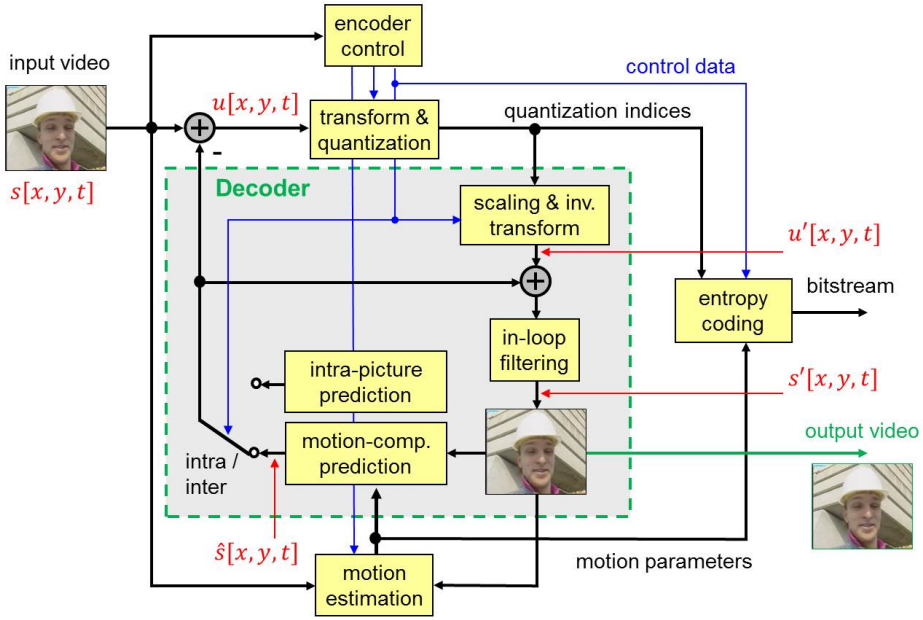
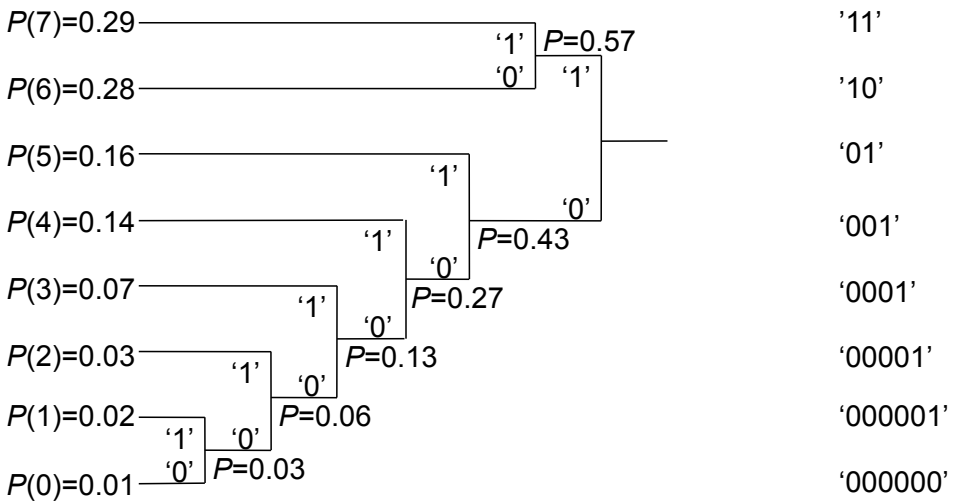
Source Coding and Compression

Heiko Schwarz



Contact:
Dr.-Ing. Heiko Schwarz
heiko.schwarz@hhi.fraunhofer.de

Lossless Coding



Outline

Part I: Source Coding Fundamentals

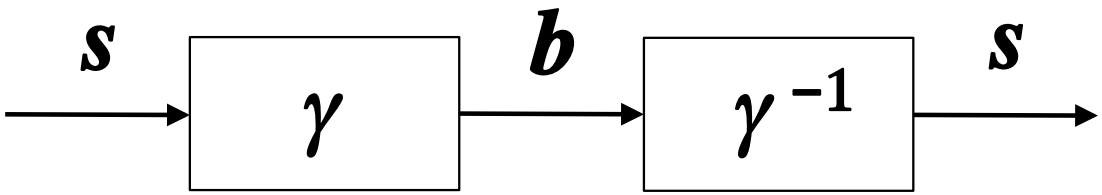
- Probability, Random Variables and Random Processes
- **Lossless Source Coding**
  - Introduction
  - Variable-Length Coding for Scalars
  - Variable-Length Coding for Vectors
  - Elias and Arithmetic Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

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 (Note: Not the case in lossy coding)
- Bit rate reduction possible, if and only if source data have 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:

$$\boldsymbol{b}^{(\ell)} = \gamma\big(\boldsymbol{s}^{(N)}\big) \tag{73}$$

  - Decoder mapping:

$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\boldsymbol{b}^{(\ell)}\big) = \gamma^{-1}\big(\gamma\big(\boldsymbol{s}^{(N)}\big)\big) \tag{74}$$

Classification of Lossless Source Codes

- **Lossless source code**
  - Encoder mapping:
$$\boldsymbol{b}^{(\ell)} = \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \tag{75}$$
  - Decoder mapping:
$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\, \boldsymbol{b}^{(\ell)} \,\big) = \gamma^{-1}\big(\, \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \,\big) \tag{76}$$
- **Fixed-to-fixed mapping**: N and ℓ are both fixed
  - Will be 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 (not discussed in lecture)
- **Variable-to-variable mapping**: ℓ and N are both variable
  - Elias and arithmetic 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 to each a_i a binary codeword $\boldsymbol{b}_i = \{b_0^i, \cdots, b_{\ell(a_i)-1}^i\}$, length $\ell(a_i) \geq 1$
- Example:
  - Alphabet $\mathcal{A} = \{x, y, z\}$
  - Encoder mapping $\gamma(a) = \begin{cases} 0 & : \quad a = x \\ 10 & : \quad a = y \\ 11 & : \quad a = z \end{cases}$
  - Message $\boldsymbol{s}$ = "xyxxzyx"
  - Bit sequence $\boldsymbol{b}$ = "0100011100"

Optimization Problem

- **Average codeword length** is given as

$$\bar{\ell} = E\{\ell(S)\} = \sum_{i=0}^{M-1} p(a_i) \cdot \ell(a_i) \tag{77}$$

- **The goal of the lossless code design problem is to minimize the average codeword length $\bar{\ell}$ 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

$$\text{if} \qquad a_k \neq a_j \qquad \text{then} \qquad \boldsymbol{b}_k \neq \boldsymbol{b}_j \tag{78}$$

  Codes that don't have that property are called **singular codes**

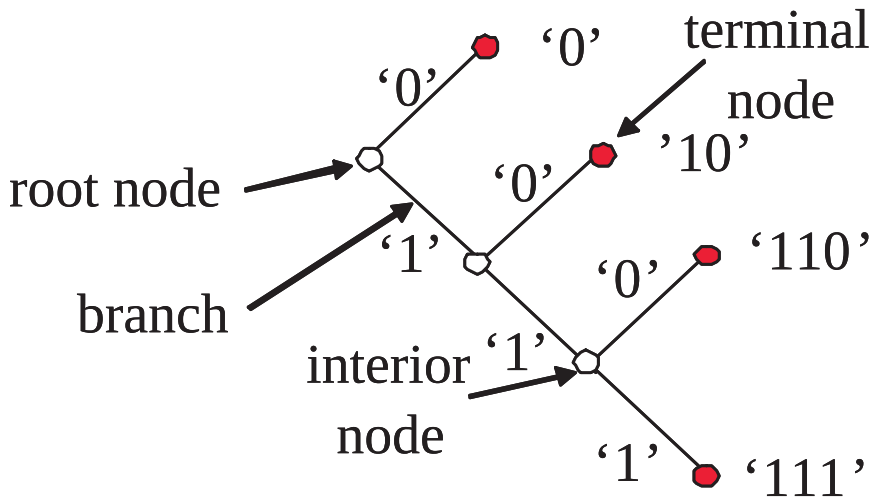
- For sequences of symbols, above constraint needs to be extended to the concatenation of multiple symbols

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

- **Prefix codes**: One class of codes that satisfies the constraint of unique decodability
  - A code is called a prefix code if no codeword for an alphabet letter represents the codeword or a prefix of the codeword for any other alphabet letter
  - It is obvious that if the code is a prefix code, then any concatenation of symbols can be uniquely decoded

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 constructed by assigning letters of the alphabet $\mathcal{A}$ to terminal nodes of a binary tree

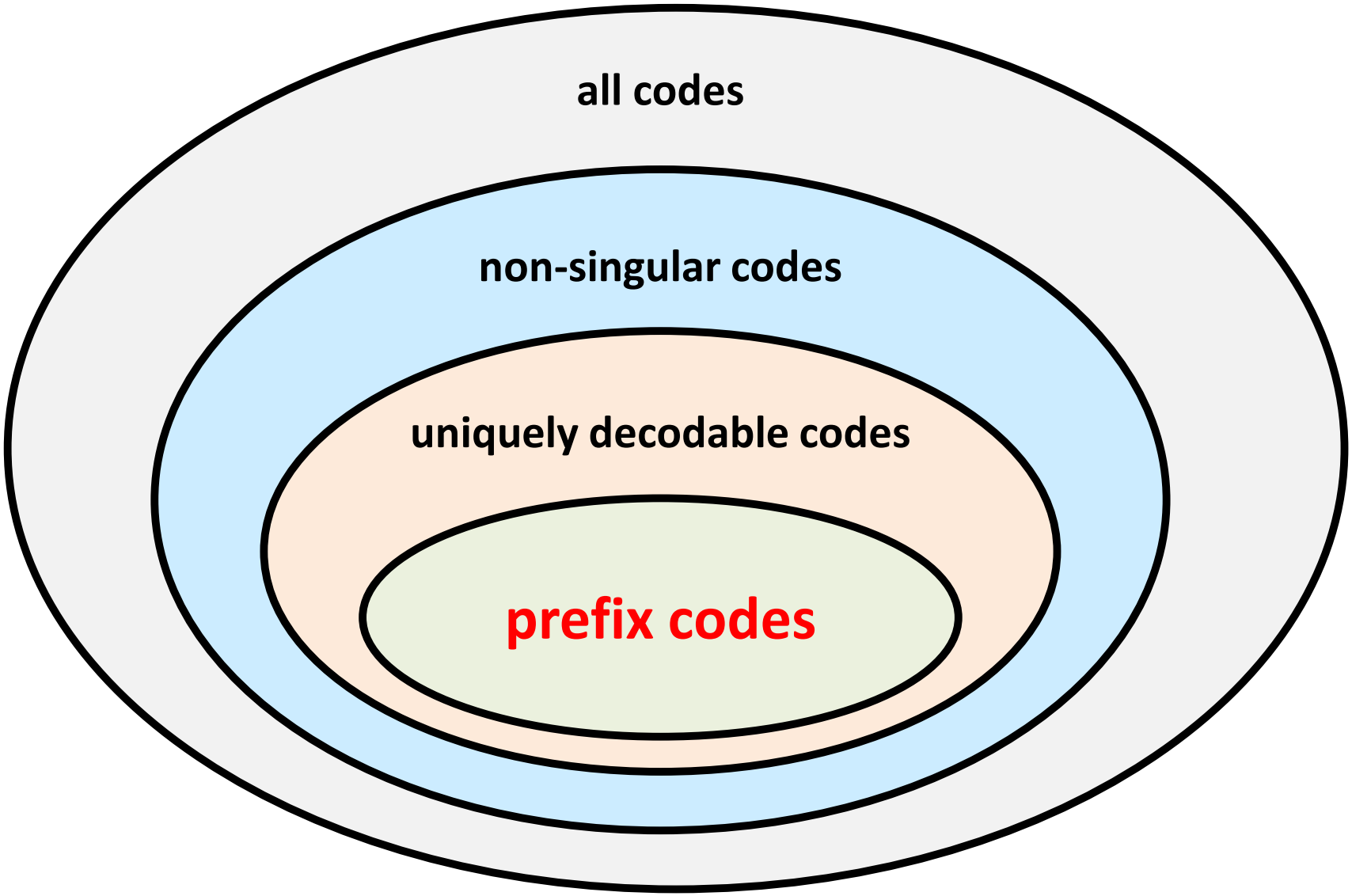
Parsing of Prefix Codes

- 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 labelled 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

- Important properties of prefix codes:
  - Prefix codes are uniquely decodable
  - Prefix codes are **instantaneously decodable**

Classification of Codes



Unique Decodability: Kraft Inequality

- Assume fully balanced tree with depth $\ell_{\max}$ (length of longest codeword)
- Codewords are assigned to nodes with codeword length $\ell(a_k) \leq \ell_{\max}$
- Each choice with $\ell(a_k) \leq \ell_{\max}$ eliminates $2^{\ell_{\max}-\ell(a_k)}$ other possibilities of codeword 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 removed terminal nodes must be less than or equal to number of terminal nodes in balanced tree with depth $\ell_{\max}$, which is $2^{\ell_{\max}}$

$$\sum_{i=0}^{M-1} 2^{\ell_{\max}-\ell(a_i)} \leq 2^{\ell_{\max}} \tag{79}$$

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

$$\text{Kraft inequality:} \qquad \zeta(\gamma) = \sum_{i=0}^{M-1} 2^{-\ell(a_i)} \leq 1 \tag{80}$$

Proof of the Kraft Inequality

- Consider

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{i_0=0}^{M-1} \sum_{i_1=0}^{M-1} \cdots \sum_{i_{L-1}=0}^{M-1} 2^{-\left(\ell(a_{i_0}) + \ell(a_{i_1}) + \cdots + \ell(a_{i_{L-1}}) \right)} \tag{81}$$

- $\ell_L = \ell(a_{i_0}) + \ell(a_{i_1}) + \cdots + \ell(a_{i_{L-1}})$ represents the combined codeword length for coding L symbols
- Let $A(\ell_L)$ denote the number of distinct symbol sequences that produce a bit sequence with the same length ℓ_L
- Let $\ell_{\max}$ be the maximum codeword length
- Hence, we can write

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \tag{82}$$

Proof of the Kraft Inequality

- We have

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \tag{83}$$

- For a uniquely decodable code, $A(\ell_L)$ must be less than or equal to 2^{ℓ_L}, since there are only 2^{ℓ_L} distinct bit sequences of length ℓ_L
- Hence, a uniquely decodable code must fulfill the inequality

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \leq \sum_{\ell_L=L}^{L \cdot \ell_{\max}} 2^{\ell_L}\, 2^{-\ell_L} = L\,(\ell_{\max} - 1) + 1 \tag{84}$$

- The left side of this inequality grows exponentially with L, while the right side grows only linearly with L
- If the Kraft inequality is not fulfilled, we can always find a value of L for which the condition (84) is violated

$\Longrightarrow$ Kraft inequality specifies a necessary condition for uniquely decodable codes

Prefix Codes and the Kraft Inequality

- Given is a set of codeword lengths $\{\ell_0, \ell_1, \cdots, \ell_{M-1}\}$ that satisfies the Kraft inequality, with $\ell_0 \leq \ell_1 \leq \cdots \leq \ell_{M-1}$
- Construction of prefix code
  - Start with fully balanced code tree of infinite depth (or depth ℓ_{M-1})
  - Choose a node of depth ℓ_0 for first codeword and prune tree at this node
  - Choose a node of depth ℓ_1 for second codeword and prune tree at this node
  - Continue this procedure until all codeword length are assigned
- Question: Is that always possible?
  - Selection of codeword ℓ_k removes $2^{\ell_i - \ell_k}$ codewords with length $\ell_i \geq \ell_k$

$\Longrightarrow$ For codeword of length ℓ_i, number of available choices is given by

$$n(\ell_i) = 2^{\ell_i} - \sum_{k=0}^{i-1} 2^{\ell_i - \ell_k} = 2^{\ell_i} \left(1 - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) \tag{85}$$

Prefix Codes and the Kraft Inequality

- Number of available choices for codeword length ℓ_i

$$n(\ell_i) = 2^{\ell_i} - \sum_{k=0}^{i-1} 2^{\ell_i - \ell_k} = 2^{\ell_i} \left(1 - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) \tag{86}$$

- Kraft inequality is fulfilled

$$\sum_{k=0}^{M-1} 2^{-\ell_k} \leq 1 \tag{87}$$

- This yields

$$n(\ell_i) \geq 2^{\ell_i} \left(\sum_{k=0}^{M-1} 2^{-\ell_k} - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) = 2^{\ell_i} \sum_{k=i}^{M-1} 2^{-\ell_k} = 1 + \sum_{k=i+1}^{M-1} 2^{\ell_i - \ell_k} \geq 1 \tag{88}$$

$\Longrightarrow$ **Can construct prefix code for any set of codeword lengths that fulfills Kraft inequality**

Practical Importance of Prefix Codes

- We have shown:
  - All uniquely decodable codes fulfill Kraft inequality
  - Possible to construct prefix code for any set of codeword lengths that fulfills Kraft inequality

$\Longrightarrow$ **There are no uniquely decodable codes that have a smaller average codeword length than the best prefix code**

- Prefix codes have further desirable properties
  - Instantaneous decodability
  - Easy to construct

$\Longrightarrow$ **All variable-length codes used in practice are prefix codes**

Lower Bound for Average Codeword Length

- Average codeword length

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i)\,\ell(a_i) = -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{2^{-\ell(a_i)}}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (89)$$

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

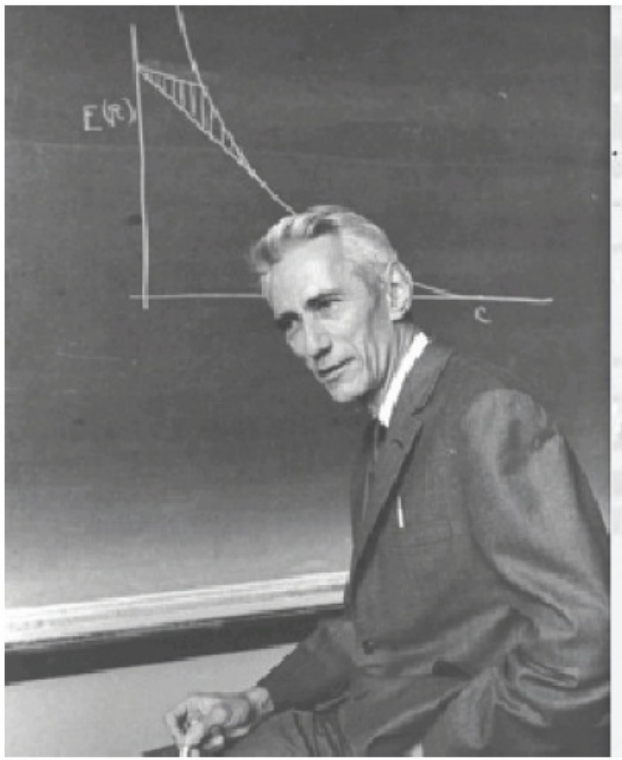
$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (90)$$

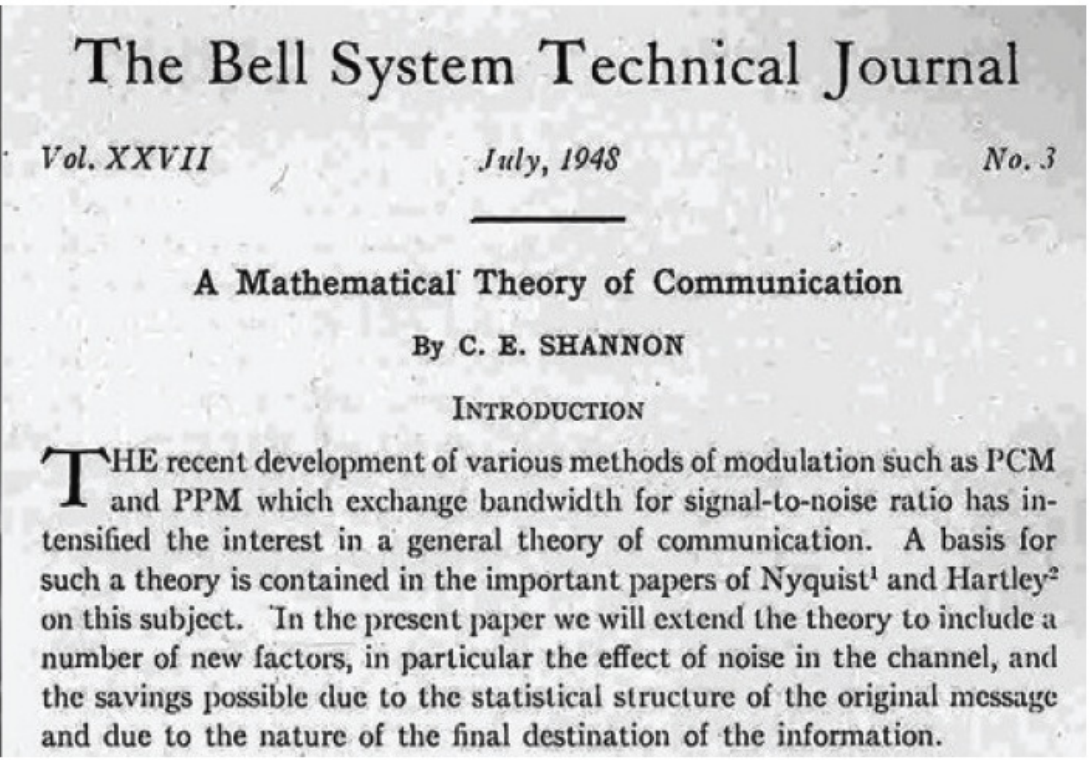
- We will show that

$$\bar{\ell} \geq -\sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) = H(S) \quad \text{(Entropy)} \quad (91)$$

Historical Reference

- C. E. SHANNON introduced entropy as an uncertainty measure for random experiments and derived it based on three postulates
- 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

- Average codeword length

$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (92)$$

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

$$-\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) \geq 0 \quad (93)$$

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

$$\begin{aligned} -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) &\geq \frac{1}{\ln 2}\sum_{i=0}^{M-1} p(a_i)\left(1 - \frac{q(a_i)}{p(a_i)}\right) \\ &= \frac{1}{\ln 2}\left(\sum_{i=0}^{M-1} p(a_i) - \sum_{i=0}^{M-1} q(a_i)\right) = 0 \end{aligned} \quad (94)$$

- Called **divergence inequality**

Entropy and Redundancy

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

$$\bar{\ell} \geq H(S) = E\{-\log_2 p(S)\} = -\sum_{i=0}^{M-1} p(a_i)\, \log_2 p(a_i) \tag{95}$$

- The measure $H(S)$ is called the **entropy** of a random variable S
- The entropy is a measure of the **uncertainty** of a random variable
- **Redundancy** of a code is given by the difference

$$\varrho = \bar{\ell} - H(S) = \sum_{i=0}^{M-1} p(a_i)\big(\ell(a_i) - \log_2 p(a_i)\big) \geq 0 \tag{96}$$

- Redundancy is zero only, if and only if
  - Kraft inequality is fulfilled with equality (codeword at all terminal nodes)
  - All probability masses are negative integer powers of two

An Upper Bound for the Minimum Average Codeword Length

- Upper bound of $\bar{\ell}$: Choose $\ell(a_i) = \lceil -\log_2 p(a_i) \rceil$, $\forall a_i \in \mathcal{A}$
- Codewords satisfy Kraft inequality (show using $\lceil x \rceil \geq x$)

$$\sum_{i=0}^{M-1} 2^{-\lceil -\log_2 p(a_i) \rceil} \leq \sum_{i=0}^{M-1} 2^{\log_2 p(a_i)} = \sum_{i=0}^{M-1} p(a_i) = 1 \tag{97}$$

- Obtained average codeword length (use $\lceil x \rceil < x + 1$)

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i) \lceil -\log_2 p(a_i) \rceil < \sum_{i=0}^{M-1} p(a_i) \left(1 - \log_2 p(a_i)\right) = H(S) + 1 \tag{98}$$

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

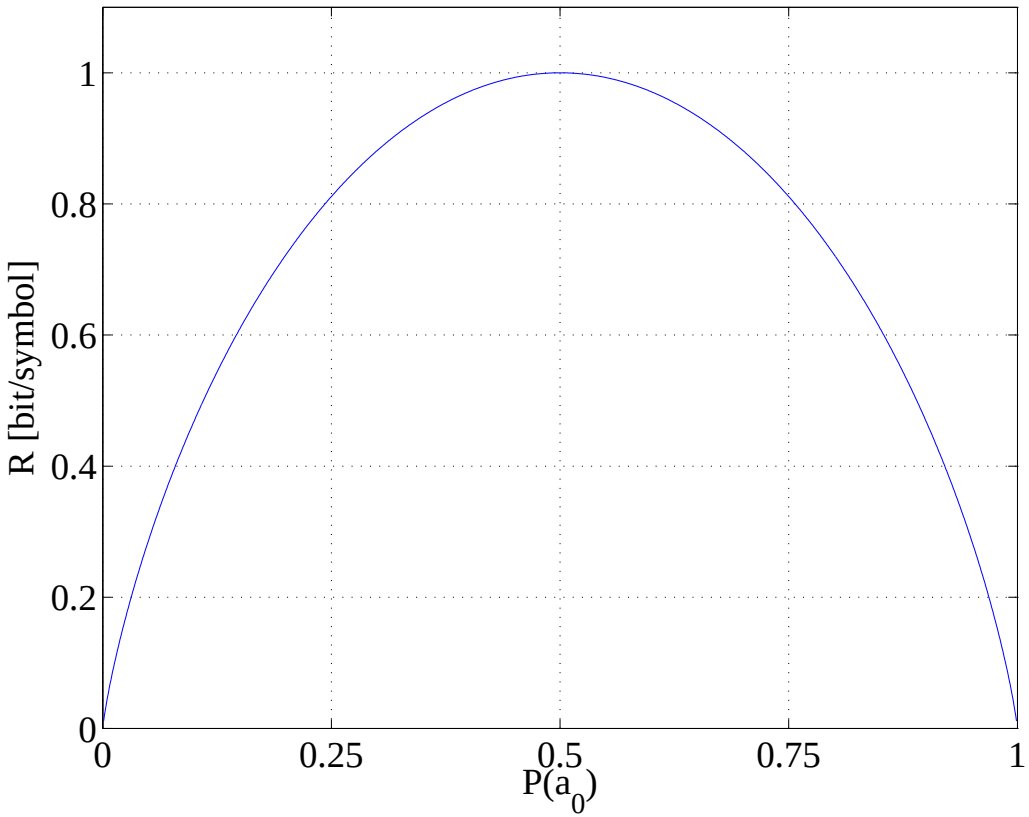
$$\boxed{H(S) \leq \bar{\ell}_{\min} < H(S) + 1} \tag{99}$$

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

$$H(S) = -p \log_2 p - (1-p) \log_2 (1-p) = H_b(p) \tag{100}$$

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



Relative Entropy or Kullback-Leibler Divergence

- Defined as

$$D(p||q) = \sum_{i=0}^{M-1} p(a_i) \log_2 \frac{p(a_i)}{q(a_i)} \tag{101}$$

- Divergence inequality (we have already proofed it)

$$D(p||q) \geq 0 \qquad \text{(equality if and only if } p = q\text{)} \tag{102}$$

- Note: $D(p||q) \neq D(q||p)$

- What does the measure $D(p||q)$ tell us?
  - Assume we have an optimal code with an average codeword length equal to the entropy for a pmf q and apply it to a pmf p
  - $D(p||q)$ is the difference between average codeword length and entropy

$$\begin{aligned} D(p||q) &= -\sum_{i=0}^{M-1} p(a_i) \log_2 q(a_i) + \sum_{i=0}^{M-1} p(a_i) \log_2 p(a_i) \\ &= \sum_{i=0}^{M-1} p(a_i) \ell_q(a_i) - H(S) = \bar{\ell}_q - H(S) \end{aligned} \tag{103}$$

Optimal Prefix Codes

Conditions for optimal prefix codes

1. For any two symbols $a_i, a_j \in \mathcal{A}$ with $p(a_i) > p(a_j)$, the associated codeword lengths satisfy $\ell(a_i) \leq \ell(a_j)$
2. There are always two codewords that have the maximum codeword length and differ only in the final bit

Justification

1. Otherwise, an exchange of the codewords for the symbols a_i and a_j would decrease the average codeword length while preserving the prefix property
2. Otherwise, the removal of the last bit of the codeword with maximum length would preserve the prefix property and decrease the average codeword length

The Huffman Algorithm

How can we generate an optimal prefix code?

- The answer to this question was given by D. A. Huffman in 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

General idea

- Both optimality conditions are obeyed if the two codewords with maximum length (that differ only in the final bit) are assigned to the letters a_i and a_j with minimum probabilities
- The two letters are then treated as a new letter with probability $p(a_i) + p(a_j)$
- Procedure is repeated for the new alphabet

The Huffman Algorithm

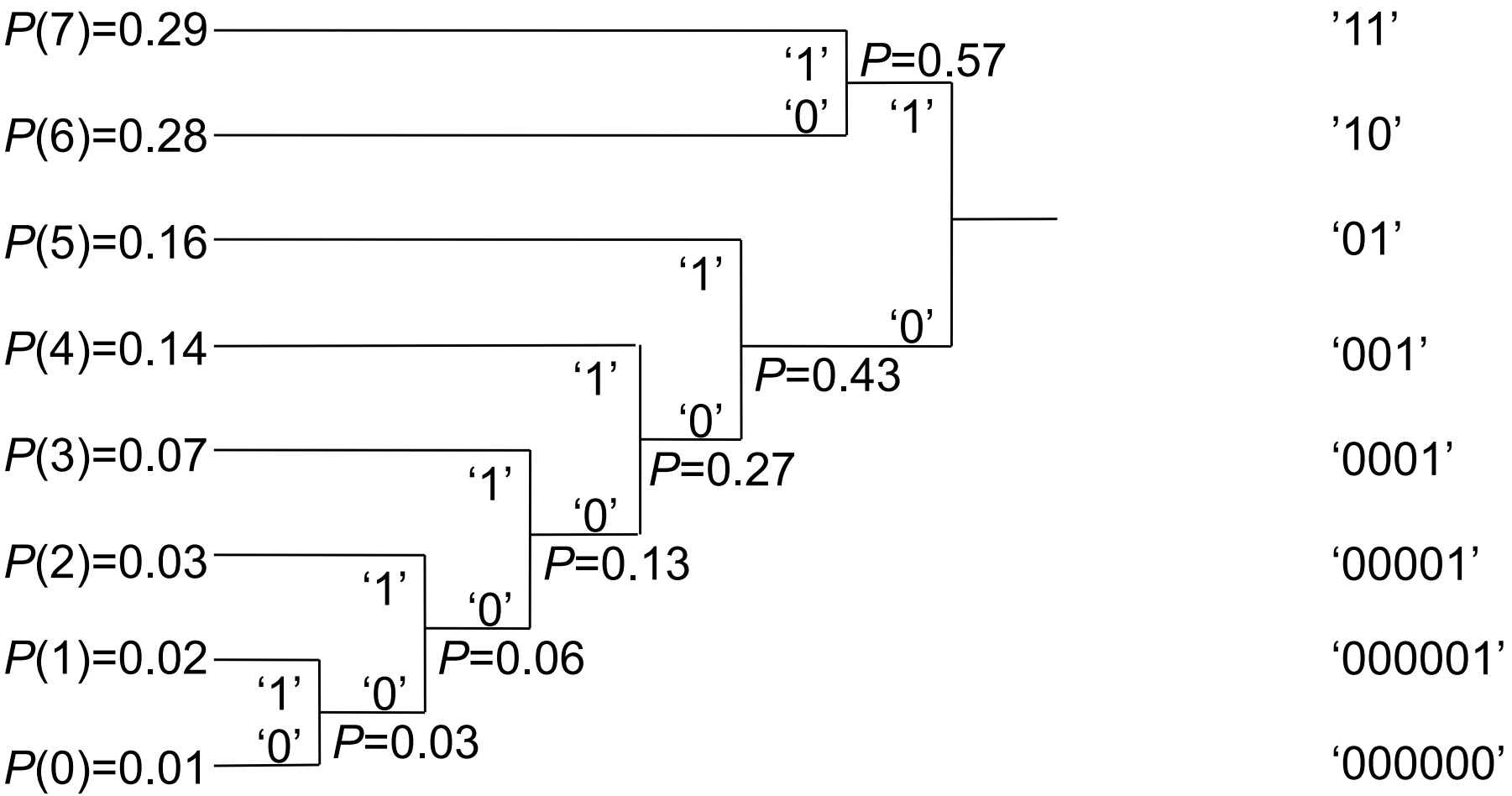
Huffman algorithm for given alphabet $\mathcal{A}$ with marginal pmf p

1. Select the two letters a_i and a_j with the smallest probabilities and create a parent node for the nodes that represent these two letters in the binary code tree
2. Replace the letters a_i and a_j by a new letter with an associated probability of $p(a_i) + p(a_j)$
3. If more than one letter remains, repeat the previous steps
4. Convert the binary code tree into a prefix code

Note: There are multiple optimal prefix codes

- Assigment of "0" and "1" to tree branches is arbitrary
- If some letters have the same probability (at some stage of the algorithm), there might be multiple ways to select two letters with smallest probabilities
- But, all optimal prefix codes have the same average codeword length

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 for Conditional Huffman Codes

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

$$H(S_n|a_k) \leq \bar{\ell}_k < H(S_n|a_k) + 1 \tag{104}$$

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

$$H(S_n|a_k) = H(S_n|S_{n-1}=a_k) = -\sum_{i=0}^{M-1} p(a_i|a_k)\, \log_2 p(a_i|a_k) \tag{105}$$

- Resulting average codeword length

$$\bar{\ell} = \sum_{k=0}^{M-1} p(a_k)\, \bar{\ell}_k \tag{106}$$

- Resulting bounds

$$\sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) \leq \bar{\ell} < \sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) + 1 \tag{107}$$

Conditional Entropy

- Lower bound is called **conditional entropy** $H(S_n|S_{n-1})$ of the random vriable S_n given random variable S_{n-1}

$$\begin{aligned} H(S_n|S_{n-1}) &= E\{-\log_2 p(S_n|S_{n-1})\} \\ &= = \sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i,a_k)\, \log_2 p(a_i|a_k), \end{aligned} \tag{108}$$

- Minimum average codeword length for conditional code is bounded by

$$\boxed{H(S_n|S_{n-1}) \le \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1} \tag{109}$$

Conditioning May Reduce Minimum Average Codeword Length

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

$$H(S_n|S_{n-1}) \leq \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1 \tag{110}$$

- Conditioning may reduce minimum average codeword length (use divergence inequality)

$$\begin{aligned} H(S) - H(S_n|S_{n-1}) &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\Big(\log_2 p(a_i) - \log_2 p(a_i|a_k)\Big) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\, \log_2 \frac{p(a_i)\, p(a_k)}{p(a_i, a_k)} \\ &\geq 0 \end{aligned} \tag{111}$$

- Note: Equality for iid process, $p(a_i, a_k) = p(a_i)\, p(a_k)$
- For the example Markov source:
  - 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

- Consider 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 (code vector instead of scalar)
- Design Huffman code for 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

$$\frac{H(S_n, \cdots, S_{n+N-1})}{N} \leq \bar{\ell}_{\min} < \frac{H(S_n, \cdots, S_{n+N-1})}{N} + \frac{1}{N} \tag{112}$$

- Define **block entropy**

$$\begin{aligned} H(S_n, \cdots, S_{n+N-1}) &= E\{-\log_2 p(S_n, \cdots, S_{n+N-1})\} \\ &= -\sum_{a_0} \cdots \sum_{a_{N-1}} p(a_0, \cdots, a_{N-1}) \, \log_2 p(a_0, \cdots, a_{N-1}) \end{aligned} \tag{113}$$

Entropy Rate

Entropy rate

- The following limit is called **entropy rate**

$$\boxed{\bar{H}(\boldsymbol{S}) = \lim_{N \to \infty} \frac{H(S_0, \cdots, S_{N-1})}{N}} \tag{114}$$

- The limit in (114) always exists for stationary sources

Fundamental lossless source coding theorem

- Entropy rate $\bar{H}(\boldsymbol{S})$: Greatest lower bound for the average codeword length $\bar{\ell}$ per symbol

$$\boxed{\bar{\ell} \geq \bar{H}(\boldsymbol{S})} \tag{115}$$

- Always asymptotically achievable with block Huffman coding for $N \to \infty$

Entropy Rate for Special Sources

- Entropy rate for iid processes

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{\sum_{n=0}^{N-1} E\{-\log_2 p(S_n)\}}{N} \\ &= \lim_{N\to\infty} E\{-\log_2 p(S_n)\} \\ &= H(S) \end{aligned} \tag{116}$$

- Entropy rate for stationary Markov processes

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0)\} + \sum_{n=1}^{N-1} E\{-\log_2 p(S_n|S_{n-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0)\}}{N} + E\{-\log_2 p(S_n|S_{n-1})\} \\ &= H(S_n|S_{n-1}) \end{aligned} \tag{117}$$

Example for Block Huffman Coding

Example:

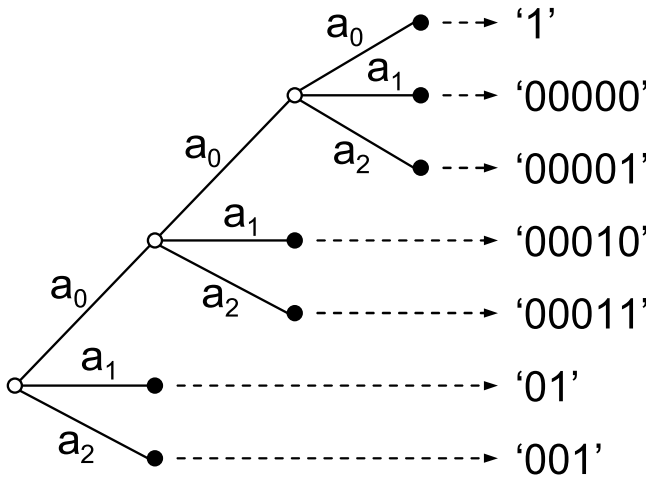
- Joint Huffman coding of 2 symbols for example Markov source
- Efficiency of block Huffman coding for different N
- Table sizes for different N

$a_i a_k$	$p(a_i, a_k)$	codewords
$a_0 a_0$	0.58	1
$a_0 a_1$	$0.03\overline{2}$	00001
$a_0 a_2$	$0.03\overline{2}$	00010
$a_1 a_0$	$0.03\overline{6}$	0010
$a_1 a_1$	$0.19\overline{5}$	01
$a_1 a_2$	$0.01\overline{2}$	000000
$a_2 a_0$	$0.02\overline{7}$	00011
$a_2 a_1$	$0.01\overline{7}$	000001
$a_2 a_2$	$0.0\overline{6}$	0011

N	$\bar{\ell}$	$N_{\mathcal{C}}$
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

$$\bar{\ell} = \frac{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, \ell_k}{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, N_k} \tag{118}$$

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

V2V Code Performance

- Example Markov process: $\bar{H}(\boldsymbol{S}) = H(S_n|S_{n-1}) = 0.7331$
- 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

Summary on Variable-Length Coding

Uniquely decodable codes

- Necessary condition: Kraft inequality
- Prefix codes: Instantaneously decodable
- Can construct prefix codes for codeword lengths that fulfill Kraft inequality

Bounds for lossless coding

- Entropy
- Conditional entropy
- Block entropy
- Entropy rate

Optimal prefix codes

- Huffman algorithm for given pmf
- Scalar Huffman codes
- Conditional Huffman codes
- Block Huffman codes
- V2V codes

Exercise 3

A fair coin is tossed an infinite number of times. Let Y_n be a random variable, with $n \in \mathbb{Z}$, that describes the outcome of the n-th coin toss. If the outcome of the n-th coin toss is head, Y_n is equal to 1; if it is tail, Y_n is equal to 0. Now consider the random process $\mathbf{X} = \{X_n\}$. The random variables X_n are determined by $X_n = Y_n + Y_{n-1}$, and thus describe the total number of heads in the n-th and $(n-1)$-th coin tosses.

(a) Determine the marginal pmf $p_{X_n}(x_n)$ and the marginal entropy $H(X_n)$. Is it possible to design a uniquely decodable code with one codeword per possible outcome of X_n that has an average codeword length equal to the marginal entropy?

(b) Determine the conditional pmf $p_{X_n|X_{n-1}}(x_n|x_{n-1})$ and the conditional entropy $H(X_n|X_{n-1})$. Design a conditional Huffman code. What is the average codeword length of the conditional Huffman code?

(c) Is the random process $\mathbf{X}$ a Markov process?

(d) Derive a general formula for the N-th order block entropy $H_N = H(X_n, \cdots, X_{n-N+1})$. How many symbols have to be coded jointly at minimum for obtaining a code that is more efficient than the conditional Huffman code developed in (b)?

(e) Calculate the entropy rate $\bar{H}(\mathbf{X})$ of the random process $\mathbf{X}$. Is it possible to design a variable length code with finite complexity and an average codeword length equal to the entropy rate? If yes, what requirement has to be fulfilled?

Exercise 4

Given is a discrete iid process $\mathbf{X}$ with the alphabet $\mathcal{A} = \{a, b, c, d, e, f, g\}$. The pmf $p_X(x)$ and 6 example codes are listed in the following table.

x	$p_X(x)$	A	B	C	D	E	F
a	$1/3$	1	0	00	01	000	1
b	$1/9$	0001	10	010	101	001	100
c	$1/27$	000000	110	0110	111	010	100000
d	$1/27$	00001	1110	0111	010	100	10000
e	$1/27$	000001	11110	100	110	111	000000
f	$1/9$	001	111110	101	100	011	1000
g	$1/3$	01	111111	11	00	001	10

(a) Develop a Huffman code for the given pmf $p_X(x)$, calculate its average codeword length and its absolute and relative redundancy.

(b) For all codes A, B, C, D, E, and F, do the following:

- Calculate the average codeword length per symbol;
- Determine whether the code is a singular code;
- Determine whether the code is uniquely decodable;
- Determine whether the code is a prefix code;
- Determine whether the code is an optimal prefix code.

(c) Briefly describe a process for decoding a symbol sequence given a finite sequence of K bits that is coded with code F.

Exercise 5

Given is a Bernoulli process $\mathbf{X}$ with the alphabet $\mathcal{A} = \{a, b\}$ and the pmf $p_X(a) = p$, $p_X(b) = 1 - p$. Consider the three codes in the following table.

Code A		Code B		Code C	
symbols	codeword	symbols	codeword	symbol	codeword
aa	1	aa	0001	a	0
ab	01	ab	001	b	1
b	00	ba	01		
		bb	1		

(a) Calculate the average codeword length per symbol for the three codes.

(b) For which probabilities p is the code A more efficient than code B?

(c) For which probabilities p is the simple code C more efficient than both code A and code B?

Exercise 6

Given is a Bernoulli process $\mathbf{B} = \{B_n\}$ with the alphabet $\mathcal{A}_B = \{0, 1\}$, the pmf $p_B(0) = p$, $p_B(1) = 1 - p$, and $0 \leq p < 1$. Consider the random variable X that specifies the number of random variables B_n that have to be observed to get exactly one "1".

Calculate the entropies $H(B_n)$ and $H(X)$.

For which value of p, with $0 < p < 1$, is $H(X)$ four times as large as $H(B_n)$?

Hint:

$$\forall_{|a|<1}, \sum_{k=0}^{\infty} a^k = \frac{1}{1-a}$$

$$\forall_{|a|<1}, \sum_{k=0}^{\infty} k\, a^k = \frac{a}{(1-a)^2}$$

Exercise 7

Proof the chain rule for the joint entropy,

$$H(X,Y) = H(X) + H(Y|X).$$

Exercise 8

Investigate the entropy of a function of a random variable X. Let X be a discrete random variable with the alphabet $\mathcal{A}_X = \{0, 1, 2, 3, 4\}$ and the binomial pmf

$$p_X(x) = \begin{cases} 1/16 & : \quad x = 0 \vee x = 4 \\ 1/4 & : \quad x = 1 \vee x = 3 \\ 3/8 & : \quad x = 2 \end{cases} .$$

(a) Calculate the entropy $H(X)$.

(b) Consider the functions $g_1(x) = x^2$ and $g_2(x) = (x-2)^2$.
Calculate the entropies $H(g_1(X))$ and $H(g_2(X))$.

(c) Proof that the entropy $H(g(X))$ of a function $g(x)$ of a random variable X is not greater than the entropy of the random variable X,

$$H(g(X)) \leq H(X)$$

Determine the condition under which equality is achieved.