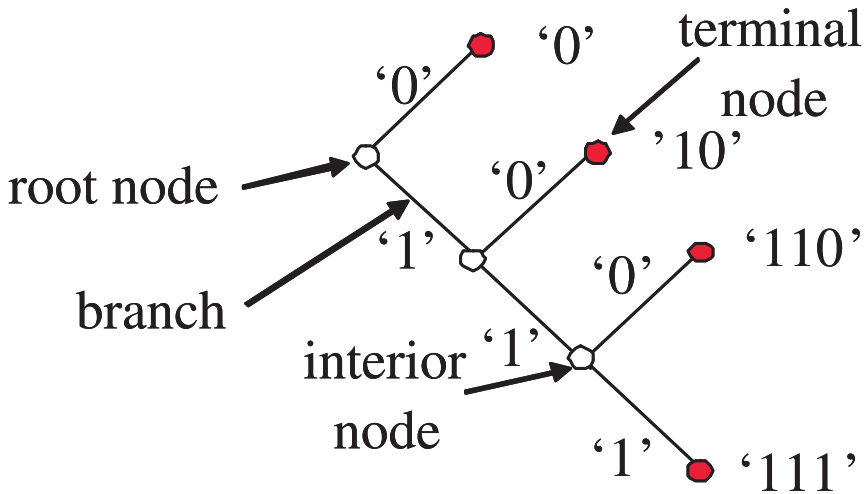
- For sequences of symbols, above constraint needs to be extended to the concatenation of multiple symbols

$\rightarrow$ *For a uniquely decodable code, a sequence of code words can only be generated by one possible sequence of source symbols.*

- One class of codes that satisfies the constraint of unique decodability is called *prefix codes*
- A code is called a prefix code if no code word is a prefix of any other code word
- It is obvious that if condition (7) is satisfied and the code is a prefix code, then a concatenation of symbols can be uniquely decodable

Binary Code Trees

- Prefix codes can be represented by trees



- A binary tree contains nodes with two branches (labelled as '0' and '1') leading to other nodes starting from a root node
- A node from which branches depart is called an interior node while a node from which no branches depart is called a terminal node
- A *prefix code* can be contracted by assigning letters of the alphabet $\mathcal{A}$ to terminal nodes of a binary tree

Parsing of Prefix Codes

- Given the code word assignment to terminal nodes of the binary tree, the parsing rule for this prefix code is given as follows
  1. Set the current node n_i equal to the root node
  2. Read the next bit b from the bitstream
  3. Follow the branch labeled with the value of b from the current node n_i to the descendant node n_j
  4. If n_j is a terminal node, return the associated alphabet letter and proceed with step 1 Otherwise, set the current node n_i equal to n_j and repeat the previous two steps
- Prefix codes are not only uniquely decodable, but also *instantaneously decodable*

Uniquely Decodability: Kraft Inequality

- Assume fully balanced tree with depth $\ell_{\max}$ (=longest code word)
- Codewords assigned to nodes with codeword length $\ell(a_k) \leq \ell_{\max}$
- Each choice with $\ell(a_k) \leq \ell_{\max}$ eliminates $2^{\ell_{\max}-\ell(a_k)}$ possibilities of code word assignment at level $\ell_{\max}$, example:
  - $\rightarrow$ $\ell_{\max} - \ell(a_k) = 0$, one option is covered
  - $\rightarrow$ $\ell_{\max} - \ell(a_k) = 1$, two options are covered
- Number of terminal nodes is less than or equal to number of terminal nodes in balanced tree with depth $\ell_{\max}$, which is $2^{\ell_{\max}}$

$$\sum_{k=0}^{K-1} 2^{\ell_{\max}-\ell(a_k)} \leq 2^{\ell_{\max}} \tag{8}$$

- A code γ may be uniquely decodable (McMillan) if

$$\text{Kraft inequality:} \qquad \zeta(\gamma) = \sum_{k=0}^{K-1} 2^{-\ell(a_k)} \leq 1 \tag{9}$$

Proof: [Cover and Thomas, 2006, p.116] or [Wiegand and Schwarz, 2011, p.25]

Lower Bound for Average Codeword Length I

- Average codeword length

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i)\,\ell(a_i) = -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{2^{-\ell(a_i)}}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (10)$$

- With the definition $q(a_i) = 2^{-\ell(a_i)} / \left(\sum_{k=0}^{M-1} 2^{-\ell(a_k)}\right)$, we obtain

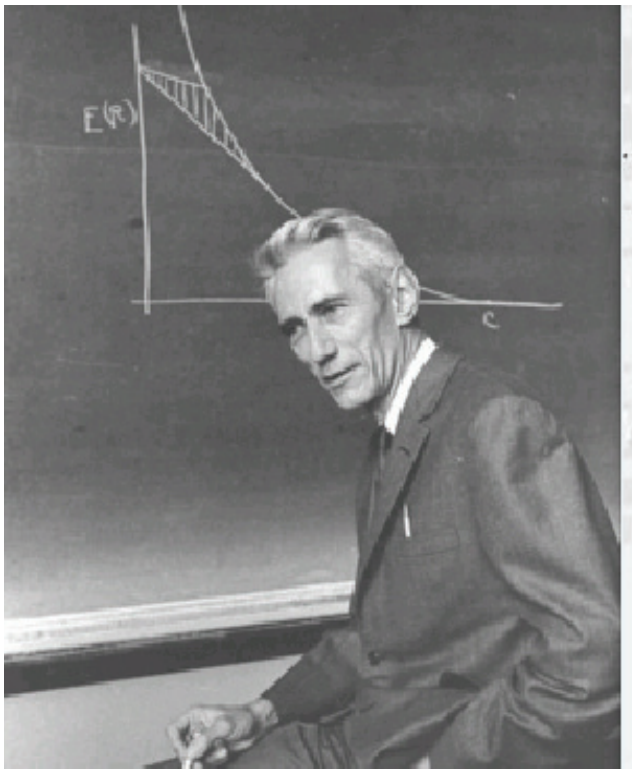
$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (11)$$

- We will show that

$$\bar{\ell} \geq -\sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) = H(S) \quad \text{(Entropy)} \quad (12)$$

Historical Reference

- C. E. SHANNON published entropy as a measure for average information
- Published 1 year later as: "The Mathematical Theory of Communication"



The Bell System Technical Journal

Vol. XXVII *July, 1948* *No. 3*

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

Lower Bound for Average Codeword Length II

- Average codeword length

$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (13)$$

- Kraft inequality $\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \leq 1$ applicable to first term

$$-\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) \geq 0 \quad (14)$$

- Inequality $\ln x \leq x - 1$ (with equality if and only if $x = 1$), applicable to second term

$$\begin{aligned} -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) &\geq \frac{1}{\ln 2}\sum_{i=0}^{M-1} p(a_i)\left(1 - \frac{q(a_i)}{p(a_i)}\right) \\ &= \frac{1}{\ln 2}\left(\sum_{i=0}^{M-1} p(a_i) - \sum_{i=0}^{M-1} q(a_i)\right) = 0 \end{aligned} \quad (15)$$

Entropy

- Average codeword length $\bar{\ell}$ for uniquely decodable codes is bounded

$$\bar{\ell} \geq H(S) = E\{-\log_2 p(S)\} = -\sum_{i=0}^{M-1} p(a_i)\, \log_2 p(a_i) \tag{16}$$

- *Redundancy* of a code is given by the difference

$$\varrho = \bar{\ell} - H(S) \geq 0 \tag{17}$$

- Redundancy is zero only, if the first and second term on the right side of (13) are equal to 0
- Upper bound of $\bar{\ell}$: choose $\ell(a_i) = \lceil -\log_2 p(a_i) \rceil$, $\forall a_i \in \mathcal{A}$

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i) \lceil -\log_2 p(a_i) \rceil < \sum_{i=0}^{M-1} p(a_i)\, (1 - \log_2 p(a_i)) = H(S) + 1 \tag{18}$$

- Bounds on minimum average codeword length $\bar{\ell}_{\min}$

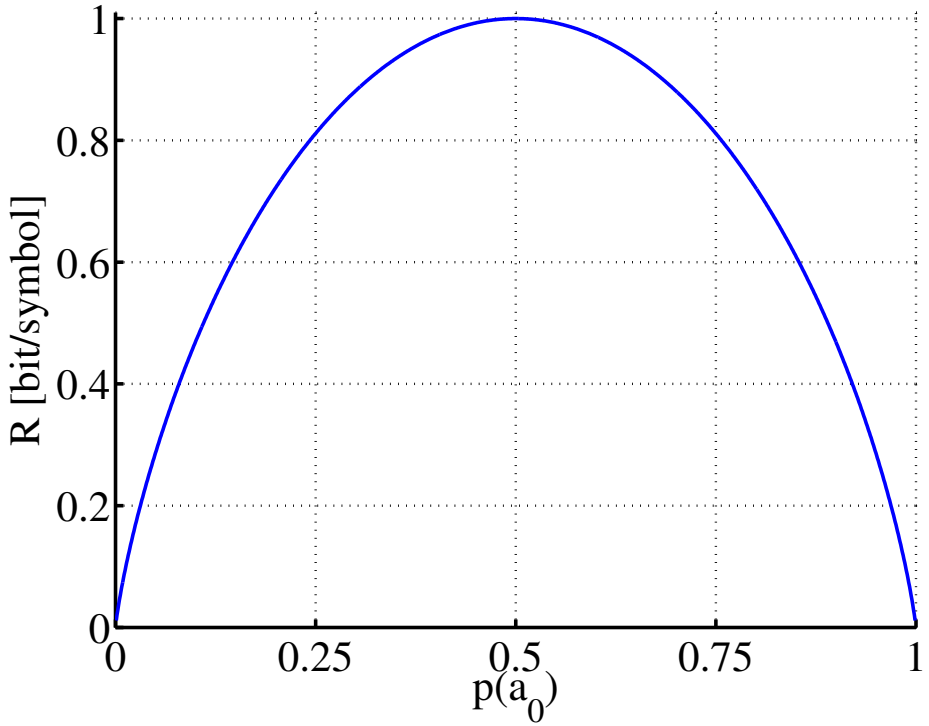
$$\boxed{H(S) \leq \bar{\ell}_{\min} < H(S) + 1} \tag{19}$$

Entropy of a Binary Source

- A binary source has probabilities $p(0) = p$ and $p(1) = 1 - p$
- The entropy of the binary source is given as

$$H(S) = -p \log_2 p - (1-p) \log_2 (1-p) = H_b(p) \tag{20}$$

with $H_b(x)$ being the so-called binary entropy function



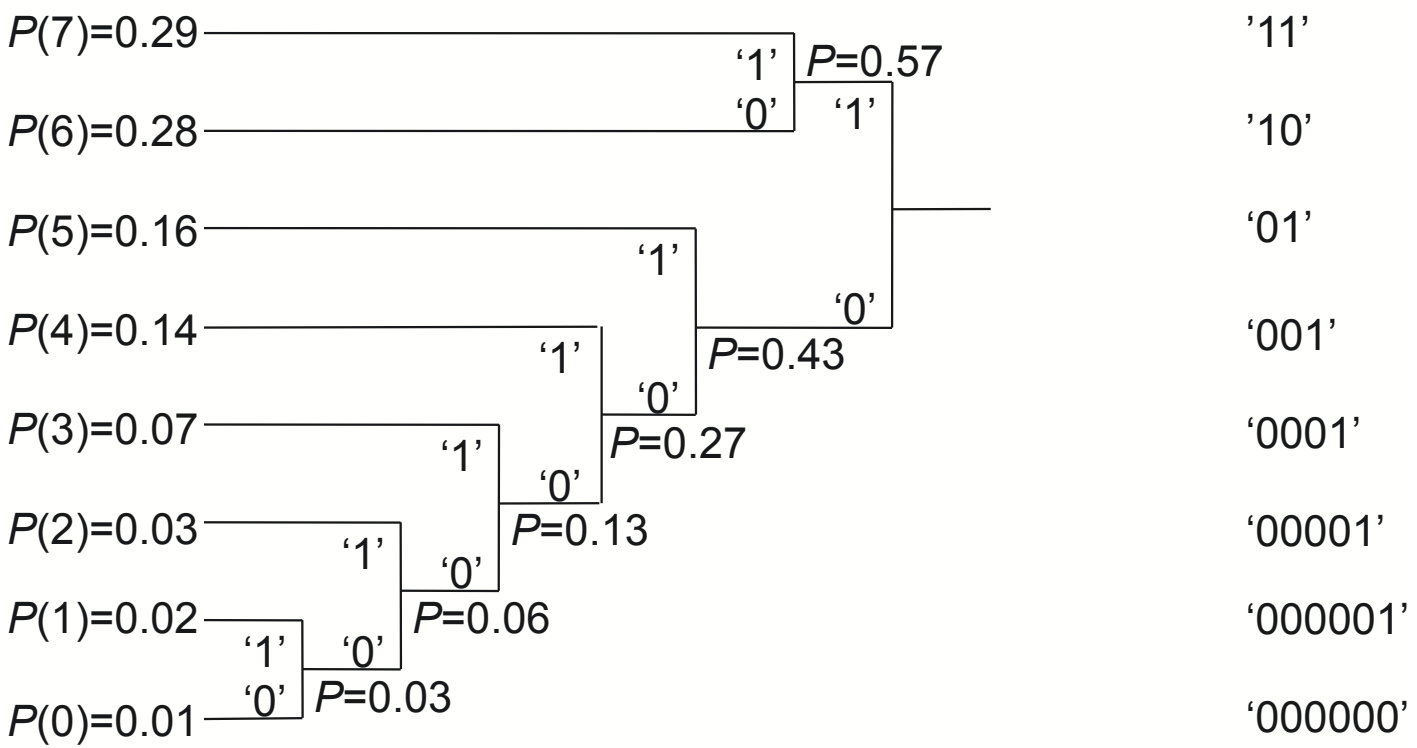
The Huffman Algorithm

- Now the question arises how to generate such a prefix code
- The answer to this question was given by D. A. HUFFMAN in 1952 [Huffman, 1952]
- The so-called Huffman algorithm always finds a prefix-free code with minimum redundancy
- For a proof that Huffman codes are optimal instantaneous codes (with minimum expected length), see [Cover and Thomas, 2006, p. 124ff]

The code tree is obtained as follows:

1. Given an ensemble representing a memoryless discrete source
2. Pick the two symbols with lowest probabilities and merge them into a new auxiliary symbol and calculate its probability
3. If more than one symbol remains, repeat the previous step
4. Convert the code tree into a prefix code

Example for the design of a Huffman code



Conditional Huffman Codes

- Random process $\{S_n\}$ with memory: design VLC for conditional pmf
- Example:
  - Stationary discrete Markov process, $\mathcal{A} = \{a_0, a_1, a_2\}$
  - Conditional pmfs $p(a|a_k) = P(S_n = a \mid S_{n-1} = a_k)$ with $k = 0, 1, 2$

a	a_0	a_1	a_2	entropy
$p(a\|a_0)$	0.90	0.05	0.05	$H(S_n\|a_0) = 0.5690$
$p(a\|a_1)$	0.15	0.80	0.05	$H(S_n\|a_1) = 0.8842$
$p(a\|a_2)$	0.25	0.15	0.60	$H(S_n\|a_2) = 1.3527$
$p(a)$	$0.6\overline{4}$	$0.2\overline{4}$	$0.\overline{1}$	$H(S) = 1.2575$

- Design Huffman code for conditional pmfs

a_i	Huffman codes for conditional pmfs			Huffman code for marginal pmf
	$S_{n-1} = a_0$	$S_{n-1} = a_1$	$S_{n-1} = a_2$	
a_0	1	00	00	1
a_1	00	1	01	00
a_2	01	01	1	01
	$\bar{\ell}_0 = 1.1$	$\bar{\ell}_1 = 1.2$	$\bar{\ell}_2 = 1.4$	$\bar{\ell} = 1.3556$

Average Codeword Length of Conditional Huffman Codes

- Average codeword length $\bar{\ell}_k = \bar{\ell}(S_{n-1}=a_k)$ is bounded by

$$H(S_n|a_k) \leq \bar{\ell}_k < H(S_n|a_k) + 1 \tag{21}$$

with *conditional entropy* of S_n given event $\{S_{n-1}=a_k\}$

$$H(S_n|a_k) = H(S_n|S_{n-1}=a_k) = -\sum_{i=0}^{M-1} p(a_i|a_k)\, \log_2 p(a_i|a_k) \tag{22}$$

- Resulting average codeword length

$$\bar{\ell} = \sum_{k=0}^{M-1} p(a_k)\, \bar{\ell}_k \tag{23}$$

- *Conditional entropy* $H(S_n|S_{n-1})$ of S_n given random variable S_{n-1}

$$H(S_n|S_{n-1}) = E\{-\log_2 p(S_n|S_{n-1})\} = \sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) \tag{24}$$

Conditioning may reduce lower bound on transmission bit rate

- Minimum average codeword length $\bar{\ell}_{\min}$

$$\boxed{H(S_n|S_{n-1}) \leq \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1} \tag{25}$$

- Conditioning May Lower Bit Rate

$$\begin{aligned} H(S) - H(S_n|S_{n-1}) &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\Big(\log_2 p(a_i) - \log_2 p(a_i|a_k)\Big) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\, \log_2 \frac{p(a_i)\, p(a_k)}{p(a_i, a_k)} \\ &\geq 0 \end{aligned} \tag{26}$$

- In the example:
  - No conditioning: $H(S) = 1.2575$, $\ell_{min} = 1.3556$
  - Conditioning: $H(S_n|S_{n-1}) = 0.7331$, $\ell_{min} = 1.1578$

Huffman Coding of Fixed-Length Vectors

- Stationary discrete random sources $\boldsymbol{S} = \{S_n\}$ with an M-ary alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$
- N symbols are coded jointly: design Huffman code joint pmf $p(a_0, \cdots, a_{N-1}) = P(S_n = a_0, \cdots, S_{n+N-1} = a_{N-1})$
- Average codeword length $\bar{\ell}_{\min}$ per symbol is bounded

$$\frac{H(S_n, \cdots, S_{n+N-1})}{N} \leq \bar{\ell}_{\min} < \frac{H(S_n, \cdots, S_{n+N-1})}{N} + \frac{1}{N} \tag{27}$$

where the *block entropy* is defined by

$$H(S_n, \cdots, S_{n+N-1}) = E\{-\log_2 p(S_n, \cdots, S_{n+N-1})\} \tag{28}$$

- The following limit is called *entropy rate*

$$\boxed{\bar{H}(\boldsymbol{S}) = \lim_{N \to \infty} \frac{H(S_0, \cdots, S_{N-1})}{N}} \tag{29}$$

Entropy Rate

- Entropy rate

$$\boxed{\bar{H}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{H(S_0, \cdots, S_{N-1})}{N}} \tag{30}$$

- The limit in (30) always exists for stationary sources [Gallager, 1968]
- Entropy rate $\bar{H}(\boldsymbol{S})$: greatest lower bound for the average codeword length $\bar{\ell}$ per symbol

$$\boxed{\bar{\ell} \geq \bar{H}(\boldsymbol{S})} \tag{31}$$

- Entropy rate for iid processes

$$\begin{aligned} \bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\left\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\right\}}{N} \\ &= \lim_{N\to\infty} \frac{\sum_{n=0}^{N-1} E\left\{-\log_2 p(S_n)\right\}}{N} = H(S) \end{aligned} \tag{32}$$

Entropy Rate for Markov Processes

- Entropy rate for stationary Markov processes

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\left\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\right\}}{N} \\ &= \lim_{N\to\infty} \frac{E\left\{-\log_2 p(S_0)\right\} + \sum_{n=1}^{N-1} E\left\{-\log_2 p(S_n|S_{n-1})\right\}}{N} \\ &= H(S_n|S_{n+1}) \end{aligned} \tag{33}$$

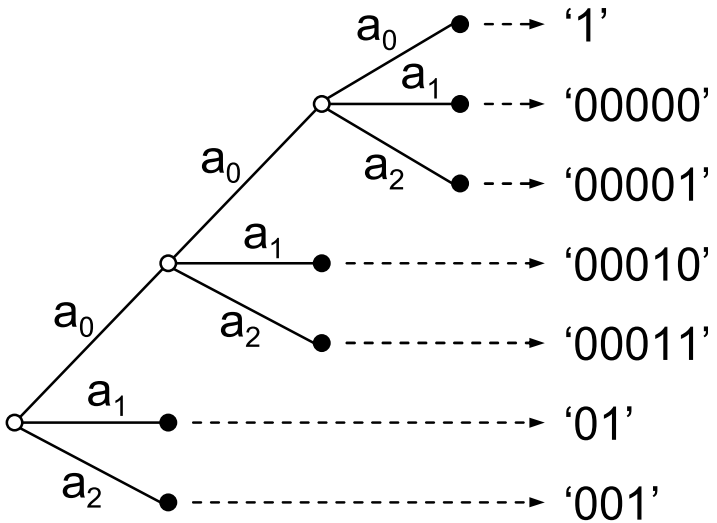
- Example: joint Huffman coding of 2 events and $\bar{\ell}$ vs. table size N_C

$a_i a_k$	$p(a_i, a_k)$	codewords
a_0a_0	0.58	1
a_0a_1	$0.03\overline{2}$	00001
a_0a_2	$0.03\overline{2}$	00010
a_1a_0	$0.03\overline{6}$	0010
a_1a_1	$0.19\overline{5}$	01
a_1a_2	$0.01\overline{2}$	000000
a_2a_0	$0.02\overline{7}$	00011
a_2a_1	$0.01\overline{7}$	000001
a_2a_2	$0.0\overline{6}$	0011

N	$\bar{\ell}$	N_C
1	1.3556	3
2	1.0094	9
3	0.9150	27
4	0.8690	81
5	0.8462	243
6	0.8299	729
7	0.8153	2187
8	0.8027	6561
9	0.7940	19683

Huffman Codes for Variable-Length Vectors

- Assign codewords to variable-length vectors: V2V codes



- Associate each leaf node $\mathcal{L}_k$ of the symbol tree with a codeword
- Use pmf of leaf nodes $p(\mathcal{L}_k)$ for Huffman design
- Average number of bits per alphabet letter

$$\bar{\ell} = \frac{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, \ell_k}{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, N_k} \tag{34}$$

where N_k denotes number of alphabet letters associated with $\mathcal{L}_k$

V2V Code Performance

- Example Markov process: $H(S_n|S_{n-1}) = 0.7331$
- Faster reduction of $\bar{\ell}$ with increasing N_C compared to fixed-length vector Huffman coding

$\boldsymbol{a}_k$	$p(\mathcal{L}_k)$	codewords
a_0a_0	0.5799	1
a_0a_1	0.0322	00001
a_0a_2	0.0322	00010
a_1a_0	0.0277	00011
$a_1a_1a_0$	0.0222	000001
$a_1a_1a_1$	0.1183	001
$a_1a_1a_2$	0.0074	0000000
a_1a_2	0.0093	0000001
a_2	0.1708	01

N_C	$\bar{\ell}$
5	1.1784
7	1.0551
9	1.0049
11	0.9733
13	0.9412
15	0.9293
17	0.9074
19	0.8980
21	0.8891

Elias Coding and Arithmetic Coding

- Drawbacks of block Huffman codes: large table sizes
- Another class of uniquely decodable codes are Elias codes and Arithmetic codes
- Mapping of a string of N symbols $\boldsymbol{s} = \{s_0, s_1, ..., s_{N-1}\}$ onto a string of M bits $\boldsymbol{b} = \{b_0, b_1, ..., b_{M-1}\}$

$$\gamma : \boldsymbol{s} \rightarrow \boldsymbol{b} \tag{35}$$

- Stream decoding or parsing maps the bit string onto the string of symbols

$$\gamma^{-1} : \boldsymbol{b} \rightarrow \boldsymbol{s} \tag{36}$$

- Complexity of code construction: linear per symbol

Mapping of the String into a Number

- Consider $\boldsymbol{S} = \{S_0, S_1, \ldots, S_{N-1}\}$, each S_i with alphabet of size K_i
- A realization of $\boldsymbol{S}$: $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$ can be represented by a unqiue real number $v \in [0, 1)$

$$v = \zeta(\boldsymbol{s}) = \sum_{i=0}^{N-1} s_i B_i \qquad \text{with} \qquad B_i = \prod_{j=0}^{i} K_j^{-1} \tag{37}$$

Note that when all $K_j = K$, the basis simplifies to $B_i = K^{-i-1}$

- Examples

$$\boldsymbol{s} = 3_{10}, 1_{10}, 2_{10}, 1_{10} \quad \rightarrow \quad v = \frac{3}{10} + \frac{1}{10 \cdot 10} + \frac{2}{10^3} + \frac{1}{10^4} = 0.3121_{10}$$

$$\boldsymbol{s} = 3_5, 1_2, 2_3, 1_2 \quad \rightarrow \quad v = \frac{3}{5} + \frac{1}{5 \cdot 2} + \frac{2}{5 \cdot 2 \cdot 3} + \frac{1}{5 \cdot 2 \cdot 3 \cdot 2} = 0.7667_{10}$$

- Assuming $v_a = \zeta(\boldsymbol{s}_a)$ and $v_b = \zeta(\boldsymbol{s}_b)$

$$\boldsymbol{s}_a > \boldsymbol{s}_b \qquad \text{if} \qquad v_a > v_b \tag{38}$$

Mapping Symbol Sequences to Intervals

- Joint pmf

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(S_0=s_0, S_1=s_1, \cdots, S_{N-1}=s_{N-1}) \tag{39}$$

- Using (38), pmf of $\boldsymbol{s}$ can be written

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(\boldsymbol{S}\leq\boldsymbol{s}) - P(\boldsymbol{S}<\boldsymbol{s}) \tag{40}$$

- Mapping of $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$ to half-open interval $\mathcal{I}_N \subset [0,1)$

$$\boxed{\mathcal{I}_N(\boldsymbol{s}) = [L_N, L_N+W_N) = \big[P(\boldsymbol{S}<\boldsymbol{s}),\ P(\boldsymbol{S}\leq\boldsymbol{s})\big)} \tag{41}$$

with

$$L_N = P(\boldsymbol{S}<\boldsymbol{s}) \tag{42}$$
$$W_N = P(\boldsymbol{S}=\boldsymbol{s}) = p(\boldsymbol{s}) \tag{43}$$

Unique Identification: The Intervals are Disjoint

- Proof by mapping of $\boldsymbol{s}^x$ onto $\mathcal{I}_N^x = [P(\boldsymbol{S} < \boldsymbol{s}^x), P(\boldsymbol{S} \leq \boldsymbol{s}^x))$ with $x = a$ or $x = b$
- Assuming $\boldsymbol{s}_a < \boldsymbol{s}_b$, it follows

$$\begin{aligned} L_N^b &= P(\boldsymbol{S} < \boldsymbol{s}_b) \\ &= P(\{\boldsymbol{S} \leq \boldsymbol{s}_a\} \cup \{\boldsymbol{s}_a < \boldsymbol{S} \leq \boldsymbol{s}_b\}) \\ &= P(\boldsymbol{S} \leq \boldsymbol{s}_a) + \underbrace{P(\boldsymbol{S} > \boldsymbol{s}_a,\, \boldsymbol{S} < \boldsymbol{s}_b)}_{\geq 0} \\ &\geq P(\boldsymbol{S} \leq \boldsymbol{s}_a) = L_N^a + W_N^a \end{aligned} \tag{44}$$

$\rightarrow$ Intervals $\mathcal{I}_N^a$ and $\mathcal{I}_N^b$ do not overlap: any number in the interval $\mathcal{I}_N(\boldsymbol{s})$ uniquely identifies $\boldsymbol{s}$

How Many Bits To Identify an Interval?

- Symbol sequence $\boldsymbol{s}$ in interval $\mathcal{I}_N(\boldsymbol{s})$ is to be transmitted
- How many bits $b_0, b_1, \ldots, b_{K-1}$ to uniquely identify $\mathcal{I}_N(\boldsymbol{s})$?

$$v = \sum_{i=0}^{m-1} b_i 2^{i-1} = 0.b_0 b_1 \cdots b_{K-1} \in \mathcal{I}_N(\boldsymbol{s}) \tag{45}$$

- Size of interval ($= p(\boldsymbol{s})$) governs number m of bits that are needed to identify the interval

$$\begin{aligned} p(\boldsymbol{s}) = 1/2 \quad &\rightarrow \quad \mathcal{B} = \{.0, .1\} \\ p(\boldsymbol{s}) = 1/4 \quad &\rightarrow \quad \mathcal{B} = \{.00, .01, .10, .11\} \\ p(\boldsymbol{s}) = 1/8 \quad &\rightarrow \quad \mathcal{B} = \{.000, .001, .010, .011, .100, .101, .110, .111\} \end{aligned}$$

- Minimum number of bits is

$$K = K(\boldsymbol{s}) = \lceil -\log_2 p(\boldsymbol{s}) \rceil \tag{46}$$

with the binary number v identifying the interval $\mathcal{I}_n$ being

$$v = \lceil L_N \cdot 2^K \rceil \cdot 2^{-K} \tag{47}$$

Analytical Derivation of Number of Bits and Bitstring

- Binary number v identifying the interval $\mathcal{I}_n$

$$v = \lceil L_N \cdot 2^K \rceil \cdot 2^{-K} \tag{48}$$

- With $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$, we obtain

$$L_N \leq v < L_N + 2^{-K} \tag{49}$$

- Using the expression for the required number of bits

$$K \geq -\log_2 p(\boldsymbol{s}) \quad \rightarrow \quad 2^{-K} \leq p(\boldsymbol{s}) = W_N \tag{50}$$

yields

$$L_N \leq v < L_N + W_N \tag{51}$$

- Hence, the representative $v = 0.b_0 b_1 \ldots b_{m-1}$ always lies inside the interval $\mathcal{I}_N(\boldsymbol{s})$

$\rightarrow$ Message $\boldsymbol{s}$ can be uniquely decoded from the transmitted bit string $\boldsymbol{b} = \{b_0, b_1, \ldots, b_{K-1}\}$ of $K(\boldsymbol{s}) = \lceil -\log_2 p(\boldsymbol{s}) \rceil$ bits

Redundancy of Elias Coding

- Average codeword length per symbol

$$\bar{\ell} = \frac{1}{N} E\{K(\boldsymbol{S})\} = \frac{1}{N} E\left\{\left\lceil -\log_2 p(\boldsymbol{S})\right\rceil\right\} \tag{52}$$

- Applying inequalities $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$, we obtain

$$\frac{1}{N} E\{-\log_2 p(\boldsymbol{S})\} \leq \bar{\ell} < \frac{1}{N} E\{1 - \log_2 p(\boldsymbol{S})\} \tag{53}$$

- Average codeword length is bounded

$$\boxed{\frac{1}{N} H(\boldsymbol{S}) \leq \bar{\ell} \leq \frac{1}{N} H(\boldsymbol{S}) + \frac{1}{N}} \tag{54}$$

Derivation of Iterative Algorithm for Elias Coding I

- Iterative construction of codewords
- Consider sub-sequences $\boldsymbol{s}^{(n)} = \{s_0, s_1, \cdots, s_{n-1}\}$ with $1 \leq n \leq N$
- Interval width W_{n+1} for the sub-sequence $\boldsymbol{s}^{(n+1)} = \{\boldsymbol{s}^{(n)}, s_n\}$

$$\begin{aligned} W_{n+1} &= P\big(\boldsymbol{S}^{(n+1)} = \boldsymbol{s}^{(n+1)}\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\ S_n = s_n\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \cdot P\big(S_n = s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \end{aligned}$$

- Iteration rule of interval width

$$\boxed{W_{n+1} = W_n \cdot p(s_n \,|\, s_0, s_1, \ldots, s_{n-1})} \tag{55}$$

- Since $p(s_n \,|\, s_0, s_1, \ldots, s_{n-1}) \leq 1$, it follows

$$W_{n+1} \leq W_n \tag{56}$$

Derivation of Iterative Algorithm for Elias Coding II

- Derivation for lower interval border L_{n+1} for the sub-sequence $\boldsymbol{s}^{(n+1)} = \{\boldsymbol{s}^{(n)}, s_n\}$

$$\begin{aligned} L_{n+1} &= P\left(\boldsymbol{S}^{(n+1)} < \boldsymbol{s}^{(n+1)}\right) \\ &= P\left(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\right) + P\left(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\, S_n < s_n\right) \\ &= P\left(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\right) + P\left(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\right) \cdot P\left(S_n < s_n \,\middle|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\right) \end{aligned}$$

- Iteration rule of lower interval boundary

$$\boxed{L_{n+1} = L_n + W_n \cdot c(s_n \mid s_0, s_1, \ldots, s_{n-1})} \tag{57}$$

with the cmf $c(\cdot)$ being defined as

$$c(s_n \mid s_0, s_1, \ldots, s_{n-1}) = \sum_{\forall a \in \mathcal{A}_n : a < s_n} p(a \mid s_0, s_1, \ldots, s_{n-1}) \tag{58}$$

- Since $W_n \cdot c(s_n \mid s_0, s_1, \ldots, s_{n-1}) \geq 0$, it follows

$$L_{n+1} \geq L_n \tag{59}$$

Elias Coding for Different Sources

- Derivation above for general case of dependent and differently distributed random variables
- For i.i.d. sources, interval refinement can be simplified

$$W_n = W_{n-1} \cdot p(s_{n-1}) \quad (60)$$
$$L_n = L_{n-1} + W_{n-1} \cdot c(s_{n-1}) \quad (61)$$

- For 1-st order Markov sources: conditional pmf $p(s_n|s_{n-1})$ and conditional cmf $c(s_n|s_{n-1})$

$$W_n = W_{n-1} \cdot p(s_{n-1}|s_{n-2}) \quad (62)$$
$$L_n = L_{n-1} + W_{n-1} \cdot c(s_{n-1}|s_{n-2}) \quad (63)$$

- For non-stationary sources, the probability $p(\cdot)$ needs to be adapted

Example: iid Source

- Example for an iid source for which an optimum Huffman code exists

symbol a_k	pmf $p(a_k)$	Huffman code	cmf $c(a_k)$
a_0='A'	$0.25 = 2^{-2}$	00	$0.00 = 0$
a_1='B'	$0.25 = 2^{-2}$	01	$0.25 = 2^{-2}$
a_2='C'	$0.50 = 2^{-1}$	1	$0.50 = 2^{-1}$

- Suppose we intend to send the symbol string $\boldsymbol{s} =' CABAC'$
- Using the Huffman code, the bit string would be $\boldsymbol{b} = 10001001$
- An alternative to Huffman coding is interval subdivision in stream entropy coding
- $p(\boldsymbol{s}) = p('C') \cdot p('A') \cdot p('B') \cdot p('A') \cdot p('C') = \frac{1}{2}\frac{1}{4}\frac{1}{4}\frac{1}{4}\frac{1}{2} = \frac{1}{256}$
- The size of the bit stream is $-\log_2 p(\boldsymbol{s}) = 8$ bits

Encoding Algorithm for Elias Codes

Encoding algorithm:

1. Given is a sequence $\{s_0, \cdots, s_{N-1}\}$ of N symbols
2. Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$
3. For each $n = 0, 1, \cdots, N-1$, determine the interval $\mathcal{I}_{n+1}$ by

$$\begin{aligned} W_{n+1} &= W_n \cdot p(s_n | s_0, \cdots, s_{n-1}) \\ L_{n+1} &= L_n + W_n \cdot c(s_n | s_0, \cdots, s_{n-1}) \end{aligned}$$

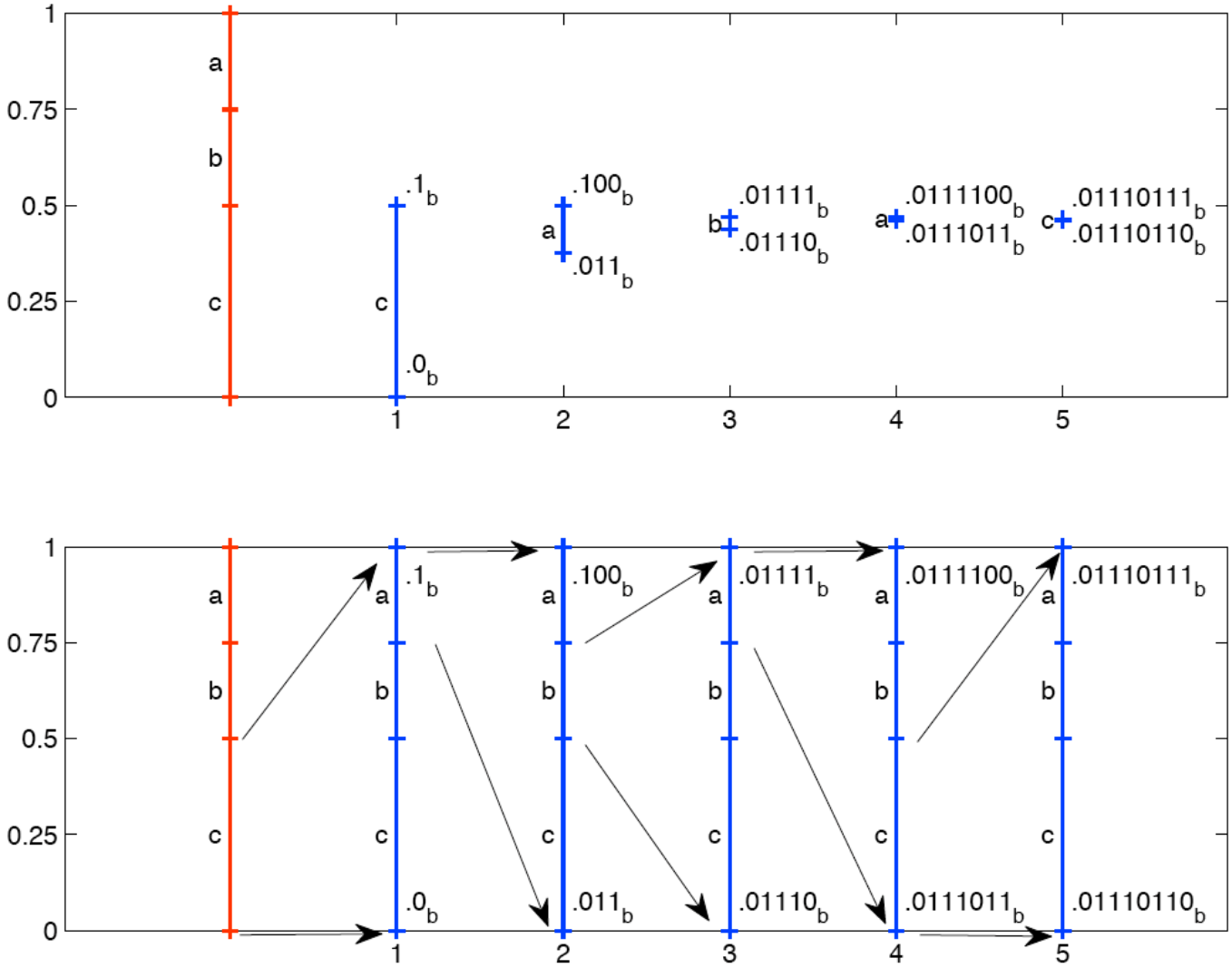
4. Determine the codeword length by $K = \lceil -\log_2 W_N \rceil$
5. Transmit the codeword $\boldsymbol{b}^{(K)}$ of K bits that represents the fractional part of $v = \lceil L_N \, 2^K \rceil \, 2^{-K}$

Example for Elias encoding

s_0='C'	s_1='A'	s_2='B'
$W_1 = W_0 \cdot p(\text{'C'}) = 1 \cdot 2^{-1} = 2^{-1} = (0.1)_b$	$W_2 = W_1 \cdot p(\text{'A'}) = 2^{-1} \cdot 2^{-2} = 2^{-3} = (0.001)_b$	$W_3 = W_2 \cdot p(\text{'B'}) = 2^{-3} \cdot 2^{-2} = 2^{-5} = (0.00001)_b$
$L_1 = L_0 + W_0 \cdot c(\text{'C'}) = L_0 + 1 \cdot 2^{-1} = 2^{-1} = (0.1)_b$	$L_2 = L_1 + W_1 \cdot c(\text{'A'}) = L_1 + 2^{-1} \cdot 0 = 2^{-1} = (0.100)_b$	$L_3 = L_2 + W_2 \cdot c(\text{'B'}) = L_2 + 2^{-3} \cdot 2^{-2} = 2^{-1} + 2^{-5} = (0.10001)_b$

s_3='A'	s_4='C'	termination
$W_4 = W_3 \cdot p(\text{'A'}) = 2^{-5} \cdot 2^{-2} = 2^{-7} = (0.0000001)_b$	$W_5 = W_4 \cdot p(\text{'C'}) = 2^{-7} \cdot 2^{-1} = 2^{-8} = (0.00000001)_b$	$K = \lceil -\log_2 W_5 \rceil = 8$
$L_4 = L_3 + W_3 \cdot c(\text{'A'}) = L_3 + 2^{-5} \cdot 0 = 2^{-1} + 2^{-5} = (0.1000100)_b$	$L_5 = L_4 + W_4 \cdot c(\text{'C'}) = L_4 + 2^{-7} \cdot 2^{-1} = 2^{-1} + 2^{-5} + 2^{-8} = (0.10001001)_b$	$v = \lceil L_5 \, 2^K \rceil \, 2^{-K} = 2^{-1} + 2^{-5} + 2^{-8}$ $\boldsymbol{b} = \text{'}\mathbf{10001001}\text{'}$

Illustration of Iteration



Decoding Algorithm for Elias codes

Decoding algorithm:

1. Given is the number N of symbols to be decoded and a codeword $\boldsymbol{b}^{(K)} = \{b_0, \cdots, b_{K-1}\}$ of K_N bits
2. Determine the interval representative v according to

$$v = \sum_{i=0}^{K-1} b_i \, 2^{-i}$$

3. Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$
4. For each $n = 0, 1, \cdots, N-1$, do the following:
   1. For each $a_i \in \mathcal{A}_n$, determine the interval $\mathcal{I}_{n+1}(a_i)$ by

$$\begin{aligned} W_{n+1}(a_i) &= W_n \cdot p(a_i|s_0, \ldots, s_{n-1}) \\ L_{n+1}(a_i) &= L_n + W_n \cdot c(a_i|s_0, \ldots, s_{n-1}) \end{aligned}$$

   2. Select the letter $a_i \in \mathcal{A}_n$ for which $v \in \mathcal{I}_{n+1}(a_i)$ and set $s_n = a_i$, $W_{n+1} = W_{n+1}(a_i)$, $L_{n+1} = L_{n+1}(a_i)$

Arithmetic Coding I

- Problem with Elias codes: precision requirement for W_N and L_N
- *Arithmetic codes*: variant of Elias codes with fixed-precision integer arithmetic
- Represent pmfs $p(a)$ and cmfs $c(a)$ by V-bit positive integers $p_V(a)$ and $c_V(a)$

$$p(a) = p_V(a) \cdot 2^{-V} \quad c(a) = c_V(a) \cdot 2^{-V} = \sum_{a_i < a} p_V(a_i) \cdot 2^{-V} \tag{64}$$

- Elias code remains decodable if intervals are always nested

$$0 < W_{n+1} \leq W_n \cdot p(s_n) \tag{65}$$

$\rightarrow$ Rounding down of $W_n \cdot p(s_n)$ at each iteration to keep fixed-length integer arithmetic (results in rate increase)

Arithmetic Coding II

- Use U-bit integer A_n and integer $z_n \geq U$

$$W_n = A_n \cdot 2^{-z_n} \tag{66}$$

- Approximate $W_0 = 1$ by $A_0 = 2^U - 1$ and $z_0 = U$
- Interval refinement for integer arithmetic

$$\begin{aligned} W_{n+1} &= W_n \cdot p(s_n) = A_n \cdot 2^{-z_n} \cdot p_V(s_n) \cdot 2^{-V} && (67)\\ \rightarrow A_{n+1} &= \left\lfloor A_n \cdot p_V(s_n) \cdot 2^{-y_n} \right\rfloor && (68)\\ \rightarrow z_{n+1} &= z_n + V - y_n && (69) \end{aligned}$$

- Restriction of A_n to

$$\underbrace{2^{U-1}}_{\text{max. precision}} \leq A_n < \underbrace{2^U}_{W_n<1} \tag{70}$$

results in

$$y_n = \left\lceil \log_2(\, A_n \cdot p_V(s_n) + 1) \right\rceil - U \tag{71}$$

Arithmetic Coding III

- Representation of the product $W_n \cdot c(s_n)$:
  first $z_n - U$ bits are zero, following $U+V$ bits represent $A_n \cdot c_V(s_n)$

$$\begin{aligned} W_n \cdot c(s_n) &= A_n \cdot c_V(s_n) \cdot 2^{-(z_n+V)} \\ &= 0.\underbrace{00000\cdots0}_{z_n-U \text{ bits}}\ \underbrace{xxxx\cdots x}_{U+V \text{ bits}}\ 00000\cdots \end{aligned}$$

- Representation of lower interval boundary

$$L_n = 0.\underbrace{aaaaa\cdots a}_{\substack{z_n-c_n-U \\ \textit{settled bits}}}\ \underbrace{0111111\cdots1}_{\substack{c_n \\ \textit{outstanding bits}}}\ \underbrace{xxxx\cdots x}_{\substack{U+V \\ \textit{active bits}}}\ \underbrace{00000\cdots}_{\textit{trailing bits}}$$

- *trailing bits*: equal to 0, but maybe changed later
- *active bits*: directly modified by the update $L_{n+1} = L_n + W_n \cdot c(s_n)$
- *outstanding bits*: may be modified by a carry from the active bits
- *settled bits*: not modified in any following interval update

Efficiency of Arithmetic Codes

- Fixed-length integer precision achieved by rounding down reduces W_n in each iteration
- Excess rate

$$\Delta\ell = \left\lceil -\log_2 W_N \right\rceil - \left\lceil -\log_2 p(\boldsymbol{s}) \right\rceil < 1 + \log_2 \frac{p(\boldsymbol{s})}{W_N} \tag{72}$$

- Upper bound for increase in codeword length per symbol

$$\Delta\bar{\ell} < \frac{1}{N} + \log_2\left(1 + 2^{1-U}\right) - \log_2\left(1 - \frac{2^{-V}}{p_{\min}}\right) \tag{73}$$

- Example:
  - number of coded symbols $N = 1000$,
  - precision for representing probabilities $V = 16$,
  - precision for representing intervals $U = 12$,
  - minimum probablity $p_{\min} = 0.02$

  $\rightarrow$ Increase in codeword length is less than 0.003 bit per symbol

Binary Arithmetic Coding

- Complexity reduction: most popular type of arithmetic coding
- Binarization of $S \in \{a_0, a_1, \ldots, a_{M-1}\}$ produces $C \in \{0, 1\}$
- Example in table: unary truncated binarization

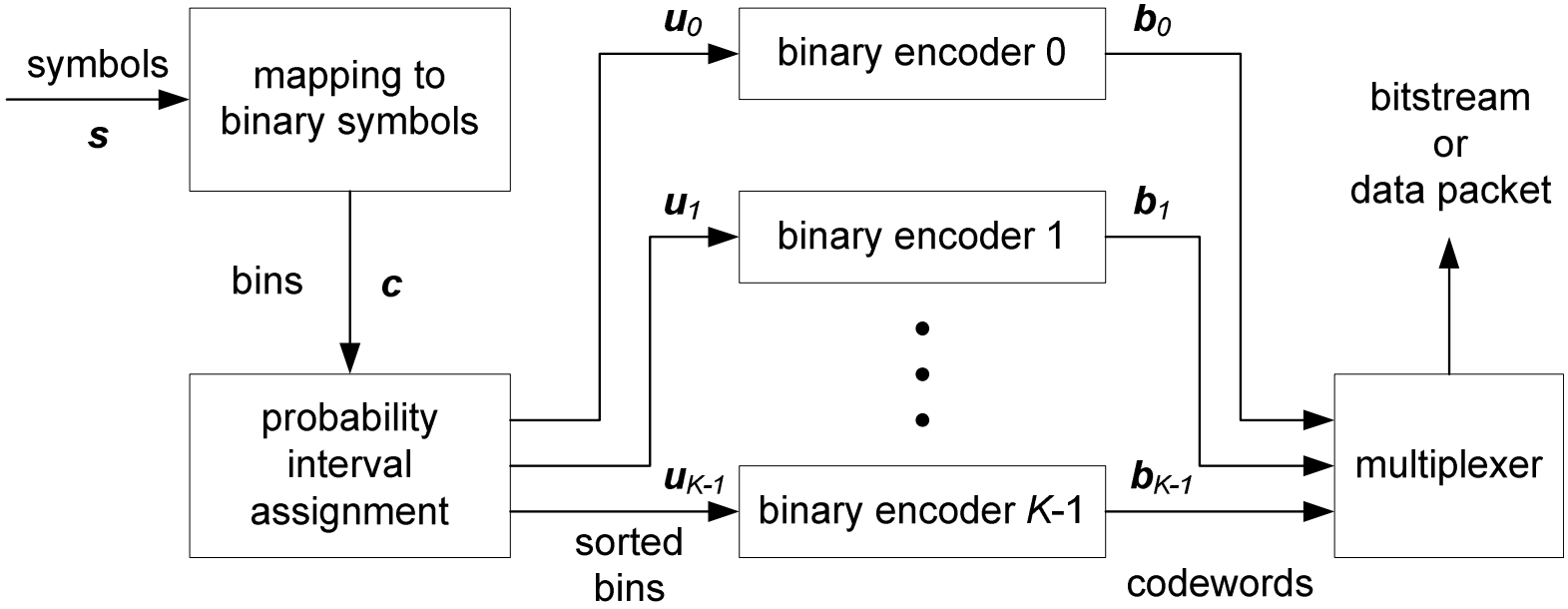
S_n	number of bins B	C_0	C_1	C_2	$\cdots$	C_{M-2}	C_{M-1}
a_0	1	1					
a_1	2	0	1				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$			
a_{M-2}	$M-2$	0	0	0	$\cdots$	0	1
a_{M-1}	$M-2$	0	0	0	$\cdots$	0	0

- Entropy and efficiency of coding unchanged due to binarization

$$\begin{aligned} H(\boldsymbol{S}) &= H(\boldsymbol{C}) \\ &= H(C_0) + H(C_1|C_0) + \ldots + H(C_{M-1}|C_0, C_1, \ldots, C_{M-2}) \end{aligned} \tag{74}$$

Probability Interval Partitioning Entropy (PIPE) Coding

- Alternative to arithmetic coding
- Quantization into probability intervals combined with variable-length coding of variable-length vectors



Algorithm for PIPE Coding

1. *Binarization*: The sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$ is converted into a sequence $\boldsymbol{c} = \{c_0, c_1, \cdots, c_{B-1}\}$ bins with pmfs; each bin c_i is characterized by a probability $P(C_i = 0)$
2. *Decomposition*: The bin sequence $\boldsymbol{c}$ is decomposed into U sub-sequences; a sub-sequence $\boldsymbol{u}_k$ contains the bins c_i with $P(C_i = 0) \in \mathcal{I}_k$ in the same order as in the bin sequence $\boldsymbol{c}$
3. *Binary Coding*: Each sub-sequence of bins $\boldsymbol{u}_k$ is coded using a distinct V2V code resulting in U codeword sequences $\boldsymbol{b}_k$
4. *Multiplexing*: The data packet is produced by multiplexing the U codeword sequences $\boldsymbol{b}_k$

Binarization: Bijective Mapping

1. Mapping of source symbols s_n to bin sequence

$$\boldsymbol{d}_n = (d_0^n, \ldots) = \gamma_d^n(s_n) \tag{75}$$

and $\boldsymbol{d}_n$ being concatenated to $\boldsymbol{d} = (d_0, d_1, \ldots, d_{B-1})$

2. Less probable bin value

$$v_{\mathrm{LPB}}^i = \begin{cases} 0, & \text{if } P(d_i = 0) \leq 0.5 \\ 1, & \text{if } P(d_i = 0) > 0.5 \end{cases} \tag{76}$$

and probability

$$p_{\mathrm{LPB}}^i = \begin{cases} P(d_i = 0), & \text{if } P(d_i = 0) \leq 0.5 \\ 1 - P(d_i = 0), & \text{if } P(d_i = 0) > 0.5 \end{cases} \tag{77}$$

3. Resulting expression for coding bins

$$b_i = d_i \oplus v_{\mathrm{LPB}}^i \tag{78}$$

provides the mapping $(s_0, s_1, .., s_{N-1}) \rightarrow (b_0, b_1, .., b_{B-1})$

Probability Interval Partitioning and Bin Assignment

- The binary values $b_i \in (b_0, b_1, .., b_{B-1})$ have associated probabilities p^i_{LPB} with

$$p^i_{\mathrm{LPB}} \in (0, 0.5] \tag{79}$$

- Partition the interval into K partitions such that

$$\bigcup_{k=0}^{K-1} \mathcal{I}_k = (0, 0.5] \quad \text{and} \quad \mathcal{I}_k \cap \mathcal{I}_j = \emptyset \ \text{ for } \ k \neq j \tag{80}$$

- Decompose $(b_0, b_1, .., b_{B-1})$ into K separate sequences

$$\boldsymbol{u}_k = (u^k_0, u^k_1, \ldots) = (b_i \ : \ b_i \in \boldsymbol{b}, \ p^i_{\mathrm{LPB}} \in \mathcal{I}_k) \tag{81}$$

What is the Loss of Probability Interval Partitioning?

- Represent all bins of interval $\mathcal{I}_k$ using one fixed probability $p_{\mathcal{I}_k} \in \mathcal{I}_k$
- Assuming optimal entropy coder for probability $p_{\mathcal{I}_k}$, rate when coding a bin with probability p is given as

$$\begin{aligned} R(p, p_{\mathcal{I}_k}) &= -p \ \log_2(p_{\mathcal{I}_k}) - (1-p) \log_2(1-p_{\mathcal{I}_k}) \\ &= H_b(p_{\mathcal{I}_k}) + (p - p_{\mathcal{I}_k}) \ H_b'(p_{\mathcal{I}_k}) \end{aligned} \tag{82}$$

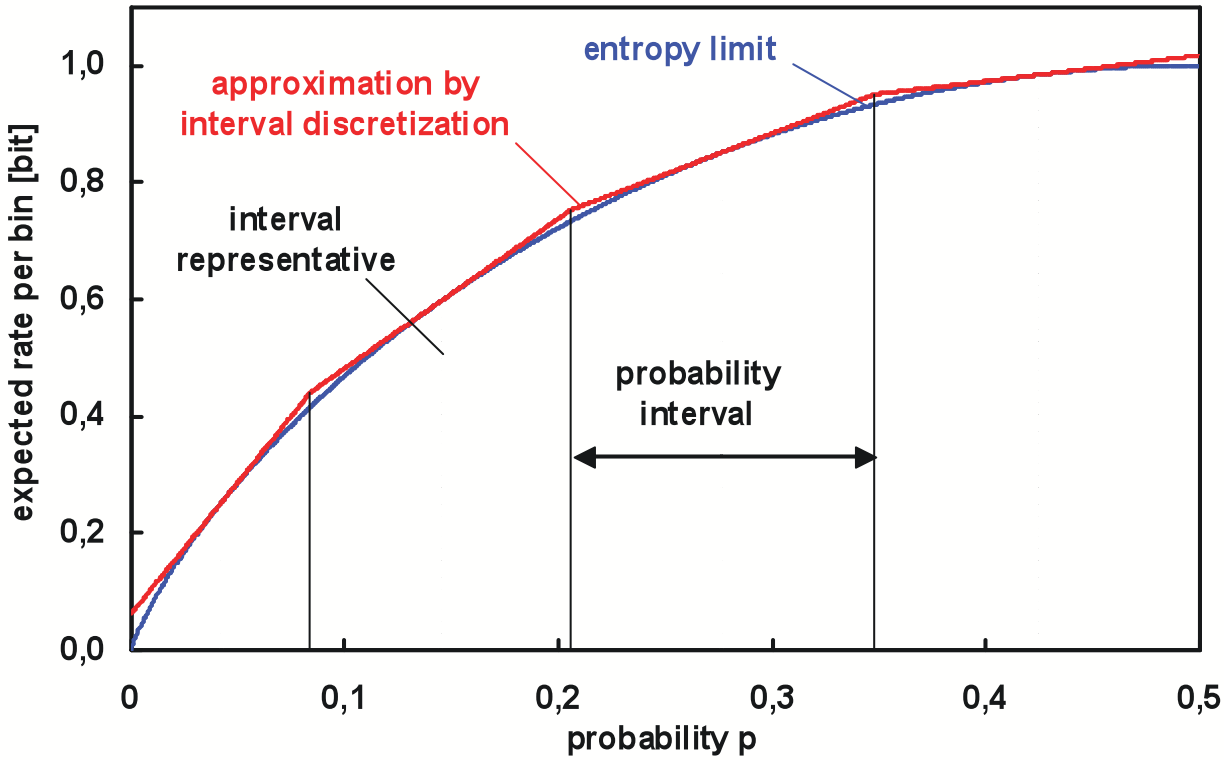
where $H_b(p)$ represents the binary entropy function

$$H_b(p) = -p \ \log_2 p - (1-p) \ \log_2(1-p) \tag{83}$$

and $H_b'(p)$ its first derivative

$$H_b'(p) = \log_2\left(\frac{1-p}{p}\right) \tag{84}$$

Example with 4 Intervals

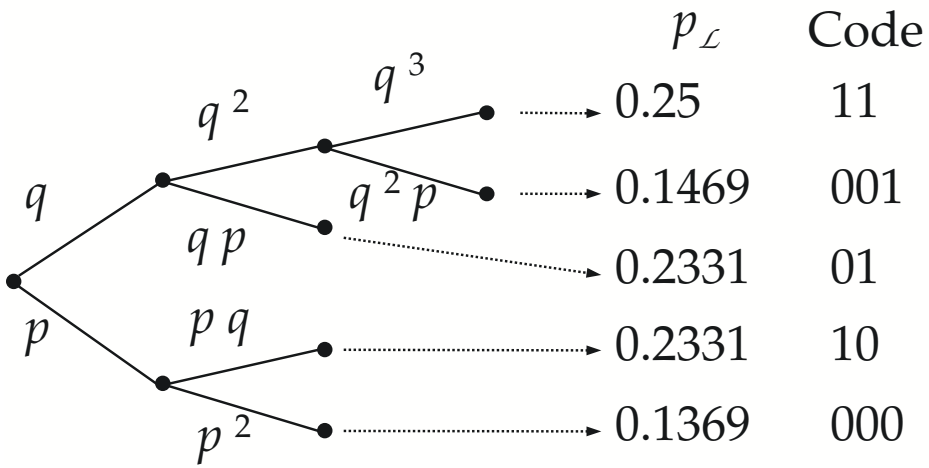


$K = 4$ intervals and p being uniformly distributed $(0, 0.5]$, redundancy is given

$$\varrho = \frac{E\{R(p, p_{\mathcal{I}_k})\}}{E\{H(p)\}} - 1 = 1\% \tag{85}$$

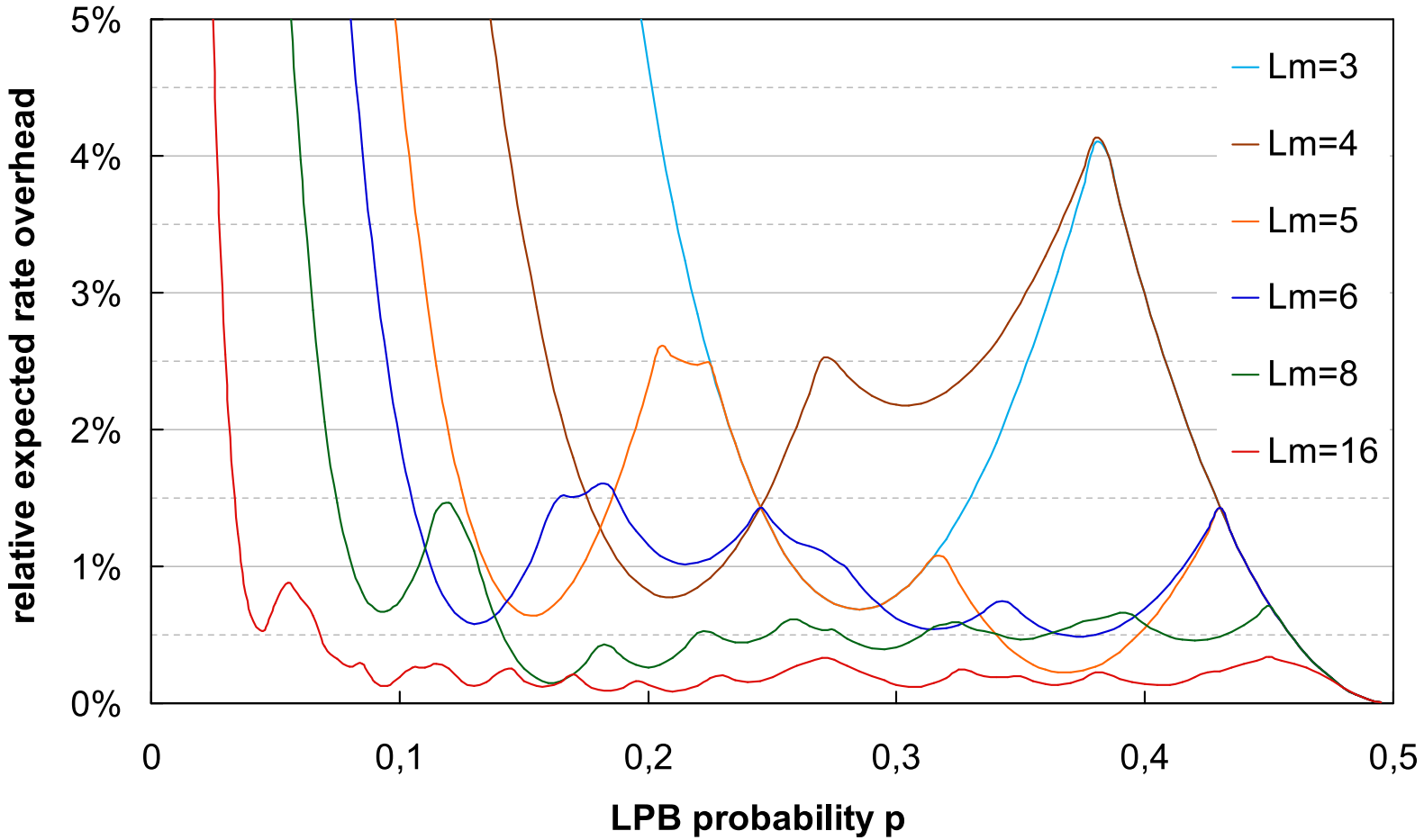
Entropy Coding for Each Probability Interval

- Assuming fixed probabilities $p = p_{\mathcal{I}_k}$ for each probability interval $\mathcal{I}_k$
- Entropy coding of the corresponding sub-sequences of coding bins $\boldsymbol{u}_k$: simplified binary arithmetic coding or variable length coding
- For variable length coding, create a variable number of bins to variable number of bits (V2V) code
- Example for $p = 1 - q = 0.37$, excess rate $\varrho = 0.2\%$

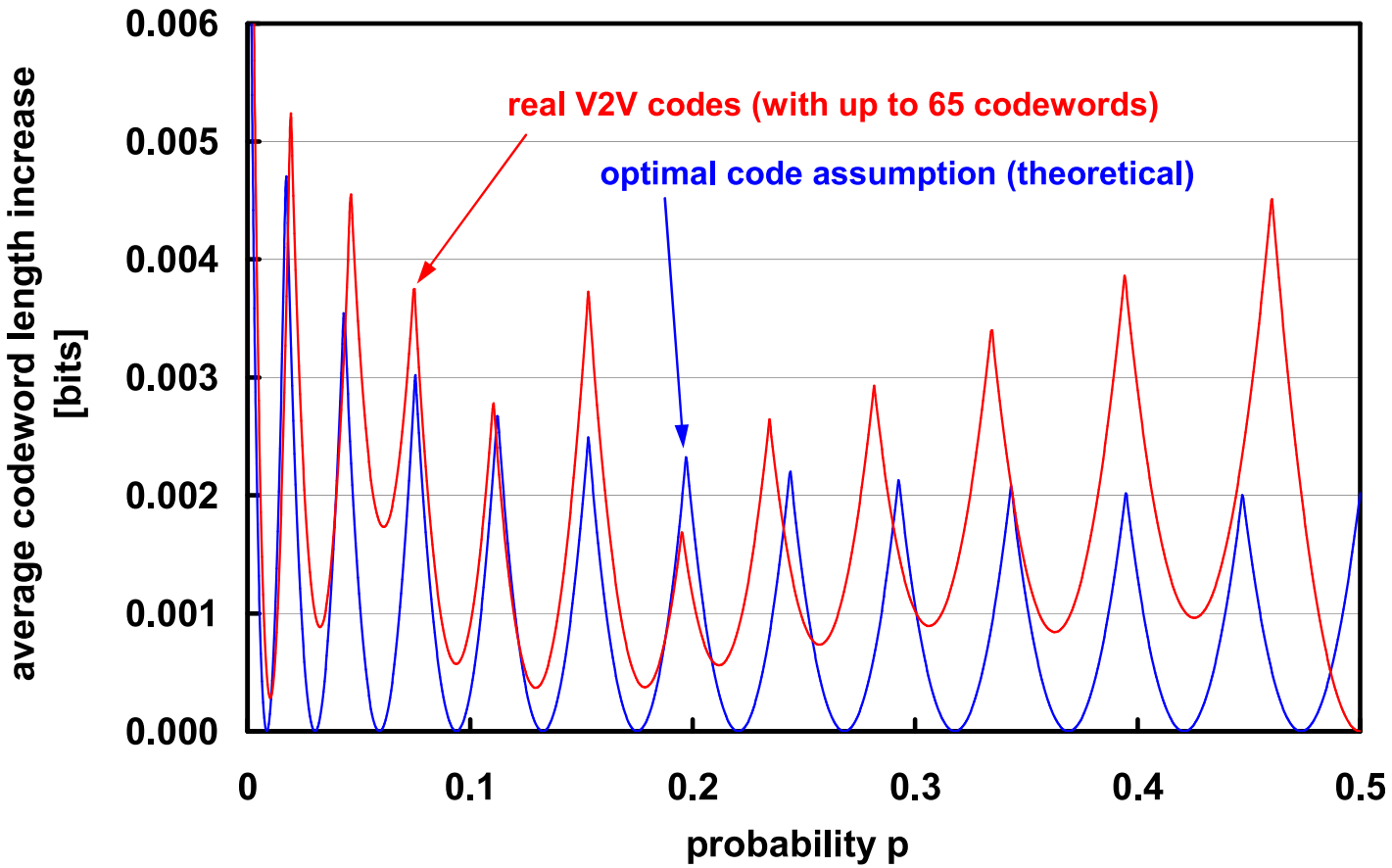


- Leaf probability $p_{\mathcal{L}}(j) = p^{x_j}\ (1-p)^{y_j}$

Excess Rate for Optimal V2V Codes



Combined Design



- Assuming uniform distribution of encoded probabilities
- Excess rate $\bar{\varrho}_{\mathrm{opt}} = 0.12\%$, $\bar{\varrho}_{\mathrm{V2V}} = 0.24\%$

Unique Decodability and Multiplexing

- PIPE coding: partition sequence of coding bins into K sub-sequences $\boldsymbol{u}_k$

$$\{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{K-1}\} = \gamma_m(\boldsymbol{b}) \tag{86}$$

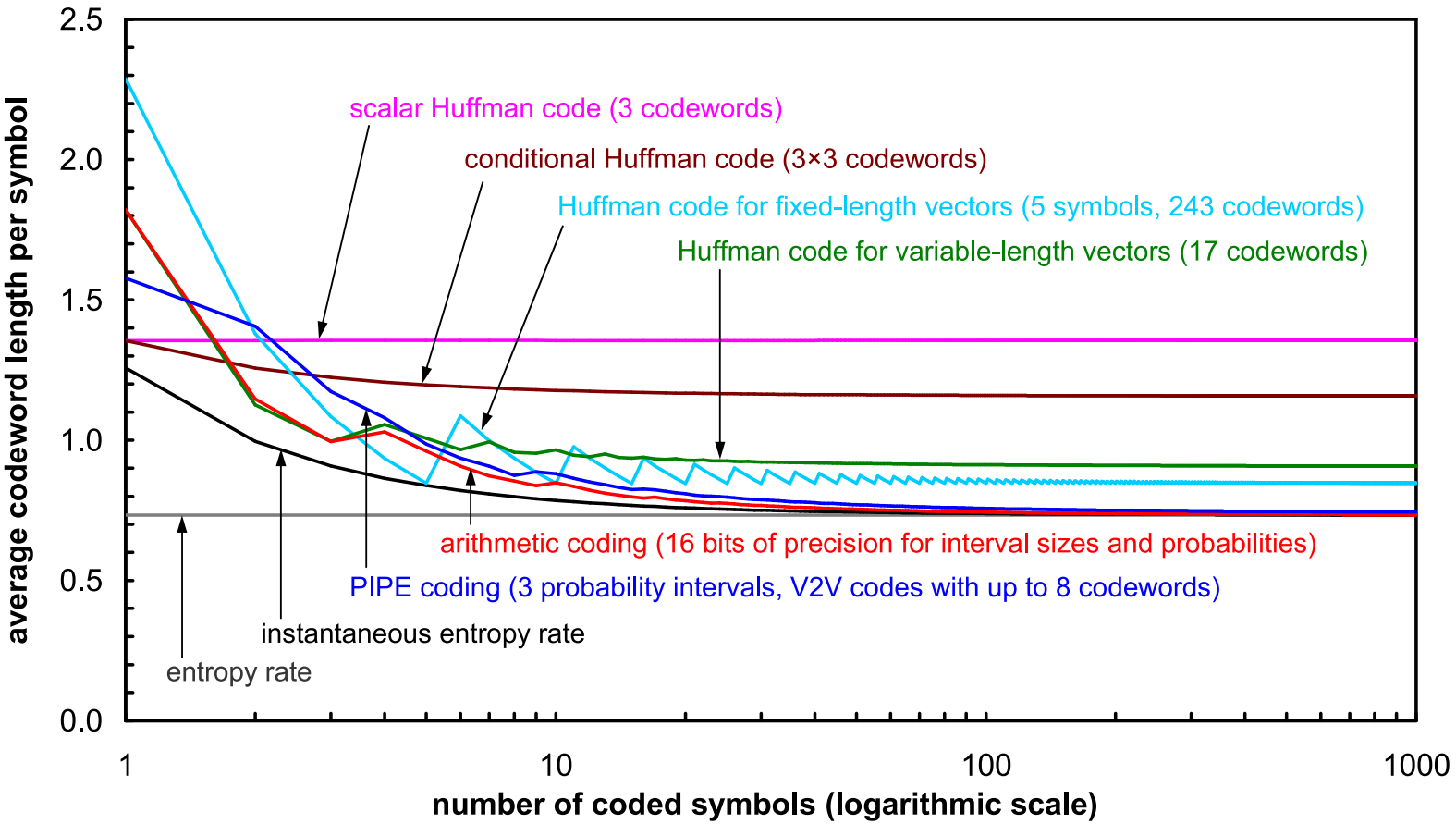
- To each sub-sequence $\boldsymbol{u}_k$, a sequence of codewords $\boldsymbol{c}_k(\boldsymbol{u}_k)$ is assigned
- A sequence of coding bins $\boldsymbol{b}$ is uniquely decodable given K sequences of codewords $\boldsymbol{c}_k(\boldsymbol{u}_k)$, if each sub-sequence is uniquely decodable and the multiplexing rule γ_m is known to the decoder
- The multiplexing rule γ_m could come in many flavors:
  - Concatenate the sub-sequences
  - Interleave codewords event driven
  - Create fixed multiplexing partitions
  - ...

Comparison of Lossless Coding Techniques

- Instantaneous entropy rate

$$\bar{H}_{\text{inst}}(\boldsymbol{S}, L) = \frac{1}{L} H(S_0, S_1, \ldots, S_{L-1}) \tag{87}$$

- Example: Markov source



Conditional and Adaptive Codes

- The question arises how sources with memory and/or with varying statistics can be efficiently entropy-coded
- One approach would be a switch Huffman code trained on the conditional probabilities
- The resulting number of Huffman code tables is often too large in practise
- Hence, conditional entropy coding is typically done using arithmetic codes
- In adaptive arithmetic coding, probabilities $p(a_k)$ are estimated/adapted simultaneously at encoder and decoder
- Statistical dependencies can be exploited using so-called context modeling techniques: conditional probabilities $p(a_k|\boldsymbol{z}_k)$ with $\boldsymbol{z}_k$ being a context/state that is simultaneously computed at encoder and decoder

Forward and Backward Adaptation

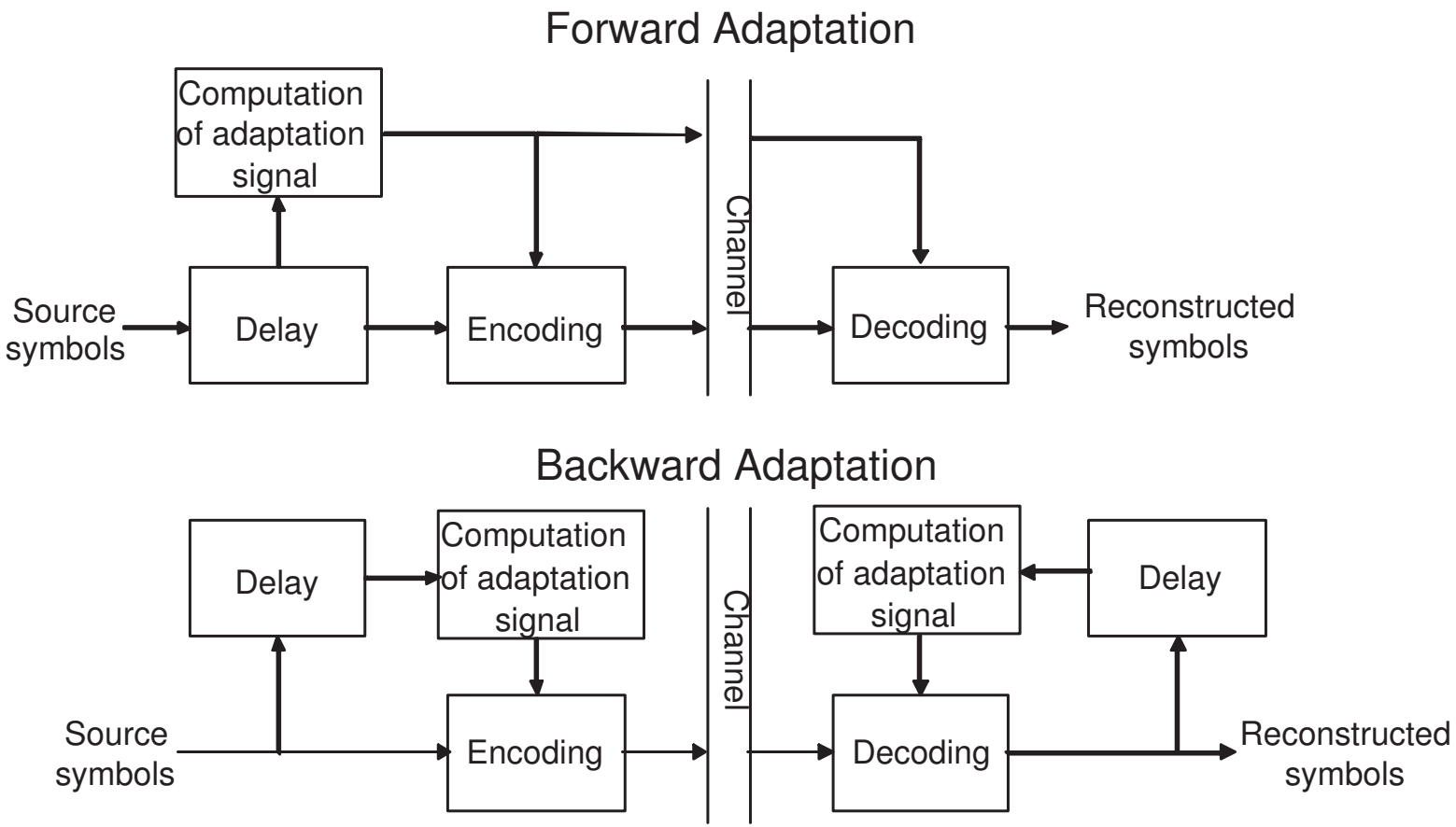
The two basic approaches for adaptation are

- *Forward adaptation*:
  - Gather statistics for a large enough block of source symbols
  - Transmit adaptation signal to decoder as side information
  - Disadvantage: increased bit-rate due to side information
- *Backward adaptation*:
  - Gather statistics simultaneously at coder and decoder
  - Drawback: error resilience

With today's packet-switched transmission systems, an efficient combination of the two adaptation approaches can be achieved:

1. Gather statistics for the entire packet and provide initialization of entropy code at the beginning of the packet
2. Conduct backwards adaptation for each symbol inside the packet in order to minimize the size of the packet

Forward and Backward Adaptation



Coding of Geometric Sources

- Geometric/exponential pdfs are very typical in source coding
- Consider a Geometric source with probability mass function

$$p(s) = 2^{-(s+1)}, s = 0, 1, 2, 3, \ldots$$

- Information content in bits

$$l(s) = -\log_2 p(s) = s + 1$$

- Optimal code with redundancy $\varrho = 0$ is 'unary' code

$$c_0 = 0, c_1 = 10, c_2 = 110, c_3 = 1110$$

- Consider geometric source with decay β

$$p(s) = (1 - \beta)\beta^s, s \geq 0$$

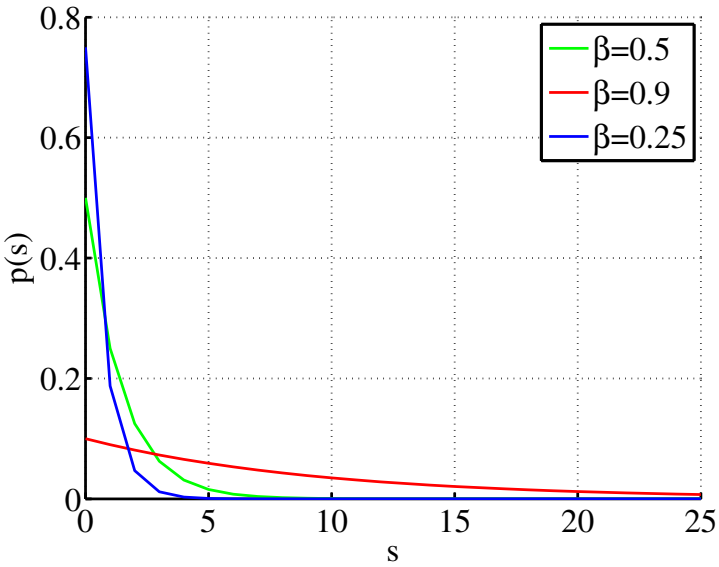
- Average codeword length when using Unary code

$$l_{av} = \sum_{s=0}^{\infty} p(s)(s+1) = \sum_{s=0}^{\infty} (1-\beta)\beta^s \cdot (s+1) = \frac{1}{1-\beta}$$

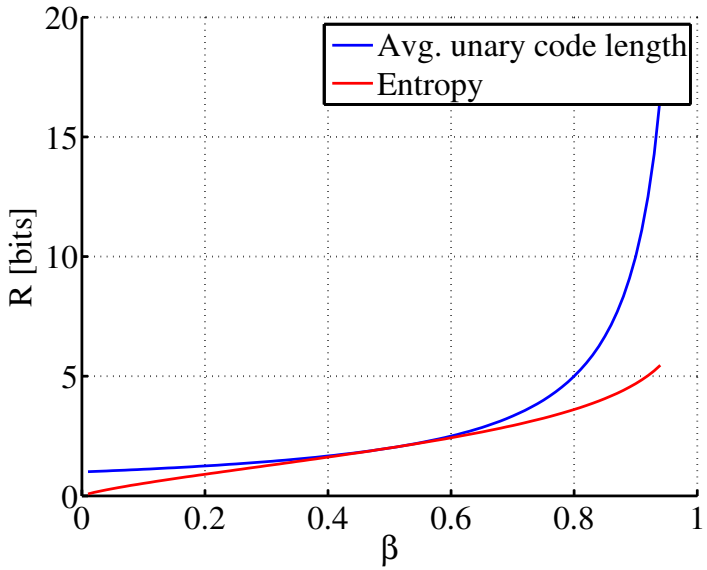
Efficiency of Unary Coding

- Entropy of geometric source with decay β:

$$H = -\log_2(1-\beta) - \frac{\beta}{(1-\beta)}\log_2\beta$$



(a) Geometric source



(b) Length using Unary code and Entropy

Coding of Geometric Sources with $0 < \beta < 2^{-1}$

- For $0 < \beta < 2^{-1}$, the unary code is still optimum single-letter VLC code, but not redundancy-free
- Information content in bits

$$l(s) = -\log_2(1-\beta) - s \cdot \log_2 \beta$$

- Optimality of symbol code is proved by the fact that the Huffman algorithm always yields the unary code (except for least probable code word)
- Term $-\log_2 \beta > 1$ for $\beta < 2^{-1}$, i.e., $l(s+1) > l(s) + 1$ and the unary code is not redundancy free
- Nearly optimal coding possible by using binary arithmetic codes
  - Binarize the geometric source using unary code
  - Encode each 'bin' using a binary arithmetic coder

Binary Arithmetic Coding of Unary Codes

- Geometric source $p(s) = \beta^s(1-\beta)$ with $0 < \beta < 2^{-1}$
- Binarize s using unary code
- Use binary arithmetic code for each 'bin'
- Each bin has probabilities: $p_b(0) = 1-\beta$, $p_b(1) = \beta$

	Bin number							
s	0	1	2	3	4	5	6	$p(s)$
0	0							$1-\beta$
1	1	0						$\beta \cdot (1-\beta)$
2	1	1	0					$\beta \cdot \beta \cdot (1-\beta)$
3	1	1	1	0				$\beta \cdot \beta \cdot \beta \cdot (1-\beta)$
4	1	1	1	1	0			$\beta \cdot \beta \cdot \beta \cdot \beta \cdot (1-\beta)$
5	1	1	1	1	1	0		$\beta \cdot \beta \cdot \beta \cdot \beta \cdot \beta \cdot (1-\beta)$
6	1	1	1	1	1	1	0	$\beta \cdot \beta \cdot \beta \cdot \beta \cdot \beta \cdot \beta \cdot (1-\beta)$
...	...							...

Summary

- Entropy describes the average information of a source
- Entropy is the lower bound for the average number of bits/symbol
- Huffman coding
  - is an efficient and simple entropy coding method
  - needs code table
  - can be inefficient for certain probabilities
  - difficult to use for exploiting statistical dependencies and time-varying probabilities
- Arithmetic coding
  - is a universal method for encoding strings of symbols
  - does not need a code table, but a table for storing probabilities
  - typically requires serial computation of interval and probability estimation update (in case probabilities are adapted)
  - approaches entropy for long strings
  - is well suited for exploiting statistical dependencies and coding time-varying probabilities

Summary (cont'd)

- PIPE coding
  - is an alternative to Arithmetic and Huffman coding
  - can exploit statistical dependencies and coding time-varying probabilities
  - allows for very low redundancy via increasing number of intervals or increasing size of V2V codes
  - allows for larger throughput than Arithmetic coding by use of V2V codes
  - multiplexing to accumulate V2V codes - requires buffering process
- Coding of Geometric Sources
  - Geometric pdf is very typical input for entropy coding
  - Unary code is optimal for representing Geometric pdf with $\beta = 2^{-1}$
  - Fast decaying geometric pdf can be coded using unary binarization followed by binary arithmetic coding