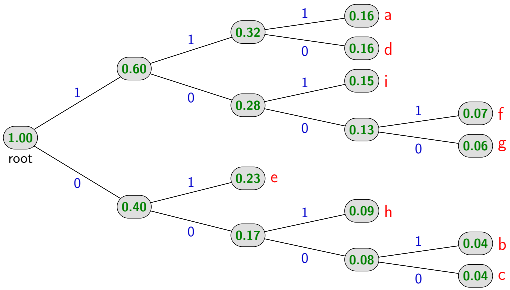
Optimal Variable-Length Codes

a_k	p_k	codewords
a	0.16	111
b	0.04	0001
c	0.04	0000
d	0.16	110
e	0.23	01
f	0.07	1001
g	0.06	1000
h	0.09	001
i	0.15	101



Last Lecture: Scalar Variable-Length Codes

Unique Decodability

- Necessary condition: Kraft-McMillan inequality for codeword lengths: $\sum_k 2^{-\ell_k} \leq 1$
- Sufficient condition: Prefix codes (can be represented as binary trees)

Prefix Codes

- Uniquely and **instantaneously decodable**, simple encoding and decoding (e.g., via binary tree)
- Can always construct prefix code for codeword lengths that satisfy Kraft-McMillan inequality

➞ **There are no better uniquely decodable codes than the best prefix codes**

Average Codeword Length and Entropy

- Average codeword length for pmf p: $\bar{\ell} = \sum_k p_k \, \ell_k$ (efficiency measure for lossless codes)
- Entropy of a source with pmf p: $H(p) = -\sum_k p_k \log_2 p_k$
- Entropy = lower bound for $\bar{\ell}$: $\bar{\ell} \geq H(p)$
- Can always construct prefix code with $H(p) \leq \bar{\ell} < H(p) + 1$ (e.g., Shannon code)

Shannon Code

$$\bar{\ell} \geq H(p)$$

$$\sum_k p_k \, \ell_k \geq -\sum_k p_k \log_2 p_k$$

➡ equality if and only if $\ell_k = -\log_2 p_k$

➡ only possible if: $\forall k, \; p_k = 2^{-n_k}$ with $n_k \in \mathbb{Z}$

Shannon Code:

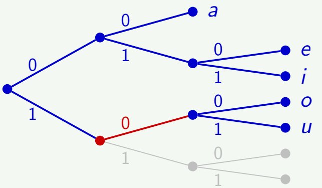
➡ Round codeword lengths to next integer: $\ell_k = \lceil -\log_2 p_k \rceil$

➡ Construct prefix code with these codeword lengths (always possible)

➡ **Typically not optimal (structural redundancies)**

Example: Shannon Code

a_k	p_k	$\ell_k = \lceil -\log_2 p_k \rceil$	codewords
a	0.40	$2 = \lceil\, 1.32\ldots \,\rceil$	00
e	0.15	$3 = \lceil\, 2.73\ldots \,\rceil$	010
i	0.15	$3 = \lceil\, 2.73\ldots \,\rceil$	011
o	0.15	$3 = \lceil\, 2.73\ldots \,\rceil$	100
u	0.15	$3 = \lceil\, 2.73\ldots \,\rceil$	101



$H \approx 2.171$

$\bar{\ell} = 2.6$

$\varrho \approx 19.8\%$

structural redundancy !

Note: Removing the redundant bit would yield $\bar{\ell} = 2.3$, $\varrho \approx 5.9\%$ (still not optimal)

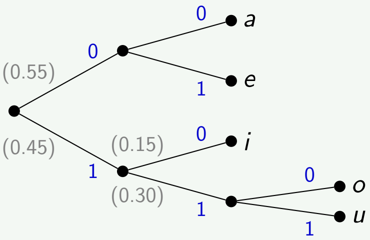
Shannon-Fano Code: Construct Full Binary Code Tree

1. Sort symbols in alphabet according to their probability masses
2. Divide sorted list into two groups A and B, so that $P(A)$ is as close to $P(B)$ as possible
3. Create a node and assign the first group A to one branch and the other group B to the other branch
4. Recursively apply steps 2 and 3 to the groups A and B until all symbols are assigned to terminal nodes

➜ **No Guarantee for Optimality** **(but yields prefix code without structural redundancy)**

Example: Shannon-Fano Code

a_k	p_k	codewords	better code
a	0.40	00	0
e	0.15	01	100
i	0.15	10	101
o	0.15	110	110
u	0.15	111	111
$H \approx 2.171$		$\bar{\ell} = 2.3$ $\varrho \approx 5.9\%$	$\bar{\ell} = 2.2$ $\varrho \approx 1.3\%$



Example: Shannon Code / Shannon-Fano Code / Optimal Code

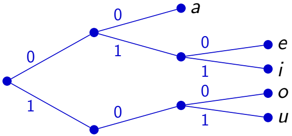
a_k	p_k	Shannon	Shannon-Fano	optimal
a	0.40	00	00	0
e	0.15	010	01	100
i	0.15	011	10	101
o	0.15	100	110	110
u	0.15	101	111	111
$H \approx 2.171$		$\bar{\ell} = 2.6$ $\varrho \approx 19.8\%$	$\bar{\ell} = 2.3$ $\varrho \approx 5.9\%$	$\bar{\ell} = 2.2$ $\varrho \approx 1.3\%$

Question

Is there a low-complex algorithm for constructing optimal prefix codes ?

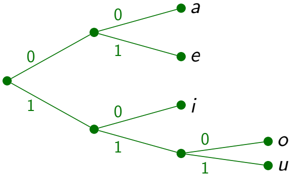
Shannon code

$\ell_k = \lceil -\log_2 p_k \rceil$



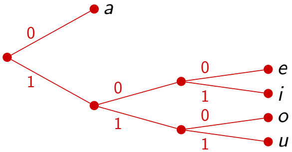
Shannon-Fano code

recursive equal probability split



optimal code

? test all full binary trees with given number of terminal nodes



Optimal Lossless Codes

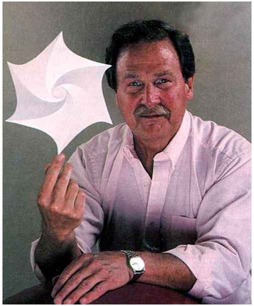
Optimal Prefix Code

- Any prefix code that achieves the minimum possible average codeword length $\bar{\ell}$ for a given pmf

➡ Each optimal prefix code is also an optimal uniquely decodable code

Construction of Optimal Prefix Codes ?

➡ Finite alphabets: **Huffman algorithm** (1952) yields a prefix code with minimum redundancy



1098 *PROCEEDINGS OF THE I.R.E.* *September*

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IRE

***Summary*—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.**

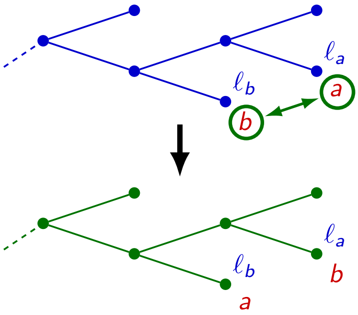
INTRODUCTION

ONE IMPORTANT METHOD of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which

will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

Exchanging Codewords in a Prefix Code

- Consider any prefix code for an alphabet $\mathcal{A}$ which includes two letters a and b with associated probabilities p_a and p_b
- a and b are associated with codewords of lengths ℓ_a and ℓ_b



What happens if we exchange the codewords for a and b?

➡ Obtain a new prefix code

➡ Average codeword length changes from $\bar{\ell}$ to $\bar{\ell}_{\text{new}}$

$$\begin{aligned}\bar{\ell}_{\text{new}} &= \bar{\ell} - (p_a\ell_a + p_b\ell_b) + (p_a\ell_b + p_b\ell_a) \\ &= \bar{\ell} - p_a(\ell_a - \ell_b) + p_b(\ell_a - \ell_b) \\ &= \bar{\ell} - (p_a - p_b)(\ell_a - \ell_b)\end{aligned}$$

➡ Different cases (for $p_a \geq p_b$):

(1) $p_a = p_b$ or $\ell_a = \ell_b$: ➡ $\bar{\ell}_{\text{new}} = \bar{\ell}$

(2) $p_a > p_b$ and $\ell_a < \ell_b$: ➡ $\bar{\ell}_{\text{new}} > \bar{\ell}$

(3) $p_a > p_b$ and $\ell_a > \ell_b$: ➡ $\bar{\ell}_{\text{new}} < \bar{\ell}$

In an optimal prefix code, we require

$$\forall a, b: \quad p_a > p_b \quad \Longrightarrow \quad \ell_a \leq \ell_b$$

Subset of Optimal Prefix Codes

Lemma (class of optimal prefix codes)

For any finite alphabet $\mathcal{A}$, there exists an optimal prefix code $\mathcal{C}$ with the following property:

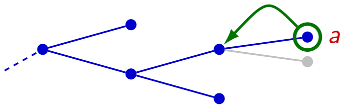
> There are two codewords that have the maximum length, differ only in the final bit, and correspond to the two least likely alphabet letters.

Proof

1. For each codeword of maximum length, the code includes a codeword of the same length that differs only in the final bit.

   In the binary tree representation, this means that the corresponding terminal node has a sibling.

   Codes without that property cannot be optimal, since a removal of the last bit of the considered codeword of maximum length would reduce the average codeword length without violating the prefix property.



$$
\begin{aligned}
\bar{\ell}_{\text{new}} &= \bar{\ell} - p_a \cdot \ell_a + p_a \cdot (\ell_a - 1) \\
&= \bar{\ell} - p_a \\
&< \bar{\ell}
\end{aligned}
$$

Subset of Optimal Prefix Codes (continued)

Lemma (class of optimal prefix codes)

For any finite alphabet $\mathcal{A}$, there exists an optimal prefix code $\mathcal{C}$ with the following property:

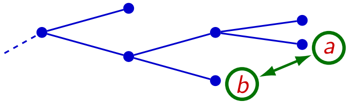
> There are two codewords that have the maximum length, differ only in the final bit, and correspond to the two least likely alphabet letters.

Proof

2 Two of the codewords of maximum length correspond to the two least likely alphabet letters.

For any two alphabet letters a and b with $p_a > p_b$, the codeword length must satisfy $\ell_a \leq \ell_b$. Otherwise, an exchange of the codewords would reduce the average codeword length without violating the prefix property.

Note: For two alphabet letters a and b with $p_a = p_b$, an exchange of the codewords does not modify the average codeword length.



$$\bar{\ell}_{\text{new}} = \bar{\ell} - (p_a - p_b)(\ell_a - \ell_b)$$

➡ $p_a > p_b,\ \ell_a > \ell_b$: $\bar{\ell}_{\text{new}} < \bar{\ell}$

➡ $p_a = p_b$: $\bar{\ell}_{\text{new}} = \bar{\ell}$

Subset of Optimal Prefix Codes (continued)

Lemma (class of optimal prefix codes)

For any finite alphabet $\mathcal{A}$, there exists an optimal prefix code $\mathcal{C}$ with the following property:

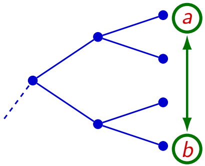
> There are two codewords that have the maximum length, differ only in the final bit, and correspond to the two least likely alphabet letters.

Proof

3. Two codewords of maximum length that differ only in the last bit are assigned to the two least likely alphabet letters.

Not necessary for optimality.

But we can always exchange any two codewords of maximum length (i.e., the same length ℓ_{max}) without impacting the average codeword length.



$$\bar{\ell}_{\text{new}} = \bar{\ell} - (p_a - p_b)(\ell_{\text{max}} - \ell_{\text{max}})$$
$$= \bar{\ell}$$

➡ **There exists on optimal prefix code $\mathcal{C}$ with the above stated property.**

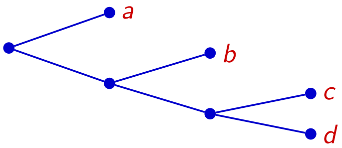
The Huffman Algorithm

Idea for Construction of Binary Code Tree

- Consider optimal prefix codes for which the two least likely symbols correspond to codewords of maximum length that differ only in the final bit

➡ Choose the two least likely symbols and create a parent node

➡ Repeat the procedure with the reduced alphabet



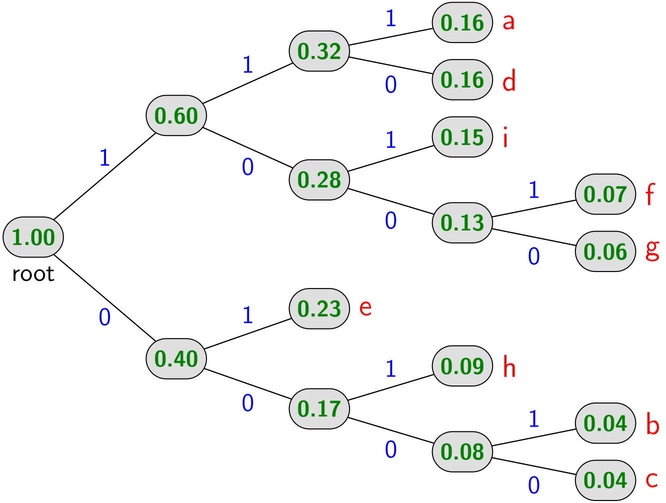
Huffman Algorithm (via construction of binary code tree)

1. Select the two letters a and b with the smallest probabilities p_a and p_b
2. Create a parent node for the two letters a and b in the binary code tree
3. Replace the letters a and b with a new letter with probability $p_a + p_b$
4. If the resulting new alphabet contains more than a single letter, repeat all previous steps with this alphabet
5. Convert the obtained binary code tree into a prefix code

Example: Construction of a Huffman Code

$$\bar{\ell} = 2.98$$
$$H(p) \approx 2.9405$$
$$\varrho \approx 0.0395 \;(1.34\,\%)$$

a_k	p_k	codewords
a	0.16	111
b	0.04	0001
c	0.04	0000
d	0.16	110
e	0.23	01
f	0.07	1001
g	0.06	1000
h	0.09	001
i	0.15	101



Average Codeword Length of Huffman Codes

All codes obtained by the Huffman algorithm are optimal prefix codes for the given pmf

- Rigorous proof (by induction) can be found in [Cover, Thomas: "Elements of Information Theory"]
- There are multiple Huffman codes (labeling of branches, same probability for multiple nodes)
- There might be optimal prefix codes that cannot be obtained by the Huffman algorithm

Bounds on Average Codeword Length

- Entropy is a lower bound for all lossless codes: $\bar{\ell} \geq H(p)$
- We showed that Shannon code has property: $\bar{\ell}_{\mathrm{Shannon}} < H(p) + 1$
- Huffman codes are optimal uniquely decodable codes: $\bar{\ell}_{\mathrm{opt}} \leq \bar{\ell}_{\mathrm{Shannon}}$

➡ **Average codeword length of Huffman codes (optimal codes) is bounded by**

$$H(p) \leq \bar{\ell}_{\mathrm{opt}} < H(p) + 1$$

Unix File Compression Utility "Pack"

Basic principle

- Design Huffman code for bytes of file to be compressed
- Transmit codeword table as part of the bitstream

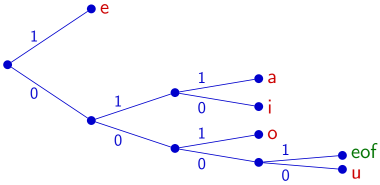
Encoding

- Determine relative frequencies of bytes in input file
- Generate Huffman code for these statistics
- Code "Huffman tree" at the beginning of bitstream
- Code bytes of input file using Huffman code

Decoding

- Decode "Huffman tree" from the compressed bitstream
- Read variable-length codewords and output bytes

using one-sided Huffman trees (with eof)



syntax of bitstream

16 bits	magic header ("1f 1e")
32 bits	number of symbols
8 bits	depth of Huffman tree
8 bits	number of terminal nodes in level 1
8 bits	number of terminal nodes in level 2
...	...
8 bits	character for 1-st terminal node
8 bits	character for 2-nd terminal node
...	...
sequence of codewords	

Huffman Coding in Practice

- Huffman coding is used in many compression tools and standards
- There are basically two variants:

1. **Sending Huffman tables as part of the bitstream**
   - ➡ Unix file compression utility **pack**
   - ➡ Image compression standard **JPEG**

2. **Huffman tables are fixed in standard**
   - ➡ Audio coding standard **MP3**
   - ➡ Video coding standards **MPEG-2 Video**, **H.263**, **MPEG-4 Visual**, **H.264 | AVC**

Lossless Coding of Sources with Memory

Example: Stationary Markov Process

- Consider stationary Markov source $\boldsymbol{S} = \{S_n\}$ with symbol alphabet $\mathcal{A} = \{\, a, b, c \,\}$

➡ Statistically properties are completely specified by conditional pmf $p(x \,|\, y) = P(S_n = x \,|\, S_{n-1} = y)$

x	$p(x \,\|\, a)$	$p(x \,\|\, b)$	$p(x \,\|\, c)$
a	0.90	0.15	0.25
b	0.05	0.80	0.15
c	0.05	0.05	0.60

calulate marginal pmf → (see exercise)

x	$p(x)$	Huffman
a	$^{29}/_{45}$	1
b	$^{11}/_{45}$	01
c	$^{5}/_{45}$	00

$$H(S_n) \approx 1.2575$$
$$\bar{\ell} \approx 1.3556 \quad (\,^{61}/_{45}\,)$$
$$\varrho \approx 0.0980 \quad (\,7.8\,\%\,)$$

- There are dependencies between successive source symbols !
- Can we exploit these dependencies for improving the coding efficiency ?

➡ **Idea**:
- Design Huffman code for each condition $(S_{n-1} = y)$
- Select codeword table for a symbol s_n based on previous symbol s_{n-1}

Conditional Huffman Codes

conditional Huffman code (assume "$S_{n-1} = a$" for first symbol)

	$S_{n-1} = a$		$S_{n-1} = b$		$S_{n-1} = c$	
x	$p(x \mid a)$	codeword	$p(x \mid b)$	codeword	$p(x \mid b)$	codeword
a	0.90	0	0.15	10	0.25	10
b	0.05	10	0.80	0	0.15	11
c	0.05	11	0.05	11	0.60	0
	$\bar{\ell}_{(S_{n-1}=a)} = 1.1$		$\bar{\ell}_{(S_{n-1}=b)} = 1.2$		$\bar{\ell}_{(S_{n-1}=c)} = 1.4$	

scalar Huffman code

x	$p(x)$	codeword
a	$^{29}/_{45}$	0
b	$^{11}/_{45}$	10
c	$^{5}/_{45}$	11
$\bar{\ell}_{\mathrm{scal}} = {}^{61}/_{45} \approx 1.3556$		

$H(S_n) \approx 1.2575$

Example: encoding: "abba" ➜ "0 10 0 10"
decoding: "010010" ➜ "abba"

➜ Average codeword length $\bar{\ell}_{\mathrm{cond}}$ for conditional Huffman code

$$\bar{\ell}_{\mathrm{cond}} = \sum_{y \in \mathcal{A}} p(y) \cdot \bar{\ell}_{(S_{n-1}=y)} = \frac{29}{45} \cdot 1.1 + \frac{11}{45} \cdot 1.2 + \frac{5}{45} \cdot 1.4 = \frac{521}{450} \approx 1.1578$$

➜ **Better than scalar Huffman code**: $\bar{\ell}_{\mathrm{cond}} < \bar{\ell}_{\mathrm{scal}}$

➜ We also have: $\bar{\ell}_{\mathrm{cond}} < H(S_n)$ ➜ **What's wrong ?**

Bounds for Conditional Huffman Codes

Consider an individual condition "$S_{n-1} = y$" (i.e., fixed value y)

- Code is constructed for the conditional probability masses $p(x \,|\, y) = \mathrm{P}(S_n = x \,|\, S_{n-1} = y)$

➡ Average codeword length $\bar{\ell}_{(S_{n-1}=y)}$ for given condition "$S_{n-1} = y$" is bounded by

$$\left(-\sum_{\forall x} p(x \,|\, y) \cdot \log_2 p(x \,|\, y) \right) \leq \bar{\ell}_{(S_{n-1}=y)} < \left(-\sum_{\forall x} p(x \,|\, y) \cdot \log_2 p(x \,|\, y) \right) + 1$$

$$H(S_n \,|\, S_{n-1} = y) \leq \bar{\ell}_{(S_{n-1}=y)} < H(S_n \,|\, S_{n-1} = y) + 1$$

➡ $H(S_n \,|\, S_{n-1} = y)$ is referred to as conditional entropy given the event $S_{n-1} = y$

Average codeword length $\bar{\ell}_{\mathrm{cond}}$ for conditional Huffman coding

- Each condition "$S_{n-1} = y$" occurs with probability $p(y) = \mathrm{P}(S_{n-1} = y) = \mathrm{P}(S_n = y)$

$$\left(\sum_{\forall y} p(y) \cdot H(S_n \,|\, S_{n-1} = y) \right) \leq \left(\sum_{\forall y} p(y) \cdot \bar{\ell}_{(S_{n-1}=y)} \right) < \left(\sum_{\forall y} p(y) \cdot H(S_n \,|\, S_{n-1} = y) \right) + \left(\sum_{\forall y} p(y) \cdot 1 \right)$$

$$H(S_n \,|\, S_{n-1}) \leq \bar{\ell}_{\mathrm{cond}} < H(S_n \,|\, S_{n-1}) + 1$$

Conditional Entropy

Conditional entropy of S_n given S_{n-1}

$$\begin{aligned} H(S_n \,|\, S_{n-1}) &= \sum_{\forall y} p(y) \cdot H(S_n \,|\, S_{n-1} = y) \\ &= \sum_{\forall y} p(y) \left(-\sum_{\forall x} p(x \,|\, y) \, \log_2 p(x \,|\, y) \right) \\ &= -\sum_{\forall x,y} p(x,y) \, \log_2 p(x \,|\, y) \end{aligned}$$

with $H(S_n|S_{n-1} = y) = -\sum_{\forall x} p(x|y) \log_2 p(x|y)$

remember: $p(x,y) = p(x \,|\, y)\, p(y)$

➡ $\boxed{H(S_n \,|\, S_{n-1}) = \mathrm{E}\Big\{ -\log_2 p(S_n \,|\, S_{n-1}) \Big\}}$

similarly: $\boxed{H(S_n) = \mathrm{E}\Big\{ -\log_2 p(S_n) \Big\}}$

Conditional Huffman Codes

$$H(S_n \,|\, S_{n-1}) \le \bar{\ell} < H(S_n \,|\, S_{n-1}) + 1$$

Scalar Huffman Codes

$$H(S_n) \le \bar{\ell} < H(S_n) + 1$$

Generalized Conditional Coding

Usage of two or more preceding symbols

- Design a Huffman code for each condition: "$S_{n-1} = a,\; S_{n-2} = b,\; \cdots$"

➡ Bounds on average codeword length: $H(S_n \,|\, S_{n-1},\, S_{n-2},\, \cdots) < \bar{\ell} < H(S_n \,|\, S_{n-1},\, S_{n-2},\, \cdots) + 1$

➡ Conditional entropy of S_n given $S_{n-1}, S_{n-2}, \cdots$

$$H(S_n \,|\, S_{n-1},\, S_{n-2},\, \cdots) = \mathrm{E}\Big\{ -\log_2 p(S_n \,|\, S_{n-1}, S_{n-2}, \cdots) \Big\} = -\sum_{x, y_1, y_2, \cdots} p(x, y_1, y_2, \cdots) \cdot \log_2 p(x \,|\, y_1, y_2, \cdots)$$

Arbitrary function of preceding symbols

- Design a Huffman code for possible value of: $C = f(S_{n-1}, S_{n-2}, \cdots)$

➡ Bounds on average codeword length: $H(S_n \,|\, C) < \bar{\ell} < H(S_n \,|\, C) + 1$

➡ Conditional entropy of S_n given $C = f(S_{n-1}, S_{n-2}, \cdots)$

$$H(S_n \,|\, C) = \mathrm{E}\Big\{ -\log_2 p(S_n \,|\, C) \Big\} = -\sum_{x,c} p_{SC}(x, c) \cdot \log_2 p_{S|C}(x \,|\, c)$$

Conditioning Does Not Increase Entropy

Consider general case with condition $C = f(S_{n-1}, S_{n-2}, \cdots)$

$$
\begin{aligned}
H(S_n \,|\, C) &= -\sum_{\forall s,c} p_{SC}(s,c) \, \log_2 p_{S|C}(s\,|\,c) \qquad \text{with} \quad p_{S|C}(s\,|\,c) = \frac{p_{SC}(s,c)}{p_C(c)} \\
&= -\sum_{\forall s,c} p_{SC}(s,c) \, \log_2 \left(\frac{p_{SC}(s,c)}{p_C(c)} \cdot \frac{p_S(s)}{p_S(s)} \right) \\
&= -\sum_{\forall s,c} p_{SC}(s,c) \, \log_2 p_S(s) - \sum_{\forall s,c} p_{SC}(s,c) \, \log_2 \left(\frac{p_{SC}(s,c)}{p_C(c)\, p_S(s)} \right) \\
&= -\sum_{\forall s} p_S(s) \, \log_2 p_S(s) - \sum_{\forall s,c} p_{SC}(s,c) \, \log_2 \left(\frac{p_{SC}(s,c)}{p_C(c)\, p_S(s)} \right) \\
&= H(S_n) - D(\, p_{SC} \,||\, p_S\, p_C \,) \qquad \text{remember:} \quad D(\, p \,||\, q \,) \ge 0
\end{aligned}
$$

➡ $\boxed{H(S_n \,|\, C) \le H(S_n)}$ $\Big($equality if and only if S_n and $C = f(S_{n-1}, S_{n-2}, \cdots)$ are independent$\Big)$

➡ **Conditioning does never increase entropy**

- Also: **Conditional coding does never increase average codeword length**

Example: Stationary Markov Process

Previous Example: Stationary Markov Source

conditional Huffman code

	$S_{n-1} = a$		$S_{n-1} = b$		$S_{n-1} = c$	
x	$p(x \mid a)$	codeword	$p(x \mid b)$	codeword	$p(x \mid c)$	codeword
a	0.90	0	0.15	10	0.25	10
b	0.05	10	0.80	0	0.15	11
c	0.05	11	0.05	11	0.60	0
	$\bar{\ell}_{(S_{n-1}=a)} = 1.1$		$\bar{\ell}_{(S_{n-1}=b)} = 1.2$		$\bar{\ell}_{(S_{n-1}=c)} = 1.4$	
	$H(S_n \mid a) \approx 0.5690$		$H(S_n \mid b) \approx 0.8842$		$H(S_n \mid c) \approx 1.3527$	

➡ average codeword length: $\bar{\ell}_{\mathrm{cond}} \approx 1.1578$

➡ conditional entropy: $H(S_n \mid S_{n-1}) \approx 0.7331$

scalar Huffman code

x	$p(x)$	codeword
a	$^{29}/_{45}$	0
b	$^{11}/_{45}$	10
c	$^{5}/_{45}$	11

$\bar{\ell}_{\mathrm{scal}} \approx 1.3556$

$H(S_n) \approx 1.2575$

➡ Conditioning reduces entropy: from 1.2575 to 0.7331

➡ Conditioning reduces average codeword length: from 1.3556 to 1.1578

Practical Example: Conditional Coding in H.264 | AVC (CAVLC)

Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = −1	nC = = −2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1

(*continued*)

Example: Binary Markov Process

Black and White Document Scans

- Binary random process $\boldsymbol{S} = \{S_n\}$:

$$\begin{aligned} S_n = 0 \quad &\rightarrow \quad \text{white sample} \\ S_n = 1 \quad &\rightarrow \quad \text{black sample} \end{aligned}$$

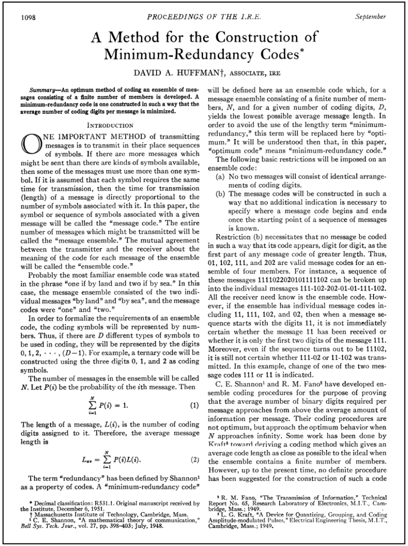
- Statistics measured over a large set of documents

$$\begin{aligned} p(0) &= 0.8 \\ p(0\,|\,0) &= 0.9 \end{aligned}$$

➡ Model: Stationary Markov process

➡ Determine remaining probabilities

$$\begin{aligned} p(1) &= 1 - p(0) && = 0.2 \\ p(1\,|\,0) &= 1 - p(0\,|\,0) && = 0.1 \\ p(0\,|\,1) &= \frac{p(1\,|\,0) \cdot p(0)}{p(1)} && = 0.4 \\ p(1\,|\,1) &= 1 - p(0\,|\,1) && = 0.6 \end{aligned}$$



1098 PROCEEDINGS OF THE I.R.E. September

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IRE

Summary—**An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.**

INTRODUCTION

ONE IMPORTANT METHOD of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be called the "message ensemble." The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the "ensemble code."

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this case, the message ensemble consisted of the two individual messages "by land" and "by sea", and the message codes were "one" and "two."

In order to formalize the requirements of an ensemble code, the coding symbols will be represented by numbers. Thus, if there are D different types of symbols to be used in coding, they will be represented by the digits $0, 1, 2, \cdots, (D-1)$. For example, a ternary code will be constructed using the three digits 0, 1, and 2 as coding symbols.

The number of messages in the ensemble will be called N. Let $P(i)$ be the probability of the ith message. Then

$$\sum_{i=1}^{N} P(i) = 1. \tag{1}$$

The length of a message, $L(i)$, is the number of coding digits assigned to it. Therefore, the average message length is

$$L_{av} = \sum_{i=1}^{N} P(i)L(i). \tag{2}$$

The term "redundancy" has been defined by Shannon[1] as a property of codes. A "minimum-redundancy code" will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 111, and 202 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 1111022020101111102 can be broken up into the individual messages 111-102-202-01-01-111-102. All the receiver need know is the ensemble code. However, if the ensemble has individual message codes including 11, 111, 102, and 02, then when a message sequence starts with the digits 11, it is not immediately certain whether the message 11 has been received or whether it is only the first two digits of the message 111. Moreover, even if the sequence turns out to be 11102, it is still not certain whether 111-02 or 11-102 was transmitted. In this example, change of one of the two message codes 111 or 11 is indicated.

C. E. Shannon[1] and R. M. Fano[2] have developed ensemble coding procedures for the purpose of proving that the average number of binary digits required per message approaches from above the average amount of information per message. Their coding procedures are not optimum, but approach the optimum behavior when N approaches infinity. Some work has been done by Kraft[3] toward deriving a coding method which gives an average code length as close as possible to the ideal when the ensemble contains a finite number of members. However, up to the present time, no definite procedure has been suggested for the construction of such a code

* Decimal classification: R531.1. Original manuscript received by the Institute, December 6, 1951.

† Massachusetts Institute of Technology, Cambridge, Mass.

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. Jour.*, vol. 27, pp. 398–403; July, 1948.

[2] R. M. Fano, "The Transmission of Information," Technical Report No. 65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass.; 1949.

[3] L. G. Kraft, "A Device for Quantizing, Grouping, and Coding Amplitude-modulated Pulses," Electrical Engineering Thesis, M.I.T., Cambridge, Mass.; 1949.

Example: Scalar and Conditional Coding

conditional Huffman code

x	$p(x\,\vert\,0)$ ($S_{n-1}=0$)	codeword ($S_{n-1}=0$)	$p(x\,\vert\,1)$ ($S_{n-1}=1$)	codeword ($S_{n-1}=1$)
0	0.9	0	0.4	0
1	0.1	1	0.6	1
	$\bar{\ell}_{(S_{n-1}=0)} = 1$		$\bar{\ell}_{(S_{n-1}=1)} = 1$	
	$H(S_n\,\vert\,0) \approx 0.4690$		$H(S_n\,\vert\,1) \approx 0.9710$	

➡ average codeword length: $\bar{\ell}_{\text{cond}} \approx 1$

➡ conditional entropy: $H(S_n\,|\,S_{n-1}) \approx 0.5694$

scalar Huffman code

x	$p(x)$	codeword
0	0.8	0
1	0.2	1

$\bar{\ell}_{\text{scal}} \approx 1$

$H(S_n) \approx 0.7219$

- Conditioning does not improve coding efficiency, even though $H(S_n\,|\,S_{n-1}) < H(S_n)$
- No codeword can be shorter than 1 bit ➡ $\bar{\ell} \geq 1$

➡ Problem for sources with probability masses $\gg 0.5$

How can we improve coding efficiency ? ➡ Joint coding of multiple symbols ?

Block Huffman Codes

Block Codes

- Code $N > 1$ successive symbols jointly (using a single codeword)

➡ Design code for N-dimensional joint pmf $p(x_1, x_2, \cdots, x_N) = \mathrm{P}(S_1 = x_1, S_2 = x_2, \cdots, S_N = x_N)$

➡ Optimal block code: Huffman algorithm

Block Huffman Coding for Black and White Document Scans

$N = 2$ symbols

$s_1 s_2$	$p(s_1, s_2)$	codewords
00	0.72	1
01	0.08	010
10	0.08	011
11	0.12	00

$$\bar{\ell}_2 = 1.44$$
$$\bar{\ell} = \bar{\ell}_2 / 2 = 0.72$$

$N = 3$ symbols

$s_1 s_2 s_3$	$p(s_1, s_2, s_3)$	codewords
000	0.648	1
001	0.072	000
010	0.032	01000
011	0.048	0101
100	0.072	001
101	0.008	01001
110	0.048	0110
111	0.072	0111

$$\bar{\ell}_3 = 1.952 \quad \Rightarrow \quad \bar{\ell} \approx 0.65$$

Block Entropy and Bounds on Average Codeword Length

Block Entropy

- Similar to marginal entropy, but for blocks of N symbols

→ **Block entropy for N successive symbols**

$$\boxed{H_N(\boldsymbol{S}) = H(S_1, S_2, \cdots, S_N) = \mathrm{E}\Big\{ -\log_2 p(S_1, S_2, \cdots, S_N) \Big\}}$$

$$= -\sum_{x_1, x_2, \cdots, x_N} p(x_1, x_2, \cdots, x_N) \cdot \log_2 p(x_1, x_2, \cdots, x_N)$$

Bounds for Block Huffman Codes

- Average codeword length $\bar{\ell}_N$ for N symbols: $H_N(\boldsymbol{S}) \le \bar{\ell}_N < H_N(\boldsymbol{S}) + 1$

→ Average codeword length $\bar{\ell}$ per symbol: $\dfrac{H_N(\boldsymbol{S})}{N} \le \bar{\ell} < \dfrac{H_N(\boldsymbol{S})}{N} + \dfrac{1}{N}$

Chain Rule for Entropies

- Conditional probabilities

$$\mathrm{P}(X \,|\, \mathcal{B}) = \frac{\mathrm{P}(X, \mathcal{B})}{\mathrm{P}(\mathcal{B})} \quad \Rightarrow \quad \mathrm{P}(X, \mathcal{B}) = \mathrm{P}(\mathcal{B}) \cdot \mathrm{P}(X \,|\, \mathcal{B})$$

➡ Chain rule for joint probability masses

$$\begin{aligned} p(x_1, x_2, \cdots, x_N) &= p(x_1, x_2, \cdots, x_{N-1}) \cdot p(x_N \,|\, x_1, x_2, \cdots, x_{N-1}) \\ &= p(x_1, x_2, \cdots, x_{N-2}) \cdot p(x_{N-1} \,|\, x_1, x_2, \cdots, x_{N-2}) \cdot p(x_N \,|\, x_1, x_2, \cdots, x_{N-1}) \\ &= p(x_1) \cdot p(x_2 \,|\, x_1) \cdot p(x_3 \,|\, x_1, x_2) \cdot \ldots \cdot p(x_N \,|\, x_1, x_2, \cdots, x_{N-1}) \end{aligned}$$

➡ **Chain rule for entropies**

$$\begin{aligned} H(S_1, S_2, \cdots, S_N) &= \mathrm{E}\Big\{ -\log_2 p(S_1, S_2, \cdots, S_N) \Big\} \\ &= \mathrm{E}\Big\{ -\log_2 \Big(p(S_1) \cdot p(S_2 \,|\, S_1) \cdot \ldots \cdot p(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \Big) \Big\} \\ &= \mathrm{E}\Big\{ -\log_2 p(S_1) \Big\} + \mathrm{E}\Big\{ -\log_2 p(S_2 \,|\, S_1) \Big\} + \ldots + \mathrm{E}\Big\{ -\log_2 p(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \Big\} \\ &= H(S_1) + H(S_2 \,|\, S_1) + H(S_3 \,|\, S_1, S_2) + \ldots + H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \end{aligned}$$

Increasing Block Size Does Not Increases Lower Bound

- Chain rule of entropies

$$\begin{aligned} H_N(\boldsymbol{S}) &= H(S_1) + H(S_2 \,|\, S_1) + H(S_3 \,|\, S_1, S_2) + \ldots + H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \\ &\geq N \cdot H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \qquad \text{(conditioning does not increase entropy)} \end{aligned}$$

- Consider one step of chain rule

$$\begin{aligned} H_N(S_1, S_2, \cdots, S_N) &= H_{N-1}(S_1, S_2, \cdots, S_{N-1}) + H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \\ N \cdot H_N(\boldsymbol{S}) &= N \cdot H_{N-1}(\boldsymbol{S}) + N \cdot H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \qquad \text{(multiplied by } N\text{)} \\ N \cdot H_N(\boldsymbol{S}) &\leq N \cdot H_{N-1}(\boldsymbol{S}) + H_N(\boldsymbol{S}) \qquad \text{(applied above inequality)} \\ (N-1) \cdot H_N(\boldsymbol{S}) &\leq N \cdot H_{N-1}(\boldsymbol{S}) \end{aligned}$$

➡ **Increasing block size never increases lower bound**

$$\boxed{\frac{H_N(\boldsymbol{S})}{N} \leq \frac{H_{N-1}(\boldsymbol{S})}{N-1}} \qquad \Big(\text{equality if and only if } \boldsymbol{S} \text{ is iid: } H(S_N \,|\, S_{N-1}, \cdots) = H(S_N)\Big)$$

Block Huffman Coding with Increasing Block Sizes

Example: Stationary Markov Source

conditional pmf

x	$p(x\|a)$	$p(x\|b)$	$p(x\|c)$
a	0.90	0.15	0.25
b	0.05	0.80	0.15
c	0.05	0.05	0.60

scalar Huffman coding:

$$\bar{\ell}_{\text{scal}} = 1.3556$$

conditional Huffman coding:

$$\bar{\ell}_{\text{cond}} = 1.1578$$

block Huffman coding

N	$\frac{H_N(\boldsymbol{S})}{N}$	$\bar{\ell} = \frac{\bar{\ell}_N}{N}$	number of codewords
1	1.2575	1.3556	3
2	0.9953	1.0094	9
3	0.9079	0.9150	27
4	0.8642	0.8690	81
5	0.8380	0.8462	243
6	0.8205	0.8299	729
7	0.8080	0.8153	2187
8	0.7987	0.8027	6561
9	0.7914	0.7940	19683

Example: Two Successive Quantization Indexes in MP3

Block Huffman Coding in MP3

- Joint coding of two successive quantization indexes
- Multiple Huffman tables specified in standard (designed offline)

Huffman code table 5

x	y	hlen	hcod
0	0	1	1
0	1	3	010
0	2	6	000110
0	3	7	0000101
1	0	3	011
1	1	3	001
1	2	6	000100
1	3	7	0000100
2	0	6	000111
2	1	6	000101
2	2	7	0000111
2	3	8	00000001
3	0	7	0000110
3	1	6	000001
3	2	7	0000001
3	3	8	00000000

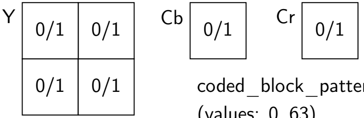
Huffman code table 6

x	y	hlen	hcod
0	0	3	111
0	1	3	011
0	2	5	00101
0	3	7	0000001
1	0	3	110
1	1	2	10
1	2	4	0011
1	3	5	00010
2	0	4	0101
2	1	4	0100
2	2	5	00100
2	3	6	000001
3	0	6	000011
3	1	5	00011
3	2	6	000010
3	3	7	0000000

Huffman code table 8

x	y	hlen	hcod
0	0	2	11
0	1	3	100
0	2	6	000110
0	3	8	00010010
0	4	8	00001100
0	5	9	000000101
1	0	3	101
1	1	2	01
1	2	4	0010
1	3	8	00010000
1	4	8	00001001
1	5	8	00000011
2	0	6	000111
2	1	4	0011
2	2	6	000101
2	3	8	00001110
2	4	8	00000111
2	5	9	000000011
3	0	8	00010011
3	1	8	00010001
3	2	8	00001111
3	3	9	000001101
3	4	9	000001010
3	5	10	0000000100
4	0	8	00001101
4	1	7	0000101
4	2	8	00001000
4	3	9	000001011
4	4	10	0000000101
4	5	10	0000000001
5	0	9	000001100
5	1	8	00000100
5	2	9	000000100
5	3	9	000000001
5	4	11	00000000001
5	5	11	00000000000

Example: Coded Block Pattern in MPEG-2



coded_block_pattern = xxxxxx (bit mask)
(values: 0..63)

Table B.9 – Variable length codes for coded_block_pattern

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51

(*continued*)

Lossless Source Coding Theorem

Entropy Rate

- Observation: Lower bound for block coding typically decreases with increasing N

➡ **Entropy rate**: Limit ($N \to \infty$) for lower bound of block coding

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

- The limit always exists for stationary random sources

Fundamental Lossless Source Coding Theorem

➡ Average codeword length for all lossless codes is bounded by

$$\bar{\ell} \geq \bar{H}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{H_N(\boldsymbol{S})}{N}$$

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

Entropy Rate for IID Sources

$$
\begin{aligned}
\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{1}{N}\, H(S_1, S_2, \cdots, S_N) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S_1) + H(S_2 \,|\, S_1) + H(S_3 \,|\, S_1, S_2) + \cdots + H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \Big) \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} H(S_k) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(N \cdot H(S) \Big) \\
&= H(S)
\end{aligned}
$$

Note: Block Huffman coding may still improve coding efficiency

Entropy Rate for Stationary Markov Sources

$$
\begin{aligned}
\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{1}{N}\, H(S_1, S_2, \cdots, S_N) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S_1) + H(S_2 \,|\, S_1) + H(S_3 \,|\, S_1, S_2) + \cdots + H(S_N \,|\, S_1, S_2, \cdots, S_{N-1}) \Big) \\
&= \lim_{N\to\infty} \frac{1}{N} \left(H(S_1) + \sum_{k=2}^{N} H(S_k \,|\, S_{k-1}) \right) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S) + (N-1) \cdot H(S_n \,|\, S_{n-1}) \Big) \\
&= \lim_{N\to\infty} \frac{H(S)}{N} + \lim_{N\to\infty} \frac{N-1}{N} \cdot H(S_n \,|\, S_{n-1}) \\
&= H(S_n \,|\, S_{n-1})
\end{aligned}
$$

Summary of Lecture: Huffman Codes

Huffman Algorithm

- Generates prefix codes with minimum redundancy for any given pmf
- Yields optimal uniquely decodable codes

Scalar Huffman codes

- Codeword table assigns codeword to each individual symbol
- Table size is equal to alphabet size

Conditional Huffman codes

- Separate codeword table for each possible condition (e.g., preceding symbol)
- Switch codeword tables during encoding and decoding

Block Huffman codes

- Code $N > 1$ successive symbols jointly
- Codeword table assigns codeword to combination of N successive symbols

Summary of Lecture: Entropy Measures and Bounds

Entropy Measures

- Scalar (or marginal) entropy: $H(S) = -\sum_a p(a) \cdot \log_2 p(a)$
- Conditional entropy: $H(S \mid C) = -\sum_{a,c} p(a,c) \cdot \log_2 p(a \mid c)$
- Block entropy: $H_N(\boldsymbol{S}) = -\sum_{x_1,\cdots,x_N} p(x_1,\cdots,x_N) \cdot \log_2 p(x_1,\cdots,x_N)$
- Entropy rate: $\bar{H}(\boldsymbol{S}) = \lim\limits_{N\to\infty} \frac{1}{N} H_N(\boldsymbol{S})$

Bounds for Lossless Coding

- Scalar Huffman coding: $H(S) \le \bar{\ell} < H(S) + 1$
- Conditional Huffman coding: $H(S \mid C) \le \bar{\ell} < H(S \mid C) + 1$
- Block Huffman coding: $\frac{1}{N} H_N(\boldsymbol{S}) \le \bar{\ell} < \frac{1}{N} H_N(\boldsymbol{S}) + \frac{1}{N}$
- **All lossless codes**: $\boxed{\bar{\ell} \ge \bar{H}(\boldsymbol{S})}$ (**fundamental lossless source coding theorem**)

Exercise 1: Huffman Code

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

x	$p_X(x)$
a	1/3
b	1/9
c	1/27
d	1/27
e	1/27
f	1/9
g	1/3

(a) Develop a Huffman code for the given pmf $p_X(x)$.

(b) Calculate the average codeword length of the developed Huffman code.

(c) Calculate the absolute and relative redundancy for the developed Huffman code.

Exercise 2: Huffman Codes and Entropy Measures

Let $\boldsymbol{Z} = \{Z_n\}$ be a binary iid process with alphabet $\{0, 1\}$ and pmf $\{0.5, 0.5\}$ (e.g., coin toss).

Let $\boldsymbol{X} = \{X_n\}$ be a random process given by $X_n = Z_{n-1} + Z_n$.

(a) Determine the marginal pmf $p_X(x)$ and the marginal entropy $H(X)$.

(b) Develop a scalar Huffman code and calculate its average codeword length.

(c) Determine the conditional pmf $p_{X_n|X_{n-1}}(x_n \,|\, x_{n-1})$ and the conditional entropy $H(X_n \,|\, X_{n-1})$.

(d) Develop a conditional Huffman code and calculate its average codeword length.

(e) Develop a block Huffman code for $N = 2$ symbols and calculate its average codeword length.

(f) Optional (more difficult):

- Derive a formula for the N-th order block entropy $H_N(X_n, \cdots, X_{n+N-1})$.
- Determine the entropy rate $\bar{H}(\boldsymbol{X})$.
- Is $\boldsymbol{X}$ a Markov process?

Exercise 3: Estimate Entropy Measures (Implementation Task)

Write a program (in a programming language of your choice) that estimates the following entropy measures based on the statistics of a given input file:

- Marginal entropy: $H(S_n)$
- 1-st order conditional entropy: $H(S_n \,|\, S_{n-1})$
- Block entropy of size $N = 2$: $H(S_n, S_{n+1})$ [calculate also $H(S_n, S_{n+1})/2$]

Assume that all files represent a sequence of 8-bit samples (i.e., each byte represents a symbol).

Test your program for the following sample files (from course website):

- white uniform noise: "whiteUniformNoise.raw"
- white Gaussian noise: "whiteGaussianNoise.raw"
- correlated Gaussian noise: "correlatedGaussianNoise.raw"
- English text file: "englishText.txt"
- 8-bit audio data: "audioData.raw"
- 8-bit image data: "imageData.raw"

What can you conclude about the potential to compress these files?

Exercise 4: Geometric Pmf (Optional)

Given is a Bernoulli process (binary iid process) $\mathbf{B} = \{B_n\} = \{B\}$ with the alphabet $\mathcal{A}_B = \{0, 1\}$ and the pmf $p_B(0) = p$, $p_B(1) = 1 - p$ with $0 < p < 1$.

Consider the random variable X that specifies the number of successive random variables B_n that have to be observed to get exactly one "1".

(a) Determine the pmf for X as function of p.

(b) Determine the entropy $H(B)$ as function of p.

(c) Determine the entropy $H(X)$ as function of $H(B)$ and p.

(d) What structure has an optimal scalar variable-length code for X and $p \leq 0.5$?

Calculate its average codeword length as function of p.

Calculate its relative redundancy as function of p.

Hints:

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